VI Masterclass: Modern Variational Inference

Sanmitra Ghosh

November 8, 2022

MRC Biostatistics Unit,
University of Cambridge
sanmitra.ghosh@mrc-bsu.cam.ac.uk

Table of contents

Recap of classical VI

Introduction

Consider a generic probabilistic model:

$$p(\theta, \mathbf{z}, \boldsymbol{y}) = p(\boldsymbol{\theta}) \prod_{n=1}^{N} p(y_n | \boldsymbol{z}_n, \boldsymbol{\theta}) p(\boldsymbol{z}_n | \boldsymbol{\theta}) \tag{1}$$

- Observations $\boldsymbol{y} = y_{1:N}$.
- Local variables/parameters $\mathbf{z} = \boldsymbol{z}_{1:N}, \quad \boldsymbol{z}_n \in \mathbb{R}^J$.
- Global variable/parameter $\boldsymbol{\theta} \in \mathbb{R}^D$.

Goal: Approximate the posterior $p(\boldsymbol{\theta}, \mathbf{z} | \boldsymbol{y})$ by a tractable distribution $q(\boldsymbol{\theta}, \boldsymbol{z}; \boldsymbol{\lambda})$.

- parameterised by $\boldsymbol{\lambda}$, the **variational parameters**.

Recap

In variational inference (VI) we try to find a suitable approximation $q(\boldsymbol{\theta}, \mathbf{z}; \boldsymbol{\lambda})$ that minimises the following Kulback-Liebler divergence:

$$\mathrm{KL}(q(\boldsymbol{\theta}, \mathbf{z}; \boldsymbol{\lambda}) || p(\boldsymbol{\theta}, \mathbf{z} | \boldsymbol{y})),$$

turning inference into an optimisation problem:

$$\boldsymbol{\lambda}^* = \underset{\boldsymbol{\lambda}}{\operatorname{argmin}} \, \mathrm{KL}(q(\boldsymbol{\theta}, \mathbf{z}; \boldsymbol{\lambda}) || p(\boldsymbol{\theta}, \mathbf{z} | \boldsymbol{y})).$$

Rather than direct minimisation of $\mathrm{KL}(q||p)$, we work with an alternative objective, the evidence lower bound (ELBO):

$$\mathcal{L}(\boldsymbol{\lambda}) = \mathbb{E}_{q(\boldsymbol{\theta}, \mathbf{z}; \boldsymbol{\lambda})}[\log p(\boldsymbol{\theta}, \mathbf{z} | \boldsymbol{y})] - \mathbb{E}_{q(\boldsymbol{\theta}, \mathbf{z}; \boldsymbol{\lambda})}[\log q(\boldsymbol{\theta}, \mathbf{z}; \boldsymbol{\lambda})] \qquad (2)$$

Recap

$$\mathcal{L}(\boldsymbol{\lambda}) = \mathbb{E}_{q(\boldsymbol{\theta},\mathbf{z};\boldsymbol{\lambda})}[\log p(\boldsymbol{\theta},\mathbf{z}|\boldsymbol{y})] - \mathbb{E}_{q(\boldsymbol{\theta},\mathbf{z};\boldsymbol{\lambda})}[\log q(\boldsymbol{\theta},\mathbf{z};\boldsymbol{\lambda})]$$

- KL is intractable; VI optimizes the ELBO instead.
  - It is a lower bound on $\log p(\boldsymbol{y})$.
  - Maximising the ELBO is equivalent to minimising the KL.
- ELBO trades off two terms.
  - The first term prefers $q(\cdot)$ to place its mass on the MAP.
  - The second term encourages $q(\cdot)$ to be diffuse.

Recap

$$\mathcal{L}(\boldsymbol{\lambda}) = \mathbb{E}_{q(\boldsymbol{\theta},\mathbf{z};\boldsymbol{\lambda})}[\log p(\boldsymbol{y}|\boldsymbol{\theta},\mathbf{z})] - \mathrm{KL}(q(\boldsymbol{\theta},\mathbf{z};\boldsymbol{\lambda})||p(\boldsymbol{\theta},\mathbf{z}))$$

- KL is intractable; VI optimizes the ELBO instead.
  - It is a lower bound on $\log p(\boldsymbol{y})$.
  - Maximising the ELBO is equivalent to minimising the KL.
- ELBO trades off two terms.
  - The first term prefers $q(\cdot)$ to place its mass on the MLE.
  - The second term encourages $q(\cdot)$ to be close to the prior.

Recap: Mean-field variational inference

- Specify the form of $q(\boldsymbol{\theta}, \mathbf{z})$.
- **Mean-field family** is fully factorised.
$$q(\boldsymbol{\theta}, \mathbf{z}; \boldsymbol{\nu}, \boldsymbol{\Phi}) = q(\boldsymbol{\theta}; \boldsymbol{\nu}) \prod_{n=1}^{N} q(\boldsymbol{z}_n; \phi_n), \quad \boldsymbol{\lambda} = (\boldsymbol{\nu}, \boldsymbol{\Phi}).$$
- Each factor is the same family as the model's complete conditional*:
$$\begin{aligned} p(\boldsymbol{\theta}|\mathbf{z}, \boldsymbol{y}) &= h(\boldsymbol{\theta}) \exp\{\boldsymbol{\eta}_g(\mathbf{z}, \boldsymbol{y})^T \boldsymbol{\tau}(\boldsymbol{\theta}) - a_g(\boldsymbol{\eta}_g(\boldsymbol{z}, \boldsymbol{y}))\} \\ q(\boldsymbol{\theta}; \boldsymbol{\nu}) &= h(\boldsymbol{\theta}) \exp\{\boldsymbol{\nu}^T \boldsymbol{\theta} - a_g(\boldsymbol{\nu})\}, \end{aligned}$$
  and
$$\begin{aligned} p(\boldsymbol{z}_{n,j}|\boldsymbol{z}_{n,-j}, \boldsymbol{y}, \boldsymbol{\theta}) &= h(\boldsymbol{z}_{n,j}) \exp\{\boldsymbol{\eta}_l(\boldsymbol{z}_{n,-j}, \boldsymbol{y}, \boldsymbol{\theta})^T \boldsymbol{\tau}(\boldsymbol{z}_{n,j}) \\ &\quad - a_g(\boldsymbol{\eta}_l(\boldsymbol{z}_{n,-j}, \boldsymbol{y}, \boldsymbol{\theta}))\} \\ q(\boldsymbol{z}_{n,j}; \phi_{n,j}) &= h(\boldsymbol{\theta}) \exp\{\phi_{n,j}^T \boldsymbol{\theta} - a_l(\phi_{n,j})\}, \end{aligned}$$
  where $h(\cdot), a(\cdot)$ are base measure and log-normalizer; $\boldsymbol{\eta}(\cdot), \boldsymbol{\tau}(\cdot)$ are natural parameter and sufficient statistics.

