Combining Bayesian inference with Neural Networks

Philip Botros

June 2020

1 Introduction

It might be confusing at first how to reconcile the principles of Bayesian Inference with the framework of neural networks. Especially given the size of modern architectures and the nature of stochastic gradient based optimization which are usually not covered in Bayesian statistics resources.

I would like to show how to integrate Bayesian thinking into modern neural networks and how to create a tractable approximation that can be computed in practice. Furthermore, this technique is out of the box amenable to standard automatic differentiation techniques only requiring minor modifications to the usual training process, called Bayes by Backprop [2].

2 Background

2.1 Maximum likelihood estimation

Usually we estimate the weights of a statistical model by performing maximum likelihood estimation (MLE), in other words, directly maximizing a certain likelihood function with respect to our model parameters:

$$\hat{\theta}_{MLE} = \underset{\theta \in \Theta}{\arg\max}\, \mathcal{L}(\mathcal{D}|\theta)$$

This implicitly assumes that all the information we need to converge to a good solution can be found in the observed data, which might be a limiting assumption.

2.2 Incorporating prior information

A follow up on this completely data-driven approach of parameter estimation is by incorporating prior information $p(\theta)$ on our weights before we have observed any data. This implies that we treat the parameters of a model as a random variable instead, following Bayes rule we get:

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}$$

We have two options now: (1) estimating the most likely parameter setting given the prior and the observed data or (2) go full Bayesian and estimate the complete posterior distribution.

2.2.1 Maximum a posteriori estimation

The first option is called maximum a posteriori estimation (MAP) and is not strictly Bayesian, since we still not obtain a posterior distribution but a point estimate:

$$\hat{\theta}_{MAP} = \arg\max_{\theta \in \Theta} p(\mathcal{D}|\theta)p(\theta)$$

Note that this coincides with the MLE estimator if our prior is completely uniform (constant for all parameter values). The nice property of MAP estimation is that we do not have to compute the marginal likelihood $p(\mathcal{D})$ since we are solving an optimization problem (we can regard $p(\mathcal{D})$ as a constant), which strongly simplifies the problem from a computational point of view. We do, however, lose the ability to marginalize over our weights.

2.2.2 Full Bayesian treatment

The second option is more involved, if we would like to perform full Bayesian inference there is usually no escaping computing the marginal likelihood. This gives us access to the predictive distribution that we are interested in for a new data point $\mathbf{x}^*$:

$$p(y^*|\mathbf{x}^*) = \mathbb{E}_{p(\theta|\mathcal{D})}[p(y^*|\mathbf{x}^*, \theta)] = \int p(y^*|\mathbf{x}^*, \theta)p(\theta|\mathcal{D})d\theta$$

A full Bayesian treatment thus corresponds to making predictions with an infinite ensemble of models, where a given prediction is weighted by the model probability. In contrast, MLE and MAP only make predictions using a single setting of the parameters, using the mode of the likelihood or posterior correspondingly. This does not properly model any uncertainty we might have with respect to this parameter setting but is a good enough approximation for a lot of cases of practical interest. Note that the MAP and posterior are equivalent if the posterior is a Dirac delta function centered at the mode.

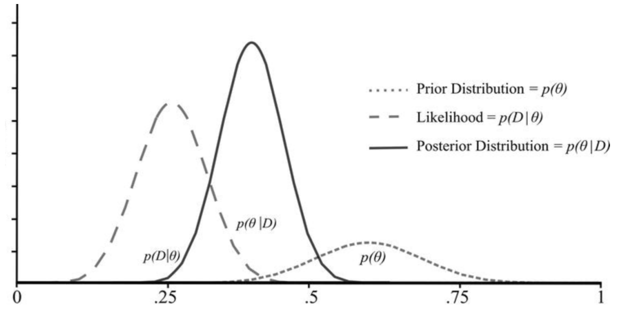


Figure 1: The introduction of the prior effectively weights the likelihood to obtain the posterior, shifting the most probable estimate to the left.

