bayes_by_backpropagation

datetime: 2024-12-03, 17:32
Field: #machine_learning #statistics
Topic: #bayesian

Motivation of Bayesian neural network

A general supervised learning problem is: Given a set of data $\mathcal{D} = \{x_n, y_n\}_{n=1}^N$, how to predict the label $\tilde{y}$ of an unseen datum $\tilde{x}$?

Ideally, if we know $p(y \mid x)$, then the prediction is simply given by $\tilde{y} = \arg\max_y p(y \mid \tilde{x})$. But the problem is we do not know $p(y \mid x)$. So we need to approximate $p(y \mid x)$ using some function / distribution approximators $p(y \mid x, w)$, with known form parameterised by w.

Then all the problem is how to find the best $w^\star$ such that $p(y \mid x, w^\star)$ is as close to $p(y \mid x)$ as possible.

Point estimate

MLE

The most straightforward approach is maximum likelihood estimate. The main idea is to find w such that the likelihood of getting the entire dataset $\mathcal{D}$ is maximised:

$$w^\star = \arg_w \max \prod_{n=1}^{N} p(y_n \mid x_n, w) = \arg_w \max p(\mathcal{D} \mid w).$$

The problem is pretty obvious: Some w are just inherently not possible to be chosen (or in another words, there are structures in the distribution of w). One example is: you are the boss, and you find that one of the worker is not at work today ($\mathcal{D}$). You want to find a reason (w) to explain it. The MLE gives that w is that they died, since it maximises the likelihood that they cannot go to work.

MAP

Now we know that MLE is problematic since it does not consider any implicit structure of w. What if we propose some prior structure for w as $p(w)$, and use the following estimate for $w^\star$:

$$w^\star = \arg_w \max \prod_{n=1}^{N} p(w \mid x_n, y_n) = \arg_w \max p(w \mid \mathcal{D}) = \arg_w \max p(\mathcal{D} \mid w) p(w).$$

Equivalence of MAP and regularisation.

Additional bayes estimator: $w^\star = \mathbb{E}_w p(w \mid \mathcal{D})$ minimises the L_2 loss, while $w^\star = \text{median}_w(p(w \mid \mathcal{D}))$.

For the unseen datum $\tilde{x}$, the prediction is given by $\hat{y} = \arg_y \max p(y \mid \tilde{x}, w^\star)$.

Here everything makes sense. But the only problem is, for arbitrary x, there is always a prediction given but we have no idea how certain the prediction is. It would be good if we can get some sort of certainty associated with the prediction so that we can decide whether we want to use this prediction depending on the certainty.

From point estimate to posterior

The core of the problem is because that we focus on the point estimates of the parameter w. Alternatively, we want to find the posterior distribution of w given observed data $\mathcal{D}$, i.e. $p(w \mid \mathcal{D})$. Then for a new data $\tilde{x}$, we can make much more predictions together with the certainty estimates.

Derivation of the objective function

But we only know the form of $p(y \mid x, w)$ and $p(w)$. In general, $p(w \mid \mathcal{D})$ is intractable. However, we can use probability measure $q(w \mid \theta)$ to approximate $p(w \mid \mathcal{D})$. We want to find $\theta^\star$ such that $q(w \mid \theta^\star)$ and $p(w \mid \mathcal{D})$ are close enough. Wait when we say "close", we are implying a metric. Here we use the classic KL divergence:

$$\begin{aligned}
\theta^\star &= \arg_{\theta\in\Theta} \min \mathrm{KL}(q(w \mid \theta) \| p(w \mid \mathcal{D})) \\
&= \arg_{\theta\in\Theta} \min \mathbb{E}_{q(w|\theta)} \left[\ln \frac{q(w \mid \theta)}{p(w \mid \mathcal{D})} \right] \\
&= \arg_{\theta\in\Theta} \min \mathbb{E}_{q(w|\theta)} \left[\ln \frac{q(w \mid \theta) p(\mathcal{D})}{p(\mathcal{D} \mid w) p(w)} \right] \\
&= \arg_{\theta\in\Theta} \min \mathrm{KL}(q(w \mid \theta) \| p(w)) - \mathbb{E}_{q(w|\theta)} \ln p(\mathcal{D} \mid w) = R(\theta, \mathcal{D}).
\end{aligned}$$

Great! Now we know the prior $p(w)$, the chosen $q(w \mid \theta)$, and the likelihood $p(\mathcal{D} \mid w)$, we **only** need to solve this optimisation problem!

