Weight Uncertainty in Neural Networks

Charles Blundell CBLUNDELL@GOOGLE.COM
Julien Cornebise JUCOR@GOOGLE.COM
Koray Kavukcuoglu KORAYK@GOOGLE.COM
Daan Wierstra WIERSTRA@GOOGLE.COM
Google DeepMind

Abstract

We introduce a new, efficient, principled and backpropagation-compatible algorithm for learning a probability distribution on the weights of a neural network, called *Bayes by Backprop*. It regularises the weights by minimising a compression cost, known as the variational free energy or the expected lower bound on the marginal likelihood. We show that this principled kind of regularisation yields comparable performance to dropout on MNIST classification. We then demonstrate how the learnt uncertainty in the weights can be used to improve generalisation in non-linear regression problems, and how this weight uncertainty can be used to drive the exploration-exploitation trade-off in reinforcement learning.

1. Introduction

Plain feedforward neural networks are prone to overfitting. When applied to supervised or reinforcement learning problems these networks are also often incapable of correctly assessing the uncertainty in the training data and so make overly confident decisions about the correct class, prediction or action. We shall address both of these concerns by using variational Bayesian learning to introduce uncertainty in the weights of the network. We call our algorithm *Bayes by Backprop*. We suggest at least three motivations for introducing uncertainty on the weights: 1) regularisation via a compression cost on the weights, 2) richer representations and predictions from cheap model averaging, and 3) exploration in simple reinforcement learning problems such as contextual bandits.

Various regularisation schemes have been developed to prevent overfitting in neural networks such as early stopping, weight decay, and dropout (Hinton et al., 2012). In this work, we introduce an efficient, principled algorithm for regularisation built upon Bayesian inference on the weights of the network (MacKay, 1992; Buntine and Weigend, 1991; MacKay, 1995). This leads to a simple approximate learning algorithm similar to backpropagation (LeCun, 1985; Rumelhart et al., 1988). We shall demonstrate how this uncertainty can improve predictive performance in regression problems by expressing uncertainty in regions with little or no data, how this uncertainty can lead to more systematic exploration than ϵ-greedy in contextual bandit tasks.

All weights in our neural networks are represented by probability distributions over possible values, rather than having a single fixed value as is the norm (see Figure 1). Learnt representations and computations must therefore be robust under perturbation of the weights, but the amount of perturbation each weight exhibits is also learnt in a way that coherently explains variability in the training data. Thus instead of training a single network, the proposed method trains an ensemble of networks, where each network has its weights drawn from a shared, learnt probability distribution. Unlike other ensemble methods, our method typically only doubles the number of parameters yet trains an infinite ensemble using unbiased Monte Carlo estimates of the gradients.

In general, exact Bayesian inference on 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. Instead we take a variational approximation to exact Bayesian updates. We build upon the work of Graves (2011), who in turn built upon the work of Hinton and Van Camp (1993). In contrast to this previous work, we show how the gradients of Graves (2011) can be made unbiased and further how this method can be used with non-Gaussian priors. Consequently, Bayes by Backprop attains performance comparable to that of dropout (Hinton et al., 2012). Our method

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37. Copyright 2015 by the author(s).

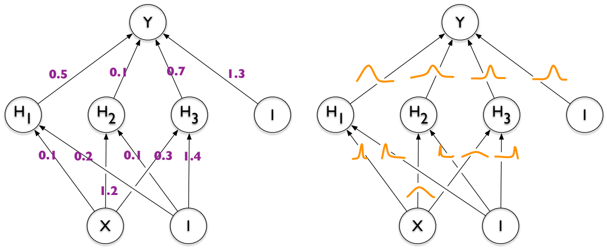

Figure 1. Left: each weight has a fixed value, as provided by classical backpropagation. Right: each weight is assigned a distribution, as provided by Bayes by Backprop.

is related to recent methods in deep, generative modelling (Kingma and Welling, 2014; Rezende et al., 2014; Gregor et al., 2014), where variational inference has been applied to stochastic hidden units of an autoencoder. Whilst the number of stochastic hidden units might be in the order of thousands, the number of weights in a neural network is easily two orders of magnitude larger, making the optimisation problem much larger scale. Uncertainty in the hidden units allows the expression of uncertainty about a particular observation, uncertainty in the weights is complementary in that it captures uncertainty about which neural network is appropriate, leading to regularisation of the weights and model averaging.

This uncertainty can be used to drive exploration in contextual bandit problems using Thompson sampling (Thompson, 1933; Chapelle and Li, 2011; Agrawal and Goyal, 2012; May et al., 2012). Weights with greater uncertainty introduce more variability into the decisions made by the network, leading naturally to exploration. As more data are observed, the uncertainty can decrease, allowing the decisions made by the network to become more deterministic as the environment is better understood.

The remainder of the paper is organised as follows: Section 2 introduces notation and standard learning in neural networks, Section 3 describes variational Bayesian learning for neural networks and our contributions, Section 4 describes the application to contextual bandit problems, whilst Section 5 contains empirical results on a classification, a regression and a bandit problem. We conclude with a brief discussion in Section 6.

2. Point Estimates of Neural Networks

We view a neural network as a probabilistic model $P(\mathbf{y}|\mathbf{x},\mathbf{w})$: given an input $\mathbf{x} \in \mathbb{R}^p$ a neural network assigns a probability to each possible output $\mathbf{y} \in \mathcal{Y}$, using the set of parameters or weights $\mathbf{w}$. For classification, $\mathcal{Y}$ is a set of classes and $P(\mathbf{y}|\mathbf{x},\mathbf{w})$ is a categorical distribution – this corresponds to the cross-entropy or softmax loss, when the parameters of the categorical distribution are passed through the exponential function then re-normalised. For regression $\mathcal{Y}$ is $\mathbb{R}$ and $P(\mathbf{y}|\mathbf{x},\mathbf{w})$ is a Gaussian distribution – this corresponds to a squared loss.

