Variational Inference and Application in Bayes

Xianbin Wang and Chenran Wang

1 Introduction

Bayesian inference is a method of statistical inference based on Bayes' rule, where we assume a prior distribution for parameter according to our belief, then have a posterior distribution for parameter. Occasionally, if we consider a conjugate prior for the parameter, the posterior distribution is easy to work with. Most of the time, we can't have a closed form for the posterior distribution, resulting in that we can't have a closed form for the Bayesian estimator. Under this circumstance, an approximation is essential for Bayesian inference. Laplace approximation [4], MCMC[1] are generated for approximation, both of the two methods generate a Markov chain sequence to approximate samples from the posterior distribution. However, it requires a lot for computation ability. Last few years, with the development of neural network and deep learning, some new ideas are generated for distribution estimation. By applying variational inference, we have a new idea for sampling from the posterior distribution using GAN [2] (Generative Adversarial Network).

2 Bayesian Inference

Bayes' rule or Bayes' theorem takes the form

$$f(\theta|y) = \frac{f(y|\theta)f(\theta)}{f(y)} \propto f(\theta)f(y|\theta),$$

where θ is a random variable with prior density function $f(\theta)$, $f(\theta)$ encodes our prior beliefs about the likely values of θ, y is a random variable with the model $\{f_\theta : \theta \in \mathbf{\Theta}\}$. Usually, θ represents the parameter and y is the observed data. We want to have a posterior distribution

for θ with the information of observations. In addition the Bayes estimator is

$$\mathbf{E}(\theta|y) = \frac{\int \theta f(y|\theta) f(\theta) d\theta}{\int f(y|\theta) f(\theta) dy}.$$

Usually, we can't have a closed form for the numerator and denominator, except that we can have a conjugate prior for θ.

Laplace approximation is a tool for posterior distribution estimation. We have the Taylor expansion for the log likelihood of the posterior density

$$\log f(\theta|y) \approx \log f(\hat{\theta}|y) - \frac{1}{2}(\theta - \hat{\theta})^T \mathcal{I}(\hat{\theta})(\theta - \hat{\theta})$$

where $\hat{\theta} = \arg\max f(\theta|y)$, $\mathcal{I}(\theta) = -\frac{\partial^2}{\partial\theta\partial\theta^T}\log f(\theta|y)$. Hence, it equals to assume the posterior distribution from a Gaussian distribution with mean $\hat{\theta}$ and covariance $\mathcal{I}(\hat{\theta})^{-1}$.

Markov chain Monte Carlo are the methods to construct a sequence of θ by iteratively sampling

$$\theta_m \sim Q(\theta|\theta_{m-1})$$

where Q is the transition density function. The sequence $\theta_1, \cdots, \theta_m, \cdots$ can be seen as the sample from the posterior distribution.

3 Variational Inference

Variational Inference is used to approximate intractable distribution. We consider a model, $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n)$. $\mathbf{x}_i$ is i.i.d generated from a random process, involving a unobserved random variable $\mathbf{Z}$, where $\mathbf{Z}$ can be seen as the latent variable. $\mathbf{Z}$ is from a distribution $f_\theta(\mathbf{Z})$, with parameter θ. $\mathbf{x}_i$ is generated from the distribution $f(\mathbf{x}|\mathbf{Z}_i)$. Most of the time, the hidden structure is intractable and we don't have any model assumption for $f_\theta(\mathbf{Z})$ and $f(\mathbf{x}|\mathbf{Z})$, and we even don't know what exactly $\mathbf{Z}$ is. Under this occasion, we can use variational inference to extract features, do dimension reduction.

