Stochastic Variational Inference - End to end

Nishant Padmanabhan

December 2025

1 Abstract

Stochastic variational inference is an algorithm used to approximate large and complex distributions by distributions belonging to a simpler variational family; i.e, a mixture of gaussians or any other common exponential family. In this article, I attempt to show three things: one, the full end to end derivation of natural gradient stochastic variational inference, the full list of assumptions used in the algorithm to make the calculations more doable and lastly, potential directions of attack in improving the algorithm.

Contents

2 Why Stochastic Variational Inference (SVI)?

Consider a dataset $D_{x,y} = \{x_1 : y_1, x_2 : y_2 \dots x_n : y_n\}$ and the set of latent random variables $\{z_1, \dots, z_m\}$. The relationship is defined by:

$$p(z|D) = \frac{P(D|z) \cdot p_0(z)}{P(D)} \tag{1}$$

In this context, $P(D|z)$ represents the likelihood, $p_0(z)$ is the prior over parameters, and $P(D)$ is the evidence. Technically, in supervised learning, the likelihood is expressed as $p(y|x, z)$, but using $p(D|z)$ is easier for now.

The Challenge of Intractability

Finding $p(z|D)$ directly is not easy. Consider an example involving a mixture of K unit-variance ($\sigma^2 = 1$) Gaussians with means $\mu = \{\mu_1, \mu_2 \dots \mu_k\}$.

- A common prior is used where $\mu_k \sim N(0, \sigma^2)$.
- Assignment variables follow a categorical distribution: $c_i \sim \text{categorical}(\frac{1}{K}, \dots, \frac{1}{K})$.
- Observations are distributed as $x_i|c_i, \mu \sim N(c_i^T \mu, 4)$.

The posterior $p(z|x) = p(\mu, c|x)$ is given by:

$$p(\mu, c|x) = \frac{P(x|\mu, c) \cdot p(\mu, c)}{P(x)} = \frac{P(\mu, c, x)}{P(x)} \tag{2}$$

To find $P(x)$, we must iterate over latent variables μ and c:

$$P(x) = \int_\mu \int_c p(\mu, c, x) dc\, d\mu = \int_\mu p(\mu) \left[\prod_{i=1}^{n} \sum_{c_i} p(c_i) P(x_i|c_i, \mu) \right] d\mu \tag{3}$$

The cost of computing $P(x)$ this way is $O(K^n)$, which is intractable or too expensive.

Approximation and Variational Inference

Our only choice is to approximate the distribution. There are two common choices:

1. **MCMC (Monte Carlo Markov Chain):** Considered the gold standard, but often takes a lot of time to run.
2. **Variational Inference (VI):** A much quicker algorithm, but not as accurate.

Our goal is to improve VI - specifically using auto-differential natural gradient stochastic methods so its accuracy approaches that of MCMC at a fraction of the training time.

VI posits a distribution $q(z)$ from a family of distributions $\mathcal{L}$, which is used to approximate $p(z|x)$. Mathematically, our goal is to find an optimum $q^*(z)$ as close to $p(z|x)$ as possible, or in other words:

$$q^*(z) = \arg\min_{q(z)\in\mathcal{L}} KL(q(z)||p(z|x)) \tag{4}$$

3 Detailed Derivation

The ELBO Derivation

Given training sets consisting of pairs (x, y). We wish to minimize a loss function capturing disagreement between the model and data:

$$KL(Q_{x,y}||P_{x,y}(\theta)) = \int q(x,y) \log \frac{q(x,y)}{p(x,y|\theta)} dxdy \tag{5}$$

$$KL(Q_{x,y}||P_{x,y}(\theta)) = \int q(x,y) \log \frac{q(y|x)q(x)}{p(y|x,\theta)p(x)} dxdy \tag{6}$$

For latent variables z and the distribution $p(z|x)$:

$$\begin{aligned} D_{KL}(q(z)||p(z|x)) &= \int q(z) \log \frac{q(z)}{p(z|x)} dz && (7)\\ &= E_q[\log q(z)] - E_q[\log p(z|x)] && (8)\\ &= E_q[\log q(z)] - E_q[\log p(x,z)] + E_q[\log p(x)] && (9)\\ &= E_q[\log q(z)] - E_q[\log p(x,z)] + log(p(x)) && (10) \end{aligned}$$

