BAYESIAN NEURAL NETWORKS - REGRESSION AND CLASSIFICATION

Simon Aertssen (SA) - s181603, Peyman Kor (PK) - s191828

Technical University of Denmark

ABSTRACT

(SA) This paper aims to investigate and reproduce the results presented in *Weight Uncertainty in Neural Networks* [1]. We investigate the novel backpropagation algorithm *Bayes by Backprop* for learning the probability distribution on the weights of a neural network. The weights are regularised by minimising a compression cost, the variational free energy [2], balancing between the complexity of the data and simple prior assumptions. First we verify that this regularization yields comparable performance to dropout on MNIST classification. We then verify how the learnt uncertainty in the weights can be used to improve generalisation in nonlinear regression problems. The project code can be found here.

.

1. INTRODUCTION

(SA) As it is noted in the field of machine learning, a machine can make the wrong decisions with a 100% confidence. In classification algorithms, this confidence is obtained by applying a softmax on the predictions, which does not accurately reflect a probability. We can build uncertainty into the model by learning probability distributions instead of learning the numerical values of the weights themselves, as illustrated in Fig. 1. This method infers the posterior distribution over the weights in the network, using simple prior assumptions. This construction has a regularizing effect as an ensemble of networks is obtained: weights are sampled from the learned distributions in every forward pass. A desired number of predictions are then made per input.

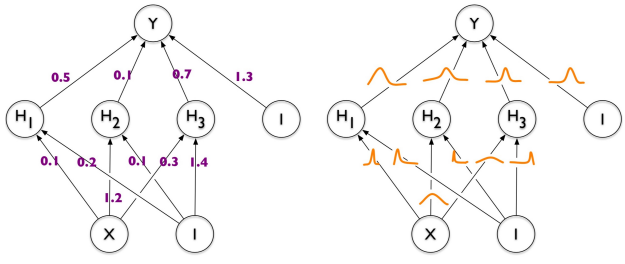

Fig. 1. Model architecture representation of feed-forward (FFNN) versus Bayesian neural networks (BNN). Uncertainty is built into the model by learning the probability density functions from which weights are sampled. [1]

2. POINT ESTIMATES VERSUS BAYES

(SA) We start by viewing the neural network as a probabilistic model $P(\mathbf{y}|\mathbf{x}, \mathbf{w})$. The objective is to predict an unknown target $\mathbf{y}^*$ from test input $\mathbf{x}^*$ using the training data $\mathcal{D} = (\mathbf{x}, \mathbf{y})$. This can be achieved by learning the weights in a neural network by maximizing the likelihood (MLE) of $\mathcal{D}$ or to maximise the posterior (MAP) by using Bayes' theorem:

$$P(\mathbf{w}|\mathcal{D}) = \frac{P(\mathcal{D}|\mathbf{w})P(\mathbf{w})}{P(\mathcal{D})} \tag{1}$$

so that the regularised estimate becomes:

$$\begin{aligned} \mathbf{w}^{\text{MAP}} &= \arg\max_{\mathbf{w}} \log P(\mathbf{w}|\mathcal{D}) \\ &= \arg\max_{\mathbf{w}} \log P(\mathcal{D}|\mathbf{w}) + \log P(\mathbf{w}) \end{aligned} \tag{2}$$

where the marginal likelihood $P(\mathcal{D})$ is left out, as it does not depend on $\mathbf{w}$. Both ML and MAP estimates of the weights can model the relations between input and output, but do not allow us to quantify the uncertainty of the network parameters. For that, we would have to know the posterior predictive distribution:

$$\begin{aligned} P(\mathbf{y}^*|\mathbf{x}^*) &= \mathbb{E}_{P(\mathbf{w}|\mathcal{D})}\left[P(\mathbf{y}^*|\mathbf{x}^*, \mathbf{w})\right] \\ &= \int P(\mathbf{y}^*|\mathbf{x}^*, \mathbf{w}) P(\mathbf{w}|\mathcal{D}) \mathrm{d}\mathbf{w} \end{aligned}$$

where (1) gives an expression for the posterior $P(\mathbf{w}|\mathcal{D})$. An analytical solution for the posterior is intractable: we would need to integrate over all possible neural network weights. We could try to sample directly from $P(\mathbf{w}|\mathcal{D})$, though this distribution has as many dimensions as neural network parameters. Both methods are unfeasible.

