STAT 547: Bayesian Workflow

Charles C. Margossian
University of British Columbia
Winter 2026
https://charlesm93.github.io/stat_547/

DRAFT

6 Variational Inference

We have seen how approximate sampling techniques, such as Markov chain Monte Carlo and importance sampling, can help us construct Monte Carlo estimators with which we can learn summaries of the posterior.

Another paradigm is to find a tractable distribution which approximates the posterior. By "tractable", I mean that the distribution is easy to manipulate: for example, it is straightforward to compute the mean and variance, and to draw samples. Variational inference (VI) is a broad class of methods to find tractable approximations to the posterior.

The task of VI is to find the best distribution q inside a pre-specified family $\mathcal{Q}$ of tractable distributions to approximate the target distribution p,

$$q^* = \text{argmin}_{q \in \mathcal{Q}} D(p||q), \tag{1}$$

where D is a divergence between p and q. Hence VI turns Bayesian inference into an optimization problem.

The two fundamental tuning choices of VI are (i) the family $\mathcal{Q}$ of approximations and (ii) the objective function which we minimize. The trade-offs for $\mathcal{Q}$ are rather intuitive: a rich family of approximations leads to a better approximation but often complicates the optimization.

The choice of objective function is subtler: there exists many ways to compare distributions. By definition, a valid divergence must be such that

$$D(p||q) = 0 \text{ if and only } p = q, \text{ and otherwise } D(p||q) > 0.$$

But this constraint only somewhat restricts our options. There are many valid divergences and, if we have a restricted family of approximations, i.e., $p \notin \mathcal{Q}$, then different divergences produce different solutions. Hence, we must examine how well different solutions approximate summaries of the posterior. At the same time, certain divergences are much more difficult to optimize than others. For example, the *total variation distance*, which we studied for MCMC, is a valid divergence but cannot be minimized through standard optimization techniques.

Before getting into the details of how VI works, let's highlight two classical applications of VI:

- **Topics model [Blei et al., 2003].** VI is used to train a probabilistic model, with which 1.8 million New York times article were sorted into topics based on the words contained in each article ("bag of words" representation of the article). Once trained, the model determines itself the topics, the words in each topic, and ascribes a distribution of topics to each article, e.g., 80% international affairs, 10% finance, $\cdots$.
- **Variational autoencoder [Kingma and Welling, 2014, Rezende et al., 2014].** This algorithm—we will see how it breaks into a probabilistic model and an inference procedure—is used to compress high-dimensional images. The model parameterizes each high-dimensional image by a low-dimensional latent variable and the image can be reconstructed by learning a neural network (the "decoder") which takes in the low-dimensional representation and outputs the reconstructed image.

6.1 Gaussian Variational Inference

A classic off-the-shelf VI algorithm is Gaussian variational inference (G-VI). We will use G-VI as a concrete example to introduce several concepts.

Suppose we have a target $p(z)$ with $z \in \mathbb{R}^d$. In practice, the space of z, denoted $\mathcal{Z}$, may not be $\mathbb{R}^d$ but as long as this space is continuous, we can apply a transformation to map $\mathcal{Z}$ to $\mathbb{R}^d$. For example, if a variable z_i is constrained to be positive, we can apply a log transformation.

In G-VI, $\mathcal{Q}$ is the family of Gaussian distributions. Each member is fully defined by the *variational parameters* $\lambda = (\nu, \Psi)$, where $\nu \in \mathbb{R}^d$ is the mean and Ψ is the covariance matrix. Often times, Ψ is taken to be a diagonal matrix, primarily to reduce the number of variational parameters from $\mathcal{O}(d^2)$ to $\mathcal{O}(d)$. This approach is called the *mean-field approximation*, although I also like the more descriptive name, *factorized approximation*. The resulting algorithm is factorized Gaussian variational inference (FG-VI).

