Weight Uncertainty in Neural Networks

Yufan Wang (yw575), Max Bronckers (mojb2), Alan Clark (ajc348)

Motivation

- Regular feed forward neural networks use **point estimates** of weights

Common Issues: Prone to Overfitting | Overly Confident | Non-Robust

Classic Regularisation Methods: Early Stopping | Weight Decay | Dropout

Bayes by Backprop (BBB)

- Instead of point estimates, **learn probability distribution on the weights**

Regularisation | Model Averaging | Exploration

- Exact Bayesian inference is intractable, instead use variational approximation to the posterior
- Find parameters θ of a distribution on the weights $q(\mathbf{w}|\theta)$

$$\theta^* = \arg\min_\theta \mathcal{F}(\mathcal{D},\theta) = \arg\min_\theta \overbrace{\mathrm{KL}[q(\mathbf{w}|\theta)||P(\mathbf{w})]}^{\text{Complexity Cost}} \overbrace{-\mathbb{E}_{q(\mathbf{w}|\theta)}[\log P(\mathcal{D}|\mathbf{w})]}^{\text{Likelihood Cost}}$$

$$= \arg\min_\theta \int q(\mathbf{w}|\theta)[log\, q(\mathbf{w}|\theta) - logP(\mathbf{w})P(\mathcal{D}|\mathbf{w})]dw$$

$$= \arg\min_\theta \mathbb{E}_{\mathbf{q}(\mathbf{w}|\theta)}[f(\mathbf{w},\theta)]$$

- Generalisation of the Gaussian reparameterisation trick

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

- Approximation of the exact cost using **Monte Carlo sampling**

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

- Inference performed using an **uncountably infinite neural networks**

$$P(\hat{\mathbf{y}}|\hat{\mathbf{x}}) = \mathbb{E}_{P(\mathbf{w}|\mathcal{D})}[P(\hat{\mathbf{y}}|\hat{\mathbf{x}},\mathbf{w})]$$

Reparameterisation Tricks

$$w_{i,j} = \mu_{i,j} + \sigma_{i,j}\epsilon_{i,j}$$

$$\epsilon_{i,j} \sim \mathcal{N}(0,1)$$

Blundell et al. [1]

$$b_{m,j} = \gamma_{m,j} + \sqrt{\delta_{i,j}}\zeta_{m,j}$$

$$\gamma_{m,j} = \sum_{i=1} a_{m,i}\mu_{i,j} \qquad \delta_{m,j} = \sum_{i=1} a_{m,i}^2\sigma_{i,j}^2 \qquad \zeta_{m,j} \sim \mathcal{N}(0,1)$$

Kingma et al. [2]