$\log p(x)$ is a constant w.r.t q, and hence can be taken outside the expectation as a constant. We define the Evidence Lower Bound ($ELBO$) as:

$$ELBO = E_q[\log p(x,z)] - E_q[\log q(z)] \tag{11}$$

$$\begin{aligned} \log p(x) &= ELBO(q) + D_{KL}(q(z)||p(z|x)) && (12)\\ &\geq ELBO(q). && (13) \end{aligned}$$

This $\log p(x)$ is a constant, therefore:

$$\arg \min_{q(z)\in\mathcal{L}} D_{KL}(q(z)||p(z|x)) = \arg \max_{q(z)\in\mathcal{L}} ELBO(q) \tag{14}$$

Usually we use the variational Mean Field approximation to represent $q(z)$. For a Bayesian mixture of Gaussians, we assume :

$$q(\mu, c) = \prod_{i=1}^{k} q(\mu_k; \eta_k, s_k) \prod_{j=1}^{n} q(c_i; \phi_i) \tag{15}$$

Exponential Family and Natural Gradients

In our derivation, we focus on natural gradients. We assume a minimal exponential family :

$$q(\theta|\eta) = h(\theta) \exp[\langle \eta, \phi(\theta) \rangle - A(\eta)] \tag{16}$$

Where:

- $\phi(\theta) \rightarrow$ vector of sufficient statistics
- $\langle \cdot, \cdot \rangle \rightarrow$ inner product
- $A(\eta) \rightarrow$ log partition function
- $h(\theta) \rightarrow$ scaling constant (assume=1)

Since we assume a minimal exponential family, he mean parameter $m = E_{q_\eta(\theta)}[\phi(\theta)]$ has a one-to-one mapping with η:

$$m = \nabla_\eta A(\eta) \tag{17}$$

The ELBO objective $\mathcal{L}(\eta)$ consists of the expected log likelihood and the KL term:

$$\mathcal{L}_{\mathcal{ELBO}}(\eta) = Eq_\eta(\theta)[\log p(D|\theta)] + E_{q_\eta(\theta)}[\log p_0(\theta)] - E_{q_\eta(\theta)}[\log q_\eta(\theta)] \tag{18}$$

For Natural Gradient updates, imagine the $ELBO$loss as a loss function $\mathcal{L}(\eta)$ for the time being:

$$\eta_{t+1} = \eta_t + \beta_t F(\eta_t)^{-1} \nabla_\eta \mathcal{L}(\eta_t) \tag{19}$$

Where:

- $F(\eta_t) = E_q[\nabla_\eta \log q \nabla_\eta \log q^T]$ is the Fisher Information Matrix.
- β_t is the learning rate.

This follows the definition from Amari (1988) .

Gradient Transformations

Standard weight updates look like:

$$\omega_{t+1} = \omega_t - \mu_t \frac{\partial L}{\partial w} \tag{20}$$

. The natural gradient update looks like :

$$\eta_{t+1} = \eta_t + \beta_t F(\eta_t)^{-1} \nabla_\eta \mathcal{L}(\eta_t) \tag{21}$$

Using the chain rule: $\nabla_\eta \mathcal{L}(\eta) = \frac{\partial m}{\partial \eta} \nabla_m \mathcal{L}(m)$. Since we assume it is a minimal exponential family, $\frac{\partial m}{\partial \eta} = \nabla^2_{\eta\eta} A(\eta)$ which is also equal to $F(\eta)$. Thus:

$$\nabla_\eta \mathcal{L}(\eta) = F(\eta) \nabla_m \mathcal{L}(\eta) \tag{22}$$

The update simplifies to:

$$\eta_{t+1} = \eta_t + \beta_t \nabla_m \mathcal{L}(\eta) \tag{23}$$

.

Back to ELBO:

$$\nabla_m(KL \text{ Term}) = E_{q_\eta(\theta)}[\phi(\theta)(\eta_o - \eta) + A(\eta) + const] \tag{24}$$

This is because

$$\begin{aligned} KL &= E_{q_\eta}[\log \frac{q_\eta(\theta)}{p_0(\theta)}] \\ &= E_{q_\eta(\theta)}[log(\frac{h(\theta)\exp(\eta_o \cdot \phi(\theta) - A(\eta_o)}{h(\theta)\exp(\eta \cdot + c\phi(\theta) - A(\eta)})] \\ &= E_{q_\eta(\theta)}[\phi(\theta)(\eta_o - \eta) + A(\eta) + const.] \end{aligned}$$

