# Assumed Density Filtering Methods For Learning Bayesian Neural Networks

**Soumya Ghosh** **Francesco Maria Delle Fave** **Jonathan Yedidia**
{soumya.ghosh, francesco.dellefave, yedidia}@disneyresearch.com
Disney Research, 222 Third Street, Cambridge, MA 02142, USA.

**Abstract**

Buoyed by the success of deep multilayer neural networks, there is renewed interest in scalable learning of Bayesian neural networks. Here, we study algorithms that utilize recent advances in Bayesian inference to efficiently learn distributions over network weights. In particular, we focus on recently proposed assumed density filtering based methods for learning Bayesian neural networks – Expectation and Probabilistic backpropagation. Apart from scaling to large datasets, these techniques seamlessly deal with non-differentiable activation functions and provide parameter (learning rate, momentum) free learning. In this paper, we first rigorously compare the two algorithms and in the process develop several extensions, including a version of EBP for continuous regression problems and a PBP variant for binary classification. Next, we extend both algorithms to deal with multiclass classification and count regression problems. On a variety of diverse real world benchmarks, we find our extensions to be effective, achieving results competitive with the state-of-the-art.

## 1 Introduction

Neural networks employing multilayer architectures are expressive models capable of capturing complex relationships between input-output pairs. Deep architectures employing rectified linear units (ReLU) (Nair and Hinton 2010), when trained on massive datasets using backpropagation (Rumelhart et al. 1986) based stochastic gradient techniques and improved regularization schemes like dropout (Srivastava et al. 2014) and drop-connect (Wan et al. 2013), demonstrate state-of-the-art performance on a variety of learning tasks spanning computer vision (Wan et al. 2013; Krizhevsky et al. 2012), natural language processing (Sutskever et al. 2014) and reinforcement learning (Mnih et al. 2015).

Fostered by these successes, there is renewed interest in Bayesian neural network models (MacKay 1992) that account for uncertainty in network parameters. These models are attractive for several reasons. They naturally provide calibrated estimates of prediction uncertainty by propagating parameter uncertainties into predictions. By retaining distributions over parameters and averaging over them rather than relying on a single point estimate, they also tend to be more robust to overfitting. Furthermore, standard stochastic gradient descent training using backpropagation requires a large number of possibly layer-specific hyperparameters, such as learning rate and momentum, to be tuned on a per dataset basis through computationally expensive procedures. In contrast, a Bayesian view of neural networks makes the rich literature on probabilistic inference available, allowing for the design of "parameter free" algorithms.

A variety of probabilistic inference algorithms including Hamiltonian Monte Carlo, variational inference, Laplace approximation, and expectation propagation, have been studied for learning weight distributions in Bayesian neural networks (Neal 1995; Hinton and Van Camp 1993; Graves 2011; MacKay 1992; Jylänki, Nummenmaa, and Vehtari 2014). However, these methods either fail to scale to large datasets and architectures and/or provide poor posterior estimates. Promising recent work has focused on improving both scalability and accuracy (Soudry, Hubara, and Meir 2014; Hernández-Lobato and Adams 2015; Balan et al. 2015; Blundell et al. 2015). The algorithms developed by (Balan et al. 2015) and (Blundell et al. 2015) rely on stochastic gradients, either for approximating the posterior through stochastic gradient Langevin dynamics (SGLD) (Welling and Teh 2011) or for optimizing the variational free energy. Although these algorithms maintain parameter uncertainty they are plagued by the same hyperparameter selection problems exhibited by classical stochastic gradient descent learning. An orthogonal direction is provided by (Soudry, Hubara, and Meir 2014; Hernández-Lobato and Adams 2015), where the authors build upon the online Bayesian inference algorithm known as assumed density filtering (ADF) (Opper 1998) to develop methods for learning Bayesian neural networks — expectation backpropagation (EBP) and probabilistic backpropagation (PBP). These algorithms are scalable, accurate, and parameter-free, and have been shown to be extremely competitive with standard backpropagation both in terms of generalization error and speed of training, for regression and classification tasks.

Despite these advantages, several factors hamper the use of PBP and EBP. First, they assume certain tractabilities which make developing extensions for handling important practical problems of multiclass classification and count regression problems difficult. Further, PBP as presented

by (Hernández-Lobato and Adams 2015) only handles continuous regression problems, while EBP (Soudry, Hubara, and Meir 2014), works only for binary classification problems and for architectures employing units with signed activations. This makes a direct comparison of the two algorithms difficult.

In this paper, we address these issues. We first develop an EBP variant capable of handling networks with rectified linear units. Next, we demonstrate how PBP can be used for binary classification problems and EBP for continuous regression. Together, these extensions allow us to perform thorough empirical evaluations of the two algorithms. Our main contribution, however, lies in extending both algorithms to the problems of multiclass classification and count regression. To model these problems, we use softmax transformed and exponentiated neural network outputs to parametrize categorical and Poisson distributions. These distributions induce additional intractabilities in the model. To cope, we develop efficient approximations that lead to both accurate posteriors and competitive results on standard benchmarks.

## 2 Neural Networks as Statistical Models

Given a dataset $\mathcal{D} = \{\mathbf{x}, \mathbf{y}\} = \{x_n, y_n\}_{n=1}^N$ of feature $x_n \in \mathbb{R}^D$ and response $y_n \in \mathcal{Y}$ pairs, we assume that there exists a noisy functional mapping between $x_n$ and $y_n$. We model this mapping via a conditional distribution $p(\mathbf{y} \mid \mathbf{x}, \mathcal{W})$ and are interested in the related problems of estimating the posterior distribution over parameters $\mathcal{W}$,

$$p(\mathcal{W} \mid \mathbf{y}, \mathbf{x}) = \frac{p(\mathbf{y}, \mathcal{W} \mid \mathbf{x})}{\int p(\mathbf{y} \mid \mathbf{x}, \mathcal{W}) p(\mathcal{W}) d\mathcal{W}}, \tag{1}$$

and predicting responses $y_*$ for new features $x_*$, using the posterior predictive distribution,

$$p(y_* \mid x_*, \mathcal{D}) = \int p(y_* \mid x_*, \mathcal{W}) p(\mathcal{W} \mid \mathbf{y}, \mathbf{x}) d\mathcal{W}. \tag{2}$$

We further assume that the conditional distribution factorizes over data instances $p(\mathbf{y} \mid \mathbf{x}, \mathcal{W}) = \prod_n p(y_n \mid x_n, \mathcal{W})$, and that each factor belongs to a parametric distribution family (e.g., Gaussian). We use a fully connected multilayer feedforward neural network (MLN) $g$, to parameterize these factors.

We use a MLN with L layers, each containing $D_l$ hidden units. All layers, with the exception of the output layer, also contain an additional bias unit. The network is parameterized by $\mathcal{W} = \{\mathrm{W}_l\}_{l=1}^L$ a collection of $D_l \times (D_{l-1} + 1)$ weight matrices, where the $+1$ accounts for the bias node. The layer outputs are denoted $\{\mathbf{z}_l\}_{l=0}^L$, with $\mathbf{z}_0$ representing the input $x_n$ and $\mathbf{z}_L = g(x_n, \mathcal{W})$ corresponding to the network output. The output of an intermediate layer $\mathbf{z}_l,\ l \in \{1, \ldots, L-1\}$ is computed via a non linear activation, $a$, of its inputs $\mathbf{u}_l = \mathrm{W}_l\ [\mathbf{z}_{l-1}, 1]^T$. Following recent successes (Nair and Hinton 2010), we restrict our attention to rectified linear activations, $z_{il} = \max(0, u_{il})$ for a neuron $i$ in layer $l$ and we constrain the output layer to have linear activations. The parametric form of the distribution $p(y_n \mid g(x_n, \mathcal{W}))$ is task dependent and is further described in subsequent sections. Finally, we endow the weights $\mathcal{W}$ with a zero mean Gaussian prior,

$$\mathcal{W} \mid \lambda \sim \prod_{l=1}^{L} \prod_{i=1}^{V_l} \prod_{j=1}^{V_{L+1}} \mathcal{N}(w_{ijl} \mid 0, \lambda^{-1}), \lambda \sim \text{Gamma}(\alpha_\lambda, \beta_\lambda) \tag{3}$$

where $w_{ijl} = \mathrm{W}_l[i, j]$ and $\lambda$ is a Gamma distributed precision parameter.