2.3 Computing posteriors in practice

How do we do this in practice? Integrating over our model parameters seems hard, luckily, we have a few options to make this easier. The most straightforward way of computing the full posterior distribution is by deriving an analytical solution. In certain cases, when the prior distribution has the same functional form as the likelihood function (conjugate prior), we can calculate the posterior in closed form.

2.3.1 Bayesian linear regression

Take a vanilla linear regression model with assumed Gaussian noise, the likelihood in this case will be:

$$p(\mathcal{D}|\theta) = \prod_i \mathcal{N}(y_i|, \theta^T \mathbf{x}_i, \sigma^2) = \mathcal{N}(\mathbf{y}|, \mathbf{X}\theta, \sigma^2 \mathbf{I})$$

Maximizing this directly will obviously lead to an estimation of θ that will represent the observed input and label pairs as closely as possible. But what if we have very little data or we assume certain properties in our real world process that might not have been properly represented in the data we have collected?

The conjugate prior of a Gaussian is a Gaussian, so if we add a prior of the same form we can derive an analytical form of the posterior, which will also be Gaussian. Let us put a prior of $p(\theta) = \mathcal{N}(\mu_0, \Sigma_0)$ on our weights, now our posterior will be:

$$\begin{aligned}
p(\theta|\mathcal{D}) &\propto p(\mathcal{D}|\theta)p(\theta) = \mathcal{N}(\mathbf{y}|, \mathbf{X}\theta, \sigma^2 \mathbf{I})\mathcal{N}(\theta|\mu_0, \Sigma_0) \\
&= \mathcal{N}(\theta|\mu_N, \Sigma_N) \\
\mu_N &= \Sigma_N \left(\Sigma_0^{-1}\mu_0 + \sigma^{-2}\mathbf{X}^T \mathbf{y}\right) \\
\Sigma_N &= \left(\Sigma_0^{-1} + \sigma^{-2}\mathbf{X}^T \mathbf{X}\right)^{-1}
\end{aligned}$$

The posterior mean is a weighted combination of the prior mean and likelihood estimate (inversely weighted by the variance), while the new covariance matrix is a sum of the prior and data variance. Furthermore, we can compute the predictive distribution in closed form as well (surprise, it will be Gaussian):

$$
\begin{aligned}
p(y^*|\mathbf{x}^*) &= \int p(y^*|\mathbf{x}^*, \theta)p(\theta|\mathcal{D})d\theta = \int \mathcal{N}(y^*|\theta^T\mathbf{x}^*, \sigma^2)\mathcal{N}(\theta|\mu_N, \sigma_N^2)d\theta \\
&= \mathcal{N}(y^*|\mu_N^T\mathbf{x}^*, \mathbf{x}^*\Sigma_N\mathbf{x}^* + \sigma^2) \\
&= \mathcal{N}(y^*|\mu_N^T\mathbf{x}^*, \underbrace{\mathbf{x}^*\Sigma_N\mathbf{x}^*}_{\text{parameter uncertainty}} + \underbrace{\sigma^2}_{\text{data uncertainty}})
\end{aligned}
$$

See Bishop for a detailed derivation of both [1]. The mean of the predictive distribution is simply the prediction using the parameters of the mean of the posterior, this is thus also equivalent to the MAP prediction in this case, since the mode of a Gaussian is equivalent to its mean. The estimated uncertainty can be decomposed in two parts, uncertainty about the parameters (which will go to zero if $n \to \infty$), and the data uncertainty, the irreducible error we have in the linear regression model, representing effects our model is not able to model.

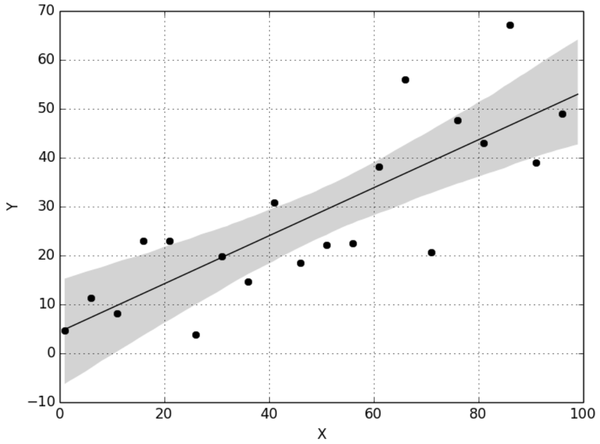


Figure 2: The predictive distribution of a Bayesian Linear Regression. Note that for every prediction we now have a corresponding confidence interval.