The power of *Bayes by Backprop* comes from using a variational approximation $q(\mathbf{w}|\theta) \approx P(\mathbf{w}|\mathcal{D})$, by using many simple distributions. Each network weight is approximated by a normal distribution with mean μ_i and standard deviation σ_i, which are denoted by $\theta = (\mu, \sigma)$. For this approximation to work, we need to minimize the difference between the distributions by using the Kullback-Leibler-divergence

$$\begin{aligned} \theta_{opt} &= \arg\min_{\theta} \mathbf{KL}\left[\, q(\mathbf{w}|\theta) \,\|\, P(\mathbf{w}|\mathcal{D}) \,\right] \\ &= \arg\min_{\theta} \mathcal{F}(\mathcal{D}, \theta) \end{aligned}$$

To show the dependance on the variational posterior we can write cost function $\mathcal{F}$ as:

$$\mathcal{F}(\mathcal{D},\theta) = \mathbb{E}_{q(\mathbf{w}|\theta)}\left[\log q(\mathbf{w}|\theta) - \log P(\mathcal{D}|\mathbf{w}) - \log P(\mathbf{w})\right] \quad (3)$$

More details on these calculations can be found in the appendix, section 7.1. However, if we now wish to compute the gradients of $\mathcal{F}$ with respect to θ, we need to take the derivative of an expectation. This can be done by a reparameterisation trick [3], where in some cases, the derivative of an expectation can be expressed as the expectation of a derivative. We can then approximate the cost by MC sampling n samples $\mathbf{w}_i$ from $q(\mathbf{w}|\theta)$:

$$\mathcal{F}(\mathcal{D},\theta) \approx \frac{1}{n}\sum_{i=1}^{n} \log q(\mathbf{w}_i|\theta) - \log P(\mathcal{D}|\mathbf{w}_i) - \log P(\mathbf{w}_i) \quad (4)$$

This is explained in more detail in the appendix, section 7.2.

3. REGULARIZATION TECHNIQUES

(SA) When maximizing the posterior (2), or minimizing the cost function (3), regularization is performed by the prior. When we take $P(\mathbf{w}) \sim \mathcal{N}(0, \lambda^{-1})$ we obtain L2-regularization in (3):

$$-\log\left(\frac{1}{\lambda\sqrt{2\pi}}\exp\left(-\frac{\mathbf{w}^2}{2\lambda}\right)\right) = \frac{1}{2}\lambda^{-1}\mathbf{w}^2 + \text{const}$$

A Laplace prior would yield L1 regularization in the same fashion. In [1] a scale mixture of two normal distributions is used as the prior, with differing variances but a mean of zero. A scale mixture prior has been found to yield better results than a normal distribution and that has been implemented in this work.

Another point of interest in BNN's is that in every forward pass the network essentially uses different weights. When using n samples per backpropagation, the network acts as an ensemble of networks. Due to the choice of $q(\mathbf{w})$, the network can also be compared to Gaussian dropout, where noise is added to the nodes in the network. Dropout has been reported to also show L2 regularization, [4].

4. CASE: CLASSIFICATION ON MNIST

(PK) For the classification problem in order to see the convergence and distribution of weights, famous MNIST digits dataset (LeCun and Cortes, 1998), consisting of 60,000 and 10,000 testing images with the 128*128 were considered. It must be mentioned that the prior distribution of BNN used in this work is scale mixture of two Gaussian densities, was defined as the below:

$$P(\mathbf{w}) = \prod_j \pi\mathcal{N}\left(\mathbf{w}_j|0,\sigma_1^2\right) + (1-\pi)\mathcal{N}\left(\mathbf{w}_j|0,\sigma_2^2\right) \quad (5)$$

Here, $\sigma_1^2 = 1$ and $\sigma_2^2 = 0.5$ were used in this case.The model architecture here is the three layers, each layer was kept in the unit size of the 400, other parameters of the model have been shown in the table below.