. Using $m = E_q[\phi(\theta)]$ we can calculate $\nabla_m(KL \text{ Term})$

$$\begin{aligned} \nabla_m(KL \text{ Term}) &= E_{q_\eta(\theta)}[\phi(\theta)(\eta_o - \eta) + A(\eta) + const] \\ &= \nabla_m(m^T(\eta - \eta_0)) + \nabla_m A(\eta) \\ &= \eta - \eta_0 - (\nabla_m)\eta)^T + \nabla_m A(\eta) \\ &= \eta_o - \eta - F(\eta)^(- 1)m + F(\eta)^-1m \\ &= \eta - \eta_0 \end{aligned}$$

The fourth line is obtained using $\nabla_m(\cdot) = F(\eta)^-1\nabla_\eta(\cdot)$ from Equation 22 and the symmetry of the Fisher Information matrix.

VOGN Final Algorithm

Plugging this into the ELBO update:

$$\eta_{t+1} = \eta_t + \beta_t(\nabla_m E_{q_\eta}[\log p(D|\theta)] + (\eta_0 - \eta_t)) \tag{25}$$

$$= (1 - \beta_t)\eta_t + \beta_t(\eta_0 + \nabla_m E_{q_\eta}[\log p(D|\theta)]) \tag{26}$$

- $F_t = \nabla_m E_{q_\eta}[\log p(D|\theta)]$

Let us now move from natural parameters to consider a Gaussian approximating family

$$q_\eta(\theta) = N(\theta; \mu, \sigma)$$

$$\eta^1 = \sigma^-1, \eta^2 = -\frac{1}{2}\sigma^-1$$

$$m^1 = \mu, m^2 = \mu\mu^T + \sigma$$

. Let the prior be a zero mean gaussian :

Then, we can show :
$\nabla_{m^1} F_t = \nabla_\mu F_t - 2[\nabla_\sigma F_t]$
$\nabla_{m^2} F_t = \nabla_\sigma F_t$

Substituting these:

$$\sigma_{t+1}^{-1} = (1-\beta_t)\sigma_t^{-1} + \beta_t(\delta I - 2\nabla_\sigma F_+) \tag{27}$$

$$\mu_{t+1} = \mu_t + \beta_t \sigma_{t+1}(\nabla_\mu F_+ - \delta\mu_t) \tag{28}$$

Using Bonnet's and Price's theorems:

$$\nabla_\mu F_t = \nabla_\mu E[\log p] = E[\nabla_\theta \log p] = -E[g(\theta)] \tag{29}$$

$$\nabla_\sigma F_t = \nabla_\sigma E[\log p] = \frac{1}{2}E[\nabla^2_{\theta\theta} \log p] = -\frac{1}{2}E[H(\theta)] \tag{30}$$

Where g is the gradient and H is the Hessian of the negative log likelihood. Using Monte Carlo approximation $\theta_t \sim q(\theta) = \mathcal{N}(\theta; \mu_t, \sigma_t I)$ to approximate the expecation terms, we get Variational Online Newton (VON) :

$$\sigma_{t+1}^{-1} = (1-\beta_t)\sigma_t^{-1} + \beta_t(NH(\theta_t) + \delta I) \tag{31}$$

$$\mu_{t+1} = \mu_t - \beta_t \sigma_{t+1}(Ng(\theta_t) + \delta_t \mu_t) \tag{32}$$

Calculating the Hessian can still be very hard, so VOGN approximates $H(\theta_t)$ to simplify calculation, by employing the following approximations/methods :

1. Use stochastic minibatch M of size M.
2. Per-sample gradient $\hat{g}(\theta_t) = \frac{1}{M}\sum_{i\in M_t} g_i(\theta_t)$.
3. Reparameterize $S_t = \frac{(\sigma_t^{-1} - \delta I)}{N}$.
4. Mean field approximation: $\sigma_t = \text{diag}(s_t)$.
5. Gauss-Newton approximation: $H(\theta_t) \approx \frac{1}{M}\sum_{i\in M_t}(g_i(\theta_t)^2)$.
6. Separate learning rates for μ, S which hopefully leads to quicker convergence.

Final Algorithm :

$$S_{t+1} = S_t + \frac{\beta_t}{M}\sum_{i\in M_t}(g_i(\theta_t)^2) \tag{33}$$

$$\mu_{t+1} = \mu_t - \gamma_t \frac{\hat{g}(\theta_t) + \frac{\delta}{N}\mu_t}{S_{t+1} + \frac{\delta}{N}} \tag{34}$$

This can now be run on Neural Networks.

