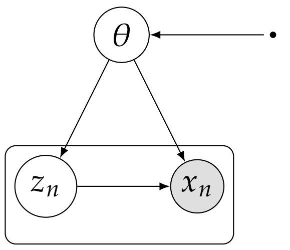
Variational Inference 10

10.1 Introduction

Variational Inference is a deterministic approximation to perform inference. This differentiates it from other approximate inference techniques, such as Markov Chain Monte Carlo, as there is no sampling involved.

Since Variational Inference is a big topic, we are just going to introduce the basic ideas of the subject.

We start by considering a joint distribution $p(x, z)$, with x observable variables and z non observable (latent) variables. This scenario includes examples like the ones explored in the Expectation Maximization chapter, where the latent variables are "coupled" with the observables even though we cannot observe them (we will call these **local latent variables** and denote them as z_n), but z this time can also be latent parameters of our distribution (**global latent variables** θ), thus encapsulating the possibility of doing inference on the parameters. Therefore z is represented as $z = (z_1, \dots, z_n, \theta)$ and our schema is represented by the following PGM:



We have observations $x_1, \dots, x_n = \underline{x}$, which will sometimes also be denoted simply as x, and we would like to compute the posterior distribution

$$p(z|\underline{x})$$

and also the model evidence

$$p(\underline{x}) = \int p(\underline{x}, z)dz.$$

This was also the context in Expectation Maximization, where we were considering the ELBO, and we discussed that we can decompose the evidence as

$$\log p(\underline{x}) = \mathscr{L}(q) + KL[q(z)||p(z|\underline{x})]$$

$$\mathscr{L}(q) = \int q(z) \left[\log p(\underline{x}, z) - \log q(z)\right] dz = \mathbb{E}_q[\log p(\underline{x}, z) - \log q(z)]$$

$$KL[q(z)||p(z|\underline{x})] = -\int q(z) \left[\log p(z|\underline{x}) - \log q(z)\right] dz$$

where q is the so called variational distribution.

In EM, we were concerned about optimizing the ELBO by a two-step optimization over q and θ. Now θ are not explicitly present, but rather trated probabilistically and included in z. Hence, we only have the variational distribution and the goal in Variational Inference is to find the best q that approximates $p(z|\underline{x})$.

Notice this: $p(\underline{x})$ is fixed because observations are fixed, it's a number. The two terms in which we decompose it are the ELBO and the KL divergence, therefore

- the q that minimizes the KL-divergence is the same as the one maximizing the ELBO $\mathcal{L}(q)$
- $\mathcal{L}(q)$ is maximum when $KL[q||p] = 0$, i.e. $q = p(z|\underline{x})$

However, in most of the cases the computation of $p(z|\underline{x})$ is intractable.

The solution to this problem is to restrict q to a tractable family of distributions. Therefore the optimal $q(z)$ is likely to be such that $KL[q||p] > 0$. The goal of Variational Inference is then to maximize $\mathcal{L}(q)$ in a suitably restricted space of variational distributions q (we will call this space Q).

Let's make a first example to see how to choose this space. Think of Q as a set of parametric distributions, i.e. $Q = \{q(z|\lambda), \lambda \in \mathcal{R}^k\}$ (might be for example Gaussian distributions, with λ representing the average and the covariance matrix of our Gaussian). The optimization problem is then on λ, hence we need to find $\text{argmax}_\lambda \mathcal{L}(q(\lambda)) = \text{argmax}_\lambda \mathcal{L}(\lambda)$, but the caveat is that this would typically be a highly non-linear and non-convex optimization. Lastly, notice that λ would typically be composed of parameters for local latent variables and global latent variables, i.e. $q(z|\lambda) = q(\theta|\lambda_\theta) \prod_i q(z_i|\lambda_i, \theta)$

10.2 Mean Field Variational Inference

Rather than choosing a parametric model for our variational distribution, we can instead opt for a different strategy.

Given that we are interested in approximating $p(z|\underline{x})$ by $q(z)$, we assume that z can be decomposed in M different blocks of variables $z = (z_1, \dots, z_M)$ and we further assume the **independence of the blocks**, i.e. that

$$q(z) = \prod_{i=1}^{M} q_i(z_i)$$

Sometimes we will denote $q_i(z_i) = q_i$ for brevity.

This is known as the **mean field** assumption (the name comes from physics and stochastic processes).

Let's start by writing the lower bound for the mean field approximation

$$\mathscr{L}(q) = \mathbb{E}\left[\log p(\underline{x}, z) - \log q(z)\right] =$$
$$\int \prod_i q_i[\log p(\underline{x}, z) - \sum_i \log q_i]dz =$$
$$= \int q_j \left[\int \log p(\underline{x}, z) \prod_{i \neq j} q_i dz_i\right] dz_j - \int q_j \log q_j dz_j + \text{const}$$
$$= \int q_j \mathbb{E}_{i \neq j}[\log p(\underline{x}, z)]dz_j - \int q_j \log q_j dz_j + \text{const}$$

where on the second row we factored out the terms depending on one factor q_j, hiding all the terms not depending on q_j in the $const$ term.