Inputs $\mathbf{x}$ are mapped onto the parameters of a distribution on $\mathcal{Y}$ by several successive layers of linear transformation (given by $\mathbf{w}$) interleaved with element-wise non-linear transforms.

The weights can be learnt by maximum likelihood estimation (MLE): given a set of training examples $\mathcal{D} = (\mathbf{x}_i, \mathbf{y}_i)_i$, the MLE weights $\mathbf{w}^{\mathrm{MLE}}$ are given by:

$$\begin{aligned}\mathbf{w}^{\mathrm{MLE}} &= \arg\max_{\mathbf{w}} \log P(\mathcal{D}|\mathbf{w}) \\ &= \arg\max_{\mathbf{w}} \sum_i \log P(\mathbf{y}_i|\mathbf{x}_i,\mathbf{w}).\end{aligned}$$

This is typically achieved by gradient descent (e.g., backpropagation), where we assume that $\log P(\mathcal{D}|\mathbf{w})$ is differentiable in $\mathbf{w}$.

Regularisation can be introduced by placing a prior upon the weights $\mathbf{w}$ and finding the maximum a posteriori (MAP) weights $\mathbf{w}^{\mathrm{MAP}}$:

$$\begin{aligned}\mathbf{w}^{\mathrm{MAP}} &= \arg\max_{\mathbf{w}} \log P(\mathbf{w}|\mathcal{D}) \\ &= \arg\max_{\mathbf{w}} \log P(\mathcal{D}|\mathbf{w}) + \log P(\mathbf{w}).\end{aligned}$$

If $\mathbf{w}$ are given a Gaussian prior, this yields L2 regularisation (or weight decay). If $\mathbf{w}$ are given a Laplace prior, then L1 regularisation is recovered.

3. Being Bayesian by Backpropagation

Bayesian inference for neural networks calculates the posterior distribution of the weights given the training data, $P(\mathbf{w}|\mathcal{D})$. This distribution answers predictive queries about unseen data by taking expectations: the predictive distribution of an unknown label $\hat{\mathbf{y}}$ of a test data item $\hat{\mathbf{x}}$, is given by $P(\hat{\mathbf{y}}|\hat{\mathbf{x}}) = \mathbb{E}_{P(\mathbf{w}|\mathcal{D})}[P(\hat{\mathbf{y}}|\hat{\mathbf{x}},\mathbf{w})]$. Each possible configuration of the weights, weighted according to the posterior distribution, makes a prediction about the unknown label given the test data item $\hat{\mathbf{x}}$. Thus taking an expectation under the posterior distribution on weights is equivalent to using an ensemble of an uncountably infinite number of neural networks. Unfortunately, this is intractable for neural networks of any practical size.

Previously Hinton and Van Camp (1993) and Graves (2011) suggested finding a variational approximation to the Bayesian posterior distribution on the weights. Variational learning finds the parameters θ of a distribution on the weights $q(\mathbf{w}|\theta)$ that minimises the Kullback-Leibler (KL) divergence with the true Bayesian posterior on the weights:

$$\begin{aligned}\theta^\star &= \arg\min_\theta \mathrm{KL}[q(\mathbf{w}|\theta)||P(\mathbf{w}|\mathcal{D})] \\ &= \arg\min_\theta \int q(\mathbf{w}|\theta) \log \frac{q(\mathbf{w}|\theta)}{P(\mathbf{w})P(\mathcal{D}|\mathbf{w})} \mathrm{d}\mathbf{w} \\ &= \arg\min_\theta \mathrm{KL}\left[q(\mathbf{w}|\theta) \mid\mid P(\mathbf{w})\right] - \mathbb{E}_{q(\mathbf{w}|\theta)}\left[\log P(\mathcal{D}|\mathbf{w})\right].\end{aligned}$$

The resulting cost function is variously known as the variational free energy (Neal and Hinton, 1998; Yedidia et al., 2000; Friston et al., 2007) or the expected lower bound (Saul et al., 1996; Neal and Hinton, 1998; Jaakkola and Jordan, 2000). For simplicity we shall denote it as

$$\begin{aligned}\mathcal{F}(\mathcal{D},\theta) = \mathrm{KL}\left[q(\mathbf{w}|\theta) \mid\mid P(\mathbf{w})\right] \\ - \mathbb{E}_{q(\mathbf{w}|\theta)}\left[\log P(\mathcal{D}|\mathbf{w})\right]. \quad (1)\end{aligned}$$

The cost function of (1) is a sum of a data-dependent part, which we shall refer to as the likelihood cost, and a prior-dependent part, which we shall refer to as the complexity cost. The cost function embodies a trade-off between satisfying the complexity of the data $\mathcal{D}$ and satisfying the simplicity prior $P(\mathbf{w})$. (1) is also readily given an information theoretic interpretation as a minimum description length cost (Hinton and Van Camp, 1993; Graves, 2011). Exactly minimising this cost naïvely is computationally prohibitive. Instead gradient descent and various approximations are used.

