Bayes by Backprop Report

Dogancan Kebude

December 2018

1 Introduction

This is a short report on implementation of *Bayes by Backprop* regression introduced in [1]. The report will not contain the details explained in the paper but instead focus on how the distributions are chosen for variational inference, how the parameters are initialized and how the backpropagation is conducted. In this implementation, we are looking for a solution to the toy regression problem explained in Sect. 5.2 in [1] with the model explained in Sect. 3.2 of the same article.

However, the model could not be trained and the probable reasons behind are explained in the discussion section.

2 Algorithmic Flow

The reader should note the algorithmic flow to have a better grasp over the derivations. For every iteration (i):

$$\epsilon \sim N(0, 1) \tag{1}$$

$$\theta = (\mu, \rho) \tag{2}$$

$$\sigma = \log\left(1 + e^{\rho}\right) \tag{3}$$

$$\mathbf{w}^{(i)} = \mu + \sigma \circ \epsilon \tag{4}$$

The above steps lead on to the standard neural network calculations and finally we calculate the loss and backpropagate to update the terms μ and θ.

3 Loss Function

As it is provided in [1], Eqn. 2, the loss function for the network is an approximation of variational free energy, a.k.a. expected lower bound (ELBO).

$$F(D, \theta) = \sum_{i=1}^{n} \log q(\mathbf{w}^{(i)}|\theta) - \log P(\mathbf{w}^{(i)}) - \log P(D|\mathbf{w}^{(i)}) \tag{5}$$

This implies a loss function that can be divided into three distributions. The implemented version chooses all three to be Gaussian. The sections below solely consist of mathematical derivations of functions to implement in code.

3.1 Variational Posterior

$$\log q(\mathbf{w}^{(i)}|\theta) = \log\left(\frac{1}{\sigma^{(i)}\sqrt{2\pi}} e^{-\frac{(\mathbf{w}^{(i)}-\mu^{(i)})^2}{2(\sigma^{(i)})^2}}\right) \tag{6}$$

$$\log q(\mathbf{w}^{(i)}|\theta) = \log 1 - \log \sigma^{(i)} - \frac{1}{2}\log 2\pi - \frac{(\mathbf{w}^{(i)}-\mu^{(i)})^2}{2(\sigma^{(i)})^2} \tag{7}$$

$$\log q(\mathbf{w}^{(i)}|\theta) = \log 1 - \log \sigma^{(i)} - \frac{1}{2}\log 2\pi - \frac{(\mathbf{w}^{(i)}-\mu^{(i)})^2}{2(\sigma^{(i)})^2} \tag{8}$$

At this point we can drop the term with 2π as it will cause the same amount of loss on all samples.

$$\log q(\mathbf{w}^{(i)}|\theta) = -\log \sigma^{(i)} - \frac{(\mathbf{w}^{(i)}-\mu^{(i)})^2}{2(\sigma^{(i)})^2} \tag{9}$$

Utilizing Eqn. 4, we get:

$$\log q(\mathbf{w}^{(i)}|\theta) = -\log \sigma^{(i)} - \frac{1}{2}\epsilon^2 \tag{10}$$

3.2 Prior

Gaussian prior for $\mathbf{w}^{(i)}$ is chosen to be unit variance.

$$\log P(\mathbf{w}^{(i)}) = \log\left(\frac{1}{\sqrt{2\pi}} e^{-\frac{(\mathbf{w}^{(i)})^2}{2}}\right) \tag{11}$$

$$\log P(\mathbf{w}^{(i)}) = \log 1 - \frac{1}{2}\log 2\pi - \frac{(\mathbf{w}^{(i)})^2}{2} \tag{12}$$

We can again drop the 2π term and conclude the prior to be:

$$\log P(\mathbf{w}^{(i)}) = -\frac{1}{2}(\mathbf{w}^{(i)})^2 \tag{13}$$

3.3 Likelihood

The log likelihood is again chosen to be Gaussian with unit variance:

$$\log P(D|\mathbf{w}^{(i)}) = \log\left(\frac{1}{\sqrt{2\pi}} e^{-\frac{(y-\hat{y})^2}{2}}\right) \tag{14}$$

$$\log P(D|\mathbf{w}^{(i)}) = \log 1 - \frac{1}{2}\log 2\pi - \frac{(y-\hat{y})^2}{2} \tag{15}$$

