Background

Point estimates for neural networks are not enough:
• No way to quantify uncertainty in predictions - results in overconfident predictions.
• Not robust - can be effectively fooled by adversarial examples.

Exact Bayesian inference completely intractable over weights:
• Functional form doesn’t allow for analytic integration.
• Huge number of weights make numerical methods intractable too.

Solution:
• Propose a fast, backpropagation-style, algorithm for learning an approximate posterior distribution over the weights.

Bayes by Backprop

• Variational Bayesian paradigm replaces integration problem with optimisation task - leverage gradient methods and auto-diff.
• Make use of Monte Carlo approximations for training and predictions.

Approximate $P(\omega|\mathcal{D})$ by minimizing KL divergence:
$$\theta^* = \text{argmin}_\theta K L[q(\omega|\theta)||P(\omega|\mathcal{D})]$$

Equivalently maximise the Evidence Lower BOund (ELBO):
$$\mathcal{F}(\theta) = \mathbb{E}_{q(\omega|\theta)}[\log P(\mathcal{D}|\omega)] - K L[q(\omega|\theta)||P(\omega)]$$

Monte Carlo approximation:
$$\mathcal{F}(\theta) \approx -\frac{1}{n} \sum_{i=1}^{n} \log q(\omega^{(i)}|\theta) - \log P(\mathcal{D}|\omega^{(i)}) - \log P(\omega)$$

Gaussian variational posterior $q(\omega^{(i)}|\theta)$:

Reparameterisation trick:
$$w = \mu + \log(1 + \exp(p)) + \epsilon$$
$$\epsilon \sim \mathcal{N}(0, 1)$$
$$p = (\mu, \rho)$$

Scale mixture prior $P(\omega)$:
$$P(\omega) = \prod_i \pi N(\omega_i|0, \sigma_1^2) + (1-\pi) N(\omega_i|0, \sigma_2^2), \sigma_1 > \sigma_2, \sigma_1 \ll 1$$

• Only double the number of parameters yet trains an infinite ensemble

1-D Regression: Visualising Uncertainty

• Simple regression task using Bayes by Backprop (BBB). We compare to predictions from a regular NN and MC Dropout NN, as well as a Gaussian process.
• Uncertainty estimates are quite conservative.
• Training done on 100 randomly sampled points from function with Gaussian noise:
$$y = x + 0.3 \sin(2\pi(x+0.2)) + 0.3 \sin(4\pi(x+0.2)) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, 0.02)$$

Bayesian Optimisation

• Taking advantage of the uncertainty information in Bayesian neural networks we can perform Bayesian optimisation.
• We maximise a very simple negative quadratic function while sequentially selecting acquisition points.
• We use Thompson sampling to pick a single function and choose the next point of be the value that maximises that function.
• After only six observations we have a pretty good model of the function.

What does the distribution over weights look like?

• BBB produces the weights with the highest variance.
• We calculate the signal-to-noise ratio (SNR) for all of the weights and see how pruning those weights with the lowest ratio affects performance - BBB is much less affected than other methods.

Conclusions

• Bayesian treatment allows for appropriate uncertainty estimation.
• Can be seen as an easy way to train an infinite ensemble of networks with only double the number of parameters.
• The induced predictive uncertainty allows for principled exploration in RL and Bayesian optimisation.
• Future directions include developing more flexible approximate posteriors and extending to different neural net architectures.

References


Table 1. MNIST classification result of SGD, dropout, BBB applied to a feed-forward NN with two 400/1200 unit layers.

<table>
<thead>
<tr>
<th>Model</th>
<th>Error Rate (%) 400 Units</th>
<th>Error Rate (%) 1200 Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vanilla SGD</td>
<td>1.84</td>
<td>1.92</td>
</tr>
<tr>
<td>MC Dropout</td>
<td>1.99</td>
<td>1.85</td>
</tr>
<tr>
<td>BBB by-Backprop</td>
<td>2.01</td>
<td>2.35</td>
</tr>
</tbody>
</table>

Table 2. Classification error on MNIST after weight pruning.

<table>
<thead>
<tr>
<th>Weights Removed (%)</th>
<th>No. of Active Weights</th>
<th>Test Error Rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>478410</td>
<td>2.59</td>
</tr>
<tr>
<td>50</td>
<td>239205</td>
<td>2.52</td>
</tr>
<tr>
<td>75</td>
<td>119603</td>
<td>2.62</td>
</tr>
<tr>
<td>95</td>
<td>23921</td>
<td>2.75</td>
</tr>
<tr>
<td>98</td>
<td>9569</td>
<td>3.34</td>
</tr>
</tbody>
</table>

Figure 1. Left: classical BP, fixed value on weights. Right: BBB, distribution over weights. Image taken from [1].

Figure 2. Regression of noisy data with credible intervals. Black crosses are training samples. Black lines are mean predictions. Pink/purple region is shows confidence. Left-to-right: Standard MLP, MC Dropout, BBB, and an RBF kernel GP. Implementations of BBB and MC Dropout built on code provided in [2].

Figure 3. 1-D regression task using Bayesian by Backprop (BBB). We compare to predictions from a regular NN and MC Dropout NN, as well as a Gaussian process.

Figure 4. Left: Comparison between weight distribution of BBB, SGD, dropout. Right: Signal-to-noise ratio of all weights.

Figure 5. Results of BBB applied to Bayesian optimisation.

Figure 6. Comparison of different approaches for estimating uncertainty in a neural network. Left: Vanilla SGD, MC Dropout, and BBB. Right: Comparison of pruning weights with low SNR results in minimal accuracy impact on BBB but leads to catastrophic failure in other methods including dropout.