Practical Bayesian Neural Networks via Adaptive Optimization Methods

Samuel Kessler [*1] **Arnold Salas** [*1] **Vincent W. C. Tan** [1] **Stefan Zohren** [1] **Stephen J. Roberts** [1]

Abstract

We introduce a novel framework for the estimation of the posterior distribution over the weights of a neural network, based on a new probabilistic interpretation of adaptive optimisation algorithms such as ADAGRAD and ADAM. We demonstrate the effectiveness of our Bayesian ADAM method, BADAM, by experimentally showing that the learnt uncertainties correctly relate to the weights' predictive capabilities by weight pruning. We also demonstrate the quality of the derived uncertainty measures by comparing the performance of BADAM to standard methods in a Thompson sampling setting for multi-armed bandits, where good uncertainty measures are required for an agent to balance exploration and exploitation.

1. Introduction

Exact Bayesian inference over the weights of a neural network is intractable as the number of parameters is very large and the functional form of a neural network does not lend itself to exact integration. For this reason, much of the research in this area has been focused on approximation techniques. Approximate inference methods that scale to large models include the Laplace approximation together with various different approximations to the Hessian of the log posterior. For instance a diagonal approximation is remarkably effective for Continual Learning (CL) (Kirkpatrick et al., 2017), the Kronecker factored Laplace approximation introduces covariances between weights of the same layer of a Neural Network (NN) (Ritter et al., 2018). Variational inference (VI) methods have also been able to scale (Graves, 2011; Blundell et al., 2015). Recent advances in reparameterisation gradients (Kingma & Welling, 2014; Figurnov et al., 2018) and automatic differentiation libraries have seen enormous advances in VI approximate inference techniques. However, VI methods require longer training times, have twice the number of parameters and are difficult to implement in comparison to Maximum a Posteriori (MAP) estimation of NNs. Natural gradient variational inference methods of (Zhang et al., 2018; Khan et al., 2018) help overcome the problem of slow convergence and difficulty in implementation by utilising the information geometry of the underlying parameter manifold which results in simple update equations. Variational Dropout (Gal & Ghahramani, 2016), has found enormous success due to ease of implementation by interpreting a dropout regularised network as Gaussian Processes (GPs) on each layer of a NN. However, an explicit approximate posterior is marginalised automatically, and thus of limited use for CL, for instance.

In this paper, we develop a novel Bayesian approach to learning for neural networks, built upon adaptive methods such as ADAGRAD (Duchi et al., 2011), RMSPROP (Tieleman & Hinton, 2012) and ADAM (Kingma & Lei, 2015). Our method relies on a new probabilistic interpretation of adaptive algorithms, that effectively shows these can readily be utilised together with a Laplace approximation of the posterior. Our proposed algorithm is also similar in spirit to the work of (Khan et al., 2018), but in contrast, their algorithm performs natural gradient variational inference, implemented within ADAM via weight perturbation of the gradient evaluation. Examples of the predictive uncertainties of our method and other common methods for approximate inference of Bayesian Neural Networks (BNNs) for a simple regression task are show in Figure 1 and discussed in Section 4.

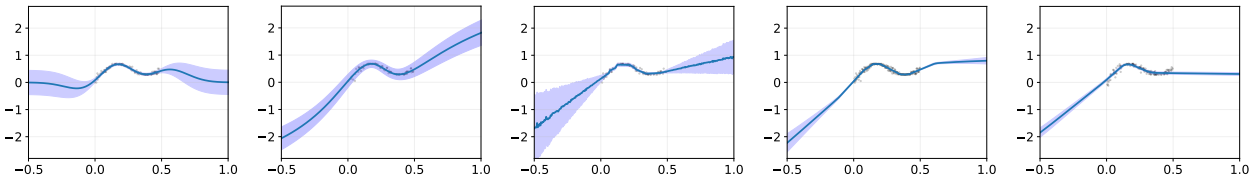
2. Preliminaries

Problem setup. Let $f(\theta)$ be a noisy objective function, a scalar function that is differentiable w.r.t. the parameters $\theta \in \Theta$, where Θ denotes the parameter space.We are interested in minimising the expected value of this function, $\mathbb{E}[f(\theta)]$, w.r.t. its parameters θ. Let $f_1(\theta), \ldots, f_T(\theta)$ denote the realisations of the stochastic function at the subsequent time steps $t \in [T]$. The stochastic nature may arise from the evaluation of the function at random subsamples (minibatches) of datapoints, or from inherent function noise.