Next we will define the function

$$\log \tilde{p}(\underline{x}, z_j) = \mathbb{E}_{i \neq j}[\log p(\underline{x}, z)] + \text{const}$$

Which means, wrapping up

$$\mathscr{L}(q_j(z_j)) = \mathbb{E}_{q_j}[\log \tilde{p}(\underline{x}, z_j) - \log q_j] + \text{ const},$$
$$\text{hence } \mathscr{L}(q_j(z_j)) = -KL[q_j || \tilde{p}(\underline{x}, z_j)] + \text{ const}$$

Now we have M lower bounds $\mathscr{L}(q_j)$, which have to be maximized for q_j, with $q_i, i \neq j$ fixed. Since the ELBO is maximized when $KL[q||p] = 0$, we know that our best approximation is indeed

$$q_j^\star(z_j) = \tilde{p}(\underline{x}, z_j) \tag{10.1}$$

Hence $\log q_j^\star(z_j) = \mathbb{E}_{i \neq j}[\log p(\underline{x}|z)] +$ const which implies that

$$q_j^\star(z_j) = \frac{\exp(\mathbb{E}_{i \neq j}[\log p(\underline{x}, z)])}{\int \exp\left(\mathbb{E}_{i \neq j}[\log p(\underline{x}, z)]\right) dz_j}$$

If we can compute analytically the expectation $\mathbb{E}_{i \neq j}[\log p(\underline{x}, z)]$ then we are in a scenario in which we can actually perform the mean field approximation. We cannot compute directly the expectation because we do not know q_i. What we do then, is that we initialize q_i to some initial distribution, then cycle through q_j, optimizing the ELBO with respect to the coordinate j and fixing all the others $q_{\neg j}$. We repeat for all the coordinates in turn (**coordinate ascent**) until convergence, which is guaranteed since the bound is convex on q_j.

Therefore, given that we know how to compute these integrals over the logarithm of the joint distribution, mean field approximation gives us a relatively easy method to approximate our posterior distribution.

However, being able to compute the integral depends on the model, which means that this approximation is not suitable for every scenario.

10.2.1 Example on Gaussian distribution

We will use mean field variational inference to go from a Gaussian to a factorized Gaussian. We have a joint distribution $p(z) = \mathcal{N}(z|\mu, \Lambda^{-1})$ where $\mu = \begin{bmatrix}\mu_1\\ \mu_2\end{bmatrix}$ and Λ is the precision matrix, so we are dealing with a 2-dimensional probability distribution. The mean field approximation implies that $q(z) = q(z_1)q(z_2)$. Therefore we need to perform the expectations.

$$\begin{aligned}\log q_1^\star(z_1) &= \mathbb{E}_{z_2}[\log p(z)] + \text{ const}\\ &= \mathbb{E}_{z_2}[-\frac{1}{2}(z_1-\mu_1)^2\Lambda_{11} - (z_1-\mu_1)\Lambda_{12}(z_2-\mu_2)] + \text{ const}\\ &= -\frac{1}{2}(z_1-\mu_1)^2\Lambda_{11} - (z_1-\mu_1)\Lambda_{12}(\mathbb{E}[z_2]-\mu_2) + \text{ const}\end{aligned}$$

This has a nice form, because, since $\log q_1^\star(z_1)$ is a quadratic form, we know that $q_1^\star(z_1)$ is Gaussian $\mathcal{N}(z_1|m_1, \Lambda_{11}^{-1})$ where $m_1 = \mu_1 - \Lambda_{11}^{-1}\Lambda_{12}(\mathbb{E}[z_2] - \mu_2)$.

By symmetry we have that

$$q_2^\star(z_2) = \mathcal{N}(z_2|m_2, \Lambda_{22}^{-1}), \qquad m_2 = \mu_2 - \Lambda_{22}^{-1}\Lambda_{21}(\mathbb{E}[z_1] - \mu_1)$$

In our case we can solve directly these equations by noticing that $\mathbb{E}[z_i] = \mu_i$ which means that $m_1 = \mu_1$ and $m_2 = \mu_2$.

Notice that these are different than the marginals of a Gaussian. The variance of each component is different than just the diagonal component of the precision matrix (when we compute the covariance in the marginalization process we need to invert the full matrix)

10.2.2 Variational Inference with direct and inverse KL

We may ask ourselves what would happen if instead of trying find the best variational distribution q such that

$$KL[q||p] \quad \text{is minimum}$$

we tried to solve the inverse problem, i.e. finding the q such that

$$KL[p||q] \quad \text{is minimum}$$

Looking at the example of the Gaussian that we introduced before we have the following qualitative results:

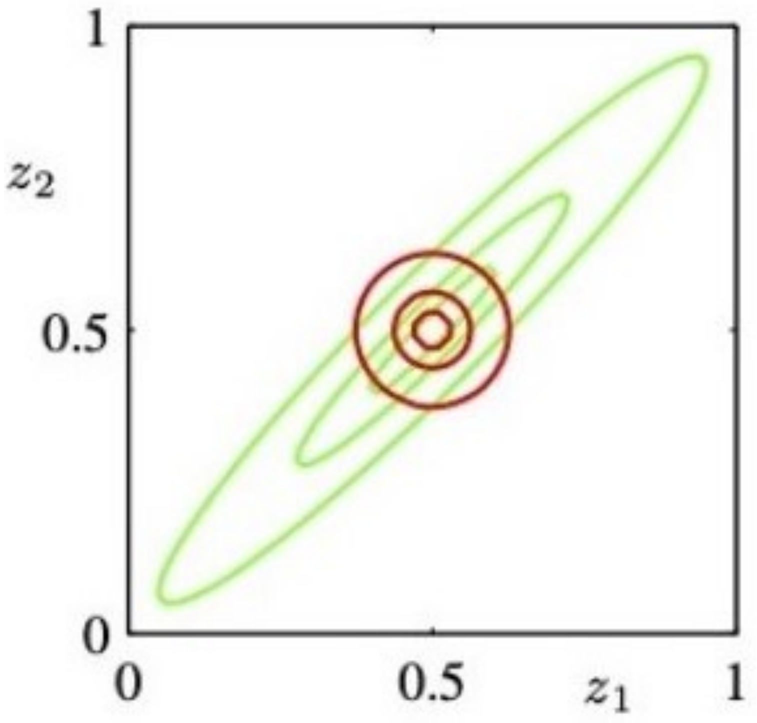
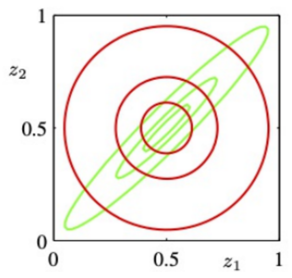
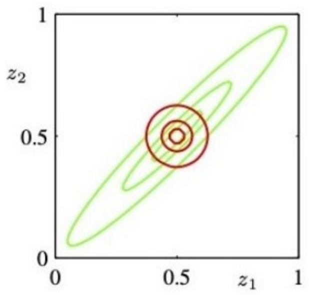


Figure 10.1: Original distribution (green) and its mean field approximation (Red)





Generally speaking, the inverse problem is intractable, as it requires to evaluate an expectation with respect to the unknown distribution p, but in our scenario, we can make it tractable by using the mean field approximation so that $q(z) = q(z_1)q(z_2)$.

In particular, the variational distribution found by minimizing $KL[p||q]$ is such that $q(z_i)$ is exactly the i_{th} marginal of the Gaussian distribution, hence it encompasses all the "original range" of our distribution (the border of the picture in red is the same of the picture in green).

This is a very common behaviour of these two approximations. The approximation of the direct KL-divergence is known as the **zero forcing** approximation scheme. If $p(z) \approx 0$ then $q(z) \approx 0$ around the mode. That's because if you take a small p and a large q you get a large $KL[q||p]$.

The approximation of the inverse KL-divergence instead is known as the **zero avoiding** approximation scheme. Which means that if $q(z)$ is non zero then $p(z)$ is non zero, for the same (but inverse) reason as before. In multidimensional distributions the variational distribution will end up overlapping different modes of our system.

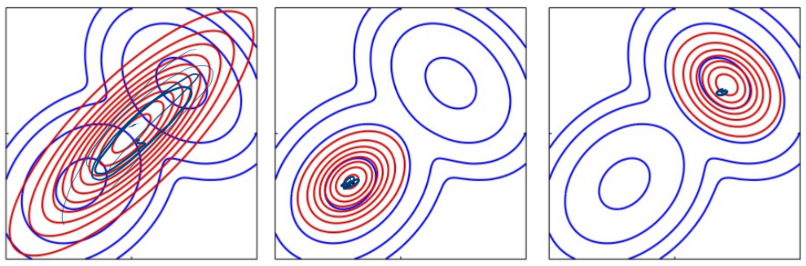
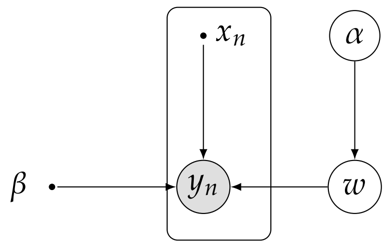


Figure 10.4: On the left, the inverse KL divergence, on the right, the direct KL divergence

10.3 Variational Linear Regression

Mean Field variational inference can be used also to make approximate inference in the context of Linear regression. In particular, we will consider the case in which we would like to put a hyperprior over the parameter α, which regulates the variance of the prior for our weights.