Eq. (1) can be now be written as an optimization problem over the space of variational parameters,

$$\lambda^* = \text{argmin}_\lambda D(p||q_\lambda). \tag{2}$$

The most common objective function in VI is the *reverse Kullback-Leibler divergence*,

$$\text{KL}(q||p) = \int (\log q(z) - \log p(z)) q(z) \text{d}z, \tag{3}$$

which compares densities over measurable sets. In your homework assignment, you showed that $\text{KL}(q||p)$ was a valid divergence, by showing that $\text{KL}(q||p) \geq 0$ and $\text{KL}(q||p) = 0$ if and only if $q = p$. Compared to other divergences, the reverse KL is relatively straightforward to analyze theoretically and to minimize.

Example: FG-VI applied to a multivariate Gaussian. Suppose p is a multivariate Gaussian target with mean μ and a non-diagonal covariance matrix Σ.

Then it can be shown analytically (as you did in your homework assignment) that the optimal variational parameters are

$$\nu = \mu \qquad \Psi_{ii} = 1/[\Sigma^{-1}]_{ii}. \tag{4}$$

In words, FG-VI recovers the mean and the marginal precisions of p. (Recall that the precision matrix is the inverted covariance matrix.) On the other hand, q misestimates other properties of p: for example, it underestimates the variance (as you showed in the homework) and the entropy.[1]

More generally, VI can recover certain properties of p even if $q \neq p$. The ability of VI to produce accurate estimators has notably been studied in asymptotic limits, where the number of observations $N \to \infty$ and where, under certain regularity conditions, the posterior becomes Gaussian per the *Bernstein-von Mises theorem* [e.g. Katsevich and Rigollet, 2024].

6.2 Variational inference in the presence of symmetry

Recent work has demonstrated VI's ability to recover properties of p when p and $\mathcal{Q}$ exhibit certain symmetries [Margossian and Saul, 2025]. Here, we'll examine how VI can recover the mean in the presence of even-symmetry.

Definition 1. *(Even- and odd-symmetry) We say a function f is even-symmetric about ν if for all $z \in \mathbb{R}^d$,*

$$f(z + \nu) = f(-z + \nu). \tag{5}$$

Similarly, we say a function f is odd-symmetric about ν if for all $z \in \mathbb{R}^d$,

$$f(z + \nu) = -f(-z + \nu). \tag{6}$$

Consider now a location family of distributions with location parameter ν and base distribution q_0, that is

$$q_\nu(z) = q_0(z - \nu), \tag{7}$$

and suppose that q_0 is even-symmetric. Examples of such distributions are the Gaussian, student-t and Laplace distributions.

Theorem 2. *Let $\mathcal{Q}$ be a location-family with even-symmetric based distribution q_0. Suppose p is even-symmetric about μ.*

Furthermore, suppose p and $\mathcal{Q}$ are such that the order of differentiation and integration can be changed when computing $\nabla_\nu KL(q_\nu||p)$.

Then $\nu = \mu$ is a stationary point of $KL(q_\nu||p)$.

Proof. We start from the definition of the KL-divergence, eq. (3), and operate a change

[1] This second result deserves to be nuanced: while the entropy is underestimated, FG-VI can still yield reasonable estimates of the entropy in certain limits [Margossian and Saul, 2023].

of variable $\zeta = z - \nu$,

$$\begin{aligned} \mathrm{KL}(q_\nu||p) &= \int (\log q_\nu(z) - \log p(z)) q_\nu(z) \mathrm{d}z \\ &= \int (\log q_0(\zeta) - \log p(\nu + \zeta)) q_0(\zeta) \mathrm{d}\zeta \\ &= -\mathcal{H}(q_0) - \log p(\nu + \zeta)) q_0(\zeta) \mathrm{d}\zeta, \end{aligned} \tag{8}$$

where $\mathcal{H}(q_0)$ is the entropy of q_0 and doesn't depend on the variational parameter ν. We now differentiate with respect ν and use the fact that we can change the order of integration and differentiation,

$$\begin{aligned} \nabla_\nu \mathrm{KL}(q_\nu||p) &= -\int \nabla_\nu \log p(\nu + \zeta) q_0(\zeta) \mathrm{d}\zeta, \\ &= -\int \nabla_\zeta \log p(\nu + \zeta) q_0(\zeta) \mathrm{d}\zeta, \end{aligned} \tag{9}$$

where in the second line we use the symmetry in the argument of ν and ζ to obtain a gradient with respect to ζ.