The simplest algorithm for this setting is the standard online gradient descent algorithm (Zinkevich, 2003), which moves the current estimate θ_t of θ in the opposite direction of the

[1]Oxford University. Correspondence to: Samuel Kessler <skessler@robots.ox.ac.uk>.

Presented at the ICML 2020 Workshop on Uncertainty and Robustness in Deep Learning.



(a) GP (b) Badam (c) MC dropout (d) BBB (e) Vadam

Figure 1. Qualitative comparison of the predictive uncertainties of our method and our principle baselines. BADAM, like the Laplace approximation, places some probability mass where data has been observed and less further away (Ritter et al., 2018).

last observed (sub)gradient value $g_t = \nabla f_t(\theta_t)$, i.e.,

$$\theta_{t+1} = \theta_t - \eta_t g_t, \tag{1}$$

where $\eta_t > 0$ is an adaptive learning rate that is typically set to $\eta/\sqrt{t}$, for some positive constant η. While the decreasing learning rate is required for convergence, such an aggressive decay typically translates into poor empirical performance.

Generic adaptive optimisation methods. We now present a framework that contains a wide range of popular adaptive methods as special cases, and highlight their differences. The presentation here follows closely that of (Reddi et al., 2018). The update rule of this generic class of adaptive methods can be compactly written in the form

$$\theta_{t+1} = \theta_t - \eta_t V_t^{-1/2} m_t, \tag{2}$$

where m_t and V_t are estimates of the (sub)gradient and inverse Hessian, respectively, of the functions $f_t(\cdot)$, based on observations up to and including iteration t. In other words, they are functions of the (sub)gradient history $g_{1:t} \equiv g_1, \ldots, g_t$, which we express as

$$m_t = \widehat{g}_t(g_{1:t}), \qquad V_t^{1/2} = \widehat{H}_t(g_{1:t}), \tag{3}$$

where $\widehat{g}_t(\cdot)$ and $\widehat{H}_t(\cdot)$ denote approximation functions for the (sub)gradient and Hessian of the loss function at iteration t, respectively. The corresponding procedure is repeated until convergence.

We first observe that the standard online gradient descent (OGD) algorithm arises as a special case of this framework if we use:

$$m_t = g_t, \qquad V_t = \mathbb{1}\,. \tag{4}$$

The key idea of adaptive methods is to choose estimator functions appropriately so as to entail good convergence. For instance, ADAGRAD (Duchi et al., 2011), employs the following estimator functions:

$$m_t = g_t, \qquad V_t = \frac{1}{t}\mathrm{diag}\left(\sum_{i=1}^{t} g_i g_i^\top\right). \tag{5}$$

In contrast to the learning rate of $\eta/\sqrt{t}$ in OGD with learning-rate decay, such a setting effectively implies a modest learning-rate decay of $\eta/\sqrt{\sum_i g_{i,j}^2}$ for $j \in [d]$. When the gradients are sparse, this can potentially lead to huge gains in terms of convergence (see (Duchi et al., 2011)).

Adaptive methods based on EWMA. Exponentially weighted moving average (EWMA) variants of ADAGRAD are popular in the deep learning community. ADADELTA (Zeiler, 2012), RMSPROP (Tieleman & Hinton, 2012) and ADAM (Kingma & Lei, 2015) are some prominent algorithms that fall in this category. The key difference between these algorithms and ADAGRAD is that they use an EWMA as the function V_t instead of a simple average. ADAM, a particularly popular variant, is based on the following estimator functions:

$$\begin{aligned} m_t &= \frac{1-\beta_1}{1-\beta_1^t}\sum_{i=1}^{t}\beta_1^{t-i} g_i, \\ V_t &= \frac{1-\beta_2}{1-\beta_2^t}\mathrm{diag}\left(\sum_{i=1}^{t}\beta_2^{t-i} g_i g_i^\top\right), \end{aligned} \tag{6}$$

where $\beta_1, \beta_2 \in [0, 1)$ are exponential decay rates. RMSPROP, which appeared in an earlier unpublished work (Tieleman & Hinton, 2012), is essentially a variant of ADAM with $\beta_1 = 0$. In practice, especially in deep-learning applications, the momentum term arising due to non-zero β_1 appears to significantly boost performance.