Our joint distribution is

$$p(y, w, \alpha) = p(y|w)p(w|\alpha)p(\alpha)$$

with each term, since we are dealing with Linear regression, defined by

$$p(y|w) = \prod_{n=1}^{N} \mathcal{N}(y_n|w^T\phi(x_n), \beta^{-1})$$
$$p(w|\alpha) = \mathcal{N}(w|0, \alpha^{-1}I)$$
$$p(\alpha) = \text{Gamma}(\alpha|a_0, b_0) = \frac{1}{\Gamma(a_0)} b_0^{a_0} \alpha^{a_0-1} e^{-b_0\alpha}$$

and our goal becomes to compute the posterior distributions for w and α using mean field variational inference, i.e. to compute $p(w, \alpha|y)$.

We will use the mean field variational distribution $q(w, \alpha) = q(w)q(\alpha)$

Let's start with

$$\begin{aligned} q^\star(\alpha) &= \mathbb{E}_w[\log p(y, w, \alpha)] + \text{ const} \\ &= \log p(\alpha) + \mathbb{E}_w[\log p(w|\alpha)] + \text{ const} \\ &= (a_0 - 1)\log\alpha - b_0\alpha + \frac{M}{2}\log\alpha - \frac{\alpha}{2}\mathbb{E}[w^T w] + \text{ const} \end{aligned}$$

Therefore what we have is, in fact, another Gamma distribution

$$\begin{aligned} q^\star(\alpha) &= \text{Gamma}(\alpha|a_N, b_N) \\ a_N &= a_0 + \frac{M}{2} \\ b_N &= b_0 + \frac{1}{2}\mathbb{E}_q[w^T w] \end{aligned}$$

We can also workout what happens for the other term of the variational distribution

$$\begin{aligned} \log q^\star(w) &= \log p(y|w) + \mathbb{E}_\alpha[\log p(w|\alpha)] + \text{ const} \\ &= -\frac{\beta}{2}\sum_{n=1}^{N}[w^T\phi(x_n) - y_n]^2 - \frac{1}{2}\mathbb{E}_\alpha w^T w + \text{ const} \end{aligned}$$

Again, notice that here we have a quadratic form, hence completing the square gives us a Gaussian distribution

$$\begin{aligned} q^\star(w) &= \mathcal{N}(w|m_N, S_N) \\ m_N &= \beta S_N \Phi^T y \\ S_N &= (\mathbb{E}[\alpha]I + \beta\Phi^T\Phi)^{-1} \end{aligned}$$

The solution for w is very similar to what we have in linear regression keeping α fixed, but now instead of just α we have its expectation in the equation for the covariance matrix.

Notice that we know what are these expectations:

$$\begin{aligned} \mathbb{E}[\alpha] &= \frac{a_N}{b_N} \\ \mathbb{E}[w^T w] &= m_N^T m_N + \text{Trace}(S_N) \end{aligned}$$

Then we can just initialize the expectation for α and then we start iterating by computing the expectation of $w^T w$ and so on.

We can also in principle compute $\mathcal{L}(q) \approx p(y)$ that approximates the model evidence, which can then be used for Bayesian model comparison.

10.4 Black box Variational Inference

As we have seen, the mean field approximation efficacy is dependent on our ability to compute the expectations that arise in the equations determining the components of our variational distribution. If we can't compute the expectations, we are not able to use the approximation at all. Here black-box (or stochastic) Variational Inference enters the picture as a viable alternative.

The general idea is to perform a **Monte-Carlo estimate** of the gradient of the ELBO with respect to the variational parameters and then perform gradient ascent with these estimates.

Our variational distribution will be tipically parametric in this scenario $q(z(\lambda))$. The lower bound is

$$\mathscr{L}(\lambda) = \mathbb{E}_{q(z|\lambda)}[\log p(x, z) - \log q(z|\lambda)]$$

We want to compute $\nabla_\lambda \mathscr{L}(\lambda)$ which we cannot compute analytically. The idea, as previously stated, is to sample this gradient, but how can we do it? We need to turn the gradient of this estimation into the estimation of a gradient in order to exploit the capabilities of Monte Carlo methods.

We have two strategies, the first one works only for special cases, while the second one, albeit more complex, is usable in general.

10.4.1 Reparameterization trick

The first solution is called "**Reparameterization trick**". It can be used if we can write $z = g_\nu(\varepsilon)$ as a certain function g of ε, with ε being some random variable coming from a distribution $\hat{q}(\varepsilon)$ which is independent of λ. We also define $\hat{\lambda} = (\lambda, \nu)$ where ν are the parameters of the function g. Then we can write

$$\mathscr{L}(\hat{\lambda}) = \mathbb{E}_{\hat{q}(\varepsilon)}[\log p(x, g_\nu(\varepsilon)) - \log q(g_\nu(\varepsilon)|\lambda)]$$

and our expectation does not depend on λ anymore. Hence we can sample from it and compute the gradient:

$$\nabla_{\hat{\lambda}} \mathscr{L}(\hat{\lambda}) = \mathbb{E}_{\hat{q}(\varepsilon)}[\nabla_{\hat{\lambda}} \log p(x, g_\nu(\varepsilon)) - \nabla_{\hat{\lambda}} q(g_\nu(\varepsilon)|\lambda)]$$

