- Weight Uncertainty in Neural Networks (Bayes by Backprop)
- Charles Blundell
- Daan Wierstra

# Motivation

Overfitting can be interpreted as making overly confident predictions on unseen data. Given a dataset, it is more appropriate to talk about the distribution of the weights, instead of their exact value. This idea is natural within a Bayesian framework.

Some ideas behind this paper

- VAE: uncertainty of latent variable, given obvervation. This work: uncertainty about which neural network is appropriate
- Naturally drive exploration in contextual bandit problems
- Training an infinite number of networks

# Background

Consider the supervised scenario in which a dataset is like $\{x^{(i)}, y^{(i)}\}_i$. We assume that the the distribution $p(y|x;w)$ can be modeled by a neural network, where $w$ is the weight of the neural network.

One approach is **MLE**, which assumes that $w$ is just a fixed parameter:

$$w^{MLE} = \arg\max_w \log p(D|w)$$

For pure Bayesian, we also assume a prior on $w$, namely $p(w)$. The prediction should be made with

$$p(y|x, D) = \mathbb{E}_{p(w|D)}[p(y|x, w)]$$

This is intractable. The problem is with $p(w|D)$ (or more precisely, $p(D)$). So often, we use **MAP**:

$$w^{MAP} = \arg\max_w \log p(D|w)p(w)$$

And this leads to L2 and L1 regularization.

However, MAP is suboptimal. Also, it does not capture the uncertainty that is inherent in the weights.

# Bayes by Backprop

The central problem:

> Given a dataset $D$, what is the distribution $p(w|D)$?

Again, this is intractable. And again, we use variational inference. That is, given $D$, we approximate $p(w|D)$ with a closed form family of distribution $q(w|\theta)$, parametrized by $\theta$.

An optimal estimation of $\theta$ will be

$$\begin{aligned}\theta^* &= \arg\min_\theta KL[q(w|\theta)||p(w|D)] \\ &= \arg\min_\theta KL[q(w|\theta)||p(w)] - \mathbb{E}_{q(w|\theta)}[p(D|w)]\end{aligned}$$

For simplicity, we write

$$F(D,\theta) = KL[q(w|\theta)||p(w)] - \mathbb{E}_{q(w|\theta)}[p(D|w)]$$

And this will be our optimization objective.

# Unbiased Monte Carlo gradients

Just the usual reparametrization trick. Gradient is approximated as

$$\nabla_\theta F(D,\theta) \approx \frac{1}{n}\sum_{i=1}^{m} \nabla_\theta \log q(w^{(i)}|\theta) - \log p(w^{(i)}) - \log p(D|w^{(i)})$$

One should note that under the reparametrization trick, $w^{(i)}$ is a function of $\theta$ and the random noise $\epsilon^{(i)}$ used for sampling. So this graident estimation is unbiased.

## Gaussian Variation Posterior

This work uses the Gaussian variational posteior. **So the number of parameters doubles**. $\sigma$ is parametrized as $\sigma = \log(1+\exp(\rho))$, so here $\theta = (\mu, \rho)$. And

$$w = \mu + \log(1+\exp(\rho)) \circ \epsilon$$

Let $f(w, \theta) = \log q(w|\theta) - \log p(w)p(D|w)$, Each step of the algorithm can be written as

1. Sample $\epsilon \sim \mathcal{N}(0, I)$
2. Compute $w = \mu + \log(1 + \exp(\rho)) \circ \epsilon$
3. Let $\theta = (\mu, \rho)$.
4. Compute $f(w, \theta)$
5. Compute the gradients of $f$ with respect to $\mu$, which is
$$\Delta_\mu = \frac{\partial f(\mathbf{w}, \theta)}{\partial \mathbf{w}} + \frac{\partial f(\mathbf{w}, \theta)}{\partial \mu}$$
6. Compute gradients of $f$ with respect to $\rho$, which is
$$\Delta_\rho = \frac{\partial f(\mathbf{w}, \theta)}{\partial \mathbf{w}} \frac{\epsilon}{1 + \exp(-\rho)} + \frac{\partial f(\mathbf{w}, \theta)}{\partial \rho}$$
7. Update the variational parameters

Note that the $\partial f / \partial w$ term is the gradient found by backpropagation. Other terms can be easily computed, and as a shift and scale to the gradients.

# Scale mixture prior

Just

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

The quantities are hyperparameters and fixed. This works better that pure Gaussian in practice.

## Minibatch Training

For each epoch we perform the gradient update suggested above. The usual way is to use minibatch to perform multiple updates, and these updates will be equivalent to one update. For each minibatch, we will compute gradients with minibatch $i$ using

$$\mathcal{F}_i^{\pi}\left(\mathcal{D}_i, \theta\right) = \pi_i \mathrm{KL}[q(\mathbf{w}|\theta)||P(\mathbf{w})] \\ -\mathbb{E}_{q(\mathbf{w}|\theta)}\left[\log P\left(\mathcal{D}_i|\mathbf{w}\right)\right]$$

Where $\sum_i \pi_i = 1$. Note $\sum_i F_i$ is $F$, so $M$ updates is equivalent to one large update. The author found that setting $\pi_i = \frac{2^{M-i}}{2^M - 1}$ is beneficial.

# Experiments

- Achieve comparable results on MNIST with dropout
- Eliminate $95\%$ of the weights according to the signal-to-noise ratio $(|\mu_i|/\sigma_i)$ does not harm the model's performance, suggesting possiblity of model condensation
- On simple curve regression, the point that the model preseve the uncertainty about unseen data is demonstrated.