3.1. Unbiased Monte Carlo gradients

Under certain conditions, the derivative of an expectation can be expressed as the expectation of a derivative:

Proposition 1. *Let ϵ be a random variable having a probability density given by $q(\epsilon)$ and let $\mathbf{w} = t(\theta, \epsilon)$ where $t(\theta,\epsilon)$ is a deterministic function. Suppose further that the marginal probability density of $\mathbf{w}$, $q(\mathbf{w}|\theta)$, is such that $q(\epsilon)d\epsilon = q(\mathbf{w}|\theta)d\mathbf{w}$. Then for a function f with derivatives in $\mathbf{w}$:*

$$\frac{\partial}{\partial\theta}\mathbb{E}_{q(\mathbf{w}|\theta)}[f(\mathbf{w},\theta)] = \mathbb{E}_{q(\epsilon)}\left[\frac{\partial f(\mathbf{w},\theta)}{\partial \mathbf{w}}\frac{\partial \mathbf{w}}{\partial\theta} + \frac{\partial f(\mathbf{w},\theta)}{\partial\theta}\right].$$

Proof.

$$\begin{aligned}\frac{\partial}{\partial\theta}\mathbb{E}_{q(\mathbf{w}|\theta)}[f(\mathbf{w},\theta)] &= \frac{\partial}{\partial\theta}\int f(\mathbf{w},\theta)q(\mathbf{w}|\theta)\mathrm{d}\mathbf{w} \\ &= \frac{\partial}{\partial\theta}\int f(\mathbf{w},\theta)q(\epsilon)\mathrm{d}\epsilon \\ &= \mathbb{E}_{q(\epsilon)}\left[\frac{\partial f(\mathbf{w},\theta)}{\partial \mathbf{w}}\frac{\partial \mathbf{w}}{\partial\theta} + \frac{\partial f(\mathbf{w},\theta)}{\partial\theta}\right]\end{aligned}$$

□

The deterministic function $t(\theta,\epsilon)$ transforms a sample of parameter-free noise ϵ and the variational posterior parameters θ into a sample from the variational posterior. Below we shall see how this transform works in practice for the Gaussian case.

We apply Proposition 1 to the optimisation problem in (1): let $f(\mathbf{w},\theta) = \log q(\mathbf{w}|\theta) - \log P(\mathbf{w})P(\mathcal{D}|\mathbf{w})$. Using Monte Carlo sampling to evaluate the expectations, a backpropagation-like (LeCun, 1985; Rumelhart et al., 1988) algorithm is obtained for variational Bayesian inference in neural networks – Bayes by Backprop – which uses unbiased estimates of gradients of the cost in (1) to learn a distribution over the weights of a neural network.

Proposition 1 is the re-parameterisation trick (Opper and Archambeau, 2009; Kingma and Welling, 2014; Rezende et al., 2014) most commonly used for latent variable models, applied to Bayesian learning of neural networks. Our work differs from this previous work in several significant ways. Bayes by Backprop operates on weights (of which there are a great many), whilst most previous work applies this method to learning distributions on stochastic hidden units (of which there are far fewer than the number of weights). Titsias and Lázaro-Gredilla (2014) considered a large-scale logistic regression task. Unlike previous work, we do not use the closed form of the complexity cost (or entropic part): not requiring a closed form of the complexity cost allows many more combinations of prior and variational posterior families. Indeed this scheme is also simple to implement and allows prior/posterior combinations to be interchanged. We approximate the exact cost (1) as:

$$\begin{aligned}\mathcal{F}(\mathcal{D},\theta) \approx \sum_{i=1}^{n} \log q(\mathbf{w}^{(i)}|\theta) - \log P(\mathbf{w}^{(i)}) \\ - \log P(\mathcal{D}|\mathbf{w}^{(i)}) \quad (2)\end{aligned}$$

where $\mathbf{w}^{(i)}$ denotes the ith Monte Carlo sample drawn from the variational posterior $q(\mathbf{w}^{(i)}|\theta)$. Note that every term of this approximate cost depends upon the *particular* weights drawn from the variational posterior: this is an instance of a variance reduction technique known as common random numbers (Owen, 2013). In previous work, where a closed form complexity cost or closed form entropy term are used, part of the cost is sensitive to particular draws from the posterior, whilst the closed form part is oblivious. Since each additive term in the approximate cost in (2) uses the same weight samples, the gradients of (2) are only affected by the parts of the posterior distribution characterised by the weight samples. In practice, we did not find this to perform better than using a closed form KL (where it could be computed), but we did not find it to perform worse. In our experiments, we found that a prior without an easy-to-compute closed form complexity cost performed best.

3.2. Gaussian variational posterior

Suppose that the variational posterior is a diagonal Gaussian distribution, then a sample of the weights $\mathbf{w}$ can be obtained by sampling a unit Gaussian, shifting it by a mean μ and scaling by a standard deviation σ. We parameterise the standard deviation pointwise as $\sigma = \log(1 + \exp(\rho))$ and so σ is always non-negative. The variational posterior parameters are $\theta = (\mu, \rho)$. Thus the transform from a sample of parameter-free noise and the variational posterior parameters that yields a posterior sample of the weights $\mathbf{w}$ is: $\mathbf{w} = t(\theta, \epsilon) = \mu + \log(1 + \exp(\rho)) \circ \epsilon$ where $\circ$ is pointwise multiplication. Each step of optimisation proceeds as follows:

1. Sample $\epsilon \sim \mathcal{N}(0, I)$.
2. Let $\mathbf{w} = \mu + \log(1 + \exp(\rho)) \circ \epsilon$.
3. Let $\theta = (\mu, \rho)$.
4. Let $f(\mathbf{w}, \theta) = \log q(\mathbf{w}|\theta) - \log P(\mathbf{w})P(\mathcal{D}|\mathbf{w})$.
5. Calculate the gradient with respect to the mean
$$\Delta_\mu = \frac{\partial f(\mathbf{w}, \theta)}{\partial \mathbf{w}} + \frac{\partial f(\mathbf{w}, \theta)}{\partial \mu}. \quad (3)$$
6. Calculate the gradient with respect to the standard deviation parameter ρ
$$\Delta_\rho = \frac{\partial f(\mathbf{w}, \theta)}{\partial \mathbf{w}} \frac{\epsilon}{1 + \exp(-\rho)} + \frac{\partial f(\mathbf{w}, \theta)}{\partial \rho}. \quad (4)$$
7. Update the variational parameters:
$$\mu \leftarrow \mu - \alpha \Delta_\mu \quad (5)$$
$$\rho \leftarrow \rho - \alpha \Delta_\rho. \quad (6)$$

Note that the $\frac{\partial f(\mathbf{w},\theta)}{\partial \mathbf{w}}$ term of the gradients for the mean and standard deviation are shared and are exactly the gradients found by the usual backpropagation algorithm on a neural network. Thus, remarkably, to learn both the mean and the standard deviation we must simply calculate the usual gradients found by backpropagation, and then scale and shift them as above.