Now suppose $\nu = \mu$. Then $p(\nu + \zeta)$, taken as a function of ζ, is even-symmetric and its gradient $\nabla_\zeta p(\nu + \zeta)$ is odd-symmetric. Furthermore, we have by assumption that $q_0(\zeta)$ is even-symmetric. Hence, the integrand is the product of an odd-symmetric and an even-symmetric distribution, and is itself odd-symmetric. Therefore, the integral goes to 0. □

Under some additional assumptions (i.e. p is log-concave[2]), we can show that this stationary point is a unique minimizer.

If p has a finite first moment, then the point of symmetry corresponds to the mean, which is then recovered by ν^*. It's worth noting that the theorem allows for several misspecification. In particular:

- q may be factorized, even though p is not.
- q and p may have different tail behaviors.

An ongoing research endeavor is to understand how VI's symmetry-matching properties generalizes to other symmetries.

6.3 Stochastic optimization for variational inference

In a Bayesian context, the target distribution is the posterior $p(z \mid y)$ and is typically only known up to a normalizing constant. Then, the variational objective becomes,

$$\begin{aligned} \mathrm{KL}(q_\lambda||p) &= \int (\log q_\lambda(z) - \log p(z, y) + \log p(y)) q_\lambda(z) \mathrm{d}z \\ &= \int (\log q_\lambda(z) - \log p(z, y)) q_\lambda(z) \mathrm{d}z + \log p(y), \end{aligned} \tag{10}$$

[2]Although I'm working on relaxing this condition...

where the laster term, $\log p(y)$, can be ignored for the purposes of optimizing λ.

Often times, the VI optimization problem is reformulated as follows,

$$\lambda^* = \text{argmax}_\lambda \int (\log p(z, y) - \log q_\lambda(z)) q_\lambda(z) \mathrm{d}z, \tag{11}$$

with the maximized objective termed the *evidence lower bound* (ELBO). The name comes from the fact that $p(y)$ is sometimes called the *evidence* and

$$\begin{aligned} & 0 & \leq \text{KL}(q_\lambda || p) \\ \iff & 0 & \leq \log p(y) - \text{ELBO} \\ \iff & \text{ELBO} & \leq \log p(y). \end{aligned} \tag{12}$$

When doing full Bayesian inference, $p(y)$ does not typically play a role, however we will see some non-Bayesian applications where $p(y)$ plays a pivotal role.

The first challenge with maximizing the ELBO is that the integral cannot be solved analytically. Instead, the ELBO is approximated via Monte Carlo and using draws from q_λ (which, in theory, is easy to sample from),

$$\text{ELBO} \approx \frac{1}{B} \sum_{b=1}^{B} \log p(z^{(b)}, y) - \log q_\lambda(z^{(b)}) q_\lambda(z^{(b)}) \mathrm{d}z, \quad z^{(b)} \sim q_\lambda. \tag{13}$$

To do gradient-based optimization, we also need to evaluate the gradient of the target. Assuming the order of integration and differentiation can be changed,

$$\nabla_\lambda \text{ELBO} = \int \nabla_\lambda (\log p(z, y) - \log q_\lambda(z)) q_\lambda(z) \mathrm{d}z. \tag{14}$$

But constructing a Monte Carlo estimator of the gradient is less straightforward, because the samples $z \sim q_\lambda$ carry a somewhat hidden dependence on λ, which we need to differentiate through. (In the integral, this dependence is made explicit in the density term $q_\lambda(z)$).