## 3 Learning Network Weight Distributions

The posterior and posterior predictive distributions described in Equations 1 and 2 can not be computed exactly and we need to resort to approximate Bayesian inference techniques. We study the ADF-based algorithms — PBP and EBP. To cope with the intractable posterior, both algorithms make a fully factorized approximation. PBP uses the following approximation,

$$\begin{aligned} q(\mathcal{W}, \lambda) &= q(\lambda \mid a_\lambda, b_\lambda) \prod_{l=1}^{L} \prod_{i=1}^{V_l} \prod_{j=1}^{V_{l-1}+1} q(w_{ijl} \mid \vartheta_{ijl}) \\ &= \text{Gamma}(\lambda \mid a_\lambda, b_\lambda) \prod_{l=1}^{L} \prod_{i=1}^{V_l} \prod_{j=1}^{V_{l-1}+1} \mathcal{N}(w_{ijl} \mid m_{ijl}, v_{ijl}), \end{aligned} \tag{4}$$

approximating the marginal posterior distribution of network weights with a set of univariate Gaussians parametrized by $\vartheta_{ijl} = \{m_{ijl}, v_{ijl}\}$. The continuous weight EBP variant[1] makes a similar but more restrictive assumption, artificially constraining the variances ($v_{ijl}$) to one. While the unconstrained approximations of PBP are expected to be more accurate, it comes at the cost of additional memory requirements. PBP requires storage of an additional variance parameter per synaptic weight. On the other hand, the cruder EBP approximation requires no more memory than traditional backpropagation-based learning. It is thus important to quantify the benefits of the more sophisticated approximation employed by PBP, both in terms of posterior estimation and generalization performance. We address these tradeoffs through careful experiments.

Assumed density filtering is an online inference algorithm that incrementally updates the posterior over $\mathcal{W}$ after observing new evidence. Consider the Bayes update to the approximate marginal posterior of weight $w_{ijl}$ after observing a new data pair $(y_n, x_n)$,

$$\tilde{q}(w_{ijl}) = \frac{1}{\mathrm{Z}} p(y_n \mid x_n, w_{ijl}) q(w_{ijl} \mid \vartheta_{ijl}^{n-1}), \tag{5}$$

where Z is the appropriate normalization constant and $\vartheta_{ijl}^{n-1}$ denotes posterior parameters after having observed $\mathcal{D}_{n-1} = \{(y_{n-1}, x_{n-1}), \ldots, (y_1, x_1)\}$. In general, the updated posterior ($\tilde{q}$) no longer has a simple parametric form and needs

[1] In this paper, we restrict ourselves to the continuous weight version of EBP which appears to both perform better than the binary version at tasks considered in this paper and is conceptually closer to PBP and backpropagation.

to be projected back to the parametric family of the approximate posterior by minimizing $\mathrm{KL}\left[\tilde{q}(w_{ijl}) \,||\, q(w_{ijl} \mid \vartheta_{ijl})\right]$ with respect to $\vartheta_{ijl}$. For Gaussian approximating families, the minimization yields the following update equations (Minka 2001b),

$$m_{ijl}^{n} = m_{ijl}^{n-1} + v_{ijl}^{n-1} \frac{\partial \ln \mathrm{Z}}{\partial m_{ijl}^{n-1}},$$
$$v_{ijl}^{n} = v_{ijl}^{n-1} - (v_{ijl}^{n-1})^2 \left[ \left( \frac{\partial \ln \mathrm{Z}}{\partial m_{ijl}^{n-1}} \right)^2 - 2 \frac{\partial \ln \mathrm{Z}}{\partial v_{ijl}^{n-1}} \right]. \quad (6)$$

For EBP only the mean updates are required since $v_{ijl}$ is constrained to 1. The posterior updates in Equation (6) require the log marginal likelihood, $\ln \mathrm{Z} = \ln \int p(y_n \mid x_n, \mathcal{W}))q(\mathcal{W}, \lambda)d\mathcal{W}d\lambda$. This quantity can be computed efficiently in a forward pass, wherein distributions are propagated forward through the network.

**Forward propagation of distributions.** Recall that the output of a neuron $i$ in layer $l$ is some transformation of its inputs $z_{il} = a(u_{il})$. For propagating distributions through the neuron, we observe that the mean ($\mu_{il}$) and variance ($\tau_{il}$) of its scaled inputs $u_{il} = \mathbf{w}_{il}^T \mathbf{z}_{l-1} / \sqrt{D_{l-1}}$ is given by,

$$\mu_{il} = \frac{1}{\sqrt{D_{l-1}}} \sum_{j=1}^{D_{l-1}} m_{ijl} \mathbb{E}\left[z_{jl-1}\right]$$
$$\tau_{il} = \frac{1}{D_{l-1}} \sum_{j=1}^{D_{l-1}} \mathbb{E}\left[w_{ijl}^2\right] \mathbb{E}\left[z_{jl-1}^2\right] - m_{ijl}^2 \mathbb{E}\left[z_{jl-1}\right]^2, \quad (7)$$

where $\mathbf{w}_{il}$ contains the $D_{l-1}$ elements of the $i^{\text{th}}$ row of $\mathrm{W}_l$. Appealing to the central limit theorem, we have $u_{il} \sim \mathcal{N}(\mu_{il}, \tau_{il})$, an approximation that is increasingly accurate with increasing $D_{l-1}$. With the Gaussian approximation in hand, we can compute the moments of the neuron output $z_{il}$ with respect to $\mathcal{N}(\mu_{il}, \tau_{il})$. For ReLU activations[2],

$$\mathbb{E}\left[z_{il}\right] = \mu_{il} \Phi\left(\frac{\mu_{il}}{\sqrt{\tau_{il}}}\right) + \tau_{il} \mathcal{N}(\mu_{il} \mid 0, \tau_{il})$$
$$\mathbb{E}\left[z_{il}^2\right] = (\mu_{il}^2 + \tau_{il}) \Phi\left(\frac{\mu_{il}}{\sqrt{\tau_{il}}}\right) + \mu_{il}\tau_{il} \mathcal{N}(\mu_{il} \mid 0, \tau_{il}), \quad (8)$$

where $\Phi(a) = \int_{-\infty}^{a} \mathcal{N}(0, 1)$ is a probit function. Assuming that the correlations between outputs of a layer can be ignored (Soudry, Hubara, and Meir 2014), we have, $\mathbf{z}_l \sim \mathcal{N}(\nu_l, \Psi_l)$, where $\nu_l = \left[\mathbb{E}\left[z_{il}\right], \ldots, \mathbb{E}\left[z_{D_l l}\right], 1\right]^T$ and $\Psi_l$ is a diagonal matrix containing $\left[(\mathbb{E}\left[z_{il}^2\right] - \mathbb{E}\left[z_{il}\right]^2), \ldots, 0\right]^T$ along the diagonal and we have appended the mean (1) and variance (0) of the bias unit. Starting with $\mathbb{E}\left[\mathbf{z}_0\right] = [x_n; 1]$ we can recursively compute the distribution of the activations $\mathbf{z}_l$ of intermediate layers, culminating in the means ($\nu_L$) and variances ($\Psi_L$) of the linear output layer $\mathbf{z}_L = g(x_n, \mathcal{W}) \sim \mathcal{N}(\nu_L, \Psi_L)$. We can thus approximate,

$$\ln \mathrm{Z} \approx \ln \int p(y_n \mid \mathbf{z}_L) \mathcal{N}(\mathbf{z}_L \mid \nu_L, \Psi_L) d\mathbf{z}_L. \quad (9)$$