Computational details

Now we have our objective function, however, how to optimise it using, say, stochastic first-order methods? We need essentially to solve two problems: 1. How to compute the gradient of the objective function. 2. How to break the objective function into a finite-sum form so that we can use batches to optimise the parameters.

Gradient of the expectation

The problem is how to compute gradient over the expectation, which contains the parameter θ of interest.

Here is the trick. Let a random variable $\epsilon \sim Q(\epsilon)$ have pdf $q(\epsilon)$ and there exist a mapping $w = t(\epsilon, \theta)$. Further we assume function $t(\cdot, \theta)$ is an one-to-one mapping from ϵ to w, with the determinant of the

Jacobian to be one, i.e. $|J(\epsilon)| = 1$. Then we have $q(w \mid \theta)dw = q(\epsilon)d\epsilon$.

IGNORE THIS ---

$$\begin{aligned}\frac{\partial \mathbb{E}_{q(w|\theta)} \ln p(\mathcal{D} \mid w)}{\partial \theta} &= \frac{\partial}{\partial \theta} \int q(w \mid \theta) \ln p(\mathcal{D} \mid w) dw \\ &= \frac{\partial}{\partial \theta} \int \ln p(\mathcal{D} \mid w) q(\epsilon) d\epsilon \\ &= \mathbb{E}_{q(\epsilon)} \left[\frac{\ln p(\mathcal{D} \mid w)}{\partial w} \frac{dw}{d\theta} \right] \\ &= \mathbb{E}_{q(\epsilon)} \left[\frac{\ln p(\mathcal{D} \mid w)}{\partial w} f(\epsilon) \right].\end{aligned}$$

START FROM HERE ---

Now our empiric risk function can be written as

$$\begin{aligned}R(\theta, \mathcal{D}) &= \mathrm{KL}(q(w \mid \theta) \| p(w)) - \mathbb{E}_{q(w|\theta)} \ln p(\mathcal{D} \mid w) \\ &= \mathrm{KL}(q(w \mid \theta) \| p(w)) - \int q(w \mid \theta) \ln(\mathcal{D} \mid w) dw \\ &= \mathrm{KL}(q(w \mid \theta) \| p(w)) - \int \ln p(\mathcal{D} \mid t(\epsilon, \theta)) q(\epsilon) d\epsilon \\ &= \mathrm{KL}(q(w \mid \theta) \| p(w)) - \mathbb{E}_{q(\epsilon)} \ln p(\mathcal{D} \mid t(\epsilon, \theta))\end{aligned}$$

In case that the KL divergence is not analytical feasible, we can apply this trick on the KL term as well.

$$\begin{aligned}R(\theta, \mathcal{D}) &= \mathrm{KL}(q(w \mid \theta) \| p(w)) - \mathbb{E}_{q(\epsilon)} \ln p(\mathcal{D} \mid t(\epsilon, \theta)) \\ &= \int q(w \mid \theta) \ln \frac{q(w \mid \theta)}{p(w)} dw - \mathbb{E}_{q(\epsilon)} \ln p(\mathcal{D} \mid t(\epsilon, \theta)) \\ &= \mathbb{E}_{q(\epsilon)} \left[\ln \frac{q(t(\epsilon, \theta) \mid \theta)}{p(t(\epsilon, \theta))} - \ln p(\mathcal{D} \mid t(\epsilon, \theta)) \right]\end{aligned}$$

So the risk function can be estimated by K-sample Monte Carlo method

$$R(\theta, \mathcal{D}) \approx \frac{1}{K} \sum_{k \in [K]} \ln \frac{q(t(\epsilon_k, \theta) \mid \theta)}{p(t(\epsilon_k, \theta))} - \ln p(\mathcal{D} \mid t(\epsilon_k, \theta)).$$