* MD Hoffman 2013.

Recap: Classical VI

Coordinate ascent variational inference (CAVI)[1]:

- Initialise $\boldsymbol{\nu}$ randomly, the repeat these steps till convergence:
    - **For each data point** y_i set local parameters:
$$\boldsymbol{\phi}_{n,j} = \mathbb{E}_q[\boldsymbol{\eta}_l(\boldsymbol{z}_{n,-j}, \boldsymbol{y}, \boldsymbol{\theta})]$$
    - Set global parameter:
$$\boldsymbol{\nu} = \mathbb{E}_q[\boldsymbol{\eta}_g(\mathbf{z}, \boldsymbol{y})]$$

- Classical VI is inefficient for big data applications.
- Cannot be applied to non-conjugate models.
- Modern VI relies upon stochastic optimisation.

[1] Blei et al. 2017

VI as stochastic optimisation

VI as gradient descent

- If gradient of the ELBO $\nabla_{\boldsymbol{\lambda}}\mathcal{L}(\boldsymbol{\lambda})$ is available:
  - We can then apply gradient descent to optimise $\boldsymbol{\lambda}$:

$$\boldsymbol{\lambda}_{t+1} \leftarrow \boldsymbol{\lambda}_t - \rho\nabla_{\boldsymbol{\lambda}}\mathcal{L}(\boldsymbol{\lambda}_t), \quad \text{with step-size } \rho.$$

- Stochastic gradient descent (SGD) to rescue.
  - Requires unbiased gradients, $\mathbb{E}[\hat{\nabla}_{\boldsymbol{\lambda}}\mathcal{L}(\boldsymbol{\lambda}_t)] = \nabla_{\boldsymbol{\lambda}}\mathcal{L}(\boldsymbol{\lambda})$
  - Requires a step-size schedule that follows the **Robbins-Monro** conditions[2]:

$$\textstyle\sum_{t=0}^{\infty}\rho_t = \infty \quad \text{and} \quad \sum_{t=0}^{\infty}\rho_t^2 < \infty.$$

  - Then the following update will converge to $\boldsymbol{\lambda}^*$

$$\boldsymbol{\lambda}_{t+1} \leftarrow \boldsymbol{\lambda}_t - \rho_t\hat{\nabla}_{\boldsymbol{\lambda}}\mathcal{L}(\boldsymbol{\lambda}_t).$$

Robbins, H. and Monro, S. (1951)

Automatic differentiation

Introduction

For a target function $f : \mathbb{R}^n \to \mathbb{R}^m$, the corresponding Jacobian's (i, j)-th entry is

$$J_{i,j} = \frac{\partial f_i}{\partial x_j}.$$

For a composite function: $f(\boldsymbol{x}) = h \circ g(\boldsymbol{x}) = h((g(\boldsymbol{x}))$, with $\boldsymbol{x} \in \mathbb{R}^n$, $g : \mathbb{R}^n \to \mathbb{R}^k$ and $h : \mathbb{R}^k \to \mathbb{R}^m$, we have the jacobian (by applying chain rule) given by

$$J = J_{h \circ g} = J_h(g(\boldsymbol{x})) \cdot J_g(\boldsymbol{x}),$$

with (i, j)-th element:

$$J_{i,j} = \frac{\partial h_i}{\partial g_1}\frac{\partial g_1}{\partial x_j} + \frac{\partial h_i}{\partial g_2}\frac{\partial g_2}{\partial x_j} + \ldots, + \frac{\partial h_i}{\partial g_k}\frac{\partial g_k}{\partial x_j}.$$

Introduction

In general if f is the composite expression of L functions

$$f = f^L \circ f^{L-1} \circ \ldots \circ f^1,$$

then the corresponding Jacobian matrix verifies

$$J = J_L \cdot J_{L-1} \cdot \ldots \cdot J_1.$$

Once we express the composite function as a computer program, then automatic differentiation (AD) can be applied for two tasks:

- Compute the action of the Jacobian on a vector $\boldsymbol{u} \in \mathbb{R}^n$ in the input space, **Forward-mode**:

  $J \cdot \boldsymbol{u}$, known as Jacobian Vector product (JVP).

- Or compute the action on a vector $\boldsymbol{w} \in \mathbb{R}^m$ in the output space, **Reverse-mode** :

  $\boldsymbol{w}^T \cdot J$, known as Vector Jacobian product (VJP).

Forward-mode

Consider the following function: $y = f(g((h(x)))$.