This motivates the *reparameterization trick*, whereby the dependence on λ and the stochastic component in z are disentangled. In particular, we want a two-step sampling procedure of the form,

$$\begin{aligned} \zeta &\sim p(\zeta) \\ z &= f_\lambda(\zeta), \end{aligned} \tag{15}$$

with f an invertible function. For example, suppose $q_\lambda(z)$ is a univariate normal with mean μ and variant ψ, and $\lambda = (\mu, \psi)$. Then, a sample from q_λ can be obtained as,

$$\begin{aligned} \zeta &\sim \text{normal}(0, 1) \\ z &= \sqrt{\psi}\zeta + \mu. \end{aligned} \tag{16}$$

Then, applying standard rules for a change of variable with respect to a probability measure, eq. (14) is rewritten as an integral with respect to ζ,

$$\nabla_\lambda \text{ELBO} = \int \nabla_\lambda (\log p(f_\lambda(\zeta), y) - \log q_\lambda(f_\lambda(\zeta)) q(\zeta) \mathrm{d}\zeta. \tag{17}$$

Our Monte Carlo estimator for the gradient is then,

$$\nabla_\lambda \widehat{\text{ELBO}} = \frac{1}{B}\sum_{b=1}^{B} \nabla_\lambda (\log p(f_\lambda(\zeta^{(b)}), y) - \log q_\lambda(f_\lambda(\zeta^{(b)})), \quad \zeta^{(b)} \sim q(\zeta). \tag{18}$$

Optimization is then performed via gradient-descent. That is, starting from an initial guess λ_0, we iteratively update our parameter estimate by following the gradient,

$$\lambda_t = \lambda_{t-1} + \epsilon_t \nabla_\lambda \widehat{\text{ELBO}}, \tag{19}$$

until some convergence criterion is met; for example, we may require that $|\nabla_\lambda \widehat{\text{ELBO}}| \leq \delta$ for some tolerance δ or check that the (estimated) ELBO remains stable after several iterations.

A crucial question is how to choose the step sizes ϵ_t. There is a rich literature on the subject—going all the way back to Newton's method,[3] which sets ϵ_t to the inverted Hessian of the objective function $g(\lambda)$,

$$\epsilon_t = \nabla^2 g(\lambda). \tag{20}$$

In standard settings—i.e., when minimizing a convex objective functions—Newton's method enjoys a quadratic convergence rate. However the cost of evaluating and inverting a second-order derivative has made Newton's method unpopular in settings where λ is high-dimensional.

Another complication in our setting is that the gradient is not evaluated exactly but stochastically. This raises the question of whether gradient-descent can converge to a solution. Proofs of convergence can be obtained by requiring that:

(i) The gradient estimator is unbiased, as is the case when drawing exact samples from $q(\zeta)$.

Interestingly, this condition is met when $B = 1$, that is we use a single sample. In my experience, running $B \ll 1$ improves the stability and convergence rate of the algorithm, and what is more samples can be drawn in parallel.

(ii) The step sizes ϵ_t decay at a rate that is neither too slow nor too fast. This condition is captured by the (somewhat mystifying) Robbins-Monroe condition,

$$\sum_{t=1}^{\infty} \epsilon_t = \infty, \quad \sum_{t=1}^{\infty} \epsilon_t^2 < \infty. \tag{21}$$

Much of the recent literature on stochastic optimization concerns finding a good learning schedule $\{\epsilon_t\}$.

There exist a few off-the-shelf optimizers and the most popular ones are currently `Adam` and `LBFGS`. These methods enjoy high-performance implementations in libraries such as `JAX` and `PyTorch`, and provide a good starting point.

[3] Newton's method is accredited to Newton himself. The wikipedia page also points to a special case of the method dating back to the Babylonian in the 16th-19th century BC!

6.4 Example: Variational inference for the SIR model

`Stan` provides a G-VI algorithm, called *automatic differentiation variational inference* (ADVI) (which unfortunately is not a very descriptive name). The algorithm proceeds as follows:

1. Transform the parameter θ to an unconstrained variable $\tilde{\theta} \in \mathbb{R}^d$, using an invertible transformation f, with $\tilde{\theta} = f(\theta)$.
2. Approximate $p(\tilde{\theta} \mid y)$ by a Gaussian $q_\lambda(\tilde{\theta})$—the Gaussian can either have a diagonal covariance matrix, meaning the Gaussian is factorized ("mean-field") or have a dense covariance matrix ("full-rank"). The approximation is found by minimizing $\text{KL}(q_\lambda(\tilde{\theta})||p(\tilde{\theta} \mid y))$.
3. Draw $\tilde{\theta} \sim q_\lambda(\tilde{\theta})$ and transform back to the original space, $\theta = f^{-1}(\tilde{\theta})$.