Let's also define the following function for easier readability

$$G(\varepsilon) = [\nabla_{\hat{\lambda}} \log p(x, g_\nu(\varepsilon)) - \nabla_{\hat{\lambda}} q(g_\nu(\varepsilon)|\lambda)]$$

In practice, we sample $\varepsilon_j \sim \hat{q}(\varepsilon)$ and the sampled gradient is just

$$\nabla_{\hat{\lambda}} \mathscr{L}(\hat{\lambda}) = \frac{1}{S} \sum_{s=1}^{S} G(\varepsilon_s)$$

Then we use Stochastic Gradient Ascent (SGA), which is the ascending version of the same algorithm that we use in Neural Network backpropagation, and we just iterate these two steps, Monte Carlo approximation of the gradient and SGA up until convergence.

The caveat is being able to perform the reparametrization trick, which is easy for Gaussians, for example, and less easy for other distributions.

Example. If we consider the parametric variational distribution to be a univariate Gaussian distribution, we can write it as:

$$q(z|\lambda) = \mathcal{N}(z|\mu, \sigma^2)$$

Now we would like to express $q(z|\lambda)$ by a distribution $\hat{q}(\varepsilon)$ independent from λ combined with a function $g_\lambda(\varepsilon)$ that depends from λ. In particular, we consider:

$$
\begin{aligned}
z &= g_\nu(\varepsilon) = \mu + \sigma \cdot \varepsilon \\
\varepsilon &\sim \hat{q}(\varepsilon) = \mathcal{N}(\varepsilon|0,1)
\end{aligned}
$$

10.4.2 Non-reparameterizable $q(z|\lambda)$

The goal is still to rewrite the gradient of the expectation as the expectation of the gradient; the general case is more complicated but an expression can be nonetheless found. Let us start from

$$
\begin{aligned}
\nabla_\lambda \mathcal{L}(\lambda) &= \nabla_\lambda \mathbb{E}_{q(z|\lambda)}[\log p(x,z) - \log q(z|\lambda)] \\
&= \nabla_\lambda \int q(z|\lambda)[\log p(x,z) - \log q(z|\lambda)]dz
\end{aligned}
$$

Using the dominated convergence theorem to get the gradient inside the integral and then applying the product rule for derivatives we get

$$
\begin{aligned}
\nabla_\lambda \mathcal{L}(\lambda) &= \int \nabla_\lambda[\log p(x,z) - \log q(z|\lambda)]q(z|\lambda)dz \\
&+ \int \nabla_\lambda q(z|\lambda)[\log p(x,z) - \log q(z|\lambda)]dz
\end{aligned}
$$

Taking a look at the first term, we see that $\nabla_\lambda[\log p(x,z)] = 0$ and we can write it as

$$
\begin{aligned}
\int \nabla_\lambda[\log p(x,z) - \log q(z|\lambda)]q(z|\lambda)dz &= -\mathbb{E}_q[\nabla_\lambda \log q(z|\lambda)] \\
&= -\mathbb{E}_q\left[\frac{\nabla_\lambda q(z|\lambda)}{q(z|\lambda)}\right] = \int \frac{\nabla_\lambda q(z|\lambda)}{q(z|\lambda)} q(z|\lambda)dz \\
&= \int \nabla_\lambda q(z|\lambda)dz \\
&= \nabla_\lambda \int q(z|\lambda)dz = \nabla_\lambda 1 = 0
\end{aligned}
$$

We still need to workout the second term. Again we use the fact that

$$
\nabla_\lambda[\log q(z|\lambda)] = \frac{\nabla_\lambda q(z|\lambda)}{q(z|\lambda)}
$$

which we rephrase as

$$
\nabla_\lambda q(z|\lambda) = \nabla_\lambda[\log q(z|\lambda)]q(z|\lambda)
$$

Recapping

$$\begin{aligned}\nabla_\lambda \mathcal{L}(\lambda) &= \int q(z|\lambda)\nabla_\lambda \log q(z|\lambda)[\log p(x,z) - \log q(z|\lambda)]dz \\ &= \mathbb{E}_q[\nabla_\lambda \log q(z|\lambda)[\log p(x,z) - \log q(z|\lambda)]]\end{aligned}$$

Then we can sample $z_s \sim q(z|\lambda)$ and have an estimate of our gradient

$$\nabla_\lambda \mathcal{L}(\lambda) \approx \frac{1}{S}\sum_{s=1}^{S} \nabla_\lambda \log q(z_s|\lambda)[\log p(x,z_s) - \log q(z_s|\lambda)]$$

Again we have an estimate of the gradient and we can use SGA to find convergence. Unfortunately this estimates of the gradient has a very high variance, which slows down the convergence of our algorithm, so in the next steps we would see strategies on how to control this variance.

10.4.3 Rao-Blackwellization

The first technique to control the variance of the stochastic estimation of the gradient of the ELBO that we have defined in black box variational inference, is known as **Rao-Blackwellization**.