More recently, (Khan et al., 2018) are able to derive similar EWMA update rules by performing natural gradient conjugate computations based on (Khan, 2017) to Bayesian neural networks using a Gaussian approximation with a mean-field approximation for the Hessian along with squared gradients. By exploiting the duality within the exponential family distributions, simple updates for variational BNNs with fast convergence are derived.

Bayesian neural networks. A Bayesian neural network (BNN) is a neural network where the weights, θ are random variables. Consider an i.i.d. data set of N feature vectors $x_1, \ldots, x_N \in \mathbb{R}^d$, with a corresponding set of out-

puts $\{y_1, \ldots, y_N\} \in \mathbb{R}$. For illustration purposes, we shall suppose that the likelihood for each datapoint is Gaussian, with mean $\mathrm{NN}(x, \theta)$ and variance σ_o^2:

$$p(y_n|x_n, \theta, \sigma^2) = \mathcal{N}(y_n|\mathrm{NN}(x_n, \theta),\, \sigma_o^2). \quad (7)$$

Similarly, we shall choose a prior distribution over the weights θ that is Gaussian of the form $p(\theta|\alpha) = \mathcal{N}(\theta|0,\, \sigma^2\, \mathbb{1})$. The resulting posterior distribution is then

$$p(\theta|\mathcal{D}, \sigma^2, \sigma_n^2) \propto p(\theta|\sigma^2)p(\mathcal{D}|\theta, \sigma_o^2), \quad (8)$$

as a consequence of the nonlinear dependence of $\mathrm{NN}(x, \theta)$ on θ, will be non-Gaussian. However, we can find a Gaussian approximation by using the Laplace approximation (MacKay, 1992).

3. Probabilistic Interpretation of Adaptive Methods

Consider a second-order Taylor expansion of the log-posterior $p(\theta|\mathcal{D})$ around the MAP θ_t:

$$\log p(\theta|\mathcal{D}) \approx \log p(\theta_t|\mathcal{D}) + \frac{1}{2}(\theta - \theta_t)^\top H(\theta - \theta_t). \quad (9)$$

The gradient upon convergence is zero and so the first order gradient in the Taylor expansion drops out. $H = \mathbb{E}\left[\nabla^2 \log p(\theta|\mathcal{D})|_{\theta=\theta_t}\right]$ and

$$\nabla^2 \log p(\theta|\mathcal{D})|_{\theta=\theta_t} = \nabla^2 \log p(\mathcal{D}|\theta)|_{\theta=\theta_t} + \sigma^2\, \mathbb{1} \quad (10)$$

where the first term on the r.h.s is a Hessian of a mean-squared error loss function, $\log p(\mathcal{D}|\theta) \propto \frac{1}{2}\sum_n (\mathrm{NN}(x_n, \theta) - y_n)^2$, where σ_o^2 is treated a nuisance parameter and set to 1. This Hessian can be approximated by the Gaussian-Newton matrix (GGN) (MacKay, 1992), (this approximation is introduced more formally in Section B). The second term in Equation (10) is the contribution from the Gaussian prior. The likelihood term is scaled by $N = |\mathcal{D}|$ as most implementations will take an average of the log-likelihood. In practice, the GGN matrix requires iterating over the training data and calculating and manipulating Jacobians which are expensive for large datasets and over-parameterized NNs, notice that one can replace it by the approximation from adaptive optimisation methods which can also be used to train the NN:

$$\mathbb{E}\left[\nabla^2 \log p(\theta|\mathcal{D})|_{\theta=\theta_t}\right] \approx V_t^{1/2}. \quad (11)$$

Exponentiation of Equation (9) yields a Gaussian functional form in θ, and hence a Gaussian approximate posterior or Laplace approximation centred on the MAP:

$$q(\theta|\mathcal{D}, g_{1:t}, \eta_t, \omega) = \mathcal{N}\left(\theta \,|\, \theta_t,\, \left(NV_t^{1/2}\right)^{-1}\right), \quad (12)$$

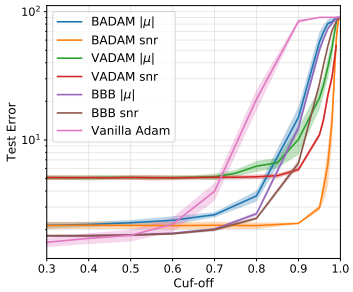


Figure 2. Weight pruning test accuracies. BADAM produces high quality uncertainties and obtains high accuracies.

where $\left(NV_t^{1/2}\right)^{-1}$ is guaranteed to be positive semi-definite, ω is a vector of hyperparameters[1] other than the learning rate, if any, that govern the underlying adaptive method. We summarise the practical implementation of this procedure through our proposed algorithm, BADAM in Section A.