3.3. Scale mixture prior

Having liberated our algorithm from the confines of Gaussian priors and posteriors, we propose a simple scale mixture prior combined with a diagonal Gaussian posterior. The diagonal Gaussian posterior is largely free from numerical issues, and two degrees of freedom per weight only increases the number of parameters to optimise by a factor of two, whilst giving each weight its own quantity of uncertainty.

We pick a fixed-form prior and do not adjust its hyperparameters during training, instead picking the them by cross-validation where possible. Empirically we found optimising the parameters of a prior $P(\mathbf{w})$ (by taking derivatives of (1)) to not be useful, and yield worse results. Graves (2011) and Titsias and Lázaro-Gredilla (2014) propose closed form updates of the prior hyperparameters. Changing the prior based upon the data that it is meant to regularise is known as empirical Bayes and there is much debate as to its validity (Gelman, 2008). A reason why it fails for Bayes by Backprop is as follows: it can be easier to change the prior parameters (of which there are few) than it is to change the posterior parameters (of which there are many) and so very quickly the prior parameters try to capture the empirical distribution of the weights at the beginning of learning. Thus the prior learns to fit poor initial parameters quickly, and makes the cost in (1) less willing to move away from poor initial parameters. This can yield slow convergence, introduce strange local minima and result in poor performance.

We propose using a scale mixture of two Gaussian densities as the prior. Each density is zero mean, but differing variances:

$$P(\mathbf{w}) = \prod_j \pi \mathcal{N}(\mathbf{w}_j | 0, \sigma_1^2) + (1 - \pi)\mathcal{N}(\mathbf{w}_j | 0, \sigma_2^2), \quad (7)$$

where $\mathbf{w}_j$ is the jth weight of the network, $\mathcal{N}(x|\mu, \sigma^2)$ is the Gaussian density evaluated at x with mean μ and variance σ^2 and σ_1^2 and σ_2^2 are the variances of the mixture components. The first mixture component of the prior is given a larger variance than the second, $\sigma_1 > \sigma_2$, providing a heavier tail in the prior density than a plain Gaussian prior. The second mixture component has a small variance $\sigma_2 \ll 1$ causing many of the weights to *a priori* tightly concentrate around zero. Our prior resembles a spike-and-slab prior (Mitchell and Beauchamp, 1988; George and McCulloch, 1993; Chipman, 1996), where instead all the prior parameters are shared among all the weights. This makes the prior more amenable to use during optimisation by stochastic gradient descent and avoids the need for prior parameter optimisation based upon training data.

3.4. Minibatches and KL re-weighting

As several authors have noted, the cost in (1) is amenable to minibatch optimisation, often used with neural networks: for each epoch of optimisation the training data $\mathcal{D}$ is randomly split into a partition of M equally-sized subsets, $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_M$. Each gradient is averaged over all elements in one of these minibatches; a trade-off between a fully batched gradient descent and a fully stochastic gradient descent. Graves (2011) proposes minimising the minibatch cost for minibatch $i = 1, 2, \dots, M$:

$$\mathcal{F}_i^{\text{EQ}}(\mathcal{D}_i, \theta) = \frac{1}{M}\text{KL}\left[q(\mathbf{w}|\theta) \mid\mid P(\mathbf{w})\right] - \mathbb{E}_{q(\mathbf{w}|\theta)}\left[\log P(\mathcal{D}_i|\mathbf{w})\right]. \quad (8)$$

This is equivalent to the cost in (1) since $\sum_i \mathcal{F}_i^{\text{EQ}}(\mathcal{D}_i, \theta) = \mathcal{F}(\mathcal{D}, \theta)$. There are many ways to weight the complexity cost relative to the likelihood cost on each minibatch. For example, if minibatches are partitioned uniformly at random, the KL cost can be distributed non-uniformly among the minibatches at each epoch. Let $\pi \in [0, 1]^M$ and $\sum_{i=1}^M \pi_i = 1$, and define:

$$\mathcal{F}_i^{\pi}(\mathcal{D}_i, \theta) = \pi_i\text{KL}\left[q(\mathbf{w}|\theta) \mid\mid P(\mathbf{w})\right] - \mathbb{E}_{q(\mathbf{w}|\theta)}\left[\log P(\mathcal{D}_i|\mathbf{w})\right] \quad (9)$$

Then $\mathbb{E}_M[\sum_{i=1}^M \mathcal{F}_i^{\pi}(\mathcal{D}_i, \theta)] = \mathcal{F}(\mathcal{D}, \theta)$ where $\mathbb{E}_M$ denotes an expectation over the random partitioning of minibatches. In particular, we found the scheme $\pi_i = \frac{2^{M-i}}{2^M-1}$ to work well: the first few minibatches are heavily influenced by the complexity cost, whilst the later minibatches are largely influenced by the data. At the beginning of learning this is particularly useful as for the first few minibatches changes in the weights due to the data are slight and as more data are seen, data become more influential and the prior less influential.

4. Contextual Bandits

Contextual bandits are simple reinforcement learning problems without persistent state (Li et al., 2010; Filippi et al., 2010). At each step an agent is presented with a context x and a choice of one of K possible actions a. Different actions yield different unknown rewards r. The agent must pick the action that yields the highest expected reward. The context is assumed to be presented independent of any previous actions, rewards or contexts.

