Weight Uncertainty in Neural Networks
Tennison Liu, Chelsea Murray, Eli Persky

### Introduction

Traditional neural networks use point estimates of the weights while Bayesian Neural Networks (BNNs) use posterior distribution over weights.

- **Poor calibration**
- **No uncertainty estimates**
- **Poor generalisation**

Exact Bayesian inference over a neural network is intractable.

- **Inference:** use variational approximation to the posterior.
- **Prediction:** ensemble of networks by repeatedly sampling weights.

### Approximate Inference

Bayes-by-Backprop (BBB) [1] objective function:

$$ F(D, \theta) = \mathbb{KL}[q(\theta)||P(\theta)] - \mathbb{E}_q[\log P(D|\theta)] $$

$$ \approx \sum_{i=1}^{n} \log q(w^{(i)}|\theta) - \log P(w^{(i)}) - \log P(D|w^{(i)}) $$

We explore single Gaussian and Mixture of Gaussians priors.

- Variational posterior q(\theta|\theta) is Gaussian, sampled using reparameterisation:
  $$ w = \mu + \sigma \odot \epsilon \quad \epsilon \sim \mathcal{N}(0, I) $$

  - Or sample activations \( \tilde{z} \) conditioned on inputs \( a \) and weights \( w \) using local reparameterisation trick (LR Trick) [2]:
    $$ b_{m,j} = \gamma_{m,j} + \sqrt{\delta_{m,j}}z_{m,j} \quad \zeta \sim \mathcal{N}(0, 1) $$
    $$ \gamma_{m,j} = \sum_{i} a_{m,i}u_{i,j} \quad \delta_{m,j} = \sum_{i} a_{m,i}^2 \sigma_{i,j}^2 $$

- Computational efficient, decreases variance of gradient estimates leading to faster convergence.

Monte Carlo (MC) Dropout [3] Bayesian interpretation of dropout i.e. draw samples at test time by repeatedly masking random weights.

### Functional Variational Inference (FVI) [4] optimisation against distributions over functions with a Gaussian Process Prior:

$$ F(D, \theta) = \mathbb{KL}[q(f)||P(f)] - \mathbb{E}_q[\log P(D|f)] $$

with

- $P(f) \sim \mathcal{GP}(0, K_{L} + K_{raw})$
- $q(f|\theta)$ NN with Gaussian weights.

### Classification

<table>
<thead>
<tr>
<th>SGD</th>
<th>MC Dropout</th>
<th>BBB Gaussian</th>
<th>BBB Mixt</th>
<th>LR Trick</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error (%)</td>
<td>1.90</td>
<td>1.26</td>
<td>1.58</td>
<td>1.22</td>
</tr>
</tbody>
</table>

Table 1. Results for MNIST classification. Models trained for 300 epochs using 10 samples in training (for BNNs).

### Reinforcement Learning

- **UCI Mushroom Bandit**: agent selects action (eat vs. not eat) with highest reward.
- Using Thompson Sampling, BBB naturally balances exploration vs. exploitation.

### Regression

- **BBB** provides improved performance over other methods while providing sensible uncertainty estimates and better calibration.
- Training a BNN can be viewed as training an infinite ensemble on neural networks while only doubling the number of parameters.
- LR and FVI provide improvements in certain situations.

### References


---

**Figure 1.** Histograms of trained weights for SGD, MC Dropout and sampled weights from BBB.

**Figure 2.** Calibration curves. Bayesian models avoid overconfidence.

**Figure 3.** Regression on synthetic data. Blue line is the median prediction; blue and orange regions show interquartile range and range.

**Figure 4.** Cumulative decisions. BBB converges to optimal decisions.

**Figure 5.** Histograms of trained weights for SGD, MC Dropout and sampled weights from BBB.

**Figure 6.** Regression with data clusters. FVI handles uncertainty between clusters.