2.3.2 Non-analytical solutions

Unfortunately, computing an analytical solution is not feasible for most problems of interest, we have to resort to sampling methods or (variational) approximations to estimate the posterior distribution.

Sampling methods disregard the idea of learning a parametric posterior and instead only want to be able to draw samples from the posterior. Based on samples from this posterior we can estimate certain moments of interest that characterize this posterior. For instance, we can approximate the mean of the predictive distribution by averaging over N samples of the posterior:

$$\mathbb{E}[p(y^*|\mathbf{x}^*)] \approx \frac{1}{N}\sum_{i=1}^{N} p(y^*|\mathbf{x}^*, \theta^{(i)})$$

However, being able to sample from high probability regions in high dimensions is a very hard problem (and slow) and this renders sampling methods impractical for a lot of problems of interest.

Variational methods on the other hand assume that we can learn a simplified known proposal distribution q that will be close enough to the actual posterior. In turn, finding this approximate distribution turns calculating an intractable integral into a tractable optimization problem. A disadvantage of this method is that there is no guarantee that the flexibility of the approximation family with respect to our real posterior will be sufficient for our problem of interest, even in the asymptotic case. In the case of learning an approximate posterior on the weights we can write:

$$\begin{aligned}
\theta^* &= \arg\min_{\theta\in\Theta} \mathrm{KL}\big(q(\mathbf{w}|\theta)||p(\mathbf{w}|\mathcal{D})\big) \\
&= \arg\min_{\theta\in\Theta} \int q(\mathbf{w}|\theta)\frac{q(\mathbf{w}|\theta)}{p(\mathbf{w})p(\mathcal{D}|\mathbf{w})} d\mathbf{w} \\
&= \arg\min_{\theta\in\Theta} \mathrm{KL}\big(q(\mathbf{w}|\theta)||p(\mathbf{w})\big) - \mathbb{E}_{q(\mathbf{w}|\theta)}[\log p(\mathcal{D}|\theta)]
\end{aligned}$$

This resulting function is commonly called the variational free energy or variational lower bound and tractable to optimize, since we removed the dependency on the actual posterior $p(\mathbf{w}|\mathcal{D})$.

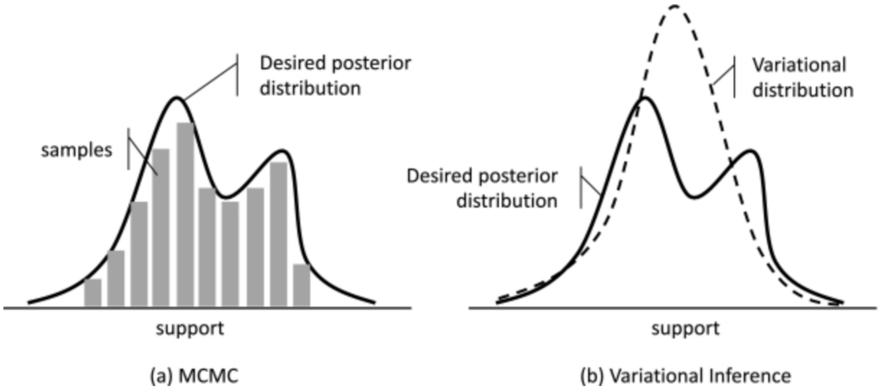

Desired posterior distribution
samples
support
(a) MCMC
Variational distribution
Desired posterior distribution
support
(b) Variational Inference

Figure 3: The difference between the approximation methods illustrated. MCMC tries to approximate the posterior by drawing samples from it. VI approximates it by calculating the parameters that will put the approximate (simplified) distribution closest to the actual posterior.

3 Bayesian Neural Networks

We can interpret a neural network as a probabilistic model that takes as input $\mathbf{x}$ and produces $p(y|\mathbf{x}, \theta)$. Unlike linear regression, where θ was a single layer of parameters, we now have a succession of layers and non-linear transformations.