Fix the independent variable, x, and compute the derivative of each sub-expression recursively:

Computational graph:

$$
\begin{aligned}
v_0 &= x \\
v_1 &= h(v_0) \\
v_2 &= g(v_1) \\
v_3 &= f(v_2) = y
\end{aligned}
$$

Differentiate inner functions:

$$
\begin{aligned}
\frac{dy}{dx} &= \frac{dy}{dv_2}\frac{dv_2}{dx} \\
&= \frac{dy}{dv_2}\left(\frac{dv_2}{dv_1}\frac{dv_1}{dx}\right) \\
&= \frac{dy}{dv_2}\left(\frac{dv_2}{dv_1}\left(\frac{dv_1}{dv_0}\frac{dv_0}{dx}\right)\right) \\
&= \frac{dy}{dv_2}\left(\frac{dv_2}{dv_1}\left(\frac{dv_1}{dv_0}\cdot 1\right)\right).
\end{aligned}
$$

- Simultaneously evaluate $v_i, \dot{v}_i$.

Forward-mode

Substitution of inner functions to evaluate the push-forward action of the Jacobian on the tangent vector $\boldsymbol{u}$, at $\boldsymbol{x}$.

$$
\begin{aligned}
J \cdot \boldsymbol{u} &= J_L \cdot J_{L-1} \cdot \ldots \cdot J_3 \cdot J_2 \cdot J_1 \cdot \boldsymbol{u} \\
&= J_L \cdot J_{L-1} \cdot \ldots \cdot J_3 \cdot J_2 \cdot \boldsymbol{u}_1 \\
&= J_L \cdot J_{L-1} \cdot \ldots \cdot J_3 \cdot \boldsymbol{u}_2 \\
&\ldots \\
&= J_L \cdot \boldsymbol{u}_{L-1},
\end{aligned}
$$

where we have the following recursion

$$
\begin{aligned}
\boldsymbol{u}_1 &= J_1 \cdot \boldsymbol{u} \\
\boldsymbol{u}_l &= J_l \cdot \boldsymbol{u}_{l-1}.
\end{aligned}
$$

- We can evaluate the Jacobian and the **JVP** of a complex function, sequentially, and in a matrix-free way.

Forward-mode: example

Consider the following function;

$$f(x_1, x_2) = y = \log(x_1) + x_1 x_2 - \sin(x_2)$$

Goal here is to compute $\frac{\partial f}{\partial x_1}$.

We start with computing derivatives of intermediate variables

$$\dot{v} = \frac{\partial v_i}{\partial x_1}.$$

Lets walk through the computational graph.

Forward-mode: example

Conside the following function;

$$f(x_1, x_2) = y = \log(x_1) + x_1 x_2 - \sin(x_2)$$

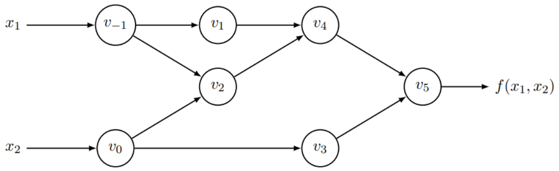


Figure 4: Computational graph of the example $f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$.

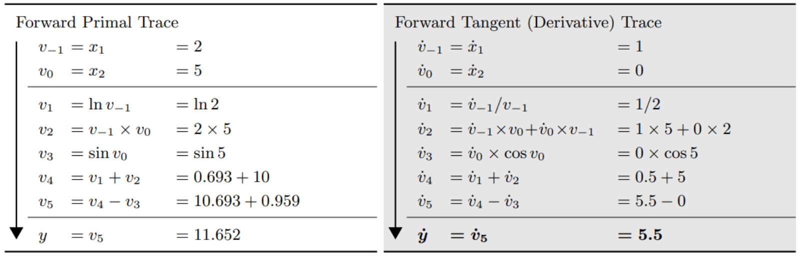
* figure from AG Baydin et al., 2018

Forward-mode: example

Conside the following function;

$$f(x_1, x_2) = y = \log(x_1) + x_1 x_2 - \sin(x_2)$$

Table 2: Forward mode AD example, with $y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$ evaluated at $(x_1, x_2) = (2, 5)$ and setting $\dot{x}_1 = 1$ to compute $\frac{\partial y}{\partial x_1}$. The original forward evaluation of the primals on the left is augmented by the tangent operations on the right, where each line complements the original directly to its left.



Forward Primal Trace			Forward Tangent (Derivative) Trace		
v_{-1}	$= x_1$	$= 2$	$\dot{v}_{-1}$	$= \dot{x}_1$	$= 1$
v_0	$= x_2$	$= 5$	$\dot{v}_0$	$= \dot{x}_2$	$= 0$
v_1	$= \ln v_{-1}$	$= \ln 2$	$\dot{v}_1$	$= \dot{v}_{-1}/v_{-1}$	$= 1/2$
v_2	$= v_{-1} \times v_0$	$= 2 \times 5$	$\dot{v}_2$	$= \dot{v}_{-1} \times v_0 + \dot{v}_0 \times v_{-1}$	$= 1 \times 5 + 0 \times 2$
v_3	$= \sin v_0$	$= \sin 5$	$\dot{v}_3$	$= \dot{v}_0 \times \cos v_0$	$= 0 \times \cos 5$
v_4	$= v_1 + v_2$	$= 0.693 + 10$	$\dot{v}_4$	$= \dot{v}_1 + \dot{v}_2$	$= 0.5 + 5$
v_5	$= v_4 - v_3$	$= 10.693 + 0.959$	$\dot{v}_5$	$= \dot{v}_4 - \dot{v}_3$	$= 5.5 - 0$
y	$= v_5$	$= 11.652$	$\boldsymbol{\dot{y}}$	$\boldsymbol{= \dot{v}_5}$	$\mathbf{= 5.5}$

* Table from AG Baydin et al., 2018

Forward-mode: Jacobian computation

We can generalise the example for $f : \mathbb{R}^n \to \mathbb{R}^m$ by setting $\boldsymbol{u} = \boldsymbol{e}_j$, the j-th unit vector. Specifically, let $u_j = 1$ for the j-th input and 0 for the rest. Then:

$$J_{.j} = J \cdot \boldsymbol{u}, \tag{3}$$

where $J_{.j}$ is the j-th column of J. We need n sweeps to compute the Jacobian.