An advantage from this proposal is that $V_t^{1/2}$ is readily available from the adaptive method, which offers computational and implementation benefits. In particular, the use of ADAM's EWMA estimates for m_t and V_t, which enable us to filter out the noise arising from the stochastic nature of the gradients, while at the same time accounting for changes in these quantities over different areas of the loss landscape. In contrast, using ADAGRAD's estimate for V_t would imply that the resulting covariance matrix in Equation (12) is constant over the trajectory of the loss landscape: a rather severe assumption. In practice, we can run ADAM and then get $V_t = \mathrm{diag}(v_t)$ post-hoc to get a cheap approximate posterior. By re-using the curvature estimates we circumvent the need to calculate an approximation of the Hessian such as the GGN approximation which involves iterating over the training dataset and evaluating Jacobians.

The estimate of the Hessian is a root mean square of gradients: $V_t^{1/2}$, contrast this with the GGN approximation of a typical Laplace approximation which does not contain the square root. The significance of the square root term is two-fold. Firstly, it ensures a reduction of the condition number for our estimate of the covariance in the high-dimensional setting. Thus, the diagonal estimate for the Hessian of the objective function has smaller eigenvalues, ensuring a more stable covariance upon inversion. Secondly, it ensures the Gaussian approximate posterior, $q(\theta|\mathcal{D}, g_{1:t}, \eta_t, \omega)$ has a more conservative covariance matrix and uncertainty estimate than the traditional Laplace which can place probability mass where the true posterior has none (Ritter et al., 2018).

[1]For example, $\omega = \emptyset$ in the case of ADAGRAD, whereas $\omega = (\beta_1, \beta_2)$ for ADAM.

Table 1. Cumulative rewards normalised by the optimal reward ± one std. The results are taken from 10 runs. VADAM outperforms on most datasets, then BADAM performs better than the other approximate inference methods.

	Mushroom	Financial	Jester	Statlog	Adult	Covertype	Census
BADAM	**0.74±0.09**	0.74±0.03	0.48±0.01	**0.97±0.02**	0.13±0.01	0.58±0.01	0.36±0.06
MC DROPOUT	0.68±0.11	0.68±0.02	0.48±0.01	0.96±0.02	0.13±0.01	0.57±0.01	0.44±0.03
BBB	0.60±0.15	0.21±0.13	0.46±0.02	0.71±0.17	0.10±0.04	0.39±0.06	0.38±0.02
VADAM	**0.76±0.03**	0.81±0.00	**0.48±0.01**	**0.99±0.00**	**0.26±0.02**	**0.74±0.01**	**0.65±0.00**
Greedy	0.64±0.18	**0.86±0.03**	**0.50±0.01**	**0.97±0.03**	0.16±0.01	0.62±0.03	0.44±0.02
Uniform	−0.93±0.09	−0.01±0.01	0.34±0.00	0.14±0.00	0.07±0.00	0.14±0.00	0.11±0.00

4. Results

We demonstrate the approximate posterior's effectiveness by pruning on MNIST and in a contextual bandit setting. We present our method in comparison to BBB, MC DROPOUT and VADAM.

In Figure 1 we introduced our method's predictive distribution for a simple regression experiment. The predictive distribution for BADAM is similar to a standard Laplace approximation in that more probability mass is placed is places close to the observations than the other methods and uncertainties widen gradually (Ritter et al., 2018). We use the Linearised Laplace trick for a stable predictive distribution (Foong et al., 2019). BBB and VADAM are expensive to run and at convergence don't produce large uncertainties far away from observations. VADAM, also requires a careful choice of an initial precision of the posterior to initialise the algorithm and its performance is sensitive to different specifications of the initial precision, which makes it difficult to train in practice.

