Nordic probabilistic AI school

Variational Inference and Optimization

Helge Langseth, Andrés Masegosa, and Thomas Dyhre Nielsen

June 14, 2022

Stochastic Gradient Ascent

A small side-step: Gradient Ascent

Why do we talk about this?

We want a way to optimize ELBO using gradient methods. If we can do Bayesian inference as optimization it will play well with, e.g., deep learning frameworks.

Gradient ascent algorithm for maximizing a function $f(\boldsymbol{\lambda})$:

1. Initialize $\boldsymbol{\lambda}^{(0)}$ randomly.
2. For $t = 1, \ldots$:
$$\boldsymbol{\lambda}^{(t)} \leftarrow \boldsymbol{\lambda}^{(t-1)} + \rho \cdot \nabla_{\boldsymbol{\lambda}} f\left(\boldsymbol{\lambda}^{(t-1)}\right)$$

$\boldsymbol{\lambda}^{(t)}$ converges to a (local) optimum of $f(\cdot)$ if:

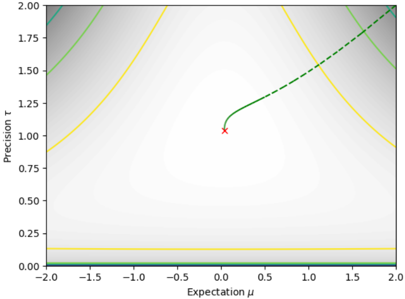
- f is “sufficiently nice”;
- The learning-rate ρ is “sufficiently small”.

Example: ML in a Gaussian model

Example: Maximum log likelihood in a Gaussian model

We have access to $N = 1000$ observations from a Gaussian distribution with unknown mean μ and precision $\tau = 1/\sigma^2$. Use $\boldsymbol{\lambda} = [\mu, \tau]^\intercal$.

$$
\begin{aligned}
f(\boldsymbol{\lambda}) &= \sum_{i=1}^{N} \log p(x_i \,|\, \boldsymbol{\lambda}) = \frac{N}{2}\log\tau - \frac{N}{2}\log(2\pi) - \frac{\tau}{2}\sum_{i=1}^{N}(x_i - \mu)^2 \\
\nabla_{\boldsymbol{\lambda}} f(\boldsymbol{\lambda}) &= \begin{bmatrix} -N\tau\mu + \tau\sum_{i=1}^{N} x_i \\ \frac{N}{2\tau} - \frac{1}{2}\sum_{i=1}^{N}(x_i - \mu)^2 \end{bmatrix}
\end{aligned}
$$



"Standard" gradient ascent is not enough for ELBO optimization

We won't be able to calculate $\nabla_{\boldsymbol{\lambda}}\mathcal{L}\left(q(\boldsymbol{\theta}\,|\,\boldsymbol{\lambda})\right)$ exactly for (at least) two reasons:

1. We may have to resolve to mini-batching (gradient from "random subset")
2. We may not be able to calculate the gradient exactly even for a mini-batch

...and Stochastic Gradient Ascent

"Standard" gradient ascent is not enough for ELBO optimization

We won't be able to calculate $\nabla_{\boldsymbol{\lambda}} \mathcal{L}\left(q(\boldsymbol{\theta} \,|\, \boldsymbol{\lambda})\right)$ exactly for (at least) two reasons:

1. We may have to resolve to mini-batching (gradient from "random subset")
2. We may not be able to calculate the gradient exactly even for a mini-batch

Stochastic gradient ascent algorithm for maximizing a function $f(\boldsymbol{\lambda})$:

If we have access to $\mathbf{g}(\boldsymbol{\lambda})$ – an **unbiased estimate** of the gradient – it still works!