4 Assumptions employed

The number of approximations/assumptions made throughout the course of the derivation were surprisingly few, given a variational inference algorithm and gaussian approximating family. That being said, here are all the assumptions made throughout the course of the derivation :

1. **Minimal Exponential Family** : For us to obtain the natural gradient step in Equation 23 of the derivation, we used the assumption that the mean parameters $m = \mathbb{E}_{q_\eta(\theta)}[\phi(\theta)]$ has a one-to-one mapping with η, and that $m = \nabla_\eta \mathbb{A}(\eta)$.

2. **Expectation approximation :** In equations 29 and 30 from the derivation, we are faced with expectations that we aren't quite sure how to calculate, so we approximate using Monte Carlo methods (the accuracy of this is expected to improve with number of iterations)

3. **Mean-field approximation :** We use mean-field approximation to approximate S_T ($= \frac{\bar{\sigma_T} 1 - \delta \mathbb{I}}{\mathbb{N}}$) as $S_T = diag(s_T)$. There are other approximations commonly used too, like K-FAC from Zhang et al.(2018) or SLANG (Mishkin et al., 2018), but this is by far the most common.

4. **Hessian approximation :** Finally, the change between Variational Online Newton(VON) and Variatonal Online Gauss Newton (VOGN) was the approximation of the Hessian. In VOGN, we approximate the Hessian as :

$$H(\theta_t) = -\nabla^2_{\theta\theta} \log p(\mathbb{D}|\theta) \tag{35}$$

$$\approx \frac{1}{\mathbb{N}} \sum_{i \in \mathbb{D}} g_i(\theta_t) g_i(\theta_t)^T \tag{36}$$

   This has the property of being positively semi-definite and other nice properties, lending itself as a useful approximation.

This is a full list of all approximations, based on these I have a few ideas on lines of attack, listed in the next section.

5 Potential Lines of Improvement

5.1 Physics-based loss function

The current algorithm is based on an *ELBO*loss function, comprising of two parts : a KL term and a likelihood term. But what if the loss function contained a physics-based component too? Perhaps something like $\mathbb{L} = ELBO + g(\theta)$, where $g(\theta)$ is some equality that needs to be satisfied for the system to be physically feasible. The exact nature of this equality would depend on the type of problem to be focused on, but the general idea remains.

5.2 Expectation Approximation

As mentioned before, when we are unable to calculate the expectations in Equations 29 and 30 of the derivation, we are forced to turn to Monte Carlo sampling methods to approximate this expectation. What if we used other methods of approximation, maybe even VI again?

5.3 Mean Field approximation

This is potentially the hardest line of approach, because if we use the variational Mean Field approximation to represent $q(z)$ where we assume :

$$q(z) = \prod_{i=1}^{N} q(z_i) \tag{37}$$

not using this approximation increases computation cost from $\mathbb{O}(N)$ to $\mathbb{O}(N^2)$. Of course this might be worth it if this led to an increase in performance, but it isn't obvious how to tweak the algorithm and if this would actually lead to improvements; perhaps we could start by looking at the K-FAC or the SLANG approximation to get ideas.

5.4 Splitting conjugate and Non conjugate terms

Conjugate terms can be calculated precisely and efficiently by methods specifically designed for the purpose; it is only non conjugate terms that need to be approximated by gradient descent. Thus, a hybrid approach can be used. That is the idea of Khan and Lin(2017)'s paper *Conjugate-Computation Variational Inference : Converting Variational Inference in Non-Conjugate Models to Inferences in Conjugate Model*.

5.5 Using momentum

Apparent from gradient descent, one can use algorithms like Vadam, Vprop or a custom designed algorithm to integrate natural momentum. These algorithms integrate Monte Carlo sampling using weight perturbation.

5.6 GGN and damping

Generalized Gauss Newton matrices are said to be more stable than directly using the Hessian (this is also positive semi-definite). Also, damping is good for the stability of the natural gradient methods in practice. This damping looks like Tikhonov regularization, trust regions, adapting damping (Levenberg-Marquardt heuristic).

These methods are pretty rough as of now, I will have to read papers like the Bayesian Learning Rule (2023) by Khan and Rue and a few more recent papers to get a better idea, but these are some hunches(last one-two suggested by AI).