Notes on Bayesian Neural Networks

Dimitrios Tanoglidis

January 2022

Some notes on the theory and practice of Bayesian Neural Networks are presented. Basic Bayesian inference is reviewed, together with the methods to approximately sample from the posterior.

1 Neural Networks with Bayesian Inference

Let us start by defining the training dataset, $\mathcal{D} = (X, Y)$, which consists of features $X = \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_N}\}$, and labels $Y = \{\mathbf{y_1}, \mathbf{y_2}, \ldots, \mathbf{y_N}\}$.

Standard neural networks map the inputs (features) to outputs through a series of nodes, arranged in layers, where the output of each node is a weighted sum of all the nodes of the previous layers, that subsequently pass through a non-linear activation function (like ReLU or sigmoid).

More complicated architectures exist, like CNNs that connect each node to only a limited number of nodes of the previous layer, but in all cases the weights (which are learned through the training process) are single values (point estimates). In other words, the weights, W, of traditional NNs are deterministic.

In **Bayesian Neural Networks** (BNNs), the point estimates are replaced by appropriate probability distributions over the weights, that can be subsequently used to estimate uncertainties in these weights and in the predictions.

The objective of training a BNN is to find the posterior distribution of the weights, given the training dataset, $p(w|X, Y)$. For that reason, according to the Bayesian recipe, a prior belief on the distribution of the weights, $p(w)$, as well as the likelihood $p(Y|X, w)$ (that connects the predictions with the learned weigths w), are specified. Then the posterior can be calculated using Bayes' theorem:

$$p(w|X, Y) = \frac{p(Y|X, w)p(w)}{p(Y|X)} \tag{1}$$

where $p(Y|X) = \int P(Y|X, w)p(w)dw$, is the **evidence**.

If the posterior has been computed, the probability distribution for the parameter y^* of a new data point with feature vector x^* is:

$$p(y^*|x^*, X, Y) = \int p(y^*|x^*, w)p(w|X, Y)dw \tag{2}$$

2 Variational Inference and the ELBO

Calculating the above posterior (1) is an intractable problem, because of the evidence factor (which has to be integrated over all values of weights in a multi-dimensional space, which requires exponential time).

Approximate solutions should be developed. A common one in many problems in (astro)physics is to sample the posterior through MCMC methods. Another one is to approximate the exact posterior distribution $p(w, \mathcal{D})$ (from now on I change X, Y to $\mathcal{D}$, for ease of writing the expressions; whenever not clear, the explicit form will be introduced again), with a well behaved, computationally tractable one, $q(w|\theta)$. The goal then of **variational inference** is to find those parameters θ such that the approximate posterior, $q(w|\theta)$, matches the true posterior, $p(w|\mathcal{D})$, as accurately as possible.

A measure of (dis)similarity between of the two distributions is the **Kullback-Leibler** (KL) divergence:

$$\boxed{\mathrm{KL}(q(w|\theta)||p(w|\mathcal{D})) \equiv \int q(w|\theta) \log \frac{q(w|\theta)}{p(w|\mathcal{D})} dw.} \tag{3}$$

Thus the inference problem can be rephrased as an optimization problem:

$$q(w|\hat{\theta}) = \mathrm{argmin}_\theta \mathrm{KL}(q(w|\theta)||p(w|\mathcal{D})) \tag{4}$$

Why is that better? To find out, let's use again Bayes' theorem to rewrite the posterior in the expression for the KL divergence:

$$\mathrm{KL}(q(w|\theta)||p(w|\mathcal{D})) = \int q(w|\theta) \log \frac{q(w|\theta)p(\mathcal{D})}{p(\mathcal{D}|w)p(w)} dw \tag{5}$$

$$= \int q(w|\theta) \log \frac{q(w|\theta)}{p(w)} dw - \int q(w|\theta) \log p(\mathcal{D}|w) dw + \log p(\mathcal{D}) \int q(w|\theta) dw \tag{6}$$

$$= \mathrm{KL}(q(w|\theta)||p(w)) - \mathbb{E}_{q(w|\theta)} \log p(\mathcal{D}|w) + \log p(\mathcal{D}) \tag{7}$$

Since the evidence does not depend on the parameters θ, minimizing the KL divergence is equal to minimizing the so-called **variational free energy**:

$$\boxed{\mathcal{F}(\mathcal{D}, \theta) = \mathrm{KL}(q(w|\theta)||p(w)) - \mathbb{E}_{q(w|\theta)} \log p(\mathcal{D}|w),} \tag{8}$$

so we don't have to calculate the evidence. Thus our loss function is $\mathcal{L} = \mathcal{F}(\mathcal{D}, \theta)$

While it is not really necessary for the definition of the optimization of the problem (I think), usually the above is expressed in terms of the **Evidence Lower Bound (ELBO)**.