Re-weighting the KL-term

In deep learning, we do not use the full batch of data to compute the gradient and update the parameters but relies on the batch-wise update. Thus, we need the loss function $\ell(\cdot)$ on each data point (x_n, y_n) so that the risk function can be expressed in the form of finite sum over all data points and we can then use batch-wise gradient descent.

$$
\begin{aligned}
R(\theta, \mathcal{D}) &\approx \frac{1}{K}\sum_{k\in[K]}\left[\ln\frac{q(t(\epsilon_k,\theta)\mid\theta)}{p(t(\epsilon_k,\theta))} - \sum_{n\in[N]}\ln p(y_n\mid x_n, t(\epsilon_k,\theta))\right] \\
\to R(\theta, \mathcal{D}) &\approx \frac{1}{NK}\sum_{k\in[K]}\left[\ln\frac{q(t(\epsilon_k,\theta)\mid\theta)}{p(t(\epsilon_k,\theta))} - \sum_{n\in[N]}\ln p(y_n\mid x_n, t(\epsilon_k,\theta))\right] \\
\to R(\theta, \mathcal{D}) &\approx \frac{1}{N}\sum_{n\in[N]}\left[\frac{1}{K}\sum_{k\in[K]}\left[\frac{1}{N}\ln\frac{q(t(\epsilon_k,\theta)\mid\theta)}{p(t(\epsilon_k,\theta))} - \ln p(y_n\mid x_n, t(\epsilon_k,\theta))\right]\right] \\
\to \ell(x_n, y_n;\theta) &= \frac{1}{K}\sum_{k\in[K]}\left[\frac{1}{N}\ln\frac{q(t(\epsilon_k,\theta)\mid\theta)}{p(t(\epsilon_k,\theta))} - \ln p(y_n\mid x_n, t(\epsilon_k,\theta))\right]
\end{aligned}
$$

If we use 1-sample Monte Carlo method such that $K = 1$, we have the loss function as

$$
\ell(x_n, y_n;\theta) = \frac{1}{N}\ln\frac{q(t(\epsilon,\theta)\mid\theta)}{p(t(\epsilon,\theta))} - \ln p(y_n\mid x_n, t(\epsilon,\theta))
$$

KL divergence of two gaussians

https://en.wikipedia.org/wiki/Multivariate_normal_distribution#Kullback–Leibler_divergence
Assume we have prior $\mathcal{N}(\mu_0, I)$ and posterior $\mathcal{N}(\mu, (\sigma^2)^\top I)$, the KL term is given by

$$
\begin{aligned}
D_{\mathrm{KL}}(\mathcal{N}(\mu,(\sigma^2)^\top I)\mid\mathcal{N}(\mu_0, I)) &= \frac{1}{2}\left(\operatorname{tr}((\sigma^2)^\top I) + \|\mu-\mu_0\|_2^2 - k + \ln\frac{\det(I)}{\det((\sigma^2)^\top I)}\right) \\
&= \frac{1}{2}\left(\sum_{i=1}^k \sigma_i^2 + \|\mu-\mu_0\|_2^2 - \ln\prod_{i=1}^k\sigma_i^2 - k\right) \\
&= \frac{1}{2}\left(\sum_{i=1}^k \sigma_i^2 + \|\mu-\mu_0\|_2^2 - \sum_{i=1}^k\ln\sigma_i^2 - k\right)
\end{aligned}
$$

Because of the independence of parameters, the KL terms for both W and b for different layers can be summed up.

```
kl_w = 0.5 * (w_var.sum() + (self.W_mu - self.prior_mu).pow(2).sum() \
    - w_var.log().sum() - np.log(self.prior_var) * self.W_mu.numel() - self.W_mu.numel())
```

I think in my implementation, the term involving `np.log(\cdot)` should be either removed (so that we stick to I as the prior), or the term $\|\mu-\mu_0\|_2^2$ can be extended to support different prior by multiplying the inverse of the prior covariance. But the experiments are done on I as prior anyway so it should not be a problem for now?

Reference

Weight Uncertainty in Neural Networks (https://arxiv.org/abs/1505.05424)