Paper_Batuhan

January 27, 2019

1 Uncertainty

Accounting for uncertainty is crucial for decision-making systems where miscalculation costs us dearly. We wouldn't really like our autonomous car to drive over people due to misclassification or our credit rating system to reject actually credible people. Usually in deep learning models, we obtain point estimates those are overly confident about their predictions. Our main discussion in this post will be how to derive a set of estimations from a neural network through Bayesian Inference and how to improve the decision-making processes accordingly. We attempt to maximize the expected profits of a lender when choosing whom to give credit throughout Lender's Club dataset at Kaggle.

Frank Knight distinguished between two types of uncertainty a long time ago. The first type he named as 'risk' being a 'quantity susceptible of measurement' which in our case corresponds to an empirical uncertainty. The second type is the 'genuine' or aleatoric uncertainty which is far out of the reach of the measurement and something that always exists (Knight, 1921). Deterministic deep learning models are able to give out measurements for epistemic uncertainty, although they are ignorant about their aleatoric uncertainty. They basically do not know what they do not know.

Indeed, it is possible to make sense of aleatoric uncertainty in deep learning models by using Bayesian Inference. But we have to be sceptical during modelling. Under the perspective of the Bayesian paradigm, each observation is an opportunity to criticize our beliefs about a given deep learning model. Using the Bayes' rule below, we can figure out how the degree of belief in a model (i.e. **the posterior function** $P(\omega|X)$) is related to the likelihood of data (i.e. ** the likelihood function $P(X|\omega)$**) , our knowledge about the data (i.e.** the prior $P(\omega)$**) and the evidence (i.e.** the marginal likelihood $P(X)$**):

$$p(\omega|X) = \frac{p(X,\omega)}{P(X)} \implies p(\omega|X) = \frac{p(X|\omega)p(\omega)}{p(X)}$$

Having defined the posterior as above, the prediction on new observations x_{new} is made through model criticism on the **posterior predictive distribution**:

$$p(x_{new}|X) = \int p(x_{new}|\omega)p(\omega|X)d\omega$$

So far, everything seems fine: As oppose to deterministic neural networks which maximize **the likelihood function** $P(X|\omega)$ and obtain only point estimates, we can calculate the **posterior predictive distribution** \$P(x_{new}|X) \$ and get a distribution of estimations for each observation. Except that it is nearly impossible in practice. The reason is that the integral above is intractable because the posterior is intractable. And posterior is intractable because the evidence is intractable.

To see this argument more clearly, let's rewrite the posterior distribution $P(\omega|X)$ by using the law of total probability:

$$p(\omega|X) = \frac{p(X|\omega)p(\omega)}{p(X)} \implies p(\omega|X) = \frac{p(X|\omega)p(\omega)}{\int p(X,\omega)d\omega}$$

The denominator of the posterior function represents a high-dimensional integral that lacks a general analytical solution. Therefore, calculating the posterior always means **approximating** it. There is a whole literature out there on how to approximate the posterior. After we are done approximating the posterior, our neural network in a regression problem will look like this:

```
In [0]: from IPython.display import Image
        from IPython.core.display import HTML
        HTML('<img src=https://blog.dominodatalab.com/wp-content/uploads/2017/03/banner.png wi
```

```
Out[0]: <IPython.core.display.HTML object>
```

Observe that for each point in the graph, we have a number of fits. The variance of fitted functions are the highest at the region where we don't have data and it is the lowest where we have a concentrated pile of data. We can also interpret it during prediction so that we have the highest aleatoric uncertainty at the region where we have the least clue about our data.

So, we explained what the posterior is. But what about the prior, our domain knowledge of the data? Does our choice of the prior matter? Not really, if we have big enough data. The importance of the prior in determining the posterior evades during the process of model criticism, where we update our beliefs (posterior) first by choosing an arbitrary prior, then in the next steps by choosing our next prior as our previous posterior. We may be careful to choose the prior as not dependent on the data (**uninformative prior**), but except that any prior will be alright for most cases **(citation needed).**

As we told before, there is wide literature on how to approximate the posterior function. We will focus on three popular methods: Hamiltonian Monte Carlo Markov Chain (Neal 1995), Variational Inference (...), and Monte Carlo Dropout (Gal, 2016).