Regression

DNN | BBB

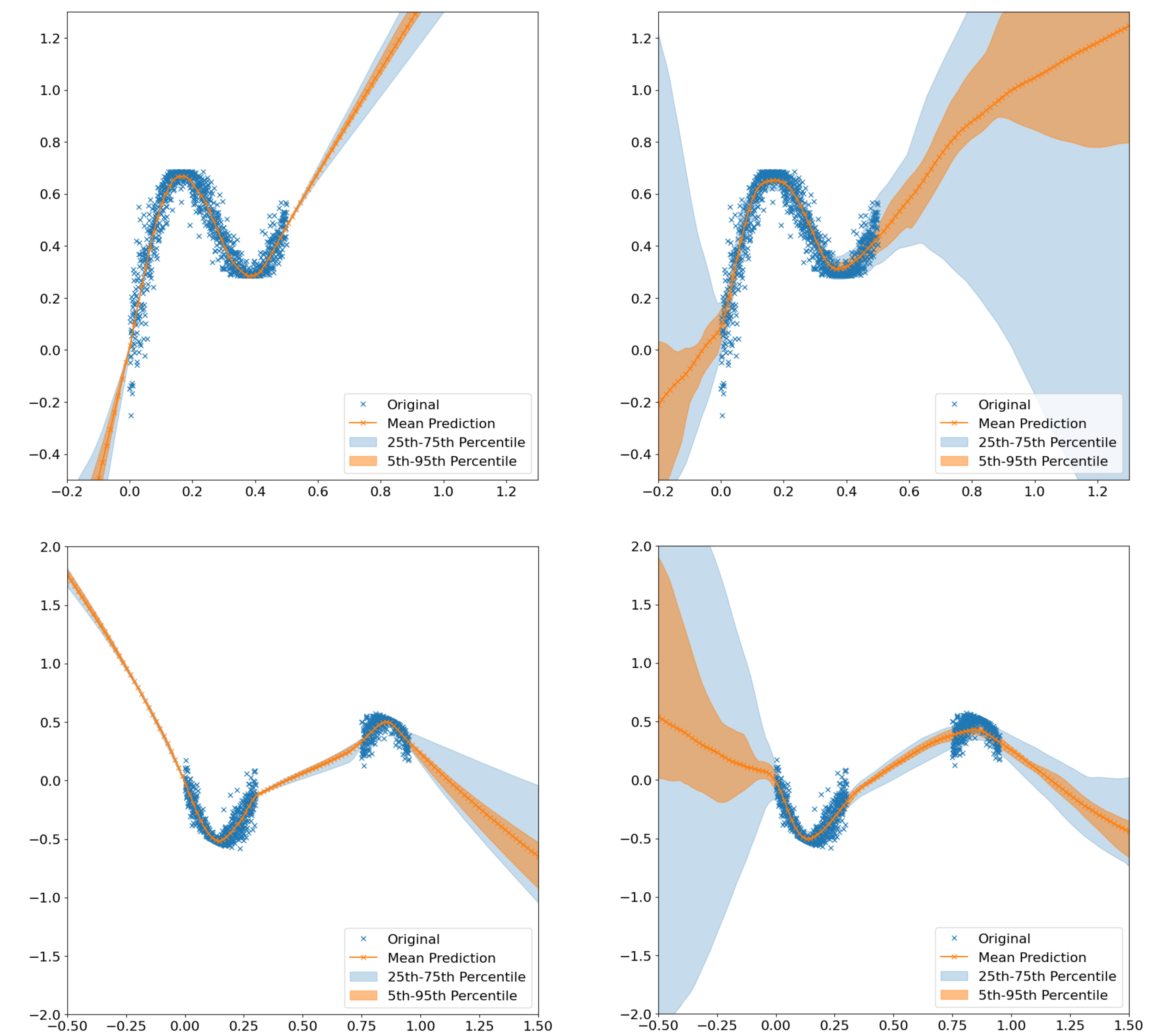

Figure 1. Regression against synthetic data using a regular DNN and BBB

- BBB exhibits significant uncertainty outside observed domain
- DNN exhibits characteristic overconfidence

MNIST

Table 1. Classification errors

Hidden Units	DNN	DNN (Dropout)	BNN
400	2.04 %	**1.33 %**	1.44 %
800	1.68 %	**1.41 %**	1.50 %
1200	1.66 %	1.52 %	**1.36 %**

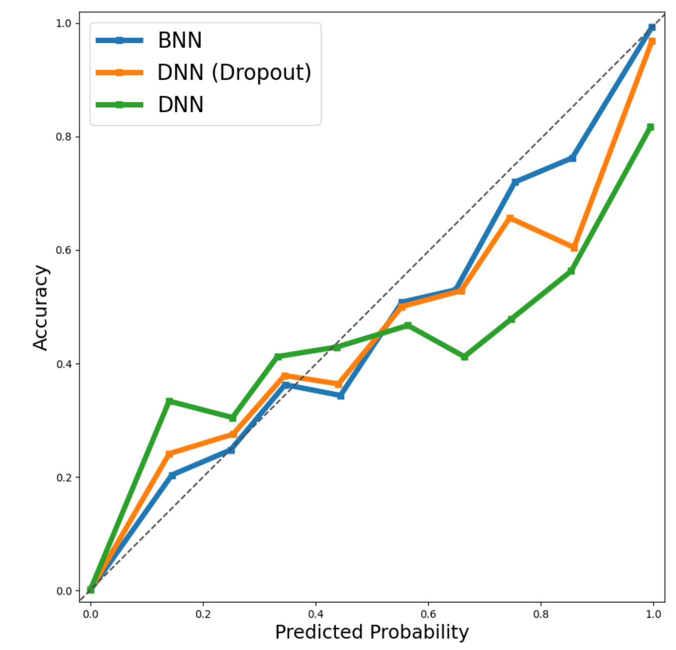

Figure 2. Reliability diagram

- BBB is **more calibrated**
- Lower degree of confidence in its predictions

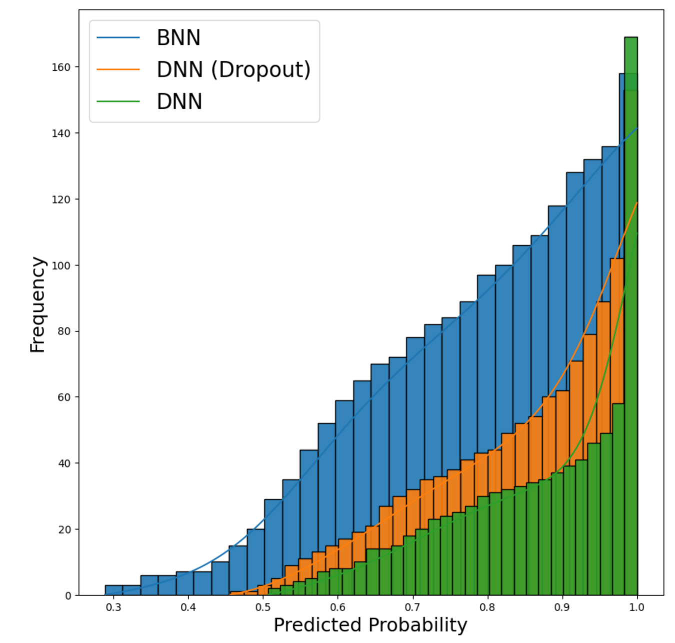

Figure 3. Cumulative distribution of confidence scores for incorrect predictions

- DNN exhibits overconfidence in its incorrect predictions

Contextual Bandits

- Dataset: UCI Mushroom Bandit
- Thompson sampling: picking an action trades-off between exploitation and exploration
- BNN expected to exhibit systematic exploration