Reverse-mode

Consider the following function: $y = f(g((h(x)))$.

Fix the dependable variable, y, and compute the derivative of each sub-expression recursively:

Computational graph:

$$v_0 = x$$
$$v_1 = h(v_0)$$
$$v_2 = g(v_1)$$
$$v_3 = f(v_2) = y$$

Differentiate outer functions:

$$\begin{aligned}\frac{dy}{dx} &= \frac{dy}{dv_1}\frac{dv_1}{dx} \\ &= \left(\frac{dy}{dv_2}\frac{dv_2}{dv_1}\right)\frac{dv_1}{dx} \\ &= \left(\left(\frac{dy}{dv_3}\frac{dv_3}{dv_2}\right)\frac{dv_2}{dv_1}\right)\frac{dv_1}{dx} \\ &= \left(\left(1 \cdot \frac{dv_3}{dv_2}\right)\frac{dv_2}{dv_1}\right)\frac{dv_1}{dx}\end{aligned}$$

- Requires two passes to evaluate $v_i, \dot{v}_i$.
- Key quantity is the adjoint: $\bar{w}_i = \frac{dy}{dv_i}$.

Forward-mode

Substitution of outer functions to evaluate the pullback action of the Jacobian on a cotangent vector $\boldsymbol{w}$.

$$\begin{aligned}\boldsymbol{w}^T \cdot J &= \boldsymbol{w}^T \cdot J_L \cdot J_{L-1} \cdot \ldots \cdot J_3 \cdot J_2 \cdot J_1 \\ &= \bar{\boldsymbol{w}}_L^T \cdot J_{L-1} \cdot \ldots \cdot J_3 \cdot J_2 \\ & \ldots \\ &= \bar{\boldsymbol{w}}_2^T \cdot J_1,\end{aligned}$$

where we have the following recursion

$$\begin{aligned}\bar{\boldsymbol{w}}_L^T &= \boldsymbol{w}^T \cdot J_L \\ \bar{\boldsymbol{w}}_{l-1}^T &= \bar{\boldsymbol{w}}_l^T \cdot J_{l-1}.\end{aligned}$$

- We can evaluate the Jacobian and the **VJP** of a complex function, sequentially, and in a matrix-free way.

Reverse-mode: example

Conside the following function;

$$f(x_1, x_2) = y = \log(x_1) + x_1 x_2 - \sin(x_2)$$

Goal here is to compute adjoints

$$\bar{w}_i = \frac{\partial y}{\partial v_i}.$$

Lets walk through the computational graph.

Forward-mode: example

Conside the following function;

$$f(x_1, x_2) = y = \log(x_1) + x_1 x_2 - \sin(x_2)$$

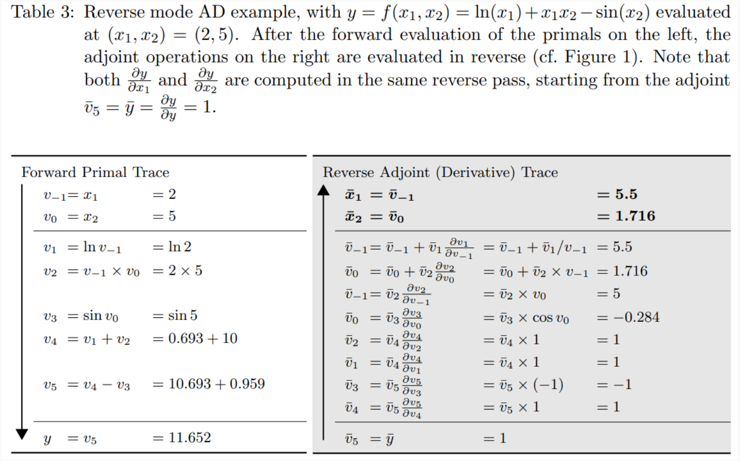


Table 3: Reverse mode AD example, with $y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$ evaluated at $(x_1, x_2) = (2, 5)$. After the forward evaluation of the primals on the left, the adjoint operations on the right are evaluated in reverse (cf. Figure 1). Note that both $\frac{\partial y}{\partial x_1}$ and $\frac{\partial y}{\partial x_2}$ are computed in the same reverse pass, starting from the adjoint $\bar{v}_5 = \bar{y} = \frac{\partial y}{\partial y} = 1$.

Forward Primal Trace		
$v_{-1} = x_1$		$= 2$
$v_0 = x_2$		$= 5$
$v_1 = \ln v_{-1}$		$= \ln 2$
$v_2 = v_{-1} \times v_0$		$= 2 \times 5$
$v_3 = \sin v_0$		$= \sin 5$
$v_4 = v_1 + v_2$		$= 0.693 + 10$
$v_5 = v_4 - v_3$		$= 10.693 + 0.959$
$y = v_5$		$= 11.652$