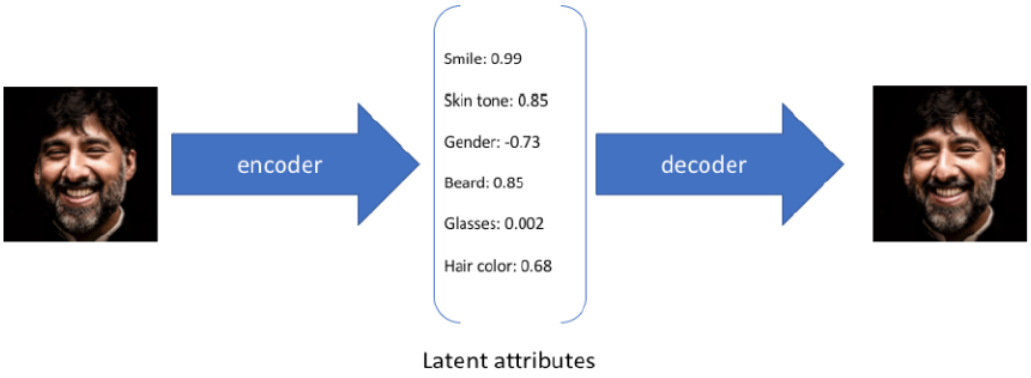

Figure 3.1: A example for intractable hidden structure

We can have a guess for latent variables, but we don't make any assumption what the latent variables are before training the model. This is the case when we use VAE [3](Variational Autoencoder) to do image analysis.

We can also have the model for $\mathbf{Z}$ and $\mathbf{x}$ before training. The detail may be quite different from VAE. But, the goal is the same, we want to train a function $q(\mathbf{Z})$ to approximate an intractable function $f_\theta(\mathbf{Z}|x)$.

The goal is to minimize the KL divergence between $q_\phi(\mathbf{Z})$ and $f_\theta(\mathbf{Z}|x)$, $D_{KL}[q_\phi(\mathbf{Z})||f_\theta(\mathbf{Z}|\mathbf{x})]$,

$$\begin{aligned} D_{KL}[q_\phi(\mathbf{Z})||f_\theta(\mathbf{Z}|\mathbf{x})] &= \mathbf{E}_{\mathbf{Z}\sim q_\phi}\big(\log\frac{q_\phi(\mathbf{Z})}{f_\theta(\mathbf{Z}|\mathbf{x})}\big) \\ &= \mathbf{E}_{\mathbf{Z}\sim q_\phi}\big(\log q_\phi(\mathbf{Z}) - \log f_\theta(\mathbf{Z}|\mathbf{x})\big) \\ &= \mathbf{E}_{\mathbf{Z}\sim q_\phi}\big(\log q_\phi(\mathbf{Z}) - \log f_\theta(\mathbf{x}|\mathbf{Z}) - \log f_\theta(\mathbf{Z}) + \log f_\theta(\mathbf{x})\big) \\ &= \mathbf{E}_{\mathbf{Z}\sim q_\phi}\big(\log\frac{q_\phi(\mathbf{Z})}{f_\theta(\mathbf{Z})}\big) - \mathbf{E}_{\mathbf{Z}\sim q_\phi}(\log f_\theta(\mathbf{x}|\mathbf{Z})) + \log f_\theta(\mathbf{x}). \end{aligned} \tag{3.1}$$

Given $\mathbf{x}$, $\log f_\theta(\mathbf{x})$ is a constant, so minimizing the $D_{KL}[q_\phi(\mathbf{Z})||f_\theta(\mathbf{Z}|\mathbf{x})]$ equals to minimizing $\mathbf{E}_{\mathbf{Z}\sim q_\phi}\big(\log\frac{q_\phi(\mathbf{Z})}{f_\theta(\mathbf{Z})}\big) - \mathbf{E}_{\mathbf{Z}\sim q_\phi}(\log f_\theta(\mathbf{x}|\mathbf{Z})) = D_{KL}(q_\phi(\mathbf{Z})||f_\theta(\mathbf{Z})) - \mathbf{E}_{\mathbf{Z}\sim q_\phi}(\log f_\theta(\mathbf{x}))$.

4 Generative Adversarial Network

GAN, generative adversarial network, includes two networks, generator and discriminator. Generator is a network to generate what we want, where the input of a generator is noise.