Consider x, y random variates and some function $J(x,y)$ of which we want to compute the expectation.

First let's define

$$\hat{J}(x) = \mathbb{E}_y[J(x,y)|x] \quad \text{hence } \mathbb{E}_x[\hat{J}(x)] = \mathbb{E}_{xy}[J(x,y)]$$

Crucially, we know, by properties of the conditional expectation, that

$$Var[\hat{J}(x)] = Var[J(x,y)] - \mathbb{E}[(J(x,y) - \hat{J}(x))^2] < Var[J(x,y)]$$

Now, let's consider a mean field factorization of $q(z|\lambda) = \prod_{i=1}^{N} q(z_i|\lambda_i)$. Since every z_i depends only on λ_i we are going to consider the gradient with respect to the single λ_i and then exploit this factorization to simplify the expression.

We need a couple more things:

- $q_{(i)}$: marginal of $q(z|\lambda)$ on the terms that form the Markov Blanket $z_{(i)}$ in p
- $p_i(x, z_{(i)})$ the product of factors of $p(x|z)$ depending on $z_{(i)}$

We are not going to perform the computation here (they are reported in the black-box variational inference paper), but we get that

$$\hat{\nabla}_{\lambda_i}[\mathcal{L}] := \mathbb{E}_{q(i)}[\nabla_{\lambda_i}\mathcal{L}(z_i)] = \mathbb{E}_{q(i)}\left[\nabla_{\lambda_i} \log q(z_i|\lambda_i)[\log p_i(x, z_{(i)}) - \log q(z_i|\lambda_i)]\right]$$

So essentially now we are taking an expectation over a *smaller* set of variables. This is playing the role of $\hat{J}(x)$ recasted on our variational inference problem, hence the variance of the estimation is reduced with respect to the original formulation of the problem.

More specifically, we need to consider samples $z_s \sim q_{(i)}(z|\lambda)$ and this distribution is just the product of the factors belonging to the Markov blanket of z_i.

10.5 Control variates

Let's first introduce the general idea of control variates. Say we have a function f of which we want to know the expectation with respect a certain distribution q. Instead of directly computating the expectation of f, we will define a new function $\hat{f}$ such that $\mathbb{E}_q[\hat{f}] = \mathbb{E}_q[f]$ and $VAR_q[\hat{f}] < VAR_q[f]$, and compute $\mathbb{E}_q[\hat{f}]$. How do we build such a $\hat{f}$?

We choose a function h such that $\mathbb{E}[h] < \infty$ and define

$$\hat{f}_a(z) = f(z) - a(h(z) - \mathbb{E}[h(z)])$$

One can trivially see that indeed $\mathbb{E}_q[\hat{f}] = \mathbb{E}[f]$ and that

$$Var[\hat{f}] = Var[f] - 2aCov[f,h] + a^2 Var[h]$$

Additionally we have the freedom to choose a and we can fix it to the value $a^\star$ that minimizes the variance, which by deriving the expression before w.r.t a and setting it to zero turns out to be

$$a^\star = \frac{COV(f,h)}{VAR(h)}$$

So the larger the covariance, the larger the reduction in the estimation of the variance.

In our scenario, we start from Rao-Blackwellization

$$f_i(z) = \nabla_{\lambda_i} \log q(z_i|\lambda_i)[\log p_i(x, z_{(i)}) - \log(z_i|\lambda_i)]$$

and we will choose

$$h_i(z) = \nabla_{\lambda_i} \log q(z_i|\lambda_i)$$

since we have seen that $\mathbb{E}[h_i(z)] = 0$.

The optimal choice $a^\star$ for a is hard to compute, because we would need to know the covariance precisely, but we can estimate it as $\hat{a}^\star$ reusing the same samples that we used to estimate the gradient.

Then, our estimation of the gradient using control variates becomes

$$\hat{\nabla}_{\lambda_i} = \frac{1}{S}\sum_{s=1}^{S} \nabla_{\lambda_i} \log q(z_i|\lambda_i)[\log p_i(x, z_s) - \log q_i(z_s|\lambda_i) - \hat{a}_i^\star]$$

where $z_s \sim q_{(i)}(z|\lambda)$.

A scheme summarizing the different kinds and applications of Variational Inference is shown in Figure 10.5.