Specifically, the ELBO is defined as the negative of the variational free energy, so:

$$\mathrm{ELBO} = \mathbb{E}_{q(w|\theta)} \log p(w|\mathcal{D}) - \mathrm{KL}(q(w|\theta)||p(w)) \tag{9}$$

Thus :

$$\mathrm{KL}(q(w|\theta)||p(w|\mathcal{D})) = -\mathrm{ELBO} + \log p(\mathcal{D}) \Rightarrow \log p(\mathcal{D}) = \mathrm{ELBO} + \mathrm{KL}(q(w|\theta)||p(w|\mathcal{D})). \tag{10}$$

Since $\mathrm{KL}(q(w|\theta)||p(w|\mathcal{D})) \geq 0$ (property of the KL divergence), we see that ELBO is indeed a lower bound of the evidence.

Thus, minimizing the KL divergence between the approximate and true posterior is the same as maximizing the ELBO.

3 How to train your BNN

The loss, eq. (10), can be evaluated by sampling weights from the variational distribution, $q(w|\theta)$. Training a neural network requires minimizing the loss, which is usually performed in an iterative way (gradient descent). To do so, one should take derivatives of the loss function, $\mathcal{L}(\mathcal{D}, \theta)$, However, this is not possible since now hte weights are stochastic MC samples. To overcome this, a few solutions have been proposed.

3.1 Reparametrization Trick and Flipout

In order for the backpropagation to work it has been proposed to use a *reparametrization* trick (RT). Instead of sampling the weights directly from the variational distribution $q(w|\theta)$, we sample some noise, ϵ, from a simple distribution, $p(\epsilon)$. The weights are then related to that noise through a deterministic function, $w = g(\epsilon, \theta)$ that is now differentiable with respect to the parameters of the initial distribution.

In the case of Gaussian variables, the noise can be drawn from the standard Normal distribution, $\epsilon \sim p(\epsilon) = \mathcal{N}(0, 1)$. Then, instead of sampling the weights from the multidimensional Gaussian $w \sim q(w|\theta) = \mathcal{N}(\mu, \sigma)$, we can write:

$$\boxed{w = \mu + \sigma \odot \epsilon} \tag{11}$$

Now, to perform backpropagation, the equations can be modified to take the derivative $\frac{\partial L}{\partial \mu}$, $\frac{\partial L}{\partial \sigma}$ and update the parameters μ, σ. We show the mathematical details of this approach (called Bayes by Backprop) in the following section.

Note that there are some problems with this reparametrization method as presented here.

First of all, by learning the parameters μ, σ, the total number of trainable paramters is doubled, which adds to the computational costs.

An even more important problem is that, because it is computationally prohibitive to sample a unique noise variable ϵ for each example in a mini-batch, in the RT implementation, the sample weights are the same for all the examples within a batch. This causes the gradients between different samples in the same batch to be correlated, thus preventing the reduction of variance during training.

Flipout has been proposed as a solution to this problem ; it solves the correlation problem by randomly multiplying each perturbation/error by ± 1 (pseudo random sign matrix).

So, in that way:

$$\boxed{w = \mu \pm \sigma \odot \epsilon}$$

.

As we have said, the sign $\pm$ is randomly chosen. This can easily vectorized and give us pseudo-independent weight perturbations.

3.2 Bayes by Backprop details

Here we present some more details on how the Bayes by Backprop algorithm works in practice.

Let us start by rewriting the loss function as:

$$\mathcal{L}(\mathcal{D}, \theta) = \int q(w|\theta) \log \frac{q(w|\theta)}{p(w)} d\theta - \int q(w|\theta) \log p(\mathcal{D}|w) d\theta \Rightarrow \tag{12}$$

$$\mathcal{L}(\mathcal{D}, \theta) = \int q(w|\theta) \left[\log q(w|\theta) - \log p(w)p(\mathcal{D}|w)\right] d\theta \Rightarrow \tag{13}$$

$$\boxed{\mathcal{L}(\mathcal{D}, \theta) = \mathbb{E}_{q(w|\theta)} \left[\log q(w|\theta) - \log p(w)p(\mathcal{D}|w)\right]} \tag{14}$$

Now, we are going to prove a proposition that can be subsequently used to derive the backpropagation equations. One can find the proof for this proposition in the original paper (weight uncertainty in neural networks).