Coding demo. We can run mean-field ADVI for the SIR influenza model. Specifically, we'll use the version of the model with a negative binomial likelihood. Since the joint distribution $p(\theta, y)$ remains unchanged, there is no need to revise the model.

In addition to approximating the posterior for the model parameters $\theta = (\gamma, \beta, \phi^{-1})$, we can also examine the posterior distribution of some derived quantities, specifically the recovery time T and the R_0 number. Table 1 shows the posterior summaries obtained with ADVI and is directly translated from the `summary()` function in `cmdStanR`.

	Mean	Median	SD	MAD	q_5	q_{95}
γ	0.537	0.536	0.0416	0.0420	0.472	0.610
β	1.75	1.75	0.0499	0.0500	1.68	1.84
ϕ^{-1}	0.148	0.129	0.0881	0.0702	0.0544	0.320
T	1.87	1.87	0.145	0.147	1.64	2.12
R_0	3.29	3.29	0.267	0.268	2.86	3.73

Table 1: *ADVI posterior summary statistics*

We compare these results to the output obtained with MCMC (Table 2).

	Mean	Median	SD	MAD	q_5	q_{95}	$\hat{R}$	ESS	ESS (tail)
γ	0.541	0.539	0.0450	0.0415	0.470	0.618	1.00	2771	2492
β	1.74	1.73	0.0537	0.0491	1.65	1.82	1.00	2298	2189
ϕ^{-1}	0.137	0.121	0.0744	0.0600	0.0520	0.276	1.00	2303	2363
T	1.86	1.85	0.155	0.143	1.62	2.13	1.00	2771	2492
R_0	3.23	3.21	0.276	0.250	2.82	3.71	1.00	2906	2324

Table 2: *MCMC posterior summary statistics*

For this particular example, the estimates of posterior summaries returned by MCMC and ADVI are in close agreement. The model is relatively simple—after all, it's only

2-dimensional—but it does involve a likelihood parameterized by an ODE. So it is a bit surprising that ADVI works so well, despite using a simple approximation.

Notice that the MCMC summary contains three additional columns for diagnostics to check that the posterior inference is reliable. Unfortunately, these diagnostics are MCMC specific and do not apply to VI.

`Stan`'s ADVI checks that the stochastic optimization converges. But unlike for MCMC, convergence of VI does not guarantee that we have accurate estimates of the posterior—merely that we found the "best" candidate $q^* \in \mathcal{Q}$ to approximate p. (And even then, convergence diagnostics for optimization are often fooled by local optimas.) If $\mathcal{Q}$ is too restrictive, then q^* may still be a poor approximation of p.

The lack of automated inference checks for VI is a gap in the workflow.

But checks exist. For example, we can use *Pareto-smoothed importance sampling* [Yao et al., 2018], as we did for leave-one-out cross-validation (Section 3). In VI, q^* is the proposed distribution and p is the target distribution. Recall that because p is only known up to a normalizing constant, we must compute self-normalized importance weights.

Even without validating the inference, we can still perform model criticism, e.g., posterior predictive checks and predictive scores on a validation set.

In many papers which use VI, you'll find that the authors only perform limited checks on the inference but always evaluate the performance of the trained model. We may then reasonably question whether the posterior is accurately estimated, at least not by MCMC standard, e.g., the kind of accuracy we would get with $\widehat{R} \leq 1.01$ and ESS ≥ 100, which corresponds to a negligible bias and standard deviation of $\sqrt{\text{Var} f(z)}/10$. Nonetheless, an imperfectly trained model can still produce useful insights and fulfill its scientific purpose.

Class discussion. Why is it important (or not important) to do inference check in Bayesian Workflow?

6.5 Example: Variational autoencoder

The variational autoencoder (VAE) is class of machine learning models/algorithms, which can be decoupled into a probabilistic model and an inference procedure.