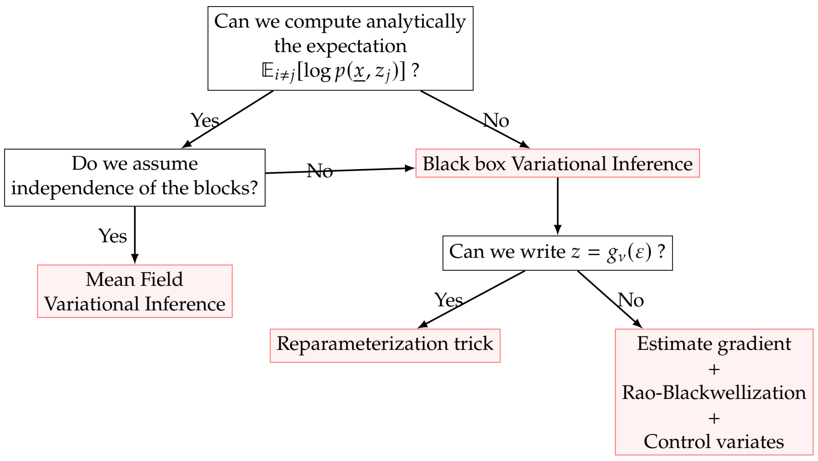

Can we compute analytically the expectation $\mathbb{E}_{i \neq j}[\log p(\underline{x}, z_j)]$?
Yes
No
Do we assume independence of the blocks?
No
Black box Variational Inference
Yes
Mean Field Variational Inference
Can we write $z = g_v(\varepsilon)$?
Yes
No
Reparameterization trick
Estimate gradient + Rao-Blackwellization + Control variates

Figure 10.5: A schematic decision diagram to identify the suitable Variational Inference scheme to use in the application at hand.

10.6 Bayesian Neural Networks

Looking at the world of deep learning, we will now try to understand how the methods we have just described can prove useful when applied to deep neural networks. These architectures are powerful function approximators which can be trained using gradient-based optimization. Sometimes this flexibility, which is their main strength, can become a weakness, e.g. if it leads to overfitting or when the data available is limited. Moreover, deep neural networks lack the uncertainty estimation on the output.

To overcome these issues we can examine neural networks under a Bayesian lens, which for its nature behaves in a probabilistic way. The model uncertainty should be introduced in the parameters $\boldsymbol{w}$. Imagine to train many times the network on the same dataset with a stochastic optimization technique: the parameters would probably be different each time, as the predictions. This procedure, used in Neural Networks ensembles, would lead to a measure of uncertainty on the weights which reflects on the uncertainty on the output, though heavily dependent on the training mechanism and on the initialization of model parameters. However, there is a simpler and more grounded way to introduce uncertainty, i.e. switching from point-wise weights to probability distributions.

This means placing a prior distribution $p(\boldsymbol{w}$ on weights and then learning the posterior distribution $p(\boldsymbol{w}|\bar{x}, \bar{y})$ with a suitable learning algorithm, using it to compute the predictive distribution:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \int_W p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w}) p(\boldsymbol{w}|\bar{x}, \bar{y}) d\boldsymbol{w}$$

where W is the space of the possible parameters.

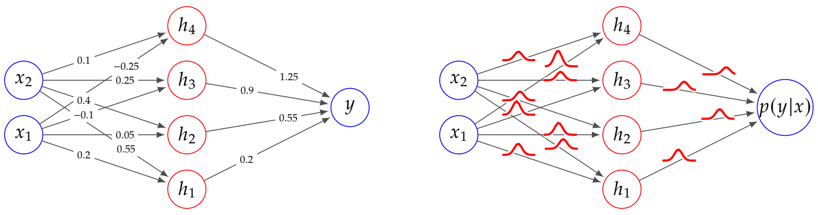


Figure 10.6: Neural network (left) VS Bayesian neural network (right). We consider that the BNN outputs a probability distribution, which is obtained by computing the predictive distribution

Remark. Looking at the predictive distribution, we can consider it as a form of Bayesian model averaging: the first term in the integral, i.e. the likelihood, corresponds to the forward pass through a neural network with a specific set of weights, multiplied by the posterior probability of that set of weights over all the possible weight values. This is equivalent to using an ensemble of an uncountably infinite number of neural networks.

Remark. The Bayesian approach allows us to regularize the learning by placing a suitable prior on the weights w: with a Gaussian prior we obtain a $L2$ regularization; with a Laplace prior a $L1$ regularization.

Computing directly $p(\boldsymbol{w}|\bar{x}, \bar{y})$, and thus the predictive distribution, is intractable as it requires learning a very high-dimensional distribution (the number of weights in a neural network is typically very large). So, we consider a variational distribution $q(\boldsymbol{w}|\theta)$ which approximates $p(\boldsymbol{w}|\bar{x}, \bar{y})$. At this point, we write the ELBO:

$$\begin{aligned}\mathscr{L}(\theta) &= \mathbb{E}_{q(\boldsymbol{w}|\theta)}[\log p(\boldsymbol{w}, \bar{x}, \bar{y}) - \log q(\boldsymbol{w}|\theta)] \\ &= \mathbb{E}_{q(\boldsymbol{w}|\theta)}[\log p(\bar{y}|\boldsymbol{w}, \bar{x}) + \log p(\boldsymbol{w}) - \log q(\boldsymbol{w}|\theta)]\end{aligned}$$

and use its opposite as the loss function of the neural network:

$$\text{Loss}(\theta) = \mathbb{E}_{q(\boldsymbol{w}|\theta)}[\log q(\boldsymbol{w}|\theta) - \log p(\bar{y}|\boldsymbol{w}, \bar{x}) - \log p(\boldsymbol{w})]$$

10.6.1 Bayes by Backprop

In order to optimize the loss function above, we need to compute the gradient with respect to the parameters θ. This is usually not tractable as we cannot interchange the gradient with the expectation, as they both act on the same parameters. So we use the reparametrization trick. In this way, we can perform the optimization by combining sampling with backpropagation, the milestone of deep learning. This algorithm is called Bayes by Backprop [8] .