Reverse Adjoint (Derivative) Trace		
$\bar{x}_1 = \bar{v}_{-1}$		$= 5.5$
$\bar{x}_2 = \bar{v}_0$		$= 1.716$
$\bar{v}_{-1} = \bar{v}_{-1} + \bar{v}_1 \frac{\partial v_1}{\partial v_{-1}}$	$= \bar{v}_{-1} + \bar{v}_1 / v_{-1}$	$= 5.5$
$\bar{v}_0 = \bar{v}_0 + \bar{v}_2 \frac{\partial v_2}{\partial v_0}$	$= \bar{v}_0 + \bar{v}_2 \times v_{-1}$	$= 1.716$
$\bar{v}_{-1} = \bar{v}_2 \frac{\partial v_2}{\partial v_{-1}}$	$= \bar{v}_2 \times v_0$	$= 5$
$\bar{v}_0 = \bar{v}_3 \frac{\partial v_3}{\partial v_0}$	$= \bar{v}_3 \times \cos v_0$	$= -0.284$
$\bar{v}_2 = \bar{v}_4 \frac{\partial v_4}{\partial v_2}$	$= \bar{v}_4 \times 1$	$= 1$
$\bar{v}_1 = \bar{v}_4 \frac{\partial v_4}{\partial v_1}$	$= \bar{v}_4 \times 1$	$= 1$
$\bar{v}_3 = \bar{v}_5 \frac{\partial v_5}{\partial v_3}$	$= \bar{v}_5 \times (-1)$	$= -1$
$\bar{v}_4 = \bar{v}_5 \frac{\partial v_5}{\partial v_4}$	$= \bar{v}_5 \times 1$	$= 1$
$\bar{v}_5 = \bar{y}$	$= 1$	

* Table from AG Baydin et al., 2018

Reverse-mode: Jacobian computation

We can generalise the example for $f : \mathbb{R}^n \to \mathbb{R}^m$ where the by setting $\boldsymbol{w} = \boldsymbol{e}_i$, the i-th unit vector. Specifically, let $u_i = 1$ for the i-th input and 0 for the rest. Then:

$$J_{i.} = \boldsymbol{w}^T \cdot J, \tag{4}$$

where $J_{i.}$ is the i-th row of J. We need m sweeps to compute the Jacobian.

Forward vs Reverse-mode

For the function $f : \mathbb{R}^n \to \mathbb{R}^m$:

- Forward mode is more efficient when $m >> n$, vice-versa for reverse mode.
- Reverse mode is most efficient when $m = 1$.
- Reverse mode is thus widely used in statistics/ machine learning since generally $m = 1$. E.g likelihood, loss function.
- Reverse mode comes with a storage cost that increases with the complexity of the computational graph.

Monte Carlo gradient estimation

Example: Bayesian logistic regression

- Data pair: $y_n, \boldsymbol{x}_n$.
- Labels: y_n
- Covariates: $\boldsymbol{x}_n \in \mathbb{R}^D$.
- Regression coefficients: $\boldsymbol{\theta} \in \mathbb{R}^D$.

$$\begin{aligned} \boldsymbol{\theta} &\sim p(\boldsymbol{\theta}) \\ p(\boldsymbol{y}|\boldsymbol{\theta}) &= \prod_{n=1}^{N} \text{Bern}(\text{logit}^{-1}(\boldsymbol{\theta}\boldsymbol{x}_n)). \end{aligned} \tag{5}$$

- If we consider a Gaussian mean-field approximation: $q(\boldsymbol{\theta}; \boldsymbol{\lambda}) := \prod_{d=1}^{D} \mathcal{N}(\mu_d, \sigma_d^2)$, where the varational parameters are $\boldsymbol{\lambda} = (\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$.
- The ELBO becomes intractable in this case due to the nonlinearity of the link function.

Monte Carlo expectations

Consider a simple non-conjugate model: $p(\boldsymbol{y}, \boldsymbol{\theta})$, like the logistic regression. The ELBO in this case:

$$\mathcal{L}(\boldsymbol{\lambda}) = \mathbb{E}_{q(\boldsymbol{\theta};\boldsymbol{\lambda})}[\log p(\boldsymbol{y}, \boldsymbol{\theta})] - \mathbb{E}_{q(\boldsymbol{\theta};\boldsymbol{\lambda})}[\log q(\boldsymbol{\theta}; \boldsymbol{\lambda})].$$

Write,

$$f(\boldsymbol{\theta}) = \log p(\boldsymbol{y}, \boldsymbol{\theta}) - \log q(\boldsymbol{\theta}; \boldsymbol{\lambda}),$$

as the cost function. To apply SGD we need to estimate the gradient of an intractable expectation:

$$\nabla_{\boldsymbol{\lambda}} \mathbb{E}_{q(\boldsymbol{\theta};\boldsymbol{\lambda})}[f(\boldsymbol{\theta})],$$

using Monte Carlo.

- Two different approaches for constructing such an estimator:
    - **Score function estimator***.
    - **Pathwise estimator**** through the reparameterisation trick.

* [Glynn 1990; Williams, 1992; Wingate+ 2013; Ranganath+ 2014; Mnih+ 2014].
** [Glasserman 1991; Fu 2006; Kingma+ 2014; Rezende+ 2014; Titsias+ 2014]

Score function estimator

Deriving the estimator:

$$\begin{aligned}\nabla_{\boldsymbol{\lambda}}\mathbb{E}_{q(\boldsymbol{\theta};\boldsymbol{\lambda})}[f(\boldsymbol{\theta})] &= \nabla_{\boldsymbol{\lambda}}\int q(\boldsymbol{\theta};\boldsymbol{\lambda})f(\boldsymbol{\theta})d\boldsymbol{\theta} = \int f(\boldsymbol{\theta})\nabla_{\boldsymbol{\lambda}} q(\boldsymbol{\theta};\boldsymbol{\lambda})d\boldsymbol{\theta}\\ &= \int q(\boldsymbol{\theta};\boldsymbol{\lambda})f(\boldsymbol{\theta})\nabla_{\boldsymbol{\lambda}}\log q(\boldsymbol{\theta};\boldsymbol{\lambda})d\boldsymbol{\lambda}, \quad \{\nabla \log q = \nabla q / q\}\\ &= \mathbb{E}_{q(\boldsymbol{\theta};\boldsymbol{\lambda})}[f(\boldsymbol{\theta})\nabla_{\boldsymbol{\lambda}}\log q(\boldsymbol{\theta};\boldsymbol{\lambda})].\end{aligned}$$