Epochs	100
Num layesr	3
Hidden units	400
Batch size	128
Learning rate	0.001
Total Learnable Paramteres	477600
Activation	relu

Table 1. Hyper-parameters for the Classification Network

Figure 2 on the right hand side shows density estimates of the weights. The Bayes by Backprop weights are sampled from the variational posterior, where sample size of the 10 was selected from the variational posterior. As we can see the Bayes Back Prop has the wider range of the weights in the model and less centered toward the zero.

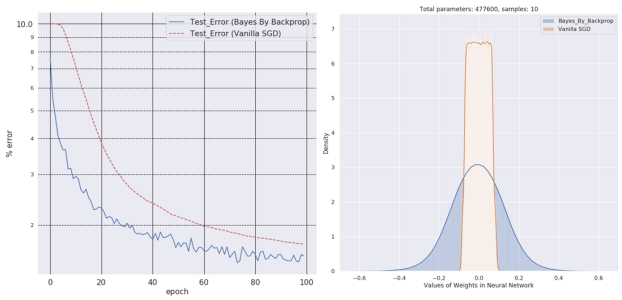

Fig. 2. Convergence rate for BNN versus SGD Vanilla (left). Distribution of weights in BNN model and SGD (right)

On the other hand we know that the main goal of having the uncertainty in the weights and having a slightly more expensive computation is to have the uncertainty in the prediction of the network. To show this concept in more sensible format, the experiment was conducted. In this experiment, one digit (in this case number 3) was considered as the "truth" then, it was rotated in 10 steps. So the first one will have the rotation of the *0* the second one *20* and so until the final rotation will have the rotation of *179*. As can be seen, when the rotation is 0, the model works in in high quality almost with extreme certainty. However, then when the rotation is happened, the model quality decreases mainly since the now the training set has different nature than the prediction where we have the rotation. So, having this experiment shows that when the prediction is in out of the training data (which probably many real examples follow this rule) the model not only make prediction, at the same time provide the probabilities to show the *reliability* of the prediction. It must be mentioned that to do this experiment, 90 samples were selected as the forward model predictions and the resulted probability shown in the Fig 3.

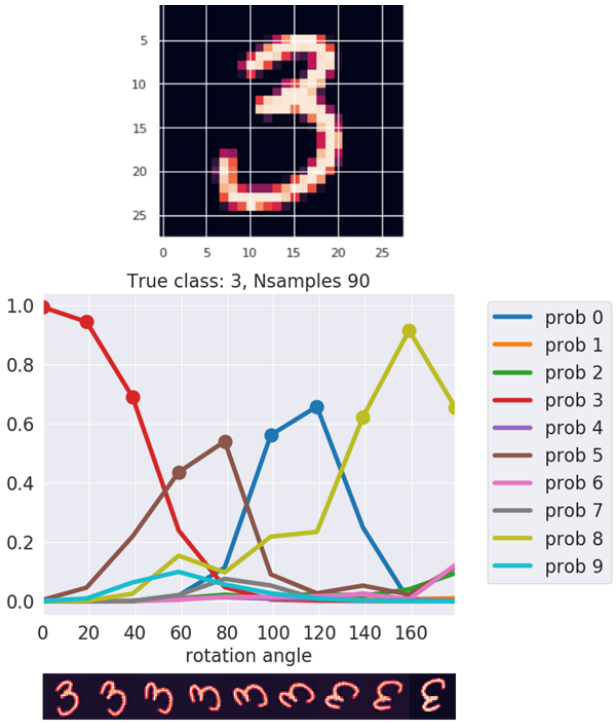

Fig. 3. Visualization of handling uncertainty in the output using the Bayesian neural network

5. CASE: NONLINEAR REGRESSION

5.1. SETUP AND HYPERPARAMETERS

(SA) Here we tried to recreate some of the results on nonlinear regression of the function

$$y = x + 0.3\sin(2\pi x) + 0.3\sin(4\pi x) + \epsilon \qquad (6)$$
$$\epsilon \sim \mathcal{N}(0, 0.02)$$

The training data consists of 500 samples over the interval $[0, 0.5]$. The testing data consists of 100 points over the interval $[-0.2, 1.2]$ to show generalization outside of the training interval. The network consists of a single hidden layer, where training used the working set of hyperparameters in Table 2. A high number of epochs was necessary to reduce the variance between models ad does show slight overfitting in Fig. 5. The number of samples n is used to compute the average cost in (4): while training, the network will receive the same input five times. The chosen prior has a mixture component of 0.3, with variances 5 and 0.5.