An agent builds a model of the distribution of the rewards conditioned upon the action and the context: $P(r|x, a, \mathbf{w})$. It then uses this model to pick its action. Note, importantly, that an agent does not know what reward it could have received for an action that it did not pick, a difficulty often known as "the absence of counterfactual". As the agent's model $P(r|x, a, \mathbf{w})$ is trained online, based upon the actions chosen, unless exploratory actions are taken, the agent may perform suboptimally.

4.1. Thompson Sampling for Neural Networks

As in Section 2, $P(r|x, a, \mathbf{w})$ can be modelled by a neural network where $\mathbf{w}$ are the weights of the neural network. However if this network is simply fit to observations and the action with the highest expected reward taken at each time, the agent can under-explore, as it may miss more rewarding actions.[1]

Thompson sampling (Thompson, 1933) is a popular means of picking an action that trades-off between exploitation (picking the best known action) and exploration (picking what might be a suboptimal arm to learn more). Thompson sampling usually necessitates a Bayesian treatment of the model parameters. At each step, Thompson sampling draws a new set of parameters and then picks the action relative to those parameters. This can be seen as a kind of stochastic hypothesis testing: more probable parameters are drawn more often and thus refuted or confirmed the fastest. More concretely Thompson sampling proceeds as follows:

1. Sample a new set of parameters for the model.
2. Pick the action with the highest expected reward according to the sampled parameters.
3. Update the model. Go to 1.

There is an increasing literature concerning the efficacy and justification of this means of exploration (Chapelle and Li, 2011; May et al., 2012; Kaufmann et al., 2012; Agrawal and Goyal, 2012; 2013). Thompson sampling is easily adapted to neural networks using the variational posterior found in Section 3:

1. Sample weights from the variational posterior: $\mathbf{w} \sim q(\mathbf{w}|\theta)$.
2. Receive the context x.
3. Pick the action a that minimises $\mathbb{E}_{P(r|x,a,\mathbf{w})}[r]$
4. Receive reward r.
5. Update variational parameters θ according to Section 3. Go to 1.

Note that it is possible, as mentioned in Section 3.1, to decrease the variance of the gradient estimates, trading off for reduced exploration, by using more than one Monte Carlo sample, using the corresponding networks as an ensemble and picking the action by minimising the average of the expectations.

Initially the variational posterior will be close to the prior, and actions will be picked uniformly. As the agent takes actions, the variational posterior will begin to converge, and uncertainty on many parameters can decrease, and so action selection will become more deterministic, focusing on the high expected reward actions discovered so far. It is

[1] Interestingly, depending upon how $\mathbf{w}$ are initialised and the mean of prior used during MAP inference, it is sometimes possible to obtain another heuristic for the exploration-exploitation trade-off: optimism-under-uncertainty. We leave this for future investigation.

Table 1. Classification Error Rates on MNIST. ⋆ indicates result used an ensemble of 5 networks.

Method	**# Units/Layer**	**# Weights**	**Test Error**
SGD, no regularisation (Simard et al., 2003)	800	1.3m	1.6%
SGD, dropout (Hinton et al., 2012)			≈ 1.3%
SGD, dropconnect (Wan et al., 2013)	800	1.3m	**1.2%***
SGD	400	500k	1.83%
	800	1.3m	1.84%
	1200	2.4m	1.88%
SGD, dropout	400	500k	1.51%
	800	1.3m	1.33%
	1200	2.4m	1.36%
Bayes by Backprop, Gaussian	400	500k	1.82%
	800	1.3m	1.99%
	1200	2.4m	2.04%
Bayes by Backprop, Scale mixture	400	500k	1.36%
	800	1.3m	1.34%
	1200	2.4m	**1.32%**

known that variational methods under-estimate uncertainty (Minka, 2001; 2005; Bishop, 2006) which could lead to under-exploration and premature convergence in practice, but we did not find this in practice.

5. Experiments

We present some empirical evaluation of the methods proposed above: on MNIST classification, on a non-linear regression task, and on a contextual bandits task.

5.1. Classification on MNIST

We trained networks of various sizes on the MNIST digits dataset (LeCun and Cortes, 1998), consisting of 60,000 training and 10,000 testing pixel images of size 28 by 28. Each image is labelled with its corresponding number (between zero and nine, inclusive). We preprocessed the pixels by dividing values by 126. Many methods have been proposed to improve results on MNIST: generative pretraining, convolutions, distortions, etc. Here we shall focus on improving the performance of an ordinary feedforward neural network *without* using any of these methods. We used a network of two hidden layers of rectified linear units (Nair and Hinton, 2010; Glorot et al., 2011), and a softmax output layer with 10 units, one for each possible label.

According to Hinton et al. (2012), the best published feedforward neural network classification result on MNIST (excluding those using data set augmentation, convolutions, etc.) is 1.6% (Simard et al., 2003), whilst dropout with an L2 regulariser attains errors around 1.3%. Results from Bayes by Backprop are shown in Table 1, for various sized networks, using either a Gaussian or Gaussian scale mixture prior. Performance is comparable to that of dropout, perhaps slightly better, as also see on Figure 2. Note that we trained on 50,000 digits and used 10,000 digits as a validation set, whilst Hinton et al. (2012) trained on 60,000 digits and did not use a validation set. We used the validation set to pick the best hyperparameters (learning rate, number of gradients to average) and so we also repeated this protocol for dropout and SGD (Stochastic Gradient Descent on the MLE objective in Section 2). We considered learning rates of 10^{-3}, 10^{-4} and 10^{-5} with minibatches of size 128. For Bayes by Backprop, we averaged over either 1, 2, 5, or 10 samples and considered $\pi \in \{\frac{1}{4}, \frac{1}{2}, \frac{3}{4}\}$, $-\log \sigma_1 \in \{0, 1, 2\}$ and $-\log \sigma_2 \in \{6, 7, 8\}$.