** Summarise the models **

2 Hamiltonian Markov Chain Monte Carlo

The idea behind Markov Chain Monte Carlo (MCMC) is to construct a Markov Chain on the state space of , whose stationary posterior distribution is the target density. Markov Chain allows us to take correlated samples and it is particularly useful to arrive at the typical set during sampling from the posterior distribution. We start MCMC from an arbitrary initial state and continue until we take samples only from the typical set. The samples from the beginning until the time we arrived to the typical set will be ignored. This period is called as burn-in phase. The MCMC forgets where it started from, and stays sampling exactly from the area we wanted.

Although the idea of MCMC approximation makes sense, there is an inherent problem with it: It is hard to be sure whether we passed burn-in phase and if so, we might not live enough to see it occurring. Even in small samples, it takes a long time for MCMC sampling to converge. (Murphy, Chapter 24)

Hamiltonian Markov Chain Monte Carlo (HMC) constructs "Markov transitions by lifting into, exploring, and projecting from the expanded space" of the typical set (Betancourt 2018). It covers the typical set very fast in comparison to the classical MCMC and provides with a physical explanation. Therefore, it is a big step towards improving the issues with duration and diagnosis of the burn-in phase of MCMC.

** the gif is here **

3 Variational Inference

The posterior is intractable. Sampling-based methods usually take so much time and computational resources that they are almost impractical to use in the deep learning setting (even though HMC made the process a lot faster). We need a shortcut. Variational Inference provides us with such shortcut while conserving Bayesian properties. The idea is to find the most similar (**citation needed**) function $q(\omega)$ to the posterior $p(\omega|x)$. In order to find it, we minimize the Kullback-Leibler divergence (KL divergence) of $q(\omega)$ from $p(\omega|x)$. In other words, we minimize $KL(q(\omega)||p(\omega|x))$. To do so, we assume that ω is parametrized by some latent variable θ (here, the latent of the latent, which we also call 'variational parameters'), and apply minimization with respect to θ.

$$\min_{\theta} KL(q(\omega;\theta)||p(\omega|x))$$

Note that KL divergence is non-symmetric, meaning that reversing the arguments will lead us to a totally different method. Also note that what we are doing here is optimizing functions to find the best functional representation of the posterior $p(\omega|x)$. This optimization belongs to the field of 'calculus of variations' (**citation needed**). To think about this optimization problem, we can rewrite below what KL divergence actually is:

$$\min_{\lambda} KL(q(\omega;\theta)||p(\omega|x)) \iff \min_{\theta} \mathop{\mathbb{E}}_{q(\omega|\theta)}[logq(\omega;\theta) - logp(\omega|x)]$$

One question immediately appears: How do we minimize the distance to the posterior if we don't even know what the posterior is? It is a nontrivial question. We can have a clue about the posterior only if we have data! Let's play with the KL divergence if we can recover the evidence $p(x)$ any where around. We will rewrite the posterior by using the Bayes's rule:

$$KL(q(\omega;\theta)||p(\omega|x)) = \mathbb{E}[logq(\omega;\theta) - logp(\omega|x)] = \mathbb{E}[logq(\omega;\theta) - log\frac{p(x,\omega)}{p(x)}] = \mathbb{E}[logq(\omega;\theta) - logp(x,\omega) + lo$$

Reorganizing the expectation gives us the evidence lower bound $ELBO(\theta)$:

$$KL(q(\omega;\theta)||p(\omega|x)) = -\mathbb{E}[logp(x,\omega) - logq(\omega;\theta)] + \mathbb{E}\,logp(x)$$

$$ELBO(\lambda) = \mathbb{E}[logp(x,\omega) - logq(\omega;\theta)]$$

As we see, $logp(x)$ does not depend on θ. That means minimizing $KL(q(\omega;\theta)||p(\omega|x))$ is the same thing as maximizing ELBO(θ) which we call the evidence lower bound. Why do we call it so?