The only requirement for the forward propagation is that the moments $\mathbb{E}\left[z_{il}\right]$ and $\mathbb{E}\left[z_{il}^2\right]$ be computable. These computations involve convolving the (squared) neuron activations with a Gaussian distribution, $\mathbb{E}\left[z_{il}\right] = \int a(u_{il}) \mathcal{N}(u_{il} \mid \mu_{il}, \tau_{il}) du_{il}$ and are easily computable for a wide variety of commonly used activations including rectified linear, sigmoidal and linear activations. Interestingly, discontinuous activations such as $sign(u_{il}) = \begin{cases} +1 & \text{if } u_{il} > 0 \\ -1 & \text{if } u_{il} < 0. \end{cases}$, pose no additional complications as long as the Gaussian convolution of the discontinuous pieces is computable. Discontinuities are smoothed away by the Gaussian convolution, allowing the algorithms to seamlessly deal with complex non linearities. Figure (1) displays the expected activations for the signed and the rectified linear activation functions. Other activation functions could be utilized, but fully exploring this space is left as future work.

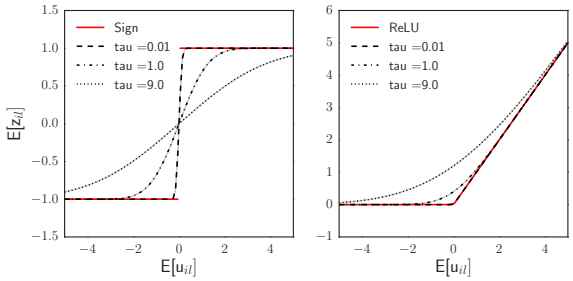


Figure 1: Expected activations of a neuron with sign (left) and rectified linear (right) non linearities. As the variance of the input decreases the expected responses tend towards the activations they approximate.

**Backward propagation of gradients.** The gradients of $\ln Z$ required in Equation (6) can be computed by reverse mode differentiation analogously to standard backpropagation. Powerful automatic differentiation tools like Theano (Bergstra et al. 2010) can be used to automate this process.

Following (Hernández-Lobato and Adams 2015), we incorporate the prior terms $\mathcal{N}(w_{ijl} \mid 0, \lambda^{-1})$ into the approximate posterior using expectation propagation (EP) (Minka 2001a). In our experimental evaluations, we replicate the settings of the original algorithms by learning the hyperparameters governing $\lambda$ via moment matching for PBP and fixing them such that $\mathbb{E}\left[\lambda\right] = \alpha_\lambda / \beta_\lambda$ is small, for EBP.

These algorithms share several resemblances with backpropagation. Similar to backpropagation, each iteration requires a forward pass followed by a backward pass. However, in the forward pass, distributions rather than point estimates are propagated through the network and the marginal probability of the target variable instead of the loss associated with the network prediction is computed. In the back-

[2]See appendix for derivations: http://www.disneyresearch.com/publication/assumed-density-filtering/

ward pass gradients of the log marginal probability are propagated backwards and used to update weight distributions.

## 4 Multiclass and Count Regression Problems

**Multiclass classification** involves categorizing features $x_n$ into one of $C$ classes. We model the labels as $C$ dimensional categorical random variables, $\mathbf{y}_n \in \{0,1\}^C$, with the $k^{th}$ element of $\mathbf{y}_n$ set to one, if $y_n = k$. This categorical distribution is parametrized by a softmax transformed output of a MLN with $C$ output units, $\mathbf{z}_L = g(x_n, \mathcal{W}) \in \mathbb{R}^C$

$$p(\mathbf{y} \mid \mathcal{W}, \mathbf{x}) = \prod_{n=1}^{N} \sigma(\mathbf{y}_n^T \mathbf{z}_L), \tag{10}$$

where $\sigma(a_j) = e^{a_j} / \sum_{k=1}^{C} e^{a_k}$ is the softmax function.

As explained in the previous section, learning the posterior distribution over network weights requires the computation of Equation (9), which for softmax likelihoods (Equation (10)) is,

$$\ln \mathrm{Z} \approx \ln \int e^{\mathbf{y}_n^T \mathbf{z}_L - \mathrm{lse}(\mathbf{z}_L)} \mathcal{N}(\mathbf{z}_L | \nu_L, \Psi_L) d\mathbf{z}_L, \tag{11}$$

where $\mathrm{lse}(\mathbf{a})$ is shorthand for the log-sum-exp function $\ln \left(\sum_j e^{a_j}\right)$. Unfortunately, this integral is analytically intractable. Furthermore, since Equation (6) needs to be computed for every $(x_n, y_n)$ pair and every epoch, any approximation we use needs to be efficient. Based on these observations, we first develop a bound using Jensen's inequality,

$$\begin{aligned} \ln \mathrm{Z} &\geq \int \mathcal{N}(\mathbf{z}_L | \nu_L, \Psi_L) \ln(e^{\mathbf{y}^T \mathbf{z}_L - \mathrm{lse}(\mathbf{z}_L)}) d\mathbf{z}_L \\ &= \mathbb{E}_{q_\phi}\left[\mathbf{y}^T \mathbf{z}_L - \mathrm{lse}(\mathbf{z}_L)\right] = \mathbf{y}^T \nu_L - \mathbb{E}_{q_\phi}\left[\mathrm{lse}(\mathbf{z}_L)\right], \end{aligned} \tag{12}$$

where for notational convenience we denote the approximate posterior $\mathcal{N}(\mathbf{z}_L | \nu_L, \Psi_L)$ as $q_\phi$. This lower bound is itself intractable, owing to the intractability of $\mathbb{E}_{q_\phi}\left[\mathrm{lse}(\mathbf{z}_L)\right]$. We cope by upper bounding the offending expectation (Blei and Lafferty 2006),

$$\mathbb{E}_{q_\phi}\left[\ln \sum_j e^{\mathbf{z}_{L_j}}\right] \leq \ln \sum_j \mathbb{E}_{q_\phi}\left[e^{\mathbf{z}_{L_j}}\right] = \ln \sum_j e^{\nu_{L_j} + \psi_{L_j}/2}, \tag{13}$$

which leads to a tractable lower bound,

$$\ln \mathrm{Z} \geq \mathbf{y}^T \nu_L - \mathrm{lse}(\nu_L + \psi_L/2), \tag{14}$$

where $\psi_L$ is a vector contains the diagonal elements of $\Psi_L$. Approximating the normalization constant with the lower bound leads to the following gradients,

$$\begin{aligned} &\frac{\partial \ln \mathrm{Z}}{\partial\, \nu_L} = (\mathbf{y} - \mathbf{t}), \quad \frac{\partial \ln \mathrm{Z}}{\partial\, \Psi_L} = -\frac{1}{2} T, \\ &\ln \mathbf{t} = (\nu_L + \psi_L/2) - \mathrm{lse}(\nu_L + \psi_L/2), \end{aligned} \tag{15}$$