6.5.1 Probabilistic model: latent variables and neural network

A *latent variable model* is a *hierarchical* model where each observation x_n has a corresponding latent variable z_n. The model also admits a "global" *hyperparameter* θ, which is common to all observations x_n. A Bayesian model can be specified by the joint distribution

$$p(x_{1:N}, z_{1:N}, \theta) = p(\theta)p(z_{1:N} \mid \theta) \prod_{n=1}^{N} p(x_{1:N} \mid z_{1:N}, \theta). \tag{22}$$

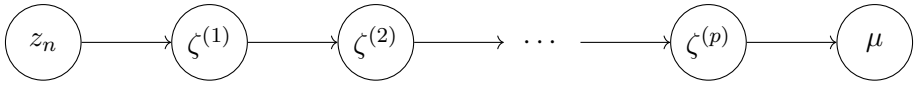

Figure 1: *A neural network is a sequence of transformations ("arrows") between layers ("nodes"). The first layer is the input layer, the final layer the output layer and all the layers in between are hidden layers. Each transformation is characterized by a linear transformation followed by a nonlinear transformation.*

In a frequentist setting, the hyperparameter θ is treated as fixed and the model is specified as a joint over $(z_{1:N}, x_{1:N})$ parameterized by θ, $p_\theta(x_{1:N}, z_{1:N})$.

In the canonical VAE, x_n is a high-dimensional image, e.g. three color channels for each pixel, and z_n is a low-dimensional representation of the image. θ is the parameter of a *decoder*, which stochastically maps z_n to x_n. One interpretation of the model is that z_n is a compression of x_n and the decoder decompresses z_n back to x_n.

For example, we may model x_n (the colors of each pixel) as normally distributed,[4] whose mean μ and covariance Σ is the output of a function,

$$\mu = f_\mu(z_n; \theta) \quad \Sigma = f_\Sigma(z_n; \theta). \tag{23}$$

In the VAE, the function f is modeled by a *neural network*. A neural network is a composition of transformations starting from an input layer and mapping to an output layer. The layers in-between are called hidden layers (Figure 1). The transformation between layers takes in an input $\zeta^{(i)}$ and outputs $\zeta^{(i+1)}$, typically by applying a linear transformation followed by a nonlinear transformation,

$$\zeta^{(i)} = \sigma\left(W^{(i)}\zeta^{(i)} + b^{(i)}\right), \tag{24}$$

where the matrix $W^{(i)}$ is called the *weights* of the network, $b^{(i)}$ is the "bias", and σ is a non-linear transformation. The weights and biases constitute the parameters of the neural network,

$$\theta = \left(W^{(1)}, W^{(2)}, \cdots, W^{(p)}, b^{(1)}, b^{(2)}, \cdots, b^{(p)}\right). \tag{25}$$

The weight matrix $W^{(i)}$ need not be a square matrix, meaning that the dimension of $\zeta^{(i)}$ and $\zeta^{(i+1)}$ can differ. In practice, the hidden layers are often taken to have a larger dimension than the input and outputs layers.

A common choice for σ is the rectified linear unit (ReLu) function,

$$\sigma(\alpha) = \max(0, \alpha). \tag{26}$$

There are many choices on the architecture of the neural network. For example what should the depth of the network be, i.e. the number of layers? What should be its

[4] A more exact model would use a categorical distribution, since the color channels are an integer over $[0, 255]$ but for a hight "count", a continuous distribution can work reasonably well.

width, i.e. dimension of intermediate layers? Which nonlinear transformation should we use?

You may ask yourself why use a neural network in the first place? Neural networks have had a tremendous success empirically and there exists *some* theory to explain this success. In particular, *universal approximation theorems* provide ideal conditions under which a neural network can approximate any function f arbitrarily well. A well-known theorem states that a neural network with one infinitely-wide hidden layer can approximate any "well-behaved" function. But despite these results, the theory is often considered to be lagging behind the practice and explaining the remarkable success of neural networks remains an active area of research.

6.5.2 Inference: Amortized variational inference

Now that we have our model, we can come up with a training procedure.

In a Bayesian setting, the task is to learn the posterior $p(\theta, z_{1:N} \mid x_{1:N})$.