By chance, our optimization objective ELBO(λ) is a lower bound to the evidence p(x) we have. That means, it is something that makes sense to maximize. To see this inequality, let's play with the logarithm of p(x) by using the law of total probability and multiply-dividing by q(ω):

$$logp(x) = log \int_{\omega} p(x,\omega) = log \int_{\omega} p(x,\omega)\frac{q(x)}{q(x)} = log \int_{\omega} q(x)\frac{p(x,\omega)}{q(x)} = log(\mathbb{E}_q[\frac{p(x,\omega)}{q(\omega)}])$$

Now we use the Jensen's Inequality $f(\mathbb{E}(X)) \geq \mathbb{E}(f(X))$ for any concave function f to show that logp(x) is always greater than or equal to our maximization objective ELBO(λ). Now we have a good justification in using the Variational Inference as a valid inference method.

$$log(\mathbb{E}_q[\frac{p(x,\omega)}{q(\omega)}]) \geq \mathbb{E}_q[log(\frac{p(x,\omega)}{q(\omega)})]$$

$$log(\mathbb{E}_q[\frac{p(x,\omega)}{q(\omega)}]) \geq \mathbb{E}[logp(x,\omega) - logq(\omega;\theta)]$$

$$log(\mathbb{E}_q[\frac{p(x,\omega)}{q(\omega)}]) \geq ELBO(\theta)$$

Variational Inference.png

We have a graphical explanation above. We maximize the evidence lower bound as to attain the nearest point to the posterior we can. There are still some Kullback Leibler divergence from the real posterior from which we cannot avoid. It is the drawback of implementing this approach. We cannot make the KL-divergence zero in general and cannot really attain the real posterior, regardless of how ambitious we are in optimization.

As you'll wonder, there are different ways to minimize ELBO(θ). In this paper, we are going to use ADVI algorithm from the PyMC3 package.

4 Monte Carlo Dropout

Scared by all those mathematical derivations of the variational inference? Good news for you: An interesting finding in deep learning research suggests that you can apply variational inference without even knowing all those derivations. And it is going to be much faster and applicable. Yarin Gal (2016) suggests that we are doing something very close to a type of variational inference each time we regularize our deterministic neural network with dropout technique. According to his PhD thesis, all we need is to apply dropout during both training and test time, as opposed to the usual application of dropout only during model training (Gal, 2016). Let's review what the usual dropout is:

Dropout picture here

If we build a complicated neural network with lots of hidden layers and cells on a limited amount of training data, our model would probably memorize the data and does not generalize well. This phenomenon is called **overfitting**. Dropout is a regularizer to avoid overfitting which disables some cells in hidden layers with some probability (Google, 2016). By doing this, dropout effectively samples from an exponential number of different networks in a tractable and feasible way (Goodfellow**dp**). Dropout is computationally cheap and it can be applied in nearly any neural network architecture with ease. Furthermore, it can be trained with stochastic gradient descent. Wenn applying dropout, we basically do not change anything in our neural network in terms of optimization.

Yarin Gal tied up several derivations of stochastic variational inference to those of the stochastic regularization techniques, including dropout. He argues that deep learning models trained with dropout have already been applying variational inference on Gaussian Processes all along (Gal, 2016). Encouraged by this finding, he suggests to use dropout during test time in order to obtain approximate samples from the posterior function $p(\omega|x)$. When we apply dropout during test time, we obtain different results each time we run the model. They are approximate samples from the posterior predictive distribution. Gal calculates unbiased estimators for the mean and the variance of the posterior predictive distribution as the following:

Consider y=$f^{\hat{\omega}}(x)$ as the output of the Bayesian NN, and t= 1,.., T are samples from the posterior predictive distribution

$$\hat{\mathbb{E}}(y) = \frac{1}{T}\sum_{t=1}^{T} f^{\hat{\omega}_t}(x)$$

and

$$\hat{\mathbb{E}}(y^T y) = \tau^{-1} I + \frac{1}{T}\sum_{t=1}^{T} f^{\hat{\omega}_t}(x)^T f^{\hat{\omega}_t}(x) - \hat{\mathbb{E}}(y)^T\,\hat{\mathbb{E}}(y)$$

Note that $f^{\hat{\omega}}(x)$ is a row vector. The mean of our posterior predictive samples is an unbiased estimator of the mean of the approximate distribution $q(\omega)$. The sample variance plus a term $\tau^{-1}I$ is also an unbiased estimator of the variance of $q(\omega)$. That means, with only a small adjustment made to the sample variance, we get the Bayesian results very handy. The adjusting term τ equals to the following:

$$\tau = \frac{(1-p)l^2}{2N\lambda}$$

Note that p here is the probability of the units not being dropped, thus $p := 1 - p_{code}$. l is the prior length-scale. This captures our belief over the function frequency. A short length-scale l corresponds to high frequency data, and a long length-scale corresponds to low frequency data. λ corresponds to the weight decay regularization term which we additionally use to regularize weight optimization.