where $T$ is a diagonal matrix whose diagonal is populated by $\mathbf{t}$. These are backpropagated through the network to obtain the gradients necessary for posterior updates (Equation (6)). The bound in Equation (14) is sometimes referred to as the "log bound", and is attractive for our purposes because of its efficiency — there are no additional free parameters to estimate via expensive numerical methods and being a zeroth order Taylor series approximation no hessians need to be computed. Although more accurate alternate bounds have been proposed (Khan 2012), approximating the lse expectation remains a challenging open problem. As an alternative, we observe that updating the posterior approximation over network weights only requires the availability of $\bigtriangledown_\phi \ln Z$. This allows us to sidestep the difficult problem of accurately bounding the lse function. We utilize recently proposed stochastic approximation techniques that directly provide unbiased estimates of $\bigtriangledown_\phi \ln Z$ through a Monte Carlo approximation (Paisley, Blei, and Jordan 2012; Kingma and Welling 2013; Rezende, Mohamed, and Wierstra 2014; Titsias and Lázaro-Gredilla 2014). Assuming mild regularity conditions we have,

$$\begin{aligned} \bigtriangledown_\phi \ln Z &= \frac{1}{Z} \bigtriangledown_\phi \mathbb{E}_{q_\phi}\left[p(\mathbf{y}_n \mid \mathbf{z}_L)\right] \\ &= \frac{1}{Z} \mathbb{E}_{q_\phi}\left[p(\mathbf{y}_n \mid \mathbf{z}_L) \bigtriangledown_\phi \ln q(\mathbf{z}_L \mid \phi)\right] \\ &\approx \frac{1}{Z}\left[\frac{1}{S}\sum_{s=1}^{S} p(\mathbf{y}_n \mid \mathbf{z}_L^{(s)}) \bigtriangledown_\phi \ln q(\mathbf{z}_L^{(s)} \mid \phi)\right], \end{aligned} \tag{16}$$

where $\{\mathbf{z}_L^{(s)}\}_{s=1}^S \sim q(\mathbf{z}_L^{(s)} \mid \phi)$. Unfortunately, this estimator is known to have high variance and further variance reduction techniques are required (Paisley, Blei, and Jordan 2012). However, since $\mathbf{z}_L$ are continuous random variables, an alternate lower variance estimator of the gradient proposed by (Kingma and Welling 2013) becomes available to us. The basic idea is to transform the random variable to be sampled ($\mathbf{z}_L$) such that the randomness is independent of the parameters ($\phi$) with respect to which gradients are desired. The following deterministic transformation can be used in our case of Gaussian distributed $\mathbf{z}_L$,

$$\epsilon^{(s)} \sim \mathcal{N}(0, \mathbf{I}), \quad \mathbf{z}_L^{(s)} = t(\phi, \epsilon^s) = \nu_L + L\epsilon^s \tag{17}$$

where $LL^T = \Psi_L$. Following this re-parameterization, computing the gradient

$$\bigtriangledown_\phi \ln Z \approx \frac{1}{Z}\left[\frac{1}{S}\sum_{s=1}^{S} \bigtriangledown_\phi p(\mathbf{y}_n \mid t(\phi, \epsilon^{(s)}))\right], \tag{18}$$

is just a matter of applying the chain rule. Backpropagating this estimator through the network provides us with the desired gradients necessary for posterior updates.

We predict class labels for unseen features using another Monte Carlo approximation to the posterior predictive distribution.

**Count Regression** involves learning a mapping between data instances $x_n$ and counts $y_n \in \mathbb{Z}_+ = \{0, 1, 2 \ldots, \}$. We model the count observations as Poisson distributed random variables,

$$p(y_n \mid \rho) = \frac{\rho^{y_n}}{y_n!} e^{-\rho}, \tag{19}$$

and parametrize the non-negative rate parameter as $\rho = e^{\mathbf{z}_L}$, with $\mathbf{z}_L = g(x_n, \mathcal{W}) \in \mathbb{R}^1$ being the output of a MLN

$g$ with a single output unit. As with softmax classification Equation (9) for Poisson regression,

$$\ln \mathrm{Z} \approx \ln \int e^{\mathbf{z}_L y_n - e^{\mathbf{z}_L} - \ln y_n!} \mathcal{N}(\mathbf{z}_L | \nu_L, \psi_L) d\mathbf{z}_L, \quad (20)$$

is intractable. Here, we use an accurate approximation which induces an exponential family posterior predictive distribution (Chan and Vasconcelos 2009; El-Sayyad 1973). The approximation is attractive because it provides straightforward procedures for predicting responses on unseen data by, for example, using the distributions mode.

Since we use an exponential link function ($e^{\mathbf{z}_L}$), our model implies that the log of the Poisson rate parameter $\ln \rho = \mathbf{z}_L$ follows a Gaussian distribution. A Gaussian $\mathcal{N}(\mu, \sigma^2)$ is well approximated by a log-Gamma$(\sigma^{-2}, \sigma^2 e^{\mu})$ distribution for large $\sigma^{-2}$. This allows us to approximate the distribution of the rate parameter as,

$$\rho \sim \text{Gamma}(\psi_L^{-1}, \psi_L e^{\nu_L}). \quad (21)$$

The Gamma-Poisson conjugacy then provides us a convenient approximation to Equation (20):

$$\begin{aligned} \ln \mathrm{Z} &\approx \ln \int_0^{\infty} \text{Poi}(y_n \mid \rho) \text{Gamma}(\rho \mid \psi_L^{-1}, \psi_L e^{\nu_L}) d\rho \\ &= \ln \frac{\Gamma(y_n + 1/\psi_L)}{\Gamma(y_n + 1)\Gamma(1/\psi_L)} \left( \frac{\psi_L e^{\nu_L}}{1 + \nu_L e^{\psi_L}} \right)^{y_n} \left( \frac{1}{1 + \psi_L e^{\nu_L}} \right)^{1/\psi_L}, \end{aligned} \quad (22)$$

where the expression inside the logarithm is just the probability mass function of a negative binomial distribution NB$(mean = e^{\nu_L}, scale = \psi_L)$ and follows from standard Gamma-Poisson conjugacy. Predictions for held out data $x_*$ involves computing, $\int \text{Poi}(y_* \mid e^{g(x_*, \mathcal{W})}) q_\phi(\mathcal{W}, \lambda) d\mathcal{W} d\lambda$. The approximations listed in Equations 21 and 22 lead to a negative binomial approximation to the posterior predictive distribution: $y_* \sim \text{NB}(e^{\mu_*}, \psi_*)$. We use the mode of the negative binomial distribution, $\lfloor (1 - \psi_*) e^{\nu_*} \rfloor$ if $\psi_* < 1$ and zero otherwise, for making predictions.

## 5 Regression and Binary Classification

**Binary Classification** We are interested in categorizing $x_n$ into one of two classes. The class labels are binary random variables $y_n = \{+1, -1\}$, modeled using probit likelihoods,

$$p(\mathbf{y} \mid \mathcal{W}, \mathbf{x}) = \prod_{n=1}^{N} \Phi(y_n \mathbf{z}_L) \quad (23)$$

where, $\mathbf{z}_L = g(x_n, \mathcal{W}) \in \mathbb{R}^1$ is the output of a MLN $g$. Probit likelihoods allow the analytical computation of ln Z,

$$\approx \ln \int \Phi(y_n \mathbf{z}_L) \mathcal{N}(\mathbf{z}_L | \nu_L, \psi_L) d\mathbf{z}_L = \ln \Phi \left( \frac{y_n \nu_L}{\sqrt{1 + \psi_L}} \right). \quad (24)$$

Our model is similar to those considered by (Soudry, Hubara, and Meir 2014) for binary classification problems. However, our algorithms also work with ReLU activations and we develop the analogous PBP extension.