Let a random variable ϵ with distribution $q(\epsilon)$. If we can write $w = t(\theta, \epsilon)$, where $t(\theta, \epsilon)$ a deterministic function. Then for a function $f(w, \theta)$:

$$\frac{\partial}{\partial \theta} \mathbb{E}_{q(w|\theta)}[f(w, \theta)] = \frac{\partial}{\partial \theta} \int f(w, \theta) q(w|\theta) dw = \tag{15}$$

$$= \frac{\partial}{\partial \theta} \int f(w, \theta) q(\epsilon) d\epsilon = \tag{16}$$

$$= \int \frac{\partial}{\partial \theta} = f(w, \theta) q(\epsilon) d\epsilon = \mathbb{E}_{q(\epsilon)} \frac{\partial}{\partial \theta} [f(w, \theta)] = \tag{17}$$

$$= \mathbb{E}_{q(\epsilon)} \left[\frac{\partial f(w, \theta)}{\partial w} \frac{\partial w}{\partial \theta} + \frac{\partial f(w, \theta)}{\partial \theta}\right] \tag{18}$$

So, finally:

$$\boxed{\frac{\partial}{\partial \theta} \mathbb{E}_{q(w|\theta)}[f(w, \theta)] = \mathbb{E}_{q(\epsilon)} \left[\frac{\partial f(w, \theta)}{\partial w} \frac{\partial w}{\partial \theta} + \frac{\partial f(w, \theta)}{\partial \theta}\right]} \tag{19}$$

We see now that the loss function can be written as:

$$\mathcal{L}(\mathcal{D}, \theta) = \mathbb{E}_{q(w|\theta)}\left[f(w, \theta)\right] \tag{20}$$

with: $f(w, \theta) = \log q(w|\theta) - \log p(w)p(\mathcal{D}|w)$.

Let the posterior parameters be $\theta = (\mu, \sigma)$. We have also said that $w = \mu \pm \sigma \odot \epsilon$.

From the proposition we have proven above, we have:

$$\frac{\partial}{\partial \mu}\mathcal{L}(\mathcal{D}, \theta) \equiv \nabla_\mu \mathcal{L} = \mathbb{E}_{q(\epsilon)}\left[\frac{\partial f(w, \theta)}{\partial w}\frac{\partial w}{\partial \mu} + \frac{\partial f(w, \theta)}{\partial \mu}\right] \tag{21}$$

and

$$\frac{\partial}{\partial \sigma}\mathcal{L}(\mathcal{D}, \theta) \equiv \nabla_\sigma \mathcal{L} = \mathbb{E}_{q(\epsilon)}\left[\frac{\partial f(w, \theta)}{\partial w}\frac{\partial w}{\partial \sigma} + \frac{\partial f(w, \theta)}{\partial \sigma}\right] \tag{22}$$

Thus, one can write the backpropagation update equation:

$$\mu = \mu - \alpha \nabla_\sigma \mathcal{L} \tag{23}$$

$$\sigma = \sigma - \alpha \nabla_\sigma \mathcal{L} \tag{24}$$

4 Epistemic and Aleatoric uncertainty

Let us start with some definitions.

Epistemic uncertainty: This refers to a type of uncertainty caused by the model. This is modelled by placing a prior distribution ovee a model's weights and then trying to capture how much these weights vary given some data.

Aleatoric uncertainty: Noise inherent in the observations, and can be reduced by more data. It is modelled by placing a distribution over the output of the model.

As we have said, the last layer of the Bayesian Neural Network is a multidimensional Gaussian with mean $\mu \in \mathbb{R}^N$ and a covariance matrix $\Sigma \in \mathbb{R}^{N(N+1)/2}$.

Then, for a given input x^*, T forward passes of the network are computed, obtaining for each one of them a mean vector μ_t and a covariance matrix Σ_t. Then, an estimator of the total covariance matrix can be written as:

$$\widehat{\mathrm{Cov}}(y^*, y^*|x^*) \approx \underbrace{\frac{1}{T}\sum_{t=1}^{T}\Sigma_t}_{\text{Aleatoric}} + \underbrace{\frac{1}{T}\sum_{t=1}^{T}(\mu_t - \bar{\mu})(\mu_t - \bar{\mu})^T}_{\text{Epistemic}}, \tag{25}$$

with $\bar{\mu} = \frac{1}{T}\sum_{t=1}^{T}\mu_t$.

The first term in the above equation captures the aleatoric, while the second term the epistemic uncertainty.

5 Tensorflow Probability and friends