Discriminator is used to distinguish candidates produced by generator from the true distribution. Here, we want the generator to generate $\mathbf{Z}$, where $\mathbf{Z} \sim q_\phi(\mathbf{Z})$, and train a generator in order to make $q_\phi(\mathbf{Z})$ close to $f_\theta(\mathbf{Z}|\mathbf{x})$. If we use $\mathbf{G}()$ to represent the generator and use $\mathbf{D}()$ to represent the discriminator, and $\mathbf{e}$ is the noise input for the generator. The goal is to make $\mathbf{G}(\mathbf{e})$ approximate $\mathbf{Z}$ from $f_\theta(\mathbf{Z}|\mathbf{x})$. We want to minimize

$$\mathbf{E}_{\mathbf{Z}\sim\mathbf{G}(\mathbf{e})}(\log\frac{q_\phi(\mathbf{Z})}{f_\theta(\mathbf{Z})}) - \mathbf{E}_{\mathbf{Z}\sim\mathbf{G}(\mathbf{e})}(\log f_\theta(\mathbf{x}|\mathbf{Z})). \tag{4.1}$$

(4.1) is the loss function for training generator. For a Bayesian problem, we want to use $\mathbf{G}(\mathbf{e})$ to approximate the sample from the posterior distribution $f_\theta(\mathbf{Z}|\mathbf{x})$, just like what Gibbs Sampler and MCMC do. Most of the time, if the prior isn't a conjugate function, we can't generate the sample easily from Gibbs sampler. But this isn't a problem for neural network. Given a good loss function, a neural network will train the generator to generate any function.

$\mathbf{Z}$ is the generator output with density function $q_\phi(\mathbf{Z})$, to approximate sample from $f_\theta(\mathbf{Z}|\mathbf{x})$. For this problem, we already know the distribution $f_\theta(\mathbf{x}|\mathbf{Z})$. So, we just need to estimate $\mathbf{E}_{\mathbf{Z}\sim\mathbf{G}(\mathbf{e})}(\log\frac{q_\phi(\mathbf{Z})}{f_\theta(\mathbf{Z})})$, the problem now is to estimate $\log\frac{q_\phi(\mathbf{Z})}{f_\theta(\mathbf{Z})}$, where $\mathbf{Z}$ is a output of the generator.

In GAN, with the loss function for discriminator,

$$\mathbf{E}_{\mathbf{Z}\sim\mathbf{G}(\mathbf{e})}\log\mathbf{D}(\mathbf{Z}) + \mathbf{E}_{\mathbf{Z}\sim f_\theta(\mathbf{Z})}\log(1-\mathbf{D}(\mathbf{Z})). \tag{4.2}$$

The output $\mathbf{D}(\mathbf{Z})$ is a estimator of the probability that $\mathbf{Z}$ is from $q_\phi(\mathbf{Z})$, which theoretically is $\frac{q_\phi(\mathbf{Z})}{q_\phi(\mathbf{Z})+f_\theta(\mathbf{Z})}$, if $\mathbf{Z} \sim \frac{1}{2}q_\phi(\mathbf{Z}) + \frac{1}{2}f_\theta(\mathbf{Z})$. As a result, $\log\frac{\mathbf{D}(\mathbf{Z})}{1-\mathbf{D}(\mathbf{Z})}$ is a estimator for $\log\frac{q_\phi(\mathbf{Z})}{f_\theta(\mathbf{Z})}$, where $\mathbf{Z} \sim \mathbf{G}(\mathbf{e})$ and $\mathbf{D}()$ is the discriminator,.

Therefore, we have a estimator for the objective function 4.1. Take $-\mathbf{E}_{\mathbf{Z}\sim\mathbf{G}(\mathbf{e})}\log\mathbf{D}(\mathbf{Z}) - \mathbf{E}_{\mathbf{Z}\sim f_\theta(\mathbf{Z})}\log(1-\mathbf{D}(\mathbf{Z}))$ as the loss function for the discriminator and $\mathbf{E}_{\mathbf{Z}\sim\mathbf{G}(\mathbf{e})}(\frac{\mathbf{D}(\mathbf{Z})}{1-\mathbf{D}(\mathbf{Z})}) - \mathbf{E}_{\mathbf{Z}\sim\mathbf{G}(\mathbf{e})}(\log f_\theta(\mathbf{x}|\mathbf{Z}))$ for the generator. $\mathbf{G}(\mathbf{e})$ is a estimator for $f_\theta(\mathbf{Z}|\mathbf{x})$.

5 Simulation

In this section, we do some simulations to check whether the method works. The first scenario, we consider the a logistic model,

$$y_i \sim \text{Bernoulli}(p_i),$$

$$p_i = \frac{1}{1 + e^{-x_i^T \beta}},$$

where $i = 1, 2, 3$, $x_i \in \mathbb{R}^{2\times 1}$. The dimension and sampler number is relatively small in order to have a better visualization.

