Probabilistic Deep Learning: Bayes by Backprop

Felix Laumann
Jul 5, 2018 · 7 min read

Having understood all the fundamentals, we can now proceed and apply them to deep learning. If you cannot follow my steps here, please go back to my previous posts, or comment to this and I'll answer you. We'll have some examples, questions and neat graphics on the way, so it won't be too arid.

> *We will not explain what deep learning or neural networks are, but if you don't feel you have a solid understanding, read Shridhar's post as an introduction or attend Andrew Ng's coursera course for a more details.*

The very base of probabilistic deep learning is understanding a neural network as a conditional model p that is parameterised by the parameters or weights θ of the network and output y when some input x is given. Mathematically, we can write this as follows:

$$p_\theta(y|x)$$

Example:
We feed an image of a red Tesla Model S into the network and it puts out "red car".

Question: Suppose we have ten possible output classes, can you guess which of the probability distributions introduced in Fundamentals 2 is the one our model p will have then?
The answer is given in the bottom of this page.

But how is that output y obtained? How is the aforementioned probability distribution p defined?

For learning this p, Bayes' theorem is utilised and we'll revise it here. As in my previous post shown, the generic formula is:

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

The probability distribution of interest is the distribution p of the parameters θ after the data D is seen. $p(D|\theta)$ is the likelihood of the occurrence of data D given a model with parameters θ, $p(\theta)$ the prior, and $p(D)$ the data distribution. Incorporating a prior *belief* in investigating a posterior state is a central characteristic of Bayesian reasoning. This prior is hence often referred to as *domain knowledge*.

Let us call a neural network with its parameters θ a model for now. Since a model is nothing else than a collection of assumptions, we can see the parameters θ of a model as a hypothesis of an underlying true function that would give us always the correct output to any given input. That is equivalent to an accuracy of 100% and we want to learn the parameters θ to come as close as possible to these 100%. We can write this as:

$$p(\text{hypothesis} \mid \text{data}) = \frac{p(\text{data} \mid \text{hypothesis})\ p(\text{hypothesis})}{p(\text{data})}$$

Our intention is to evaluate a hypothesis on data, i.e. we calculate the posterior probability distribution p(hypothesis | data) by scoring the probability that it gives the data based on the hypothesis, i.e. the likelihood p(data | hypothesis); we multiply it with a prior belief p(hypothesis), and normalise it by the distribution of the data p(data). How well the hypothesis represents the data is the crucial aspect since it can be interpreted as the *goodness of fit* of a model to the data and is therefore the most common objective of any probabilistic model.

Bayesian inference

Built on this, Bayesian inference allows, as a second instance, to predict an output y^* for a new data point x^* by integrating over all parameters θ, as in

$$p(y^\star|x^\star, \mathcal{D}) = \int p(y^\star|x^\star, \theta)\ p(\theta|\mathcal{D}) d\theta.$$

The pity is, we must deal with an integral now, and as you can imagine, computing this integral could take years, because of the very complex form of the probability distribution p . We speak in such cases of *intractability* and must find appropriate approximation techniques.

Bayes by Backprop

Blundell, et al. (2015) introduced *Bayes by Backprop* that will most likely be seen as a break-through in probabilistic deep learning in some time. It symbolises a practical solution how the issue of the aforementioned intractability can be adequately solved.

Our subsequent process is underpinned by suggestions of Graves (2011) and Hinton and Van Camp (1993), based on the speed of computation, to use a variational rather than a Monte Carlo scheme to find the approximate Bayesian posterior distribution. First of all, that means we define a simplified approximate distribution q with its variational parameters θ that shall be as similar as possible to the underlying true distribution p that is intractible:

$$q_\theta(y|\mathcal{D}) \sim p(y|\mathcal{D})$$

$$q_\theta(w|\mathcal{D}) \sim p(w|\mathcal{D})$$

This is realised by minimising the Kullback-Leibler (KL) divergence between *p* and *q*, what can be seen as an optimisation problem:

$$\theta_{opt} = \arg\min_{\theta} \mathrm{KL}\Big[q_\theta(w|\mathcal{D}) \| p(w|\mathcal{D})\Big]$$

