Where is the uncertainty in Bayesian Neural Networks?

Moritz Kniebel

June 21, 2020

1 Uncertainty in Neural Networks

Predictions of (deep) Neural Networks (NNs) are usually point estimates. This creates problems, as the associated uncertainties are unknown. Especially in safety-critical applications (e.g. self-driving cars) having a posterior distribution can improve decision-making.
Bayesian Neural Networks describe methods that apply uncertainty to Neural Networks. These methods provide a principled framework for several applications, including uncertainty estimation. The uncertainty information is included in the posterior distribution over the weights. With this information every output and weight can be quantified with a bound of confidence. A system (e.g. self-driving car) can benefit from uncertainty estimates in decision-making (e.g. to brake, or not to brake). Therefore it's important to know, how much the system can rely on the model's prediction. Consequently, it is desirable, that a trained model has high confidence in - for the system - critical regions.
Using Bayes theorem the posterior over the weights can be formulated. But since it includes the data-likelihood it is only computable for small networks. For bigger networks, in most cases, the posterior gets analytically intractable, due to size and non-linearity of NNs. Addressing this problem, there are Bayesian methods to approximate the posterior. These approximations are mostly normal distributions, commonly referenced as Gaussians. Two of the most common are the ones used in this thesis:

- Laplace Approximations of weights (MacKay 1992)
  - For more complex networks: A Scalable Laplace Approximation for Neural Networks (Ritter, Botev, and Barber 2018)
- Variational Inference (Graves 2011)

To this date, the location of the uncertainty in NNs hasn't been explored extensively. However, locating the uncertainty within BNNs could have important implications for their training or Bayesian methods within Deep Learning in general. Potential improvements include the training procedure, which could focus on more certain parts of the network first. As well as changing the model's architecture. For example, if a weight, or even a layer doesn't get identified during training, maybe it shouldn't be part of the network at all.

2 Description of the main methods

2.1 Laplace approximation of the weights in Neural Networks

With the Laplace approximation the true posterior of the weights $p(W|\mathcal{D})$ can be approximated with a Gaussian, where W are the weights and $\mathcal{D}$ is the data. The Gaussian is centered at the mode W_{MAP} of the true posterior, where MAP denotes the maximum a posteriori estimate of the weights. The mode W_{MAP} can be found by using nonlinear optimization methods. The curvature information of the Gaussian is given by the Hessian H w.r.t. the Loss, evaluated at W_{MAP}. With this information the true posterior over the weights can be approximated as:

$$p(W|\mathcal{D}) \approx \mathcal{N}((W_{\text{MAP}}), H^{-1}) \,,$$

where $\mathcal{N}$ is the normal distribution.

2.1.1 Kronecker-factored Approximate Curvature (KFAC)

Looking at state of the art Neural Networks, with millions of weight parameters, the method of (MacKay 1992) reaches its limit. The full Hessian w.r.t the loss E gets enormously large, which makes the computation infeasible. That's why (Ritter, Botev, and Barber 2018) approximate the Hessian by a block matrix of independent Kronecker factorized matrices, which are more easily invertible. Doing this they can approximate the posterior distribution over the weights as a Gaussian. The sample Hessian H_λ of each layer λ can be approximated with a Kronecker product:

$$[H_\lambda]_{(a,b)(c,d)} \equiv \frac{\delta^2 E}{\delta W^\lambda_{a,b} \delta W^\lambda_{b,c}} = a^{\lambda-1}_b a^{\lambda-1}_d [\mathcal{H}_\lambda]_{a,c} \,,$$

where a_λ are the activation values of layer λ and $\mathcal{H}$ is the pre-acitvation Hessian of layer λ:

$$[\mathcal{H}_\lambda]_{a,b} = \frac{\delta^2 E}{\delta h^\lambda_a \delta h^\lambda_b} \,,$$

where h_λ is the pre-acitvation of layer λ. Additionally let's denote the covariance of the incoming activations $a_{\lambda-1}$ as:

$$[\mathcal{Q}_\lambda]_{c,d} = (a^{\lambda-1}_c a^{T\lambda-1}_d)$$

This makes it possible to express the sample Hessian of each layer as:

$$H_\lambda = \frac{\delta^2 E}{\delta \text{vec}(W_\lambda) \delta \text{vec}(W_\lambda)} = (a_{\lambda-1} a^T_{\lambda-1}) \otimes \frac{\delta^2 E}{\delta h_\lambda \delta h_\lambda} = \mathcal{Q}_\lambda \otimes \mathcal{H}_\lambda,$$

where $\otimes$ denotes the Kronecker-product. With those kronecker factors the posterior of the weights in layer λ can be approximated as:

$$W_\lambda \sim \mathcal{MN}(W^*_\lambda, \bar{\mathcal{Q}}^{-1}, \bar{\mathcal{H}}^{-1}) \,,$$

where $\mathcal{MN}$ is the matrix Gaussian distribution.