In this simulation, $x_1 = (1, 1.5)^T, x_2 = (1, -1.5)^T, x_3 = (1, -0.5)^T$, where the first dimension represents the intercept. $y_1 = 1, y_2 = 1, y_3 = 1$. β has a prior, Normal$(0, 2\mathbf{I}_{2\times 2})$. We make a comparison with true posterior density function.

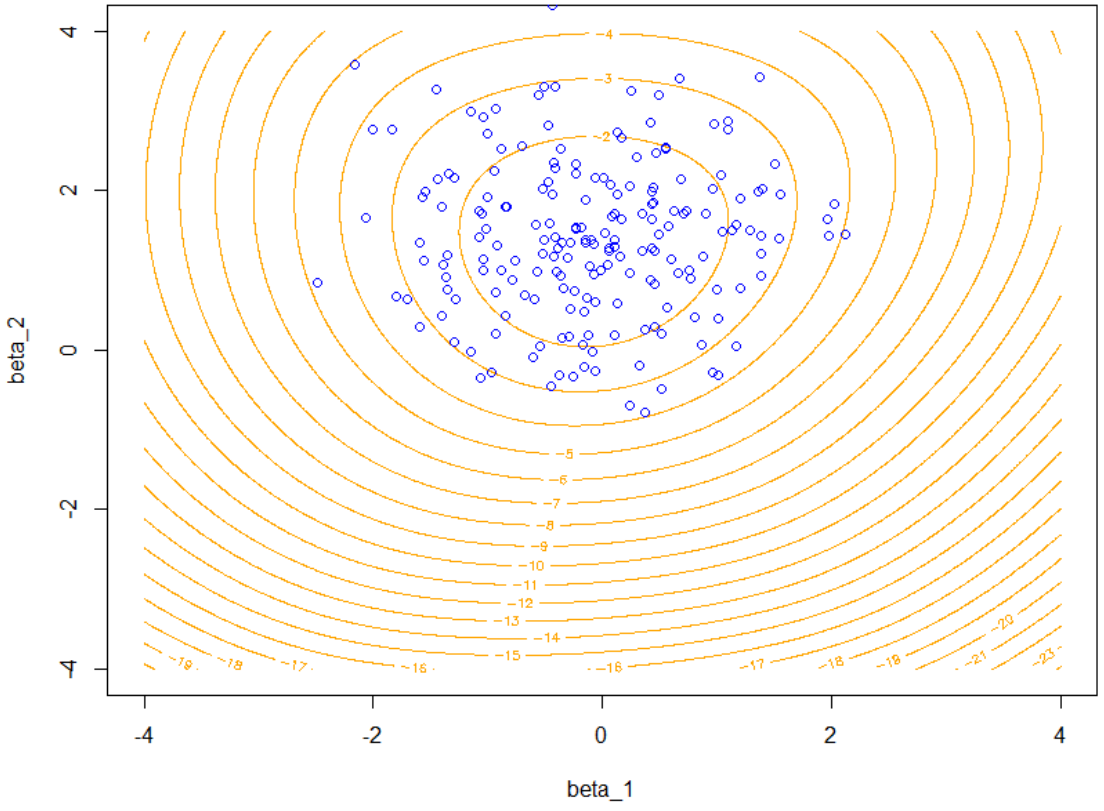

Figure 5.1: Sample from generator and true posterior density function contour plot