Substituting the expression for $f(\boldsymbol{\theta})$ we have gradient of ELBO as:

$$\nabla_{\boldsymbol{\lambda}}\mathcal{L}(\boldsymbol{\lambda}) = \mathbb{E}_{q(\boldsymbol{\theta};\boldsymbol{\lambda})}[\nabla_{\boldsymbol{\lambda}}\log q(\boldsymbol{\theta};\boldsymbol{\lambda})(\log p(\boldsymbol{y},\boldsymbol{\theta}) - \log q(\boldsymbol{\theta};\boldsymbol{\lambda}))],$$

and a noisy gradient estimate as:

$$\hat{\nabla}_{\boldsymbol{\lambda}}\mathcal{L}(\boldsymbol{\lambda}) = \frac{1}{S}\sum_{s=1}^{S}[\nabla_{\boldsymbol{\lambda}}\log q(\boldsymbol{\theta}^{(s)};\boldsymbol{\lambda})(\log p(\boldsymbol{y},\boldsymbol{\theta}^{(s)}) - \log q(\boldsymbol{\theta}^{(s)};\boldsymbol{\lambda}))], \quad (6)$$

where $\boldsymbol{\theta}^{(s)} \sim q(\boldsymbol{\theta};\boldsymbol{\lambda})$.

Properties of the estimator

- Unbiasedness
  - Unbiased when interchange between differentiation and integration is valid.
  - Use of Dominated convergence theorem*.
  - Assumptions usually hold in machine learning applications.
- Variance
  - Variance depends on parameter dimensionality.
    $$\mathbb{V}[\hat{\nabla}_{\boldsymbol{\lambda}}\mathcal{L}(\boldsymbol{\lambda})] = \mathbb{E}_{q(\boldsymbol{\theta};\boldsymbol{\lambda})}\left[\left(\nabla_{\boldsymbol{\lambda}} \log q(\boldsymbol{\theta};\boldsymbol{\lambda}) f(\boldsymbol{\theta})\right)^2\right] - \left(\mathbb{E}_{q(\boldsymbol{\theta};\boldsymbol{\lambda})}[\hat{\nabla}_{\boldsymbol{\lambda}}\mathcal{L}(\boldsymbol{\lambda})]\right)^2$$
  - Variance depends on the cost function $f(\boldsymbol{\theta})$, multiplicatively.
  - Practically impossible to use this estimator due to high variance.

Solution: Control variates.

* Shakir Mohamed 2020

Control variates

Replace $f(\boldsymbol{\theta})$ with $\tilde{f}(\boldsymbol{\theta})$ where $\mathbb{E}[\tilde{f}(\boldsymbol{\theta})] = \mathbb{E}[f(\boldsymbol{\theta})]$. Choose as follows:

$$\tilde{f}(\boldsymbol{\theta}) = f(\boldsymbol{\theta}) - a(h(\boldsymbol{\theta}) - \mathbb{E}[h(\boldsymbol{\theta})]),$$

where $h(\cdot)$ is a chosen function with known expectation, a is a coefficient. Variance of $\tilde{f}(\boldsymbol{\theta})$ given by

$$\mathbb{V}[\tilde{f}] = \mathbb{V}[f] - 2a\,\mathrm{Cov}(f, h) + a^2\mathbb{V}[h].$$

Minimising the above gives the optimal value for the coefficient:

$$a^* = \frac{\mathrm{Cov(f, h)}}{\mathbb{V}[h]} = \sqrt{\frac{\mathbb{V}[f]}{\mathbb{V}[h]}}\,\mathrm{Corr}(f, h),$$

which implies that we should choose a $h(\cdot)$ that is correlated with $f(\cdot)$.

Control variates

In practise we choose,

$$h(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\lambda}} \log q(\boldsymbol{\theta}; \boldsymbol{\lambda}).$$

Important property of score function:

$$\begin{aligned}\mathbb{E}_{q(\boldsymbol{\theta};\boldsymbol{\lambda})}[\nabla_{\boldsymbol{\lambda}} \log q(\boldsymbol{\theta}; \boldsymbol{\lambda})] &= \int q(\boldsymbol{\theta}; \boldsymbol{\lambda}) \frac{\nabla_{\boldsymbol{\lambda}} q(\boldsymbol{\theta}; \boldsymbol{\lambda})}{q(\boldsymbol{\theta}; \boldsymbol{\lambda})} d\boldsymbol{\theta} \\ &= \nabla_{\boldsymbol{\lambda}} \int q(\boldsymbol{\theta}; \boldsymbol{\lambda}) d\boldsymbol{\theta} = \nabla_{\boldsymbol{\lambda}} 1 = \mathbf{0}.\end{aligned} \tag{7}$$

Many of the other techniques from Monte Carlo can help:

- Importance Sampling, Quasi Monte Carlo, Rao-Blackwellization.

Black-box variational inference (BBVI)

- BBVI* is variational inference with the score estimator.
- Only need to evaluate the $\log p(\boldsymbol{y}|\boldsymbol{\theta})$, truly black-box.
- Allows discrete latent variables and implicit likelihoods**.
- Variance stabilisation is often difficult in practise.

* Ranganath et al., 2014 ** D Tran 2017

Pathwise estimator: The reparameterisation trick

Law of the Unconscious Statistician (LOTUS):

$$\mathbb{E}_{q(\boldsymbol{\theta};\boldsymbol{\lambda})}[f(\boldsymbol{\theta})] = \mathbb{E}_{p(\boldsymbol{\epsilon})}[f(g(\boldsymbol{\lambda};\boldsymbol{\epsilon}))], \tag{8}$$

where $p(\boldsymbol{\epsilon})$ is a simpler base distribution and $\boldsymbol{\theta} = g(\boldsymbol{\lambda};\boldsymbol{\epsilon})$ output of a deterministic transformation.