Classification on MNIST. To assess the quality of the obtained uncertainties we follow weight-pruning on MNIST. Given a posterior mean μ and posterior variance σ^2, we sort the weights by their signal-to-noise ($|\mu|/\sigma$) ratio and discard the fraction p of weights with the lowest values, by setting these weights to zero. As a baseline, we perform pruning on a model with $\Sigma = 1$. BADAM can produce high quality uncertainties seen as the pruning via signal-to-noise drop in the accuracy is more robust to pruning than all other baselines and it has a high accuracy for without pruning. Experimental details can be found in Section D.2.

Contextual Bandits. We demonstrate the effectiveness of the BADAM uncertainties by using the BNN in a contextual multi-armed bandit setting where the agent requires a good measure uncertainty for decision making. The contextual multi-armed bandit problem proceeds as follows, at time t our agent will receive a context $X_t \in \mathbb{R}^d$ and will need to decide which action $a_t \in \mathcal{A}$ to pick to maximise a total reward $r = \sum_{t=1}^{T} r_t$. Our agent learns a function $f : (X_t, a_t) \to r_t \in \mathbb{R}$. Thompson sampling provides a Bayesian framework for our agent to manage the exploration exploitation dilemma (Thompson, 1933). In Thompson sampling at each iteration our model samples from its prior and then greedily selects the action which maximises the reward. The model then receives feedback for the selected decision and updates its prior. No feedback is observed for actions which are not selected. The challenge in this setting is that at time t our agent's approximate prior is used to estimate r_t and pick an arm. The rewards r_j, $j \leq t$ are not i.i.d. and this feedback loop together with an approximate posterior can lead to disagreements between the model's posterior and the true posterior (Riquelme et al., 2018). This in turn can result in large cumulative regrets. Since the posterior uncertainty construction and representation learning in the BADAM algorithm are separate we postulate good performance (Riquelme et al., 2018). Details of the experimental setup can be found in Section D.3.

The results in Table 1 show that the rewards from BADAM are comparable to MC DROPOUT and outperforms BBB. VADAM performs best, at each iteration of VADAM as there is an injection of noise which could increase exploration and explain the good performance. The greedy bandit performs well as in (Riquelme et al., 2018), however it underperforms BADAM on the Mushroom bandit.

5. Conclusion

In this paper, we introduce a straightforward and cheap approach to posterior inference in BNNs, derived from a new probabilistic interpretation of adaptive methods. In particular, we discuss how to refine this framework to ADAM. Finally, we demonstrate empirically the performance of the posterior distribution on MNIST classification and on contextual bandit problems. BADAM is computationally efficient, like MC DROPOUT, does not require post-hoc computation of a covariance matrix like the Laplace approximation and has the additional advantage of producing an explicit approximate posterior distribution like (Blundell et al., 2015; Khan et al., 2018), without the complexity of implementing and tuning stochastic variational inference.

References

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight Uncertainty in Neural Networks. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

Duchi, J. C., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

Figurnov, M., Mohamed, S., and Mnih, A. Implicit Reparameterization Gradients. In *Neural Information Processing Systems*, 2018.

Foong, A. Y. K., Li, Y., Miguel Hernández-Lobato, J., and Turner, R. E. 'In-Between' Uncertainty in Bayesian Neural Networks. In *ICML 2019 Workshop on Uncertainty and Robustness in Deep Learning*, 2019.

Gal, Y. and Ghahramani, Z. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *International Conference on Machine Learning*, 2016.

Graves, A. Practical Variational Inference for Neural Networks. In *Advances in Neural Information Processing Systems 24*, 2011.

Khan, L. Conjugate-Computation Variational Inference : Converting Variational Inference in Non-Conjugate Models to Inferences in Conjugate Models. 2017.

Khan, M. E., Nielsen, D., Tangkaratt, V., Lin, W., Gal, Y., and Srivastava, A. Fast and Scalable Bayesian Deep Learning by Weight-Perturbation in Adam. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

Kingma, D. P. and Lei, J. B. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, 2015.

Kingma, D. P. and Welling, M. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations*, 2014.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. Overcoming catastrophic forgetting in neural networks. *arXiv: 1612.00796v2*, 2017.

Lawrence, N. D. *Variational Inference in Probabilistic Models*. PhD thesis, 2000.

MacKay, D. J. C. A Practical Bayesian Framework for Backpropagation Networks. *Neural Computation*, 4:448–472, 1992.

Martens, J. New insights and perspectives on the natural gradient method. Technical report, 2014.