In a frequentist setting, we learn θ by maximizing the marginal likelihood,

$$\theta^* = \text{argmax}_\theta \; p_\theta(x_{1:N}) = \int \prod_{n=1}^{N} p_\theta(x_{1:N} \mid z_{1:N}, \theta) p(z_{1:N}) \mathrm{d}z_{1:N}. \tag{27}$$

Notice the unknown $z_{1:N}$ is still treated as a random variable and so the procedure can be described as a hybrid between a Bayesian and a frequentist approach.

In practice, the integral on the R.H.S of eq. (27) is intractable. To approximate $p_\theta(x_{1:N})$, we will use the ELBO. Applying eq. (12) to our problem, we have that

$$\text{ELBO}(\theta, \lambda) = p_\theta(x_{1:N}) - \text{KL}(q_\lambda(z_{1:N}) || p_\theta(z_{1:N} \mid x_{1:N})), \tag{28}$$

where λ are the variational parameters for a family of approximations $\mathcal{Q}$. To have a concrete example in mind, you may once again take $\mathcal{Q}$ to be the family of Gaussians.

The ELBO provides a lower-bound on $p_\theta(x_{1:N})$ and the gap in this bound is given by $\text{KL}(q_\lambda(z_{1:N}) || p_\theta(z_{1:N} \mid x_{1:N}))$. This suggests a joint optimization problem over θ and the variational parameters λ,

$$\begin{aligned} \theta^*, \lambda^* &= \text{argmax}_{\theta,\lambda} \; \text{ELBO}(\theta, \lambda) \\ &= \text{argmax}_{\theta,\lambda} \; \left[p_\theta(x_{1:N}) - \text{KL}(q_\lambda(z_{1:N}) || p_\theta(z_{1:N} \mid x_{1:N})) \right]. \end{aligned} \tag{29}$$

If $\mathcal{Q}$ is rich enough, then we can drive $\text{KL}(q_\lambda(z_{1:N}) || p_\theta(z_{1:N} \mid x_{1:N}))$ to 0 and optimize the true objective function. However, $\mathcal{Q}$ is often restricted and so the KL can usually not be driven to 0, meaning minimize an approximation of $p_\theta(x_{1:N})$ rather than $p_\theta(x_{1:N})$ itself.

Conditional on θ, each observation x_n only depends on z_n,

$$p(x_{1:N} \mid z_{1:N}) = \prod_{n=1}^{N} p(x_n \mid z_n). \tag{30}$$

We will also assume that the priors on $z_{1:N}$ factorizes, $p(z_{1:N}) = \prod_n p(z_n)$.

If the likelihoods $p(x_n \mid z_n)$ follow the same distribution and similarly for the priors $p(z_n)$, then the posterior $p(z_n \mid x_n)$ can be expressed as a function of x_n and z_n,

$$p(z_n \mid x_n) = g(z_n, x_n). \tag{31}$$

This suggests that, instead of learning a variational parameter λ_n for each factor $q(z_n)$, we can instead *amortize* the procedure and learn a function f such that,

$$\lambda_n = f(x_n). \tag{32}$$

If $\mathcal{Q}$ is the family of Gaussian, then f maps x_n to a posterior mean and covariance matrix. Once again we can approximate f with a neural network. This neural network is often termed the *encoder*, because it maps the original image x_n to a low-dimensional representation z_n.

There are several arguments for using amortized VI:

- The number of variational parameters is determined by the size of the encoder neural network, rather than by the dimension of the observations.
- Information is pooled across observations, empirically leading to faster optimization.
- The learned function, $\hat{f}$ can be used to compress images in a validation set.

On the other hand, amortization can also have drawbacks:

- Amortization restricts the family $\mathcal{Q}$ of approximation—this is most obvious if we use a simple encoder, for example a linear function—and the solution can be suboptimal relative to standard VI. This sub-optimality is known as the *amortization gap*.

Amortized VI can also be used to do full Bayesian inference on the joint posterior $p(\theta, z_{1:N} \mid x_{1:N})$ [e.g Margossian and Blei, 2024].