An example of an one-liner*:

$$q(\boldsymbol{\theta};\boldsymbol{\lambda}) = \mathcal{N}(\boldsymbol{\theta};\boldsymbol{\mu},\boldsymbol{\Sigma}), \quad p(\boldsymbol{\epsilon}) = \mathcal{N}(\boldsymbol{\theta};\mathbf{0},\mathbf{1}), \quad g(\boldsymbol{\lambda};\boldsymbol{\epsilon}) = \boldsymbol{\mu} + \boldsymbol{L}\boldsymbol{\epsilon},$$

where $\boldsymbol{L}\boldsymbol{L}^T = \boldsymbol{\Sigma}$. There can be other such sampling paths.

- Also known as non-centred parameterisation** in Monte Carlo parlance.
- Doesn't work for distributions whose sampling and density evaluation paths are different, e.g Gamma.

* Devroye 1996. ** Papaspiliopoulos et al., 2007

Pathwise estimator: the reparameterisation trick

Deriving the estimator:

$$\begin{aligned}\nabla_{\boldsymbol{\lambda}}\mathbb{E}_{q(\boldsymbol{\theta};\boldsymbol{\lambda})}[f(\boldsymbol{\theta})] = \nabla_{\boldsymbol{\lambda}}\int q(\boldsymbol{\theta};\boldsymbol{\lambda})f(\boldsymbol{\theta})d\boldsymbol{\theta} = \nabla_{\boldsymbol{\lambda}}\int f(g(\boldsymbol{\lambda};\boldsymbol{\epsilon}))p(\boldsymbol{\epsilon})d\boldsymbol{\epsilon} \\ = \mathbb{E}_{p(\boldsymbol{\epsilon})}[\nabla_{\boldsymbol{\lambda}}f(g(\boldsymbol{\lambda};\boldsymbol{\epsilon}))] \\ = \mathbb{E}_{p(\boldsymbol{\epsilon})}[\nabla_{\boldsymbol{\theta}}f(\boldsymbol{\theta})\nabla_{\boldsymbol{\lambda}}g(\boldsymbol{\lambda};\boldsymbol{\epsilon})].\end{aligned}$$

Now substituting back the expression for the cost we have the ELBO's gradient as:

$$\nabla_{\boldsymbol{\lambda}}\mathcal{L}(\boldsymbol{\lambda}) = \mathbb{E}_{p(\boldsymbol{\epsilon})}[\nabla_{\boldsymbol{\theta}}(\log p(\boldsymbol{y},\boldsymbol{\theta}) - \log q(\boldsymbol{\theta};\boldsymbol{\lambda}))\nabla_{\boldsymbol{\lambda}}g(\boldsymbol{\lambda};\boldsymbol{\epsilon})],$$

where $\boldsymbol{\theta} = g(\boldsymbol{\lambda};\boldsymbol{\epsilon})$, and a noisy gradient estimate as:

$$\hat{\nabla}_{\boldsymbol{\lambda}}\mathcal{L}_(\boldsymbol{\lambda}) = \frac{1}{S}\sum_{s=1}^{S}[\nabla_{\boldsymbol{\theta}}(\log p(\boldsymbol{y},\boldsymbol{\theta}^{(s)}) - \log q(\boldsymbol{\theta}^{(s)};\boldsymbol{\lambda}))\nabla_{\boldsymbol{\lambda}}g(\boldsymbol{\lambda};\boldsymbol{\epsilon}^{(s)})], \quad (9)$$

where $\boldsymbol{\epsilon}^{(s)} \sim p(\boldsymbol{\epsilon})$

Properties of the estimator

- Unbiasedness
  - Unbiased when interchange between differentiation and integration is valid.
  - Valid as long as the cost is differentiable.
- Variance
  - Variance independent of parameter dimensionality.
    $$\mathbb{V}[\hat{\nabla}_{\boldsymbol{\lambda}}\mathcal{L}(\boldsymbol{\lambda})] = \mathbb{V}_{p(\boldsymbol{\epsilon})}[h(\boldsymbol{\epsilon})] = \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\left(h(\boldsymbol{\epsilon}) - \mathbb{E}_{p(\boldsymbol{\epsilon})}[h(\boldsymbol{\epsilon})]\right)^2\right],$$
    where $h(\boldsymbol{\epsilon}) := \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}) \nabla_{\boldsymbol{\lambda}} g(\boldsymbol{\lambda}; \boldsymbol{\epsilon})$.
  - Variance independent of the cost function $f(\boldsymbol{\theta})$.

Automatic differentiation variational inference

Variational inference with the pathwise estimator

Setup:

- Probabilistic model $p(\boldsymbol{y}, \boldsymbol{\theta})$.
- Consider a Gaussian mean-field approximation: $q(\cdot; \boldsymbol{\lambda}) := \prod_{d=1}^{D} \mathcal{N}(\mu_d, \sigma_d^2)$.
- If $\text{supp}(\boldsymbol{\theta}) \in \mathbb{R}_{>0}$, then use a differentiable transformation $\mathcal{T} : \mathbb{R}_{>0} \to \mathbb{R}$, e.g $\exp(\cdot)$.