If we train this in the usual way, i.e. with a cross-entropy loss for classification or a mean squared error loss for regression (any suitable error function E), we recover the MLE estimator (denoting our network by f_θ):

$$\hat{\theta}_{MLE} = \arg\max_{\theta \in \Theta} \mathcal{L}(\mathcal{D}|\theta) = \arg\min_{\theta \in \Theta} \sum_i - \log E(f_\theta(\mathbf{x}_i), y_i)$$

Another option regularly employed is to add L1 or L2 regularization, in which case we perform MAP estimation on the parameters of our network. L1 or L2 regularization are equivalent to placing a Laplacian or Gaussian prior on the weights respectively.

3.1 Fully Bayesian NN

However, as before, we do not want a point estimation of our neural network but a full posterior distribution over our weights. By obtaining this we have access to our predictive distribution which, in theory, will give us proper uncertainty bounds on our predictions. Recall the form of the predictive distribution:

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

Note that we are taking the expectation with respect to the weights of our neural network. Computing this in exact form is intractable for the most neural networks of practical size.

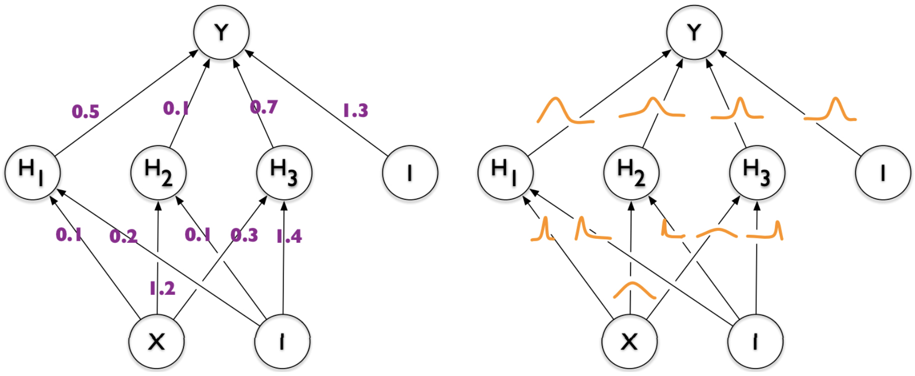


Figure 4: Left: a standard neural network with a scalar value per weight. Right: a Bayesian neural network with a distribution per weight [2]

3.2 Bayes by Backprop

To circumvent the intractability of the posterior Blundell et al. [2] resort to variational inference. Remember that the variational lower bound is given by:

$$\mathcal{F} = \mathrm{KL}\big(q(\mathbf{w}|\theta)||p(\mathbf{w})\big) - \mathbb{E}_{q(\mathbf{w}|\theta)}[\log p(\mathcal{D}|\theta)]$$

How can we compute this quantity in a stochastic gradient based optimization scheme, where especially the term on the right seems challenging, the expectation of the likelihood with respect to the variational distribution? The authors simply choose to approximate it by Monte Carlo sampling:

$$\mathcal{F} \approx \sum_{i=1}^{n} \log q(\mathbf{w}^{(i)}|\theta) - \log p(\mathbf{w}^{(i)}) - \log p(\mathcal{D}|\mathbf{w}^{(i)})$$

Where, under certain conditions with respect to the chosen form of our posterior distribution, we can backpropagate the loss obtained using our sampled representation of our weights through the network in a similar vein as the reparametrization trick of the VAE.

3.3 Gaussian example

To make it more concrete, assume a diagonal Gaussian posterior. We can make a sample of our posterior weights $\mathbf{w}^{(i)}$ differentiable by performing our sampling in the following way:

1. Sample $\epsilon \sim \mathcal{N}(0, I)$
2. Let $\mathbf{w}^{(i)} = \mu + \sigma \odot \epsilon$

Basically, using this formulation, we remove the stochasticity from our model parameters outside of our graph to ϵ. Now we can simply backpropagate our loss with respect to our variational parameters μ and σ given that we regard a sample of ϵ as fixed using $\mathcal{F}$.