**Continuous Regression** Here the responses $y_n \in \mathbb{R}^1$ are real random variables, and we model them as Gaussian distributed random variables,

$$p(\mathbf{y} \mid \mathcal{W}, \mathbf{x}, \gamma) = \prod_{n=1}^{N} \mathcal{N}(y_n \mid g(x_n, \mathcal{W}), \gamma^{-1}) \quad (25)$$

The mean of the Gaussian is parametrized by $z_L = g(x_n, \mathcal{W}) \in \mathbb{R}^1$ and $\gamma \sim \text{Gamma}(\alpha_\gamma, \beta_\gamma)$ controls the variance. This regression model was used in conjunction with PBP in (Hernández-Lobato and Adams 2015). We additionally learn it using EBP.

## 6 Experiments

In this section, we empirically evaluate the proposed extensions. We begin by comparing EBP — extended to incorporate rectified linear units — and PBP on continuous regression and binary classification problems replicating the datasets and architectures described in the original works. Next, we present experiments for vetting the multiclass and count regression extensions.

For regression, we use the UCI[3] datasets used by (Hernández-Lobato and Adams 2015). To replicate the experimental settings, we use networks with one $50$ unit hidden layer for all but the two largest datasets, (*Protein* and *Year Prediction*, datasets 7 and 10 in Figure (2)) using 100 hidden units instead. All datasets are standardized to have features with zero mean and unit standard deviation. The datasets are split such that the train and test sets follow a $90/10$ ratio. We repeat the random splitting process $10$ times for each dataset (except *Year Prediction*, where computational concerns limit us to a single experiment) and report the average performance across the splits after training for $100$ epochs. We measure error using root mean squared error (RMSE). Figure (2) presents the mean test errors achieved by the two algorithms along with associated standard deviations. We find that PBP significantly outperforms EBP on most regression datasets. The RMSE score achieved by PBP averaged over all datasets and splits is $2.65 \pm 2.81$, compared to EBP's $3.87 \pm 4.03$. We also find a similar trend holds for predictive log likelihoods, PBP's $-1.34 \pm 2.15$ compared to EBP's $-2.11 \pm 2.20$, indicating a better fit to data. Experiments with deeper architectures are available in the appendix and are consistent with these observations.

Next, for binary classification we use the datasets made available by (Soudry, Hubara, and Meir 2014), summarized in the appendix. Again, following the authors' settings, we use an architecture with a single hidden layer containing $120$ rectified linear units. We then follow the experimental protocol used for regression. The training and test error percentage is available in Figure 2. Consistent with regression, we find that PBP achieves both a lower error rate of $0.07 \pm 0.06$ and a higher test log likelihood $-0.33 \pm 0.45$, when compared to EBP's $0.27 \pm 0.12$ and $-0.50 \pm 0.15$.

[3] https://archive.ics.uci.edu/ml/datasets.html, see appendix for dataset details.

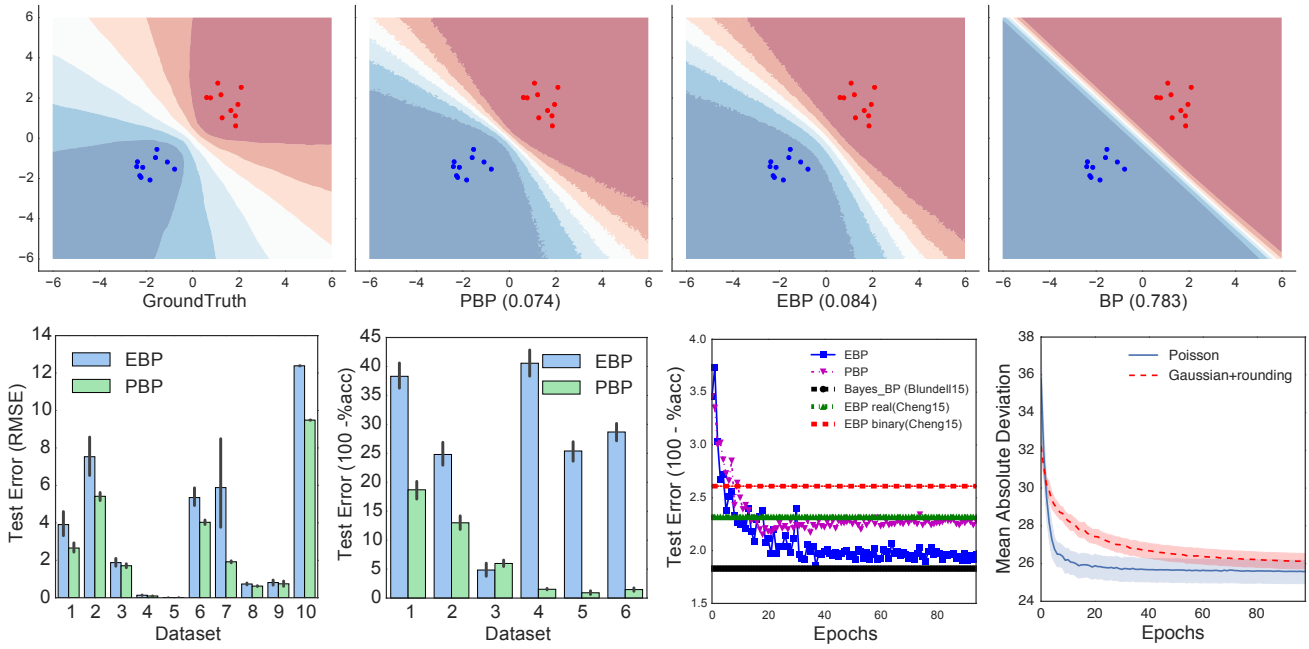


Figure 2: **Top**: Posterior distributions inferred by different approaches on toy data. The numbers in parenthesis represent average point wise KL divergence from the ground truth. **Bottom**: Test errors for different tasks. From left to right, we have RMSE errors on regression datasets, 0-1 error for binary classification datasets, classification performance on MNIST and test mean absolute deviation for count regression.

For multiclass problems we have two available approximations — log bound and stochastic. In preliminary experiments, we found the stochastic approximation to be more effective and only consider it in this section. Following (Balan et al. 2015), we first compare the posterior predictive densities learned by EBP and PBP on synthetic data. We generate ten 2D data points from two well separated classes and train a network with a single ten ReLU hidden layer and a two dimensional softmax output layer. We place a vague Gaussian prior on the weights $\mathcal{N}(w \mid 0, 100)$. We also compute the posterior predictive density using standard backpropagation and the gold standard "ground-truth" density by averaging over $20,000$ samples from the No-U-Turn sampler (Hoffman and Gelman 2014) after a burn-in of $5000$. The resulting distributions are visualized in Figure (2), with darker hues indicating higher probability of belonging to the appropriate class. For quantitative evaluations, Figure (2) also includes point wise KL divergence over an evenly spaced discrete grid spanning $[-6, 6] \times [-6, 6]$ between the ground-truth and the competing methods. Both ADF algorithms incorporate uncertainty in parameter estimates and perform significantly better than backpropagation which tends to be overconfident in its predictions even far from the observed data. The more flexible approximation employed by PBP produces marginal improvements over EBP. In general, both EBP and PBP tend to underestimate the uncertainty, a byproduct of performing multiple ADF passes over the data wherein the same data points are repeatedly incorporated into the approximate posterior without discounting for having previously observed them (Minka 2001a).

