Weight Uncertainty in Neural Networks

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, Daan Wierstra

ICML 2015

(with application to bandits?)

Lack of emphasis on performance improvement

Idea

Standard MLE/MAP training obtains point estimates for each weight:

$$\begin{aligned}\hat{\boldsymbol{w}}^{\mathrm{MLE}} &= \arg\max_{\boldsymbol{w}} \log p(D|\,\boldsymbol{w}) \\ \hat{\boldsymbol{w}}^{\mathrm{MAP}} &= \arg\max_{\boldsymbol{w}} \log p(D|\,\boldsymbol{w}) + p(\boldsymbol{w})\end{aligned}$$

Bayesian approach obtains distributions over each weight*:

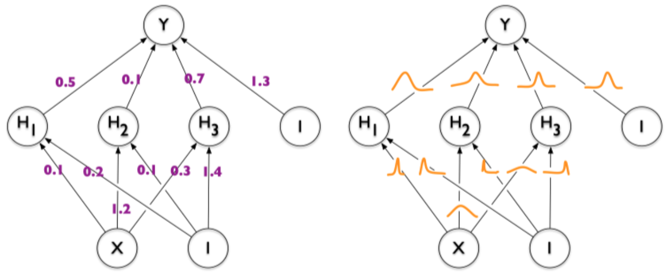


Figure 1. Left: each weight has a fixed value, as provided by classical backpropagation. Right: each weight is assigned a distribution, as provided by Bayes by Backprop.

*with fully factored approximate posterior (known as variational Bayes).

Theory: variational Bayes objective

Let $p(\boldsymbol{w}|\, D)$ be the full posterior. We want to approximate this posterior by $q_{\boldsymbol{\theta}}(\boldsymbol{w})$.

$$\begin{aligned}
\boldsymbol{\theta}^* &= \arg\min_{\boldsymbol{\theta}} \mathrm{KL}(q_{\boldsymbol{\theta}}(\boldsymbol{w})||\, p(\boldsymbol{w}|\, D)) \quad \text{(the variational Bayes objective)} \\
&= \arg\min_{\boldsymbol{\theta}} \mathrm{KL}(q_{\boldsymbol{\theta}}(\boldsymbol{w})||\, p(\boldsymbol{w}, D)) \quad \text{(talk about relationship to MCMC)} \\
&= \arg\min_{\boldsymbol{\theta}} \int q_{\boldsymbol{\theta}}(\boldsymbol{w}) \boxed{\log \frac{q_{\boldsymbol{\theta}}(\boldsymbol{w})}{p(\boldsymbol{w}, D)}} d\boldsymbol{w} \\
&= \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{w}\sim q_{\boldsymbol{\theta}}(\boldsymbol{w})}[f(\boldsymbol{\theta}, \boldsymbol{w})]\, d\boldsymbol{w} \quad (f(\boldsymbol{\theta}, \boldsymbol{w}) = \boxed{\underline{\log q_{\boldsymbol{\theta}}(\boldsymbol{w}) - \log p(\boldsymbol{w})}} \boxed{- \log p(D|\, \boldsymbol{w})})
\end{aligned}$$

Gradient optimization of the objective.

$$\boldsymbol{\theta}' = \boldsymbol{\theta} - \underline{\nabla_{\boldsymbol{\theta}}\{\mathbb{E}_{\boldsymbol{w}\sim q_{\boldsymbol{\theta}}(\boldsymbol{w})}[f(\boldsymbol{\theta}, \boldsymbol{w})]\}}$$

But how should we evaluate this gradient of expectation? As expectation of gradient!

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}}\{\mathbb{E}_{\underline{\boldsymbol{w}\sim q_{\boldsymbol{\theta}}(\boldsymbol{w})}}[\underline{f(\boldsymbol{\theta}, \boldsymbol{w})}]\} &= \nabla_{\boldsymbol{\theta}}\{\mathbb{E}_{\underline{\boldsymbol{\varepsilon}\sim q_0(\boldsymbol{\varepsilon})}}[f(\boldsymbol{\theta}, r(\boldsymbol{\theta}, \boldsymbol{\varepsilon}))]\} \quad \text{(reparametrization trick)} \\
&= \mathbb{E}_{\boldsymbol{\varepsilon}\sim q_0(\boldsymbol{\varepsilon})}[\nabla_{\boldsymbol{\theta}}[f(\boldsymbol{\theta}, r(\boldsymbol{\theta}, \boldsymbol{\varepsilon}))]] \\
&\approx \sum_{i=1}^{N} \nabla_{\boldsymbol{\theta}}[f(\boldsymbol{\theta}, r(\boldsymbol{\theta}, \boldsymbol{\varepsilon}_i))] \quad (\boldsymbol{\varepsilon}_i \sim q_0(\boldsymbol{\varepsilon}); \text{talk about autodiff})
\end{aligned}$$