Reddi, S. J., Kale, S., and Kumar, S. On the Convergence of Adam and Beyond. In *6th International Conference on Learning Representations*, 2018.

Riquelme, C., Tucker, G., and Snoek, J. Deep Bayesian Bandits Showdown: An Empirical Comparison of Bayesian Deep Networks for Thompson Sampling. *International Conference on Learning Representations, ICLR.*, 2018.

Ritter, H., Botev, A., and Barber, D. A Scalable Laplace Approximation For Neural Networks. In *International Conference on Learning Representations*, 2018.

Thompson, W. R. On The Likelihood That One Unknown Probability Exceeds Another in View of the Evidence of Two Samples. *Biometrika*, 25(3-4):285–294, 12 1933. ISSN 0006-3444. doi: 10.1093/biomet/25.3-4.285.

Tieleman, T. and Hinton, G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4 (2):26–31, 2012.

Zeiler, M. D. ADADELTA: An Adaptive Learning Rate Method. *CoRR*, abs/1212.5701, 2012.

Zhang, G., Sun, S., Duvenaud, D., and Grosse, R. Noisy Natural Gradient as Variational Inference. In *International Conference on Machine Learning*, 2018.

Zinkevich, M. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning*, pp. 928–936, 2003.

Appendices

A. BADAM Algorithm

The specific approach whereby we can use ADAM to obtain cheap estimates of the posterior are illustrated in Algorithm 1.

Algorithm 1 BADAM: Bayesian Learning of Neural Networks via ADAM

Input: $\theta_1 \in \mathbb{R}^d$, global learning rate η, exponential decay rates β_1, β_2, constant ϵ
Set $m_0 = v_0 = 0$
for $t = 1$ **to** $T - 1$ **do**
$g_t = \nabla f_t(\theta_t)$
$m_t = \frac{\beta_1 m_{t-1} + (1-\beta_1) g_t}{1-\beta_1^t}$ and $v_t = \frac{\beta_2 v_{t-1} + (1-\beta_2) g_t g_t^\top}{1-\beta_2^t}$
$\theta_{t+1} = \theta_t - \eta m_t / (v_t^{1/2} + \epsilon)$ (element-wise division)
end for
Output: final weight distribution $\mathcal{N}\left(\theta \,|\, \theta_t,\ (N \mathrm{diag}(v_t^{1/2}))^{-1}\right)$

B. The Generalised Gauss Newton Matrix

The Hessian of the log-likelihood in Equation (10) is the Hessian of a mean-squared error loss function, $\log p(\mathcal{D}|\theta) \propto \frac{1}{2}\sum_n (\mathrm{NN}(x_n, \theta) - y_n)^2$, ($\sigma_o^2$ is treated a nuisance parameter and set to 1). The Hessian can be written as:

$$\nabla_\theta^2 \log p(\mathcal{D}|\theta) = \sum_n \nabla_\theta \mathrm{NN}(x_n, \theta) \nabla_\theta \mathrm{NN}(x_n, \theta)^\top + \sum_n (\mathrm{NN}(x_n, \theta) - y_n) \nabla_\theta^2 \mathrm{NN}(x_n, \theta), \tag{13}$$

The second term involves a residual term which is assumed to be small for NNs if it is able to fit the data well. Hence the Hessian can be approximated by a sum of outer products of Jacobians of the NN predictions. Hence taking expectations like in Equation (10):

$$\mathbb{E}_{p_\theta(y|x_n)}\left[\nabla_\theta^2 \log p(\mathcal{D}|\theta)\right] = \mathbb{E}_{p_\theta(y|x_n)}\left[\sum_n \nabla_\theta \mathrm{NN}(x_n, \theta) \nabla_\theta \mathrm{NN}(x_n, \theta)^\top\right] \tag{14}$$

$$= \sum_n \nabla_\theta \mathrm{NN}(x_n, \theta) \nabla_\theta \mathrm{NN}(x_n, \theta)^\top, \tag{15}$$

since the Jacobians do not depend on y and then

$$H = \sum_n \nabla_\theta \mathrm{NN}(x_n, \theta) \nabla_\theta \mathrm{NN}(x_n, \theta)^\top + \sigma^2 \mathbb{1}\,. \tag{16}$$

This is referred to as the Gaussian-Newton matrix (GGN) (MacKay, 1992), its inverse is the covariance of traditional Laplace approximation of the posterior of a BNN. The GGN matrix corresponds Fisher information for exponential family distributions (Martens, 2014).