2.2 Variational Inference

Variational Inference tackles the problem of intractability of the posterior over the weights $p(W|\mathcal{D})$, where W are the weights and $\mathcal{D}$ is the data. The method approximates the posterior by using a variational distribution $q(W|\theta)$, parameterized with θ, of the same functional form as the true posterior. The approximating distribution $q(W|\theta)$ should be as close as possible to the true posterior $p(w|\mathcal{D})$, making this an optimization-problem. This is done by minimizing the Kullback-Leibler divergence between $p(W|\mathcal{D})$ and $q(W|\theta)$, while the parameters in $q(W|\theta)$ are estimated.
Using this method, $q(W|\theta)$ learns a good representation of the data.

2.2.1 The evidence lower bound (ELBO) and reparametrization trick

Variational Inference can be optimized using ELBO and the reparametrization trick.

Trying to minimize the difference between $p(W|\mathcal{D})$ and $q(W|\theta)$ by using the KL divergence depends directly on the intractable posterior, making the problem unsolvable. Maximizing the evidence lower bound (ELBO), however, achieves the same as minimizing the KL. This equivalence can be seen through the decomposition of the log evidence $\log p(\mathcal{D})$:

$$\log p(\mathcal{D}) = \mathcal{L}(\mathcal{D}, \theta) + KL(q(W|\theta)||p(W|\mathcal{D})) \,,$$

where $\mathcal{L}(\mathcal{D}, \theta)$ is the evidence lower bound (ELBO).
Since the ELBO and KL add up to the log evidence and the KL is non-negative, KL is minimal when ELBO is maximal. The key is, that ELBO only depends on $q(W|\theta)$, making it possible to solve the problem of minimizing the difference between $p(W|\mathcal{D})$ and $q(W|\theta)$. Hence the new objective is to maximize the ELBO.

The reparametrization trick makes it possible to optimize a models parameters, since we can not back-propagate through a stochastic node. By moving the parameters outside of the distribution function, a gradient can be calculated for the parameters.

2.3 Discussion of other possible Bayesian methods

There are more possible Bayesian methods that yield uncertainty estimates from Neural Networks. In this thesis, only Laplace approximation and Variational Inference will used, since they are best suited for our purposes.

- (Lakshminarayanan, Pritzel, and Blundell 2017) are moving away from the usage of BNNs to integrate uncertainty into NNs. Instead, they are making use of NN ensembles. These ensembles consist of several NNs and their predictions are averaged to a joint output.

- (Gal and Ghahramani 2015): By developing a new theoretical framework, they use dropout NNs to model the uncertainty.

- (Blundell et al. 2015) developed a backprop-compatible algorithm, which regularizes the networks weights, by minimising the variational free energy $\mathcal{F}$.

3 Research Goals/ Contents of the thesis

Main goals:

1. Making use of the above methods [(MacKay 1992) and (Graves 2011)] the uncertainty will be located in a Neural Network with simple architecture.
2. Locating the uncertainty over the weights for one layer of the network, by applying two different methods (Laplace and VI).
3. To make this more understandable a visualization of the above will be elaborated.
4. Move on to a more complex network and try to transfer the findings from the previous steps.

Possible extensions:

- Experiments to measure the influence of the size of the weights to the resulting uncertainty.
- Experiments to observe, how the uncertainty behaves during training.
- Adding a third method to receive uncertainty (e.g. KFAC)

References

Blundell, Charles, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. 2015. *Weight Uncertainty in Neural Networks.* arXiv: 1505.05424 [stat.ML].

Gal, Yarin, and Zoubin Ghahramani. 2015. *Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning.* arXiv: 1506.02142 [stat.ML].

Graves, Alex. 2011. "Practical Variational Inference for Neural Networks." In *Advances in Neural Information Processing Systems 24*, edited by J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, 2348–2356. Curran Associates, Inc. http://papers.nips.cc/paper/4329-practical-variational-inference-for-neural-networks.pdf.

Lakshminarayanan, Balaji, Alexander Pritzel, and Charles Blundell. 2017. "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles." In *Advances in Neural Information Processing Systems 30*, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, 6402–6413. Curran Associates, Inc. http://papers.nips.cc/paper/7219-simple-and-scalable-predictive-uncertainty-estimation-using-deep-ensembles.pdf.

MacKay, David J. C. 1992. "A Practical Bayesian Framework for Backpropagation Networks." *Neural Computation* 4 (3): 448–472. doi:10.1162/neco.1992.4.3.448. eprint: https://doi.org/10.1162/neco.1992.4.3.448. https://doi.org/10.1162/neco.1992.4.3.448.

Ritter, Hippolyt, Aleksandar Botev, and David Barber. 2018. "A Scalable Laplace Approximation for Neural Networks." In *International Conference on Learning Representations.* https://openreview.net/forum?id=Skdvd2xAZ.