where r is a *deterministic* map such that if $\underline{\boldsymbol{\varepsilon} \sim q_0(\boldsymbol{\varepsilon})}$ (no $\boldsymbol{\theta}$ in it!) then $\underline{r(\boldsymbol{\theta}, \boldsymbol{\varepsilon})} \sim \underline{q_{\boldsymbol{\theta}}(\boldsymbol{w})}$.

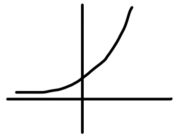
Algorithm 1: bayes-by-backprop

Scale mixture prior.
$$P(\mathbf{w}) = \prod_j \pi \mathcal{N}(\mathbf{w}_j|0, \sigma_1^2) + (1-\pi)\mathcal{N}(\mathbf{w}_j|0, \sigma_2^2), \quad (7)$$
with $\sigma_1 > \sigma_2$

Choice of variational family. Diagonal Gaussian (parameter count only increases by a factor of two), but many others should work. (talk about limitations)

Reparametrization function. All vectors have dim being $|\boldsymbol{w}|$ of the neural network.

$$\boldsymbol{w} = r(\boldsymbol{\theta} = (\boldsymbol{\mu}, \boldsymbol{\rho}), \boldsymbol{\varepsilon}) = \underline{\boldsymbol{\mu}} + \underline{\mathrm{softplus}(\boldsymbol{\rho})} \circ \underline{\boldsymbol{\varepsilon}} \qquad \boxed{\boldsymbol{\varepsilon} \sim N(0, \boldsymbol{I})}$$



Algorithm. Recall from last step that the <u>gradient of the training objective</u> is

$$\underline{\nabla_{\boldsymbol{\theta}}\{\mathbb{E}_{\boldsymbol{w}\sim q_{\boldsymbol{\theta}}(\boldsymbol{w})}[f(\boldsymbol{\theta}, \boldsymbol{w})]\}} = \underline{\mathbb{E}_{\boldsymbol{\varepsilon}\sim q_0(\boldsymbol{\varepsilon})}[\nabla_{\boldsymbol{\theta}}[f(\boldsymbol{\theta}, r(\boldsymbol{\theta}, \boldsymbol{\varepsilon}))]]}$$

and $\frac{1}{N}\sum_{i=1}^{N} \nabla_{\boldsymbol{\theta}}[\underline{f(\boldsymbol{\theta}, r(\boldsymbol{\theta}, \boldsymbol{\varepsilon}_i))}]$ $\quad (\boldsymbol{\varepsilon}_i \sim q_0(\boldsymbol{\varepsilon}))$ is an unbiased estimator of the gradient.

1. Sample $\underline{\boldsymbol{\varepsilon} \sim N(0, \boldsymbol{I})}$. (more can be sampled to reduce variance; for brievity use 1 sample)
2. $\underline{\boldsymbol{w} = \boldsymbol{\mu} + \mathrm{softplus}(\boldsymbol{\rho}) \circ \boldsymbol{\varepsilon}}$
3. Compute $\underline{L = f(\boldsymbol{\theta} = (\boldsymbol{\mu}, \boldsymbol{\rho}), \boldsymbol{w})}$; note that in autodiff software, $\boldsymbol{w}$ can be differentiated thru
4. Call <u>L.backward()</u> to compute gradient wrt $(\boldsymbol{\mu}, \boldsymbol{\rho})$.
5. Call <u>optimizer.step()</u> to apply negated gradient to parameters.