Epochs	1000
Hidden nodes	64
Batch size	64
Learning rate	0.01
n	5
Activation	relu

Table 2. Hyperparameters for the two-layer network for the regression case.

5.2. RESULTS

(SA) During testing, 1000 samples were taken per input. The predictions are presented in Fig. 4. We can see how the uncertainty that was built into the model gives us more info on the spread of the predicted mean, which is low in the training interval.

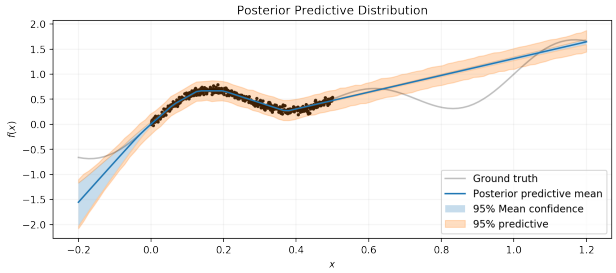

Fig. 4. Regression case, showing the epistemic (blue) and aleatoric (orange) uncertainty.

5.3. COMPARISON

(SA) A comparison was made between Bayesian networks, Dropout networks and networks with L2-regularization, after the discussion in Section 3. The test error is presented in Fig. 5; though these results vary significantly between different training rounds.

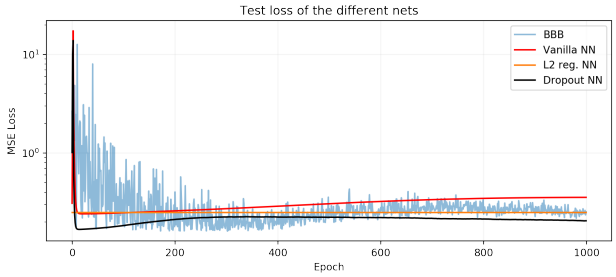

Fig. 5. Comparison of test loss for different nets. In [1], the BNN reaches the same error order as a dropout network, we cannot reproduce these results with great confidence.

As we can see, the BNN shows a highly stochastic performance. This can be reduced by increasing the number of samples when evaluating (4), but which greatly impacts performance. The vanilla NN (no regularization) quickly overfits. The claim that a BNN and a dropout network reach the same order of test error was not reproducible.

5.4. DISCUSSION

(SA) BNN's are very slow learners, as only a high number of epochs has been found to adequately generalize outside of the training interval, and the need for sampling impacts performance significantly. The results presented in Fig. 4 and 5 are highly stochastic and are difficult to reproduce off-the-bat. While it is interesting to see a quantified uncertainty on the predictions of the network, the advantages of BNN's are outweighed by their stochastic and slow character.

6. SUMMARY

(PK) The initial goal of this project was to understand the probabilistic methods in machine learning. The relevant literature and the document regarding the implementing the Bayesian Neural Network was investigated as the method for probabilistic inference from the prior and the training data set. It was found that especially in the classification problem, the BNN could be very helpful in order to quantifying the uncertainty in the model when faced with the data in different style in testing. Providing uncertainty on the output is especially matter of the interest when dealing with high stake decisions where each action has high impact. From practical point of view, we could make a few observation in the model building phase:

- We found that using the scale mixture Gaussian prior could lead the better convergence in the network when compared with the simple Gaussian prior in BNN.
- When it comes to BNN, it needs more computational power compared to the Vanilla neural network, yet from the convergence perspective BNN as well could learn as fast as the conventional neural network (In terms of test error versus epoch)
- At the end when we make sample from the weights, especially in the classification case, the weights are more diverse than classical neural network also are more decentralized from the zero mean.
- Generally, we can see that BNN provides a new framework for regularization while capturing uncertainty in the model, however subjectivity in the prior selection and expensive computational power are major two drawbacks of the BNN.
- Further continue this research, doing more epoch and evaluate the rates, using another data-sets rather than data in this study and also implementing the CNN in the framework of BNN are among the suggestions for further work.