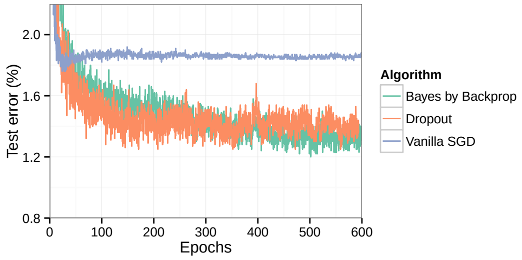

Figure 2. Test error on MNIST as training progresses.

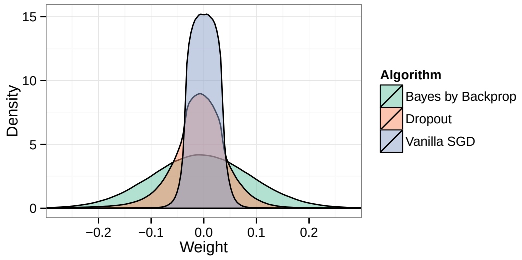

Figure 3. Histogram of the trained weights of the neural network, for Dropout, plain SGD, and samples from Bayes by Backprop.

Figure 2 shows the learning curves on the test set for Bayes by Backprop, dropout and SGD on a network with two layers of 1200 rectified linear units. As can be seen, SGD converges the quickest, initially obtaining a low test error and then overfitting. Bayes by Backprop and dropout converge at similar rates (although each iteration of Bayes by Backprop is more expensive than dropout – around two times slower). Eventually Bayes by Backprop converges on a better test error than dropout after 600 epochs.

Figure 3 shows density estimates of the weights. The Bayes by Backprop weights are sampled from the variational posterior, and the dropout weights are those used at test time. Interestingly the regularised networks found by dropout and Bayes by Backprop have a greater range and with fewer centred at zero than those found by SGD. Bayes by Backprop uses the greatest range of weights.

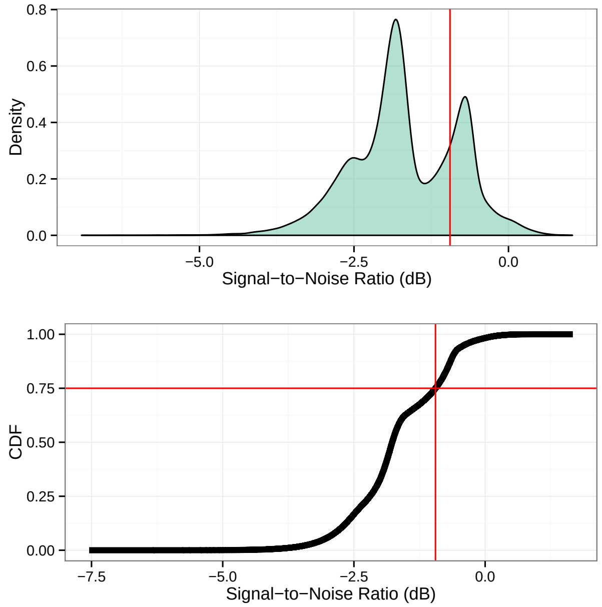

Figure 4. Density and CDF of the Signal-to-Noise ratio over all weights in the network. The red line denotes the 75% cut-off.

In Table 2, we examine the effect of replacing the variational posterior on some of the weights with a constant zero, so as to determine the level of redundancy in the network found by Bayes by Backprop. We took a Bayes by Backprop trained network with two layers of 1200 units[2] and ordered the weights by their signal-to-noise ratio ($|\mu_i|/\sigma_i$). We removed the weights with the lowest signal to noise ratio. As can be seen in Table 2, even when 95% of the weights are removed the network still performs well, with a significant drop in performance once 98% of the weights have been removed.

In Figure 4 we examined the distribution of the signal-to-noise relative to the cut-off in the network uses in Table 2. The lower plot shows the cumulative distribution of signal-to-noise ratio, whilst the top plot shows the density. From the density plot we see there are two modalities of signal-to-noise ratios, and from the CDF we see that the 75% cut-off separates these two peaks. These two peaks coincide with a drop in performance in Table 2 from 1.24% to 1.29%, suggesting that the signal-to-noise heuristic is in fact related to the test performance.

Table 2. Classification Errors after Weight pruning

Proportion removed	# Weights	Test Error
0%	2.4m	1.24%
50%	1.2m	1.24%
75%	600k	1.24%
95%	120k	1.29%
98%	48k	1.39%

It is interesting to contrast this weight removal approach to obtaining a fast, smaller, sparse network for prediction after training with the approach taken by distillation (Hinton et al., 2014) which requires an extra stage of training to obtain a compressed prediction model. As with distillation, our method begins with an ensemble (one for each possible assignment of the weights). However, unlike distillation, we can simply obtain a subset of this ensemble by using the probabilistic properties of the weight distributions learnt to gracefully prune the ensemble down into a smaller network. Thus even though networks trained by Bayes by Backprop may have twice as many weights, the number of parameters that actually need to be stored at run time can be far fewer. Graves (2011) also considered pruning weights using the signal to noise ratio, but demonstrated results on a network 20 times smaller and did not prune as high a proportion of weights (at most 11%) whilst still maintaining good test performance. The scale mixture prior used by Bayes by Backprop encourages a broad spread of the weights. Many of these weights can be successfully pruned without impacting performance significantly.