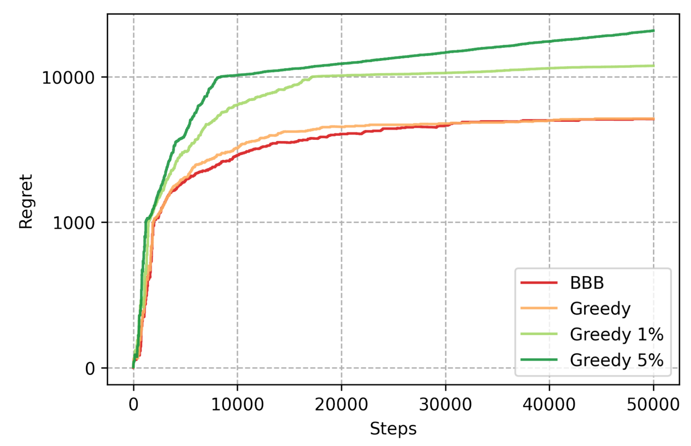

Figure 4. Comparison of cumulative regret for various agents on the mushroom bandit task (step size = 5000, gamma = 0.5, lr = 1e-4)

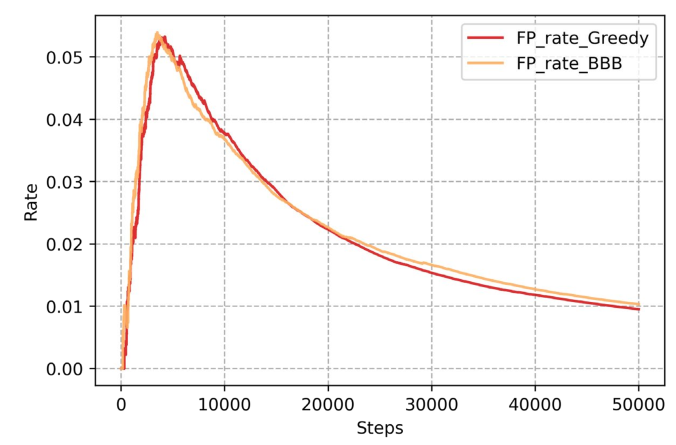

Figure 5. Comparison of false positive rate for greedy and BBB agents

Observations

- Significantly slower training times than SGD; LRT helps
- Do not observe the original paper's weight distributions

Next Steps

Weight Pruning | Laplacian Prior | Adversarial Examples

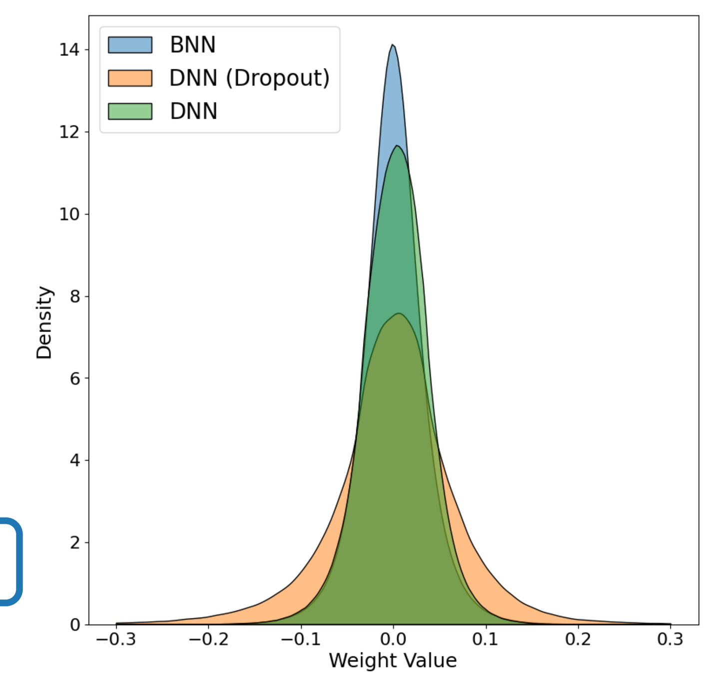

Figure 6. Weight distributions

References

[1] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra, "Weight Uncertainty in Neural Networks," *32nd Int. Conf. Mach. Learn. ICML 2015*, vol. 2, pp. 1613–1622, May 2015

[2] Diederik P. Kingma, Tim Salimans, and Max Welling, "Variational Dropout and the Local Reparameterization Trick," *Adv. Neural Inf. Process. Syst.*, vol. 2015-January, pp. 2575–2583, Jun. 2015

MPhil in Machine Learning and Machine Intelligence

March 2022