6.6 Variational inference beyond the Gaussian approximation

Many VI algorithms rely on a family $\mathcal{Q}$ of approximations which is not Gaussian. In many applications, it is common to construct a bespoke family $\mathcal{Q}$ to match the characteristics of the target p [e.g., Blei et al., 2003]. But this approach requires a large algorithmic effort from the user and works poorly in Bayesian workflow, since every revision of the model requires a revision of the variational family $\mathcal{Q}$.

The ideal of *black box VI* is to use a family which works well across a broad range of models and does not require bespoke manipulations from the user. The Gaussian approximation is a reasonable starting point:

- Many Bayesian models have Gaussian components in the likelihood or prior.
- Under certain conditions, the Bernstein-von Mises theorem tells us that as we accumulate data the posterior becomes more Gaussian.

- The resulting variational optimization problem tends to be manageable, especially if the approximation is factorized.

But in many applications, we can improve on the Gaussian approximation. Here I'll very briefly review some strategies:

- **Pathfinder VI [Zhang et al., 2022].** Pathfinder offers a slight twist on traditional VI, but in end effect, it produces a normal approximation.

  The *multi-pathfinder* runs Pathfinder VI I times in parallel and generates I normal approximations. If p is far from Gaussian, different runs of the optimizer produce different solutions. Combining these solutions produces a mixture of normals and the relative weight of each mixture component is determined by an importance sampling scheme.

- **Normalizing flow.** Normalizing flows are defined by an initial draw $\zeta \sim q_0$ which is then transformed via a learned function $z = f(\zeta)$. Usually, we'll pick q_0 to be a simple distributon, for example a Gaussian. This simple distribution is then transformed into a more sophisticated distribution using f.

  The final approximation to p is,

  $$q(z) = q(f(\zeta)) \left| J_{f^{-1}} \right|, \tag{33}$$

  where $J_{f^{-1}}$ is the Jacobian of f^{-1},

  $$J_{f^{-1},ij} = \frac{\partial}{\partial z_j} f^{-1}(\zeta)_i. \tag{34}$$

  We can elect f to be a highly flexible function, once again using a neural network. However, in order to evaluate $\log q(z)$ when computing the ELBO and its gradient, we must be able to evaluate the Jacobian determinant $\left| J_{f^{-1}} \right|$. This imposes constraints on the neural network architecture. In particular, the final function f needs to be invertible.

  Naturally, the more "complicated" f is, the more difficult the variational optimization problem—both in terms of computational cost per step and number of steps required to find a minima (in fact, once neural networks are involved, we can expect the objective function to be non-convex and have many local minimas).

References

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

A. Katsevich and P. Rigollet. On the approximation accuracy of gaussian variational inference. *Annals of Statistics*, 52(4):1384–1409, 2024.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR) 2014*, 2014.

C. Margossian and L. K. Saul. Variational inference in location-scale families: Exact recovery of the mean and correlation matrix. In *Proceedings of The 28th International Conference on Artificial Intelligence and Statistics*, volume 258 of *Proceedings of Machine Learning Research*, pages 3466–3474. PMLR, 2025.

C. C. Margossian and D. M. Blei. Amortized variational inference: When and why? In *Uncertainty in Artificial Intelligence*, volume 244 of *Proceedings of Machine Learning Research*, pages 2434–2449, 2024.

C. C. Margossian and L. K. Saul. The shrinkage-delinkage trade-off: An analysis of factorized gaussian approximations for variational inference. In *Proceedings of the Thirty-Ninth Conference on Uncertainty in Artificial Intelligence*, volume 216 of *Proceedings of Machine Learning Research*, pages 1358–1367. PMLR, 2023.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *International Conference on Machine Learning*, 2014. arXiv:1401.4082.

Y. Yao, A. Vehtari, D. Simpson, and A. Gelman. Yes, but did it work?: Evaluating variational inference. In *International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5581–5590, 2018.

L. Zhang, B. Carpenter, A. Gelman, and A. Vehtari. Pathfinder: Parallel quasi-newton variational inference. *Journal of Machine Learning Research*, 23:1–49, 2022. URL `http://jmlr.org/papers/v23/21-0889.html`.