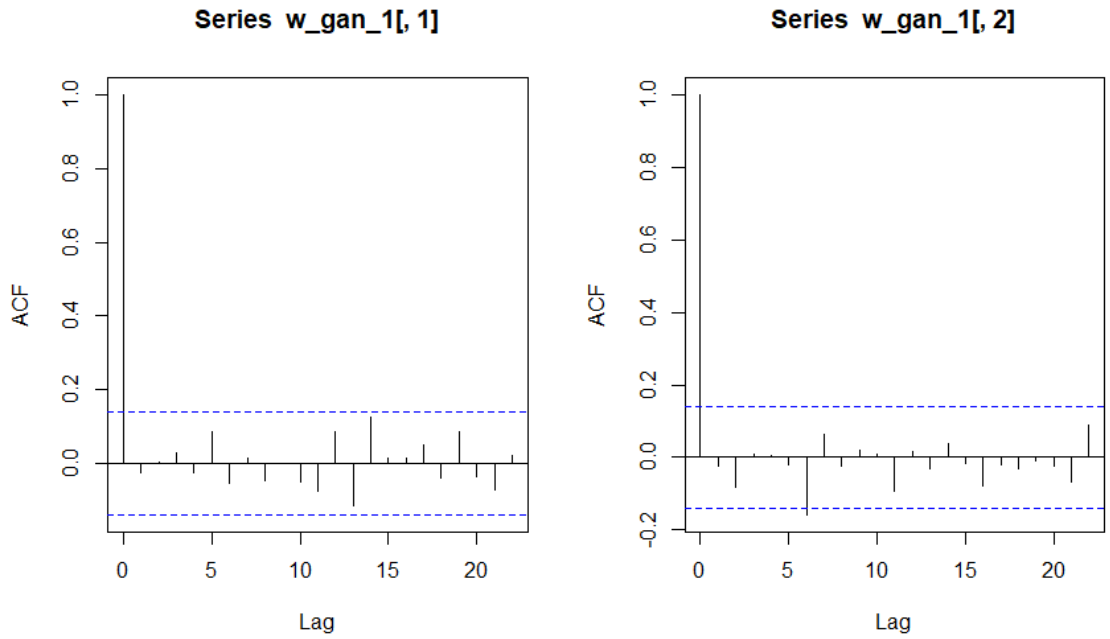

Figure 5.2: ACF plot for β_1 and β_2

The orange contour plot is the contour plot for true log posterior density function, the blue points is generated from generator. We can see that the performance of generator is good.

Another simulation is a linear regression model,

$$y_i = x_i^T \beta + \varepsilon,$$

where $y_i \in \mathbb{R}$, $x_i \in \mathbb{R}^{2\times 1}$, $\beta \in \mathbb{R}^{2\times 1}$, $\varepsilon \sim$ Normal$(0, \frac{1}{2})$. $y_1 = 0.5$, $y_2 = -1$, $y_3 = 1$, $x_1 = (0.5, 1)^T$, $x_2 = (-0.5, 1)^T$, $x_3 = (1, 1)^T$. β has a prior, Normal$(0, 2\mathbf{I}_{2\times 2})$.

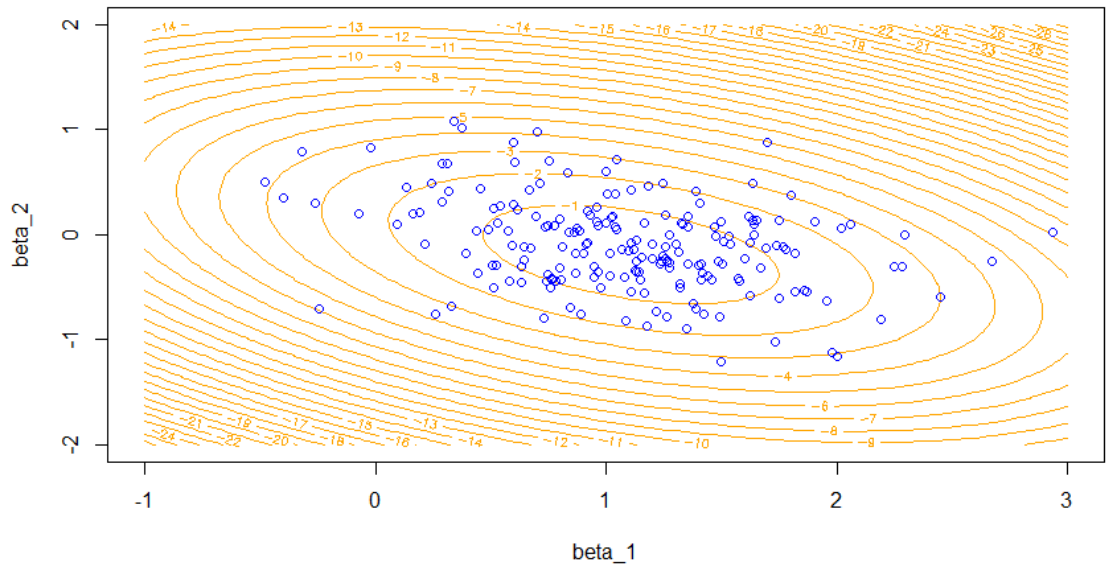

Figure 5.3: Sample from generator and true posterior density function contour plot