5.2. Regression curves

We generated training data from the curve:

$$y = x + 0.3\sin(2\pi(x+\epsilon)) + 0.3\sin(4\pi(x+\epsilon)) + \epsilon$$

where $\epsilon \sim \mathcal{N}(0, 0.02)$. Figure 5 shows two examples of fitting a neural network to these data, minimising a conditional Gaussian loss. Note that in the regions of the input space where there are no data, the ordinary neural network reduces the variance to zero and chooses to fit a particular function, even though there are many possible extrapolations of the training data. On the left, Bayesian model averaging affects predictions: where there are no data, the confidence intervals diverge, reflecting there being many possible extrapolations. In this case Bayes by Backprop prefers to be uncertain where there are no nearby data, as opposed to a standard neural network which can be overly confident.

5.3. Bandits on Mushroom Task

We take the UCI Mushrooms data set (Bache and Lichman, 2013), and cast it as a bandit task, similar to Guez (2015,

[2] We used a network from the end of training rather than picking a network with a low validation cost found during training, hence the disparity with results in Table 1. The lowest test error observed was 1.12%.

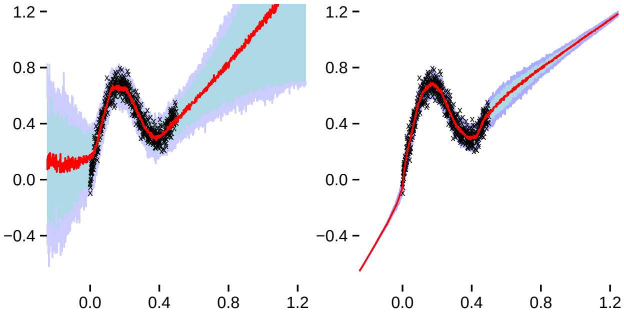

Figure 5. Regression of noisy data with interquatile ranges. Black crosses are training samples. Red lines are median predictions. Blue/purple region is interquartile range. Left: Bayes by Backprop neural network, Right: standard neural network.

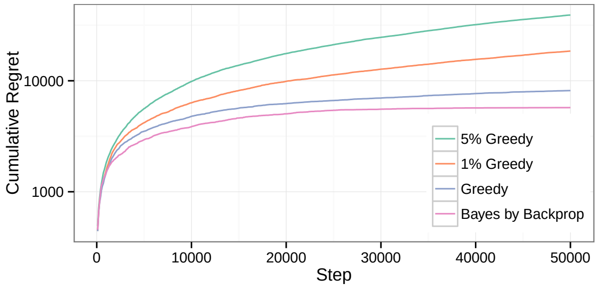

Figure 6. Comparison of cumulative regret of various agents on the mushroom bandit task, averaged over five runs. Lower is better.

Chapter 6). Each mushroom has a set of features, which we treat as the context for the bandit, and is labelled as edible or poisonous. An agent can either eat or not eat a mushroom. If an agent eats an edible mushroom, then it receives a reward of 5. If an agent eats a poisonous mushroom, then with probability $\frac{1}{2}$ it receives a reward of -35, otherwise a reward of 5. If an agent elects not to eat a mushroom, it receives a reward of 0. Thus an agent expects to receive a reward of 5 for eating an edible reward, but an expected reward of -15 for eating a poisonous mushroom.

Regret measures the difference between the reward achievable by an oracle and the reward received by an agent. In this case, an oracle will always receive a reward of 5 for an edible mushroom, or 0 for a poisonous mushroom. We take the cumulative sum of regret of several agents and show them in Figure 6. Each agent uses a neural network with two hidden layers of 100 rectified linear units. The input to the network is a vector consisting of the mushroom features (context) and a one of K encoding of the action. The output of the network is a single scalar, representing the expected reward of the given action in the given context. For Bayes by Backprop, we sampled the weights twice and averaged two of these outputs to obtain the expected reward for action selection. We kept the last 4096 reward, context and action tuples in a buffer, and trained the networks using randomly drawn minibatches of size 64 for 64 training steps ($64 \times 64 = 4096$) per interaction with the Mushroom bandit. A common heuristic for trading-off exploration vs. exploitation is to follow an ε-greedy policy: with probability ε propose a uniformly random action, otherwise pick the best action according to the neural network.

Figure 6 compares a Bayes by Backprop agent with three ε-greedy agents, for values of ε of 0% (pure greedy), 1%, and 5%. An ε of 5% appears to over-explore, whereas a purely greedy agent does poorly at the beginning, greedily electing to eat nothing, but then does much better once it has seen enough data. It seems that non-local function approximation updates allow the greedy agent to explore, as for the first $1,000$ steps, the agent eats nothing but after approximately $1,000$ the greedy agent suddenly decides to eat mushrooms. The Bayes by Backprop agent explores from the beginning, both eating and ignoring mushrooms and quickly converges on eating and non-eating with an almost perfect rate (hence the almost flat regret).

6. Discussion

We introduced a new algorithm for learning neural networks with uncertainty on the weights called Bayes by Backprop. It optimises a well-defined objective function to learn a distribution on the weights of a neural network. The algorithm achieves good results in several domains. When classifying MNIST digits, performance from Bayes by Backprop is comparable to that of dropout. We demonstrated on a simple non-linear regression problem that the uncertainty introduced allows the network to make more reasonable predictions about unseen data. Finally, for contextual bandits, we showed how Bayes by Backprop can automatically learn how to trade-off exploration and exploitation. Since Bayes by Backprop simply uses gradient updates, it can readily be scaled using multi-machine optimisation schemes such as asynchronous SGD (Dean et al., 2012). Furthermore, all of the operations used are readily implemented on a GPU.

Acknowledgements The authors would like to thank Ivo Danihelka, Danilo Rezende, Silvia Chiappa, Alex Graves, Remi Munos, Ben Coppin, Liam Clancy, James Kirkpatrick, Shakir Mohamed, David Pfau, and Theophane Weber for useful discussions and comments.