7. APPENDIX

7.1. KL DIVERGENCE AND THE COST FUNCTION

The KL divergence between the variational distribution $q(\mathbf{w}|\theta)$ and the true posterior $P(\mathbf{w}|\mathcal{D})$ is defined as

$$\begin{aligned}\mathbf{KL}\left[\, q(\mathbf{w}|\theta) \parallel P(\mathbf{w}|\mathcal{D}) \,\right] &= \int q(\mathbf{w}|\theta) \log \frac{q(\mathbf{w}|\theta)}{P(\mathbf{w}|\mathcal{D})} d\mathbf{w} \\ &= \mathbb{E}_{q(\mathbf{w}|\theta)} \log \frac{q(\mathbf{w}|\theta)}{P(\mathbf{w}|\mathcal{D})}\end{aligned}$$

Applying Bayes' rule to $P(\mathbf{w}|\mathcal{D})$ we obtain:

$$\begin{aligned}&\mathbb{E}_{q(\mathbf{w}|\theta)} \left[\log \frac{q(\mathbf{w}|\theta)}{P(\mathcal{D}|\mathbf{w})P(\mathbf{w})} P(\mathcal{D}) \right] \\ &= \mathbb{E}_{q(\mathbf{w}|\theta)} \left[\log q(\mathbf{w}|\theta) - \log P(\mathcal{D}|\mathbf{w}) - \log P(\mathbf{w}) \right] + \log P(\mathcal{D}) \\ &= \mathbf{KL}\left[q(\mathbf{w}|\theta) \| q(\mathbf{w}) \right] - \mathbb{E}_{q(\mathbf{w}|\theta)} \left[\log P(\mathcal{D}|\mathbf{w}) \right] + \log P(\mathcal{D})\end{aligned}$$

where $\log P(\mathcal{D})$ does not depend on $\mathbf{w}$ and can be taken out of the expectation. We take the first two terms as the variational free energy $\mathcal{F}(\mathcal{D}, \theta)$, which is our chosen cost function:

$$\mathbf{KL}\left[q(\mathbf{w}|\theta) \| P(\mathbf{w}|\mathcal{D}) \right] = \mathcal{F}(\mathcal{D}, \theta) + \log P(\mathcal{D})$$

The negative variational free energy is also known as the evidence lower bound on $\log P(\mathcal{D})$ because the Kullback-Leibler divergence is always non-negative, and because the minimum of $\mathcal{F}(\mathcal{D}, \theta)$ with respect to θ doe not depend on $\log P(\mathcal{D})$.

7.2. THE REPARAMETERISATION TRICK

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 $\mathbf{w}$:*

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

Proof.

$$\begin{aligned}\frac{\partial}{\partial\theta}\mathbb{E}_{q(\mathbf{w}|\theta)}[f(\mathbf{w}, \theta)] &= \frac{\partial}{\partial\theta} \int f(\mathbf{w}, \theta) q(\epsilon) \mathrm{d}\epsilon \\ &= \int \frac{\partial}{\partial\theta} f(\mathbf{w}, \theta) q(\epsilon) \mathrm{d}\epsilon \\ &= \mathbb{E}_{q(\epsilon)}\left[\frac{\partial}{\partial\theta} f(\mathbf{w}, \theta)\right] \qquad (7)\end{aligned}$$

since f and $\frac{\partial f(\mathbf{w}, \theta)}{\partial \mathbf{w}}$ are continuous in $\mathbf{w}$ and θ in an open neighborhood of $[\mathbf{w}] \times [0, +\infty]$, according to Leibniz' rule. Additionally, (7) is equal to

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

8. REFERENCES

[1] C. Blundell, J. Cornebise, K. Kavukcuoglu & D. Wierstra. *Weight Uncertainty in Neural Networks*. Google DeepMind. 21st May 2015. arXiv:1505.05424v2 [stat.ML].

[2] R. M. Neal and G. E. Hinton. *A view of the EM algorithm that justifies incremental, sparse, and other variants*. Learning in graphical models, pages 355 - 368. Springer, 1998.

[3] M. Opper and C. Archambeau. *The variational Gaussian approximation revisited*. Neural computation, 21(3):786–792, 2009.

[4] P. Baldi, P. Sadowski. *Understanding Dropout*. University of California. NIPS 2013.

The project code can be found at https://github.com/Peymankor/BNN-Project