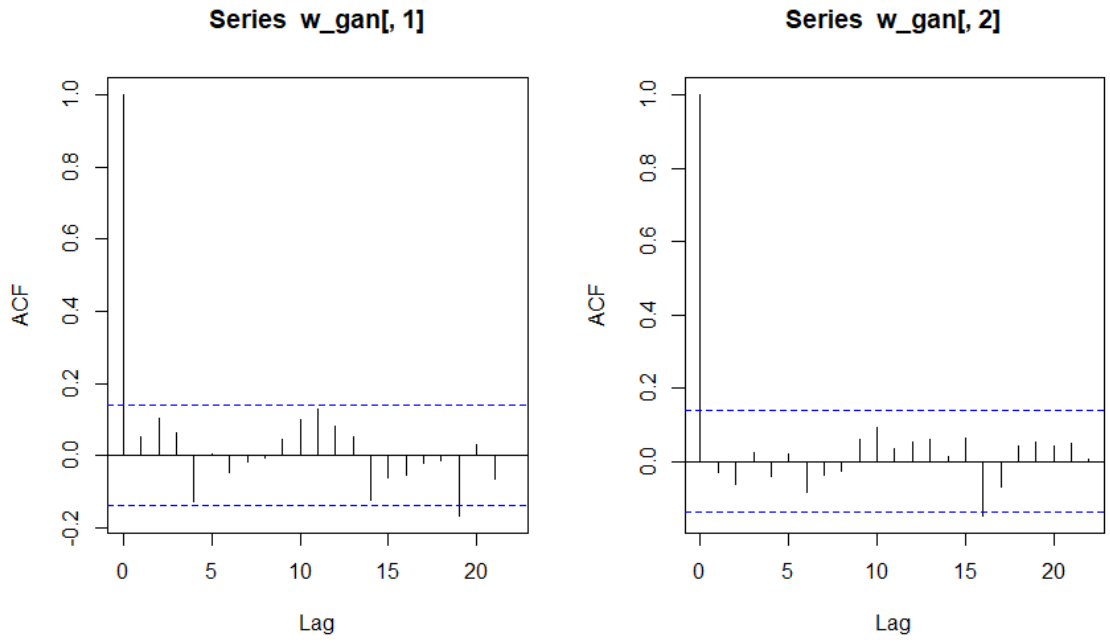

Figure 5.4: ACF plot for β_1 and β_2

Figure 5.1 and 5.3 shows that method works well in sampling. And the posterior samples are independent because of the independent inputs, which is better than MCMC. The auto correlation plot proves that.

6 Discussion

In the simulation section, we have two applications in posterior distribution estimation. Both of the two examples show that the method works well, and has an advantage over MCMC. However, we don't apply this method to a more intractable hierarchical model with a high dimension latent variable. So, the performance in high dimension situation needs to be checked in the future. In addition, another popular topic in variational inference is VAE (variational autoencoder). Different from our model, we don't know the exact meaning of the latent variable in VAE. An application of VAE is image compression. With already trained encode and decoder, we can compress a series image into a relatively low dimension. And another difference from our model is the randomness of the latent variable, that is, with the same input of the encoder, we can get different latent variables. Actually, the encoder just trains the parameter for distribution, and the latent variable is randomly generated by the distribution with rained parameter.

References

[1] Stephen Brooks. Markov chain monte carlo method and its application. *Journal of the royal statistical society: series D (the Statistician)*, 47(1):69–100, 1998.

[2] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 2672–2680, Cambridge, MA, USA, 2014. MIT Press.

[3] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[4] Håvard Rue, Sara Martino, and Nicolas Chopin. Approximate bayesian inference for latent gaussian models by using integrated nested laplace approximations. *Journal of the royal statistical society: Series b (statistical methodology)*, 71(2):319–392, 2009.