Dropping the 2π term:

$$\log P(D|\mathbf{w}^{(i)}) = -\frac{1}{2}(y - \hat{y})^2 \tag{16}$$

3.4 Final Loss Function

The three subsections above lead our loss function for regression to become:

$$F(D, \theta) = \sum_{i=1}^{n} -\log \sigma^{(i)} - \frac{1}{2}\epsilon^2 + \frac{1}{2}(\mathbf{w}^{(i)})^2 + \frac{1}{2}(y - \hat{y})^2 \tag{17}$$

For any $D^{(i)}$ and $\theta^{(i)}$ pair, the episodic loss function is, hence:

$$F(D^{(i)}, \theta^{(i)}) = \frac{1}{n}[-\log \sigma^{(i)} - \frac{1}{2}\epsilon^2 + \frac{1}{2}(\mathbf{w}^{(i)})^2] + \frac{1}{2}(y - \hat{y})^2 \tag{18}$$

4 Gradients

The gradients should be calculated for μ and ρ parameters for optimization through gradient descent.

4.1 Gradient with respect to μ

Since variational posterior terms in Eqn. 18 do not contain a μ, their partial derivatives are not shown below.

$$\frac{\partial F(D^{(i)}, \theta^{(i)})}{\partial \mu^{(i)}} = \frac{1}{n}\mathbf{w}^{(i)} + \frac{\partial P(D^{(i)}|\mathbf{w}^{(i)})}{\partial \mathbf{w}^{(i)}} \frac{\partial \mathbf{w}^{(i)}}{\partial \mu^{(i)}} \tag{19}$$

Where $\frac{\partial P(D^{(i)}|\mathbf{w}^{(i)})}{\partial \mathbf{w}^{(i)}}$ is the standard backpropagation through the neural network. The network gradients will be shown later.

Also, as can be seen from Eqn. 4:

$$\frac{\partial \mathbf{w}^{(i)}}{\partial \mu^{(i)}} = 1 \tag{20}$$

$$\frac{\partial F(D^{(i)}, \theta^{(i)})}{\partial \mu^{(i)}} = \frac{1}{n}\mathbf{w}^{(i)} + \frac{\partial P(D^{(i)}|\mathbf{w}^{(i)})}{\partial \mathbf{w}^{(i)}} \tag{21}$$

4.2 Gradient with respect to ρ

All terms of Eqn. 18 except ϵ contain a ρ element inside and hence will be handled one by one for a clearer view:

4.2.1 The $\sigma^{(i)}$ Term

$$\frac{\partial[\log \sigma^{(i)}]}{\partial \rho^{(i)}} = \frac{\partial[\log \sigma^{(i)}]}{\partial \sigma^{(i)}} \frac{\partial \sigma^{(i)}}{\partial u^{(i)}} \frac{\partial u^{(i)}}{\partial \rho^{(i)}} \tag{22}$$

where $u^{(i)} = 1 + e^{(\rho)(i))}$

$$\frac{\partial[\log \sigma^{(i)}]}{\partial \rho^{(i)}} = \frac{1}{\sigma^{(i)}} \frac{1}{1 + e^{(\rho^{(i)})}} e^{(\rho^{(i)})} \tag{23}$$

$$\frac{e^{(\rho^{(i)})}}{1 + e^{(\rho^{(i)})}} = \frac{1}{\frac{1+e^{(\rho^{(i)})}}{e^{(\rho^{(i)})}}} = \frac{1}{1 + e^{(-\rho^{(i)})}} \tag{24}$$

$$\frac{\partial[\log \sigma^{(i)}]}{\partial \rho^{(i)}} = \frac{1}{\sigma^{(i)}} \frac{1}{1 + e^{(-\rho^{(i)})}} \tag{25}$$