Next, we focus on the MNIST hand written digit dataset which contains $60,000$ training and $10,000$ test images. Without using accuracy enhancing methods like generative pre-training, data augmentation or convolutions we focus on measuring the performance of a vanilla feedforward neural networks on this dataset. We employ a two hidden layer architecture, each containing 400 rectified linear units and train for a hundred epochs. In addition to internal comparisons, we also compare against the Bayes by backprop method of (Blundell et al. 2015)[4] and the method reported in (Cheng et al. 2015), which does not provide well calibrated probabilities, who use identical architectures. The results are displayed in Figure (2). Interestingly, on MNIST both EBP and PBP perform quite well and are better than (Cheng et al. 2015) while being comparable to (Blundell et al. 2015). Furthermore, unlike (Blundell et al. 2015), our algorithms are parameter free, and do not require a separate validation set to tune hyperparameters. Finally, we note that in recent, as yet unpublished work (Balan et al. 2015) report improved results using their SGLD based algorithm. A thorough comparison with their work is left for the future.

Finally, we test our count regression model on a real world bike rental demand forecasting problem (Fanaee-T and Gama 2013). Given a combination of historical usage patterns and weather data from the Capital Bikeshare program in Washington, D.C., logged on an hourly basis between 2011 and 2012, the goal is to forecast hourly bike rental demand. A sensible alternative to count regression is to use Gaussian regression (Equation (25)) and to round the continuous predictions to the nearest integer. Moreover, the

[4]The numbers are reproduced from their paper

continuous regression model in Equation (25) has access to a precision $\gamma$ parameter that the Poisson model does not, allowing it to capture over-dispersion effects that the Poisson model can not. This extra flexibility makes the Gaussian regression with rounding a strong benchmark to compare against. On a network with a single hidden layer containing $100$ units, we train both models using PBP for a $100$ epochs on ten $90/10$ splits. Average test error, measured via mean absolute deviation, as a function of epochs is displayed in Figure (2). We find that our Poisson model achieves lower errors, requires fewer training epochs and achieves higher predictive log likelihoods $-3.2108 \pm 0.03$ compared to $-5.12 \pm 0.04$ achieved by the Gaussian model.

## 7 Conclusion

In this paper, we developed extensions to PBP and EBP, two scalable online algorithms for training Bayesian neural networks, to multiclass and count regression problems and found the proposed extensions to be effective, producing competitive results. We also developed an EBP variant for ReLU and continuous regression problems and a PBP variant for binary classification allowing us to compare the two algorithms. We found that on most datasets the more sophisticated posterior approximation employed by PBP leads to better generalization performance and posterior estimates.

## References


Balan, A. K.; Rathod, V.; Murphy, K.; and Welling, M. 2015. Bayesian dark knowledge. *arXiv* abs/1506.04416.

Bergstra, J.; Breuleux, O.; Bastien, F.; Lamblin, P.; Pascanu, R.; Desjardins, G.; Turian, J.; Warde-Farley, D.; and Bengio, Y. 2010. Theano: a CPU and GPU math expression compiler. In *SciPy*, 1–7.

Blei, D., and Lafferty, J. 2006. Correlated topic models. In *NIPS*. 147–154.

Blundell, C.; Cornebise, J.; Kavukcuoglu, K.; and Wierstra, D. 2015. Weight uncertainty in neural networks. In *ICML*.

Chan, A. B., and Vasconcelos, N. 2009. Bayesian Poisson regression for crowd counting. In *ICCV*, 545–551.

Cheng, Z.; Soudry, D.; Mao, Z.; and Lan, Z. 2015. Training binary multilayer neural networks for image classification using expectation backpropagationn. *arXiv preprint arXiv:1503.03562*.

El-Sayyad, G. 1973. Bayesian and classical analysis of poisson regression. *J. Royal Stat. Soc. Series B (Methodological)* 445–451.

Fanaee-T, H., and Gama, J. 2013. Event labeling combining ensemble detectors and background knowledge. *Prog. in AI* 1–15.

Graves, A. 2011. Practical variational inference for neural networks. In *NIPS*, 2348–2356.

Hernández-Lobato, J. M., and Adams, R. 2015. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *ICML*, 1861–1869.

Hinton, G. E., and Van Camp, D. 1993. Keeping the neural networks simple by minimizing the description length of the weights. In *COLT*, 5–13.

Hoffman, M. D., and Gelman, A. 2014. The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo. *JMLR* 15(1):1593–1623.

Jylänki, P.; Nummenmaa, A.; and Vehtari, A. 2014. Expectation propagation for neural networks with sparsity-promoting priors. *JMLR* 15(1):1849–1901.

Khan, M. 2012. Variational learning for latent Gaussian model of discrete data. *Ph.D. Thesis, Dept. of Computer Science, UBC*.

Kingma, D. P., and Welling, M. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Krizhevsky, A.; Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*, 1097–1105.

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

Minka, T. P. 2001a. Expectation propagation for approximate bayesian inference. In *UAI '01*, 362–369.

Minka, T. P. 2001b. *A family of algorithms for approximate Bayesian inference*. Ph.D. Dissertation, MIT.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533.

Nair, V., and Hinton, G. E. 2010. Rectified linear units improve restricted Boltzmann machines. In *ICML*, 807–814.

Neal, R. M. 1995. *Bayesian Learning for Neural Networks*. Ph.D. Dissertation, University of Toronto.

Opper, M. 1998. On-line learning in neural networks. chapter A Bayesian Approach to On-line Learning, 363–378.

Paisley, J.; Blei, D. M.; and Jordan, M. I. 2012. Variational Bayesian inference with stochastic search. In *ICML*.

Rezende, D. J.; Mohamed, S.; and Wierstra, D. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 1278–1286.

Rumelhart, D. E.; Rumelhart, D. E.; Hinton, G. E.; and Williams, R. J. 1986. Learning representations by back-propagating errors. *Nature* 323(Oct):533–536+.

Soudry, D.; Hubara, I.; and Meir, R. 2014. Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights. In *NIPS*. 963–971.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: A simple way to prevent neural networks from overfitting. *JMLR* 15(1):1929–1958.

Sutskever, I.; Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *NIPS*, 3104–3112.

Titsias, M., and Lázaro-Gredilla, M. 2014. Doubly stochastic variational Bayes for non-conjugate inference. In *ICML*, 1971–1979.

Wan, L.; Zeiler, M.; Zhang, S.; Cun, Y. L.; and Fergus, R. 2013. Regularization of neural networks using dropconnect. In *ICML*, 1058–1066.

Welling, M., and Teh, Y. W. 2011. Bayesian learning via stochastic gradient Langevin dynamics. In *ICML*, 681–688.

# Supplementary Material for Assumed Density Filtering Methods for Scalable Learning of Bayesian Neural Networks

## 1 Rectified Linear Units

In the forward pass, we need to compute $\mathbb{E}[z_{il}]$ and $\mathbb{E}\left[z_{il}^2\right]$. Here we derive the expression in Equation 8 of the main text.