Automatic Differentiation Variational Inference

Noisy ELBO:

$$\hat{\nabla}_{\boldsymbol{\lambda}}\mathcal{L}_{(}\boldsymbol{\lambda}) = \frac{1}{S}\sum_{s=1}^{S}[\nabla_{\boldsymbol{\theta}}(\log p(\boldsymbol{y}, \boldsymbol{\theta}^{(s)}) - \log q(\boldsymbol{\theta}^{(s)}; \boldsymbol{\lambda}))\nabla_{\boldsymbol{\lambda}} g(\boldsymbol{\lambda}; \boldsymbol{\epsilon}^{(s)})].$$

ADVI[*] algorithm, repeat until convergence:

1. Set $t = 1$
2. Initialise variational parameters $\mu_1^t, \ldots, \mu_d^t, \sigma_1^t, \ldots, \sigma_d^t$.
3. $\boldsymbol{\epsilon}^{(s)} \sim \mathcal{N}(\boldsymbol{\theta}; \mathbf{0}, \mathbf{1}), \quad \forall s.$
4. $\xi_d^{(s)} = g(\epsilon_d; (\mu_d^t, \sigma_d^t)) = \mu_d^t + \sigma_d^t \epsilon_d^{(s)}, \quad \forall (d, s).$
5. $\theta_d^{(s)} = \mathcal{T}(\xi_d^{(s)}), \forall d, s.$ If $\text{supp}(\theta_d) \in \mathbb{R}$ then $\mathcal{T}$ is the identity function.
6. Evaluate $\hat{\nabla}_{\boldsymbol{\lambda}}\mathcal{L}_{(}\boldsymbol{\lambda}_t)$, where $\boldsymbol{\lambda}_t = (\mu_1^t, \ldots, \mu_d^t, \sigma_1^t, \ldots, \sigma_d^t)$.
7. Apply one step of SGD: $\boldsymbol{\lambda}_{t+1} \leftarrow \boldsymbol{\lambda}_t + \rho_t \hat{\nabla}_{\boldsymbol{\lambda}_t}\mathcal{L}(\boldsymbol{\lambda}_t)$. $t \leftarrow t + 1$.
8. GOTO step 3.

Evaluate all the gradients using **automatic differentiation**.

* Kucukelbir et al., 2016

ADVI for massive datasets

If conditional independence exists, then the log likelihood term in the ELBO can be approximated with

$$\sum_{i=1}^{n} p(y_i|\boldsymbol{\theta}) \approx \frac{M}{n} \sum_{i \in \mathcal{I}_{\mathcal{M}}} p(y_i|\boldsymbol{\theta}), \tag{10}$$

where $\mathcal{I}_{\mathcal{M}}$ is a mini-batch of indices of length M, where $M < n$.

- Variational inference with Monte Carlo gradients and sub-sampled likelihood is known as doubly-stochastic*.

* M Titsias 2014

Beyond mean-field approximation

Structured mean-field[1], introduce dependency:

$$q(\boldsymbol{\theta}; \boldsymbol{\lambda}) = \prod_d q(\theta_d | \{\theta_j\}_{j \neq d}; \boldsymbol{\lambda}) \tag{11}$$

Autoregressive[2]:

$$q(\boldsymbol{\theta}; \boldsymbol{\lambda}) = \prod_d q(\theta_d | \theta_{<d}; \boldsymbol{\lambda}) \tag{12}$$

Mixture[3]

$$q(\boldsymbol{\theta}; \boldsymbol{\lambda}) = \sum_r \gamma_r q_r(\theta_r; \lambda_r) \tag{13}$$

Full-rank Gaussian[4]:

$$q(\boldsymbol{\theta}; \boldsymbol{\lambda}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{14}$$

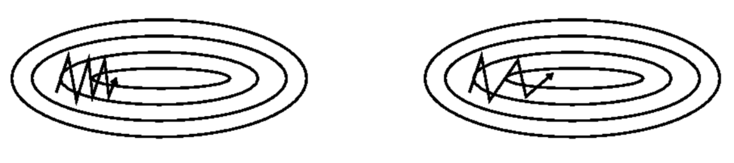
- When combined with $\mathcal{T}$ these become expressive.

[1] Saul and Jordan, 1996. [2] Gregor+ 2015. [3] Tran+ 2016. [4] Kucukelbir+ 2016

ADVI: some practical tips

- Use exact gradient of the entropy: $\mathbb{H}_q := \nabla_{\boldsymbol{\lambda}} \mathbb{E}_{q(\boldsymbol{\theta};\boldsymbol{\lambda})}[q(\boldsymbol{\theta};\boldsymbol{\lambda})]$.
- Optimise log standard deviation $\log \sigma_d$ for mean-field.
- Similarly use log Cholesky parameterisation for full-rank Gaussian.
- Initialise with a small value of σ_d.
- Avoid vanilla SGD.

Avoid vanilla SGD: Use Momentum and adaptive learning rate


(a) SGD without momentum (b) SGD with momentum

- Momentum* accelerates SGD in the relevant direction and dampens oscillations.
- Fraction γ of the update vector of the past time step added.

$$\begin{aligned} \boldsymbol{v}_t &\leftarrow \gamma \boldsymbol{v}_{t-1} + \rho \hat{\nabla}_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{\lambda}_{t-1}) \\ \boldsymbol{\lambda}_t &\leftarrow \boldsymbol{\lambda}_{t-1} - v_t \end{aligned} \tag{15}$$

- Adapt learning rate to each individual parameter, e.g **AdaGrad****, **ADAM**** etc:

* Ning Qian 1999. ** [Duchi et al., 2011, Kingma et al., 2015]

ADVI

- Need differentiable likelihood $p(\boldsymbol{y}|\boldsymbol{\theta})$, not truly black-box.
- Doesn't allows discrete latent variables and implicit likelihoods.
  - Although the Gumbell-softmax* or concrete-reparameterisation** can be applied for discrete variables.
- Variance is much more stable than score function estimator (BBVI).
- Have become the engine behind Bayesian deep learning.

* CJ Maddison et al., 2016 ** E Jang et al., 2016

Practicals

Practicals with ADVI

- First notebook: **linear regression**.
  https://colab.research.google.com/drive/1R1XjFn-uihoDfhMa-o-5qUqfBC0aZ3yz?usp=sharing
- Second notebook: **logistic regression**.
  https://colab.research.google.com/drive/1j0cn7Irfpr1IW6uO65F4VpHMvXc6LfGm?usp=sharing