4.2.2 The $\mathbf{w}^{(i)}$ Term

$$\frac{\partial[\frac{1}{2}(\mathbf{w}^{(i)})^2]}{\partial \rho^{(i)}} = \mathbf{w}^{(i)} \frac{\partial \mathbf{w}^{(i)}}{\partial \sigma^{(i)}}) \frac{\partial \sigma^{(i)}}{\partial \rho^{(i)}} = \mathbf{w}^{(i)} \frac{\epsilon}{1 + e^{(-\rho^{(i)})}} \tag{26}$$

4.2.3 The Likelihood Term

$$\frac{\partial P(D^{(i)}|\mathbf{w}^{(i)})}{\partial \rho^{(i)}} = \frac{\partial P(D^{(i)}|\mathbf{w}^{(i)})}{\partial \mathbf{w}^{(i)}} \frac{\partial \mathbf{w}^{(i)}}{\partial \rho^{(i)}} \tag{27}$$

As it can be seen from Eqn. 26:

$$\frac{\partial \mathbf{w}^{(i)}}{\partial \rho^{(i)}} = \frac{\epsilon}{1 + e^{(-\rho^{(i)})}} \tag{28}$$

$$\frac{\partial P(D^{(i)}|\mathbf{w}^{(i)})}{\partial \rho^{(i)}} = \frac{\partial P(D^{(i)}|\mathbf{w}^{(i)})}{\partial \mathbf{w}^{(i)}} \frac{\epsilon}{1 + e^{(-\rho^{(i)})}} \tag{29}$$

4.2.4 Complete gradient with respect to ρ

$$\frac{\partial F(D^{(i)}, \theta^{(i)})}{\partial \rho^{(i)}} = \frac{1}{n}[(-\frac{1}{\sigma^{(i)}} + \mathbf{w}^{(i)}\epsilon) \frac{1}{1 + e^{(-\rho^{(i)})}}] + \frac{\partial P(D^{(i)}|\mathbf{w}^{(i)})}{\partial \mathbf{w}^{(i)}} \frac{\epsilon}{1 + e^{(-\rho^{(i)})}} \tag{30}$$

4.3 Network Backpropagation

The forward pass of the utilized network utilizes ReLU as the activation function, hence the forward pass of the network can be described as:

$$x_{train} * \mathbf{w}_0 + b_0 = z_0 \tag{31}$$

$$ReLU(z_0) = a_0 \tag{32}$$

$$a_0 * \mathbf{w}_1 + b_1 = z_1 \tag{33}$$

$$ReLU(z_1) = a_1 \tag{34}$$

...

$$ReLU(z_{n-1}) = a_{n-1} \tag{35}$$

$$a_{n-1} * \mathbf{w}_n + b_n = \hat{y} \tag{36}$$

Hence, the backward pass looks as follows:

$$\frac{\partial P(D^{(i)}|\mathbf{b}_n^{(i)})}{\partial \mathbf{b}_n^{(i)}} = -(y - \hat{y}) \tag{37}$$

$$\frac{\partial P(D^{(i)}|\mathbf{w}_n^{(i)})}{\partial \mathbf{w}_n^{(i)}} = -(y - \hat{y})a_{n-1} \tag{38}$$

$$\frac{\partial P(D^{(i)}|\mathbf{b}_{n-1}^{(i)})}{\partial \mathbf{b}_{n-1}^{(i)}} = -(y - \hat{y})\mathbf{w}_n^{(i)}[z_{n-1} > 0] \tag{39}$$

$$\frac{\partial P(D^{(i)}|\mathbf{w}_{n-1}^{(i)})}{\partial \mathbf{w}_{n-1}^{(i)}} = -(y - \hat{y})\mathbf{w}_n^{(i)}[z_{n-1} > 0]a_{n-2} \tag{40}$$

...

$$\frac{\partial P(D^{(i)}|\mathbf{b}_0^{(i)})}{\partial \mathbf{b}_0^{(i)}} = -(y - \hat{y})\mathbf{w}_n^{(i)}[z_{n-1} > 0]\mathbf{w}_{n-1}^{(i)}[z_{n-2} > 0]...\mathbf{w}_1^{(i)}[z_0 > 0] \tag{41}$$

$$\frac{\partial P(D^{(i)}|\mathbf{w}_0^{(i)})}{\partial \mathbf{w}_0^{(i)}} = -(y - \hat{y})\mathbf{w}_n^{(i)}[z_{n-1} > 0]\mathbf{w}_{n-1}^{(i)}[z_{n-2} > 0]...\mathbf{w}_1^{(i)}[z_0 > 0]x_{train} \tag{42}$$

Since the μ and ρ parameters are defined for all weights and biases, the gradients should be calculated for all weights and all μ, ρ pairs.