We know that $z_{il} = max(0, u_{il})$. The expectation can be computed as follows,

$$\begin{aligned}\mathbb{E}[z_{il}] &= \int_{-\infty}^{+\infty} \max(0, u_{il}) \mathcal{N}(u_{il} \mid \mu_{il}, \tau_{il}) du_{il} \\ &= \int_{0}^{\infty} u_{il} \mathcal{N}(u_{il} \mid \mu_{il}, \tau_{il}) du_{il}\end{aligned} \tag{1}$$

Dropping the subscripts and substituting $m = \frac{u-\mu}{\tau^{1/2}}$ we have,

$$\begin{aligned}&\mathbb{E}[z_{il}] = \int_{\frac{-\mu}{\sqrt{\tau}}}^{\infty} (\mu + \tau^{1/2} m) \frac{exp(-m^2/2)}{\sqrt{2\pi}} dm \\ &= \mu \int_{\frac{-\mu}{\sqrt{\tau}}}^{\infty} \frac{exp(-m^2/2)}{\sqrt{2\pi}} dm + \sqrt{\tau} \int_{\frac{-\mu}{\sqrt{\tau}}}^{\infty} m \frac{exp(-m^2/2)}{\sqrt{2\pi}} dm \\ &= \mu \Phi\left(\frac{\mu}{\sqrt{\tau}}\right) + \tau \mathcal{N}(\mu \mid 0, \tau)\end{aligned} \tag{2}$$

Next, we show how to compute the second moment.

$$\mathbb{E}\left[z_{il}^2\right] = \int_{0}^{\infty} u_{il}^2 \mathcal{N}(u_{il} \mid \mu_{il}, \tau_{il}) du_{il} \tag{3}$$

Again dropping the subscripts and substituting $m = \frac{u-\mu}{\tau^{1/2}}$ we have,

$$\begin{aligned}&\mathbb{E}\left[z_{il}^2\right] = \int_{\frac{-\mu}{\sqrt{\tau}}}^{\infty} (\mu + \tau^{1/2} m)^2 \frac{exp(-m^2/2)}{\sqrt{2\pi}} dm \\ &= \mu^2 \Phi\left(\frac{\mu}{\sqrt{\tau}}\right) + \frac{2\mu\sqrt{\tau}}{\sqrt{2\pi}} exp(-\mu^2/2\tau) + \tau \int_{\frac{-\mu}{\sqrt{\tau}}}^{\infty} m^2 \frac{e^{-m^2/2}}{\sqrt{2\pi}} dm \\ &= \mu^2 \Phi\left(\frac{\mu}{\sqrt{\tau}}\right) + \frac{2\mu\sqrt{\tau}}{\sqrt{2\pi}} exp(-\mu^2/2\tau) - \\ &\frac{\sqrt{\tau}\mu}{\sqrt{2\pi}} exp(-\mu^2/2\tau) + \tau \Phi\left(\frac{\mu}{\sqrt{\tau}}\right)\end{aligned} \tag{4}$$



Rearranging terms we have,

$$\mathbb{E}\left[z_{il}^2\right] = (\mu^2 + \tau)\Phi\left(\frac{\mu}{\sqrt{\tau}}\right) + \mu\tau\mathcal{N}(\mu \mid 0, \tau) \tag{5}$$

## 2 Multiclass posterior predictive distribution

The posterior predictive distribution for a new feature $x_*$ can be calculated through a Monte Carlo approximation.

$$\begin{aligned}p(y_* \mid x_*, \mathcal{D}) &= \int p(y_* \mid x_*, \mathcal{W}) p(\mathcal{W}, \lambda \mid \mathbf{y}, \mathbf{x}) d\mathcal{W} d\lambda \\ &\approx \int p(y_* \mid x_*, \mathcal{W}) q(\mathcal{W}, \lambda \mid \mathbf{y}, \mathbf{x}) d\mathcal{W} d\lambda \\ &= \int p(y_* \mid x_*, \mathcal{W}) q(\mathcal{W} \mid \vartheta) d\mathcal{W} \\ &\approx \int \sigma(\mathbf{z}_L) \mathcal{N}(\mathbf{z}_L \mid \nu_L, \Psi_L) d\mathbf{z}_L \\ &\approx \frac{1}{S} \sum_s \mathbf{z}_L^s, \quad \mathbf{z}_L^s \sim \mathcal{N}(\mathbf{z}_L \mid \nu_L, \Psi_L)\end{aligned} \tag{6}$$

Our experiments used $S = 100$ samples.

## 3 Continuous regression and Binary classification experiments

### Descriptions of datasets

We used ten UCI regression datasets for comparing PBP against rectified linear EBP. Table 1 summarizes the characteristics of the different datasets. The order of the presented datasets correspond to the labeling 1 through 10 used in the main paper For binary classification, we used text classification datasets summarized in Table 2.

### Test log likelihoods

In Figure 1 we present the per dataset test log-likelihoods achieved by PBP and EBP on continuous regression and binary classification. Notice that on a large majority PBP achieves higher predictive log-likelihoods.

### Multi layer experiments

We also compared the performance of EBP and PBP on multilayer architectures. Table 3 we summarize results from 2 and 5 layer networks on regression datasets.

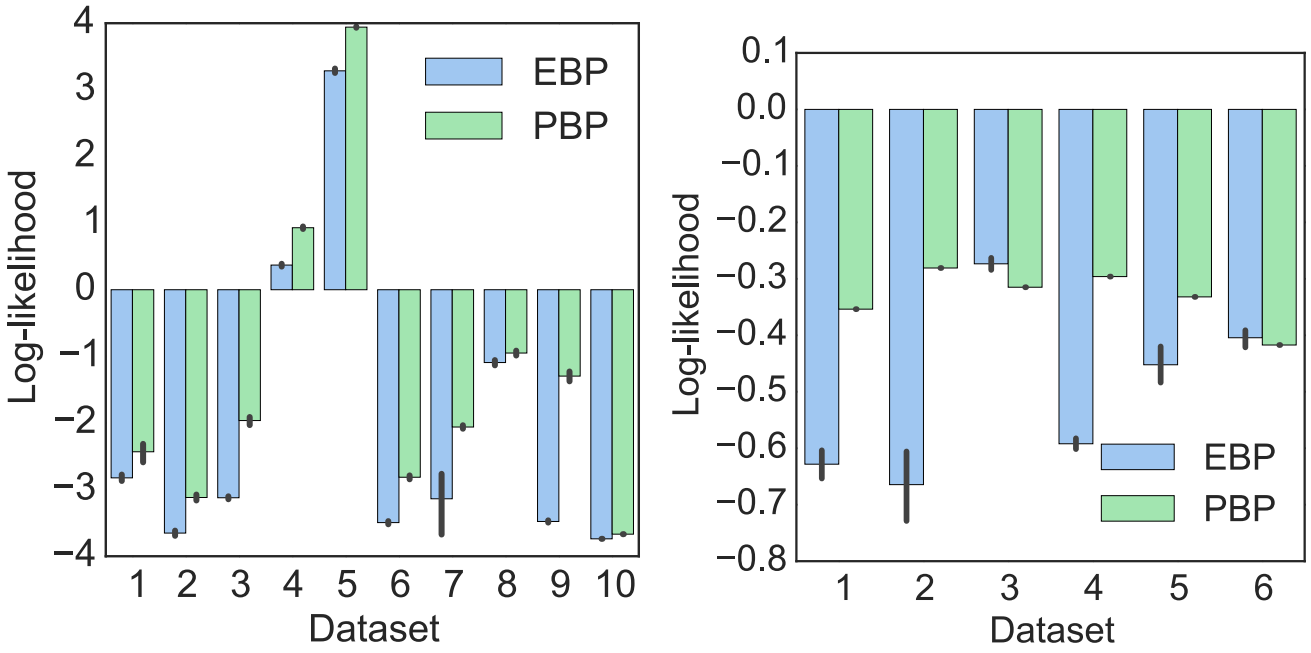


Figure 1: Test log likelihoods on regression (left) and classification datasets(right)