We start by assuming that the variational posterior distribution is a Gaussian distribution with diagonal covariance matrix (but in principle we could use any reparameterizable distribution). We notice that we need to require $\boldsymbol{\sigma}$ to be non-negative and so we parametrize it as:

$$\boldsymbol{\sigma} = \log(1 + \exp(\boldsymbol{\rho}))$$

In this case,

$$\boldsymbol{\theta} = (\boldsymbol{\mu}, \boldsymbol{\rho})$$

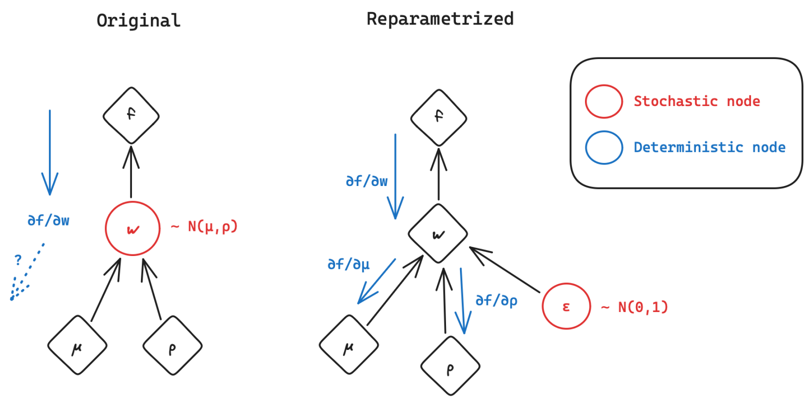
and we can reparametrize $\boldsymbol{w}$ as:

$$\boldsymbol{w} = \boldsymbol{\mu} + \log(1 + \exp(\boldsymbol{\rho})) \cdot \varepsilon$$

with $\varepsilon \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$.

In this way, we are now able to sample $\boldsymbol{w}$ by sampling from a simple distribution which does not depend on the parameters that we are optimizing. So, after taking the expectation, we are interested in minimizing

$$\mathbb{E}_\varepsilon[f(\boldsymbol{w},\boldsymbol{\theta})] = \mathbb{E}_\varepsilon[\log p(\bar{y}|\bar{x},\boldsymbol{w}) + \log p(\boldsymbol{w}) - \log q(\boldsymbol{w}|\theta)]$$



We can then perform the optimization step by step by repeating the following procedure:

1. Sample $\varepsilon \sim \mathcal{N}(\mathbf{0},\boldsymbol{I})$
2. Compute the weights as $\boldsymbol{w} = \boldsymbol{\mu} + \log(1+\exp(\boldsymbol{\rho}))\cdot\varepsilon$
3. Execute the forward pass
4. Compute the unbiased Monte Carlo gradients with respect to the parameters and calculate the update steps (backpropagation):

$$\begin{aligned}\Delta_\mu &= \nabla_w f(\boldsymbol{w},\boldsymbol{\theta})\cdot\nabla_\mu \boldsymbol{w} + \nabla_\mu f(\boldsymbol{w},\boldsymbol{\theta})\\ &= \nabla_w f(\boldsymbol{w},\boldsymbol{\theta}) + \nabla_\mu f(\boldsymbol{w},\boldsymbol{\theta})\\ \Delta_\rho &= \nabla_w f(\boldsymbol{w},\boldsymbol{\theta})\cdot\nabla_\rho \boldsymbol{w} + \nabla_\rho f(\boldsymbol{w},\boldsymbol{\theta})\\ &= \nabla_w f(\boldsymbol{w},\boldsymbol{\theta})\frac{\varepsilon}{1+\exp(-\boldsymbol{\rho})} + \nabla_\rho f(\boldsymbol{w},\boldsymbol{\theta})\end{aligned}$$

5. Update the variational parameters:

$$\begin{aligned}\boldsymbol{\mu} &\leftarrow \boldsymbol{\mu} - \alpha\Delta_\mu\\ \boldsymbol{\rho} &\leftarrow \boldsymbol{\rho} - \alpha\Delta_\sigma\end{aligned}$$

   where α is the learning rate.

Remark. Notice that $\nabla_w f(\boldsymbol{w},\boldsymbol{\theta})$, which is present in both Δ_μ and Δ_ρ is the usual gradient found by backpropagation: Bayes by Backprop just scales and shifts it.

Remark. As usual in deep learning, minibatches are used in the training process. In this case, the KL cost has to be re-weighted in a proper way.