<u>Minibatching.</u>

Other results omitted:

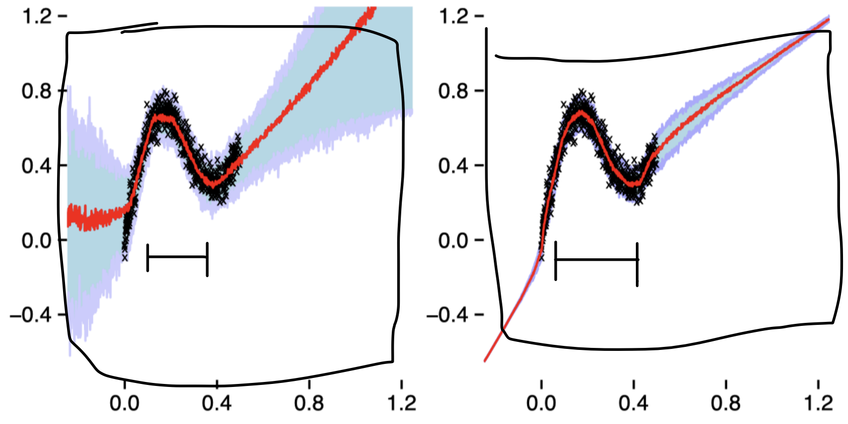


Figure 5. Regression of noisy data with interquatile ranges. Black crosses are training samples. Red lines are median predictions. Blue/purple region is interquartile range. Left: Bayes by Backprop neural network, Right: standard neural network.

Blue/purple region: sampling from the posterior predictive

Algorithm 2: bayes-by-backprop for contextual bandits

2-arm unconditional bandits

Repeat:

- Sample two means to compare
- Pick the action with the higher mean
- Receive reward
- Update the parameters (of the normal distribution)

2-arm conditional bandits with neural network approximators

Repeat:

- Receive context x
- Sample two parameter vectors from approximate posterior
- Compute expected reward under sampled parameters, pick action with higher mean
- Receive reward
- Update the parameters $\underline{\text{(buffer)}}$

Bandit experiment: method

Mushroom task:

- Converted from UCI mushroom dataset
- Each mushroom has a set of features, and is marked edible or inedible
- Reward structure per step:
  - Edible mushroom: reward of 5
  - Inedible mushroom: 1/2 chance reward of -35, 1/2 chance reward of 5
  - Doesn't eat: reward of 0

Cumulative regret: difference between reward achieved by an oracle and the reward reiceved by an agent; the oracle would correctly eat every edible mushroom and not eat every inedbile one.

Neural network:

- Input: mushroom features x (context), action $\overset{a}{x}$ (one-hot encoded)
- Architecture: 2 hidden linears of 100 rectified linear units
- Output: a single scalar; the expected reward of a given action in the given context

Baselines: epsilon-greedy; neural network + random choice

Bandit experiment: results

Implications: maybe able to deal with more complicated features (e.g., images)

Potential issues (?): variational inference is still an approximation

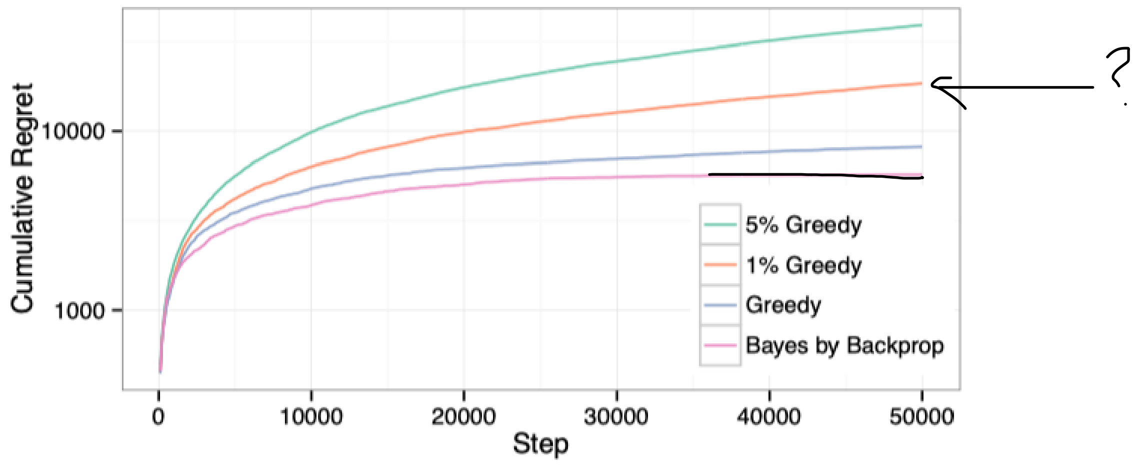


Figure 6. Comparison of cumulative regret of various agents on the mushroom bandit task, averaged over five runs. Lower is better.

The Bayes by Backprop agent explores from the beginning, both eating and ignoring mushrooms and quickly converges on eating and non-eating with an almost perfect rate (hence the almost flat regret).