References

Shipra Agrawal and Navin Goyal. Analysis of Thompson sampling for the multi-armed bandit problem. In *Proceedings of the 25th Annual Conference On Learning*

Theory (COLT), volume 23, pages 39.1–39.26, 2012.

Shipra Agrawal and Navin Goyal. Further optimal regret bounds for Thompson sampling. In *Proceedings of the 16th International Conference on Artificial Intelligence and Statistics Learning (AISTATS)*, pages 99–107, 2013.

Kevin Bache and Moshe Lichman. *UCI Machine Learning Repository*. University of California, Irvine, School of Information and Computer Sciences, 2013. URL `http://archive.ics.uci.edu/ml`.

Christopher M Bishop. Section 10.1: variational inference. In *Pattern Recognition and Machine Learning*. Springer, 2006. ISBN 9780387310732.

Wray L Buntine and Andreas S Weigend. Bayesian backpropagation. *Complex systems*, 5(6):603–643, 1991.

Olivier Chapelle and Lihong Li. An empirical evaluation of Thompson sampling. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2249–2257, 2011.

Hugh Chipman. Bayesian variable selection with related predictors. *Canadian Journal of Statistics*, 24(1):17–36, 1996.

Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, et al. Large scale distributed deep networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1223–1231, 2012.

Sarah Filippi, Olivier Cappe, Aurlien Garivier, and Csaba Szepesvri. Parametric bandits: The generalized linear case. In *Advances in Neural Information Processing Systems*, pages 586–594, 2010.

Karl Friston, Jérémie Mattout, Nelson Trujillo-Barreto, John Ashburner, and Will Penny. Variational free energy and the Laplace approximation. *Neuroimage*, 34 (1):220–234, 2007.

Andrew Gelman. Objections to Bayesian statistics. *Bayesian Analysis*, 3:445–450, 2008. ISSN 1931-6690. doi: 11.1214/08-BA318.

Edward I George and Robert E McCulloch. Variable selection via gibbs sampling. *Journal of the American Statistical Association*, 88(423):881–889, 1993.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier networks. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics Learning (AISTATS)*, volume 15, pages 315–323, 2011.

Alex Graves. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2348–2356, 2011.

Karol Gregor, Ivo Danihelka, Andriy Mnih, Charles Blundell, and Daan Wierstra. Deep AutoRegressive networks. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pages 1242–1250, 2014.

Arthur Guez. *Sample-Based Search Methods For Bayes-Adaptive Planning*. PhD thesis, University College London, 2015.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NIPS 2014 Deep Learning and Representation Learning Workshop*, 2014.

Geoffrey E Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the 16th Annual Conference On Learning Theory (COLT)*, pages 5–13. ACM, 1993.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580*, July 2012.

Tommi S. Jaakkola and Michael I. Jordan. Bayesian parameter estimation via variational methods. *Statistics and Computing*, 10(1):25–37, 2000.

Emilie Kaufmann, Nathaniel Korda, and Rémi Munos. Thompson sampling: An asymptotically optimal finite-time analysis. In *Proceedings of the 23rd Annual Conference on Algorithmic Learning Theory (ALT)*, pages 199–213. Springer, 2012.

Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2014. arXiv: 1312.6114.

Yann LeCun. Une procédure d'apprentissage pour réseau à seuil asymmetrique (a learning scheme for asymmetric threshold networks). In *Proceedings of Cognitiva 85, Paris, France*, pages 599–604, 1985.

Yann LeCun and Corinna Cortes. *The MNIST database of handwritten digits*. 1998. URL `http://yann.lecun.com/exdb/mnist/`.

Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 661–670, New York, NY, USA,

2010. ACM. ISBN 978-1-60558-799-8. doi: 10.1145/1772690.1772758.

David JC MacKay. A practical Bayesian framework for backpropagation networks. *Neural computation*, 4(3): 448–472, 1992.

David JC MacKay. Probable networks and plausible predictions-a review of practical Bayesian methods for supervised neural networks. *Network: Computation in Neural Systems*, 6(3):469–505, 1995.

Benedict C May, Nathan Korda, Anthony Lee, and David S. Leslie. Optimistic Bayesian sampling in contextual-bandit problems. *The Journal of Machine Learning Research*, 13(1):2069–2106, 2012.

Thomas P Minka. *A family of algorithms for approximate Bayesian inference*. PhD thesis, Massachusetts Institute of Technology, 2001.

Thomas P Minka. Divergence measures and message passing. Technical report, Microsoft Research, 2005.

Toby J Mitchell and John J Beauchamp. Bayesian variable selection in linear regression. *Journal of the American Statistical Association*, 83(404):1023–1032, 1988.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, pages 807–814, 2010.

Radford M Neal and Geoffrey E Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer, 1998.

Manfred Opper and Cédric Archambeau. The variational Gaussian approximation revisited. *Neural computation*, 21(3):786–792, 2009.

Art B. Owen. *Monte Carlo theory, methods and examples*. 2013.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pages 1278–1286, 2014.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5, 1988.

Lawrence K Saul, Tommi Jaakkola, and Michael I Jordan. Mean field theory for sigmoid belief networks. *Journal of artificial intelligence research*, 4(1):61–76, 1996.

Patrice Y Simard, Dave Steinkraus, and John C Platt. Best practices for convolutional neural networks applied to visual document analysis. In *Proceedings of the 12th International Conference on Document Analysis and Recognition (ICDAR)*, volume 2, pages 958–958. IEEE Computer Society, 2003.

William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, pages 285–294, 1933.

Michalis Titsias and Miguel Lázaro-Gredilla. Doubly stochastic variational bayes for non-conjugate inference. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1971–1979, 2014.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann L Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1058–1066, 2013.

Jonathan S Yedidia, William T Freeman, and Yair Weiss. Generalized belief propagation. In *Advances in Neural Information Processing Systems (NIPS)*, volume 13, pages 689–695, 2000.