C. Predictive Distributions

The predictive distribution over unseen data x^* can be obtained through sampling from the posterior:

$$p(y^*|x^*, \mathcal{D}) = \int p(y^*|x^*, \theta) p(\theta|\mathcal{D}) d\theta \approx \frac{1}{M}\sum_{i=1}^M p(y^*|x^* \theta^{(i)}), \tag{17}$$

where $\theta^{(i)} \sim q(\theta|\mathcal{D})$. Sampling from the Laplace approximation can cause serious underfitting (Lawrence, 2000). Regularisation over a validation set can be performed to mitigate against underfitting by tuning N and the prior variance σ^2. In practice the scaling N can be treated as a hyperparameter (Kirkpatrick et al., 2017). Alternatively N can be treated as "pseudo-observations" (Ritter et al., 2018). By adding dropout too neural network each new pass through the data will change the loss function landscape. With this in mind a heuristic it is useful to regularise the posterior by tuning N according to a validation dataset. Alternatively the predictive distribution can be approximated by linearizing the output about the MAP to generate stable, θ_t (Foong et al., 2019). The predictive distribution for BADAM uses this see Figure 1.

D. Experimental details

D.1. Toy Regression Experiments

Data generation. Data is generated from the function $y = x + 0.3\sin(2\pi(x+\epsilon)) + 0.3\sin(4\pi(x+\epsilon)) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, 0.02)$. Data for training is sampled from the range $(0.0, 0.5)$ while data used for evaluating the testing was sampled from the range $(-0.5, 1.2)$. The training set has 200 samples.

Model architecture. A two layer NN with ReLU activations and hidden state sixes of 50 neurons are used. For training BADAM we use and L^2 regularisation size of 10^{-4} and $N = 200$ (in equation 12) is the number of points in the training set . the predictive distribution uses the Linearised-Laplace to obtain a closed form Gaussian solution to the predictive distribution (Foong et al., 2019). The BADAM network is trained for $20,000$ epochs with a learning rate of 0.001.

D.2. MNIST Weight Pruning Experiments

The BADAM network 2 layers of size 400, uses dropout with a rate of 0.2. For weight pruning it is the ordering of the weights which matters and so N simply acts to scale the denominators of the snr, $|\mu|/\sigma$, therefore $N = 60,000$ which is the training set size. Interestingly setting no L^2 regularisation produced the best uncertainties. We found that using a larger value of $\beta_2 = 0.99999$ produces good uncertainties. We run BADAM with default settings for 100 epochs to achieve a good solution.

D.3. Contextual Bandits Experiments

Model step. We use the experimental setup described in an implementation provided by (Riquelme et al., 2018) for evaluation of our BADAM algorithm. A detailed description of the datasets used for our multi-armed bandit experiments can also be found in the appendix of (Riquelme et al., 2018). We compare to BBB, MC DROPOUT, VADAM an NN which greedily picks each action (named Greedy in Tables 1), and as a baseline an agent which uniformly samples actions. The neural networks architectures used for all networks are the same: 2 layers with 100 units each and ReLU activations, they regress contexts in $\mathbb{R}^d$ to outputs in $\mathbb{R}^{|\mathcal{A}|}$. BADAM weights are initialized with $U[-0.3, 0.3]$, and use gradient clipping such that the 2-norm of the gradients are not greater than 5. In terms of hyperparameters, all networks use an initial learning rate of 0.1 with an inverse decaying schedule. The number of points seen N is set dynamically as the $t \times$ batch_size, where t is the number of rounds of the bandit.

Multi-armed bandits experimental details. The experimental setup for the multi-armed bandits proceeds as follows: at each round a new context from the dataset is presented to the bandit algorithm, we go through the dataset once. Each action is initially selected 3 times so that each agent has some initial information to learn from. Subsequently, actions are greedily chosen, and only the reward for the chosen action will be backpropagated upon training. In terms of training, all observed contexts, actions and rewards are stored in a buffer. The buffer is sampled to create batches used for training. Training occurred every $t_f = 20$ rounds for $t_s = 50$ minibatches using a batch size of 512 for all neural bandits.

The full datasets are used for all bandits experiments apart from the Census and Covertype datasets which are very large, we use a subset of $n = 10000$ points from these two datasets.