We can, in theory, use any prior we want as long as its density is easy to compute and differentiable with respect to our model parameters. To keep it simple however, the authors propose using a standard Gaussian prior or a more complicated scale mixture prior of two Gaussians:

$$p(\mathbf{w}) = \prod_{j} \pi \mathcal{N}(w_j|0, \sigma_1^2) + (1-\pi)\mathcal{N}(w_j|0, \sigma_2^2)$$

To see the benefits of the latter and to illustrate the power of using proper priors note that we know have the freedom to encode our preferences into the weights of our model using σ_1^2 and σ_2^2. For instance, by setting one σ^2 close to zero we can encode our preference to have a sparse network such that we can prune nodes after training for a reduced memory footprint and quicker inference.

For clarity, let us write out what the contribution to the variational free energy will be for one particular example k using a sampled representation i and a standard Gaussian prior and a Gaussian posterior:

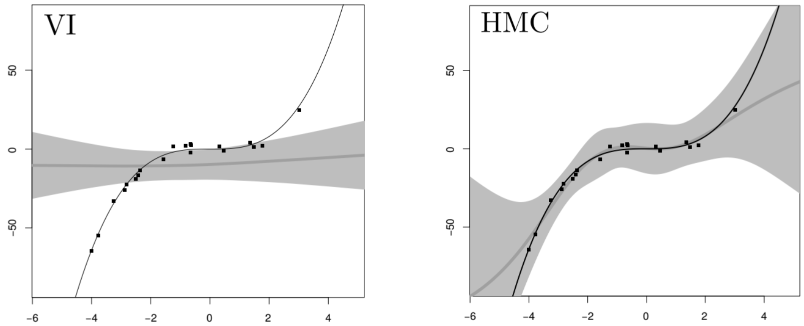
$$\mathcal{F}_k^{(i)} \approx \sum_j \log \mathcal{N}(w_j^{(i)}|\mu_j, \sigma_j^2) - \sum_j \log \mathcal{N}(w_j^{(i)}|0, 1) - E(f(\mathbf{x}_k|\mathbf{w}^{(i)}), y_k)$$

Where the last term E will be a suitable error function, i.e. the cross entropy for classification. This shows that the final objective function is nothing more than: (1) the log probability of the sampled weights under the corresponding parameters of that particular weight, (2) the log probability of the sampled weights under the chosen prior and (3) the classification loss. Note that we can compute the first two terms by summing the individual log probabilities since we assume the weights to be independent.

Also, by having a distribution over our weights our model has to be robust to minor perturbations of the weights, leading to better regularization. Dropout uses the same principle, but adds noise to the activations instead of the weights [7].

4 Limitations

Assuming independence between the weights is a very restrictive assumption, as we expect the real posterior to be highly complicated. This can be seen in Figure 5, where the simplified variational posterior is not expressive enough to model the data generating process correctly. On the other hand, the sampling based method (HMC) is able to not only model the data uncertainty, but also the function itself relatively correct.



(a) Posterior uncertainty under variational inference

(b) Posterior uncertainty under Hamiltonian Monte Carlo

Figure 5: Difference in quality of the estimated posterior. We see that the sampling based method (HMC) outperforms the variational inference one both on accuracy and uncertainty estimates. [4]

More recent work has focused on improving the expressivity of the approximate posterior, [5] use normalizing flows to transform the simple base (Gaussian) distribution into a more expressive one. Another approach uses a low rank approximation of the full covariance to improve on the restrictive diagonal covariance typically used [6]. Although this did improve the quality of the posterior, [3] showed that there are still fundamental issues with variational Bayesian methods, only capturing uncertainty at one mode, thereby failing to produce the multimodal distribution we are looking for.

References

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer-Verlag, Berlin, Heidelberg, 2006.

[2] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

[3] Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*, 2019.

[4] José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pages 1861–1869, 2015.

[5] Christos Louizos and Max Welling. Multiplicative normalizing flows for variational bayesian neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2218–2227. JMLR. org, 2017.

[6] Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. A simple baseline for bayesian uncertainty in deep learning. In *Advances in Neural Information Processing Systems*, pages 13132–13143, 2019.

[7] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.