| | Dataset | RMSE EBP 2layer | RMSE PBP 2layer | RMSE EBP 5layer | RMSE PBP 5layer |
|---|---|---|---|---|---|
| 1 | Boston | $3.14 \pm 0.93$ | $\mathbf{2.79 \pm 0.16}$ | $9.33 \pm 1.0$ | $3.28 \pm 0.15$ |
| 2 | Concrete | $5.30 \pm 0.77$ | $\mathbf{5.24 \pm 0.11}$ | $6.33 \pm 0.91$ | $6.96 \pm 0.16$ |
| 3 | Energy Efficiency | $1.38 \pm 0.17$ | $\mathbf{0.90 \pm 0.04}$ | $3.54 \pm 3.03$ | $1.08 \pm 0.06$ |
| 4 | Kin8nm | $0.07 \pm 0.22$ | $\mathbf{0.07 \pm 0.00}$ | $0.18 \pm 0.09$ | $0.10 \pm 0.00$ |
| 5 | Naval Propulsion | $0.007 \pm 0.00$ | $\mathbf{0.003 \pm 0.00}$ | $0.007 \pm 0.00$ | $0.006 \pm 0.00$ |
| 6 | Power Plant | $4.21 \pm 0.23$ | $\mathbf{4.03 \pm 0.03}$ | $4.56 \pm 0.25$ | $4.48 \pm 0.04$ |
| 7 | Protein Structure | $\mathbf{2.14 \pm 0.17}$ | $4.25 \pm 0.02$ | $2.04 \pm 0.15$ | $3.47 \pm 0.04$ |
| 8 | Wine | $0.71 \pm 0.06$ | $\mathbf{0.64 \pm 0.00}$ | $0.82 \pm 0.04$ | $0.65 \pm 0.01$ |
| 9 | Yacht | $1.14 \pm 0.45$ | $\mathbf{0.85 \pm 0.05}$ | $5.58 \pm 5.77$ | $1.92 \pm 0.23$ |
| 10 | Year Prediction | 9.21 | 8.21 | NA | 8.93 |

Table 3: RMSE test error rates for EBP and PBP using 2 layer and 5 layer architectures.

## 4 Multiclass Experiments

Here we present additional results for the multiclass experiments.

### Log bound vs Stochastic approximation

We evaluated the differences between log bound and stochastic approximations by measuring their performance on three multi class datasets – MNIST, UCI HAR, a six class human activity recognition dataset and Sensorless Drive Diagnosis Data Set, a 11 class dataset for detecting malfunctioning components. On each dataset we trained a network with 2 hidden layers of 400 units each and trained for a 100 epochs using both PBP and EBP. Figure 2 summarizes the performance of the two approximations, averaged over all datasets and the two algorithms (EBP and PBP). This clearly demonstrates the superior performance of the stochastic approximation. For this experiment, we used 100 samples, but a similar trend holds even with a single sample.

### Stochastic approximation quality

First, we study the effect of the number of stochastic samples $S$ on the ln $Z$ approximation quality. We generate synthetic 10 and 100 class datasets by sampling Gaussian distributed $x_n$ and $\mathcal{W}$ and then conditionally sampling our multiclass model to generate $y_n$. We ensure that each class contains $\approx 100$ examples. We then train on these datasets for 50 epochs, using both PBP and EBP for different settings of $S = \{1, 10, 100\}$ repeating the process 20 times. The average training log likelihoods achieved by the different approximations are displayed in Figure 3. As expected, the variance decreases with increasing S. We find that approximations with $S = 1$ exhibit high variance but the variance decreases rapidly with increasing $S$, with $S = 100$ exhibiting only marginally lower variance than $S = 10$. The dis-

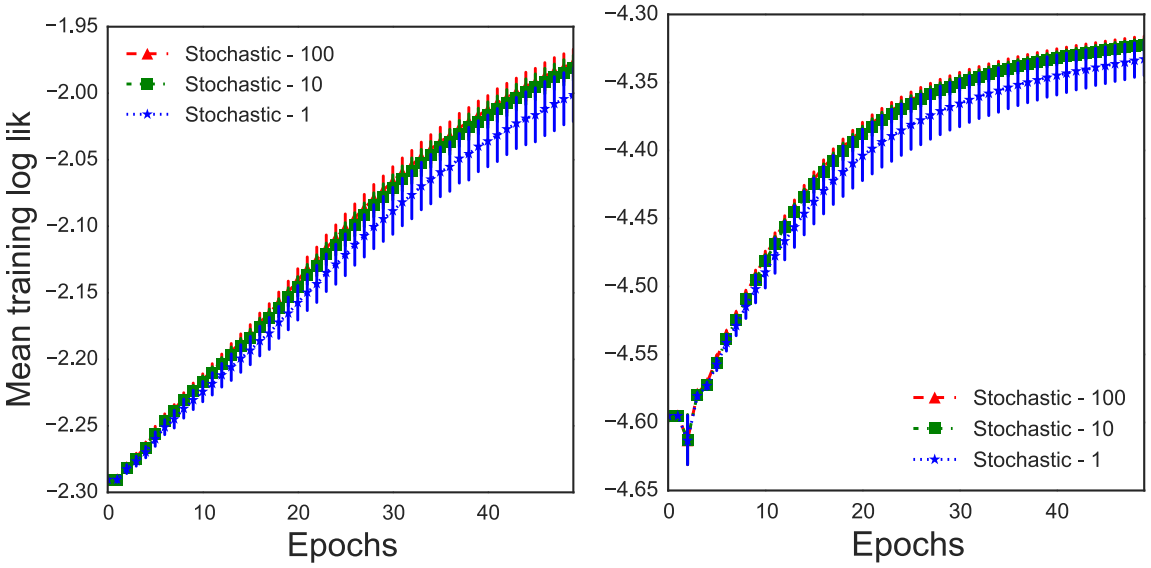


Figure 3: Training log likelihood variance on synthetic data, with 10 (left) and 100 (right) classes.

| | Dataset | $N$ | d |
|---|---|---|---|
| 1 | Boston | 506 | 13 |
| 2 | Concrete Compression Strength | 1030 | 8 |
| 3 | Energy Efficiency | 768 | 8 |
| 4 | Kin8nm | 8192 | 8 |
| 5 | Naval Propulsion | 11,934 | 16 |
| 6 | Combined Cycle Power Plant | 9568 | 4 |
| 7 | Protein Structure | 9568 | 4 |
| 8 | Wine Quality Red | 1599 | 11 |
| 9 | Yacht Hydrodynamics | 308 | 6 |
| 10 | Year Prediction MSD | 515,345 | 90 |

Table 1: Continuous regression datasets

| | Dataset | $N$ | d |
|---|---|---|---|
| 1 | 20News group comp vs HW | 1943 | 29409 |
| 2 | 20News group elec vs med | 1971 | 38699 |
| 3 | Spam or ham d0 | 2500 | 26580 |
| 4 | Spam or ham d1 | 2500 | 27523 |
| 5 | Reuters news I8 | 2000 | 12167 |
| 6 | Reuters news I6 | 2000 | 11463 |

Table 2: Binary classification datasets

played results are for PBP, EBP performs similarly.

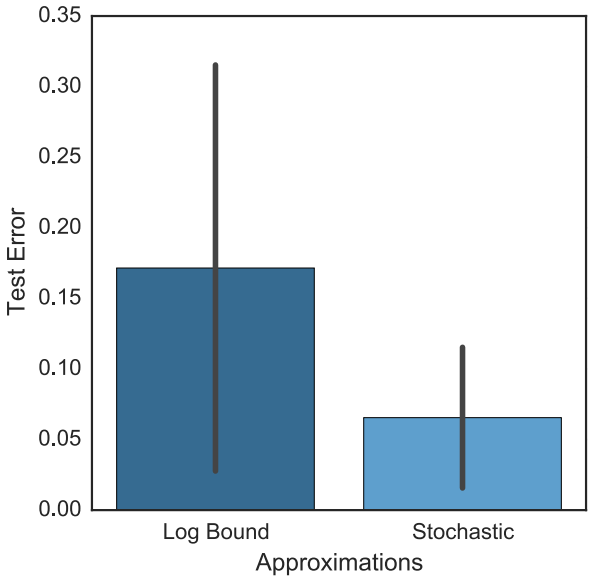


Figure 2: Performance of Log bound vs Stochastic approximations, averaged over algorithms (EBP and PBP) and datasets.