A well-explaining graphical intuition for this procedure is given by Graves (2011):

So, let's now look further into the KL-divergence and how it is defined:

$$\mathrm{KL}\Big[q_\theta(w|\mathcal{D}) \| p(w|\mathcal{D})\Big] = \int q_\theta(w|\mathcal{D}) \log \frac{q_\theta(w|\mathcal{D})}{p(w|\mathcal{D})} dw$$

And here we face another pity: we have another integral and this symbolises another intractability. So, what to do now? We approximate again. We already have approximated the underlying true distribution *p* with a variational distribution *q* and we know that we can always sample, i.e. use Monte Carlo methods, from any intractable distribution. So, why do we not sample from the variational distribution *q* when we see tiny pieces of the true distribution *p* what happens while we see data? This

exact step is the quintessence of *Bayes by Backprop:* first, we approximate the underlying true distribution p with an approximate distribution q which shape is represented by parameters θ that can be learnt, and second sample from that q while seeing data.

Hence, we arrive at a tractable objective function:

$$\theta_{opt} = \arg\min_{\theta} \sum_{i=1}^{n} \log q_\theta(w^{(i)}|\mathcal{D}) - \log p(w^{(i)}) - \log p(\mathcal{D}|w^{(i)})$$

$$\text{with sample } w^{(i)} \text{ from } q_\theta(w|\mathcal{D})$$

Local Reparamerisation Trick

Let us once again remember that we want to implement the above procedure in a neural network, and therefore must calculate derivatives of the parameters being learnt, i.e. for us, derivatives of distributions. For doing so, the local reparameterisation trick (Kingma et al., 2015) is deployed which "moves" the parameters to be learnt, namely the mean μ and the standard deviation σ in case of a Gaussian distribution, out of the distribution function for any weight w. We define that ϵ as a sample of a standard Gaussian distribution, multiply it with the standard deviation σ and add the mean μ.

$$\theta = (\mu, \sigma^2)$$

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

$$f(\epsilon) = w = \mu + \sigma \cdot \epsilon$$

Doing so, we have these two parameters of interest incorporated in every weight value and can both calculate the derivative of it and re-write it into a probability distribution.

Our parameters are then updated, i.e. learnt, according to:

$$\frac{\partial f}{\partial} \quad \frac{\partial f}{}$$

$$\Delta\mu = \frac{\partial f}{\partial w} + \frac{\partial f}{\partial \mu}$$

$$\Delta\sigma = \frac{\partial f}{\partial w}\ \frac{\epsilon}{\sigma} + \frac{\partial f}{\partial \sigma}$$

$$\mu \leftarrow \mu - \alpha\Delta\mu$$

$$\sigma \leftarrow \sigma - \alpha\Delta\sigma$$

$$\theta^{opt} = (\mu^{opt}, \sigma^{opt})$$

That was all you need to know to apply Bayes' theorem in any deep neural network. *Bayes by Backprop* was originally only applied to a feed-forward neural network, but was in follow-up work also implemented in a recurrent and a convolutional neural network by Fortunato et al. (2017) and by me and colleagues in Shridhar et al. (2018), respectively.

References

Blundell, C., Cornebise, J., Kavukcuoglu, K., & Wierstra, D. (2015). Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*.

Fortunato, M., Blundell, C., & Vinyals, O. (2017). Bayesian recurrent neural networks. *arXiv preprint arXiv:1704.02798*.

Graves, A. (2011). Practical variational inference for neural networks. In *Advances in neural information processing systems* (pp. 2348–2356).

Hinton, G. E., & Van Camp, D. (1993). Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory* (pp. 5–13). ACM.

Kingma, D. P., Salimans, T., & Welling, M. (2015). Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems* (pp. 2575–2583).

Shridhar, K., Laumann, F., Llopart Maurin, A., Olsen, M., Liwicki, M. (2018). Bayesian Convolutional Neural Networks with Variational Inference. *arXiv preprint arXiv:1806.05978*

Answer: categorical distribution

Special thanks to Kumar Shridhar and Jesper Wohlert for their valuable comments. Follow both of them, they do phenomenal work!