5 Update

Following the algorithmic flow in Sect. 2, we end up with calculating the loss and then gradients to update μ and ρ pairs at each iteration as follows:

$$\mu = \mu - \eta \frac{\partial F(D^{(i)}, \theta^{(i)})}{\partial \mu^{(i)}} \tag{43}$$

$$\rho = \rho - \eta \frac{\partial F(D^{(i)}, \theta^{(i)})}{\partial \rho^{(i)}} \tag{44}$$

Where η is the learning rate.

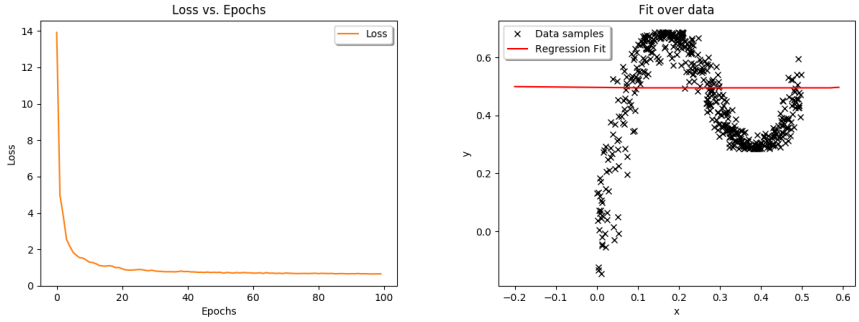
6 Parameter Initialization

The initial μ and ρ parameters are sampled from zero mean Gaussians with 0.01 variance for the weights and from uniform distributions between -0.01 and 0.01 for biases.

7 Discussion

The *Bayes by Backprop* implementation unfortunately ended up non-trainable. The training generally ends up with a linear fit that goes through the middle of the samples. The possible reasons could be:

- Bad implementation of Loss
- Bad implementation of Gradients
- Bad initialization
- Bad hyperparameter tuning



(a) Loss vs. Epochs

(b) Regression fit over samples

Figure 1: The training results for hidden layer sizes 32 and 64, with $1e-3$ learning rate, sample count 500 and 100 epochs.

Going through the possible reasons one by one:

7.1 Bad Implementation of Loss

The choice of distributions for loss might have caused the non-trainable network. Unit variance choice for the prior and the likelihood seems to be poor however due to lack of time the gradients with variances could not be implemented. A better choice for variances might be to utilize the data's variance: As it is explained in Sect. 5.2 of [1], a 0.02 variance noise is applied to the data.

7.2 Bad Implementation of Gradients

The calculated gradients could have been problematic. However, a gradient check through eager execution of Tensorflow for autograd utilization proved this to not be the case: The differences between Tensorflow gradients and the implemented gradients are either exactly zero or on the $< 1e-15$ scale.

7.3 Bad Initialization

As no initialization tips were provided in the original paper, several different initializations were tried. For μ, the initialization should not matter much and the initialization in Sect. 6 seems to work. However, for ρ, the initialization matters and a small negative offset (e.g. around -5) seems to work better. Still, I could not validate this through mathematics and I am unsure if this really is the case.

7.4 Bad Hyperparameter Tuning

Several number of samples, epoch counts, layer sizes and learning rates were tried. However, due to processor limitations, the validation got too slow at large layer sizes (i.e. $> 150,000$ weights). The fastest and best fit seems to be a three layer network with hidden layer sizes of 32 and 64, a learning rate of $1e-3$ with 500 data samples and 100 epochs as it works faster and generates a loss that is ~ 0.2. However, having a loss that small and a regression that is not a good fit for the data, points towards a bad loss implementation as explained in Sect. 7.1.

References

[1] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 1613–1622. JMLR.org, 2015.