1. Initialize $\boldsymbol{\lambda}^{(0)}$ randomly.
2. For $t = 1, \ldots$:

$$\boldsymbol{\lambda}^{(t)} \leftarrow \boldsymbol{\lambda}^{(t-1)} + \rho_t \cdot \mathbf{g}\left(\boldsymbol{\lambda}^{(t-1)}\right)$$

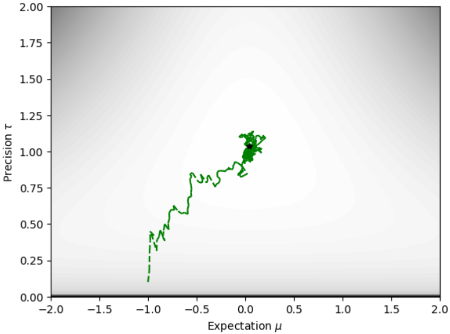
$\boldsymbol{\lambda}_t$ converges to a (local) optimum of $f(\cdot)$ if:

- f is "sufficiently nice";
- $\mathbf{g}(\boldsymbol{\lambda})$ is a random variable with $\mathbb{E}[\mathbf{g}(\boldsymbol{\lambda})] = \nabla_{\boldsymbol{\lambda}} f(\boldsymbol{\lambda})$ and $\mathrm{Var}[\mathbf{g}(\boldsymbol{\lambda})] < \infty$.
- The learning-rates $\{\rho_t\}$ is a Robbins-Monro – sequence:
  - $\sum_t \rho_t = \infty$
  - $\sum_t \rho_t^2 < \infty$

Example: Maximum log likelihood in a Gaussian model

We consider the same maximum likelihood problem, but instead of the gradient based on the full sample, we only have a **mini-batch of a single example** x_t at iteration t:

$$\mathbf{g}(\boldsymbol{\lambda} \mid x_t) = N \cdot \begin{bmatrix} -\tau\mu + \tau x_t \\ \frac{1}{2\tau} - \frac{1}{2}\left(x_t - \mu\right)^2 \end{bmatrix}$$



Black Box Variational Inference

Main idea: Cast inference as an optimization problem

Optimize the ELBO by stochastic gradient ascent over the parameters $\boldsymbol{\lambda}$. If that works, Bayesian inference can be **seamlessly integrated** with building-blocks from other gradient-based machine learning approaches (like deep learning).

Algorithm: Maximize $\mathcal{L}(q) = \mathbb{E}_q\left[\log \frac{p(\boldsymbol{\theta}, \mathcal{D})}{q(\boldsymbol{\theta}|\boldsymbol{\lambda})}\right]$ by gradient ascent

- Initialization:
  - $t \leftarrow 0$;
  - $\hat{\boldsymbol{\lambda}}_0 \leftarrow$ random initialization;
  - $\{\rho_t\} \leftarrow$ a Robbins-Monro sequence.
- Repeat until negligible improvement in terms of $\mathcal{L}(q)$:
  - $t \leftarrow t + 1$;
  - $\hat{\boldsymbol{\lambda}}_t \leftarrow \hat{\boldsymbol{\lambda}}_{t-1} + \rho_t \nabla_{\boldsymbol{\lambda}} \mathcal{L}(q)|_{\hat{\boldsymbol{\lambda}}_{t-1}}$;

Important issue:

Can we calculate $\nabla_{\boldsymbol{\lambda}} \mathcal{L}(q)$ efficiently without adding new restrictive assumptions?

BBVI - calculating the gradient

The algorithm requires that we can find

$$\nabla_{\boldsymbol{\lambda}} \mathcal{L}(q) = \nabla_{\boldsymbol{\lambda}} \mathbb{E}_{\boldsymbol{\theta} \sim q} \left[\log \frac{p(\boldsymbol{\theta}, \mathcal{D})}{q(\boldsymbol{\theta} \,|\, \boldsymbol{\lambda})} \right].$$

Tricky: How can we move the gradient inside the expectation?

Use these properties to simplify the equation:

1. $\nabla_{\boldsymbol{\lambda}} \left(f(\boldsymbol{\theta}, \boldsymbol{\lambda}) \cdot g(\boldsymbol{\theta}, \boldsymbol{\lambda}) \right) = f(\boldsymbol{\theta}, \boldsymbol{\lambda}) \cdot \nabla_{\boldsymbol{\lambda}} g(\boldsymbol{\theta}, \boldsymbol{\lambda}) + g(\boldsymbol{\theta}, \boldsymbol{\lambda}) \cdot \nabla_{\boldsymbol{\lambda}} f(\boldsymbol{\theta}, \boldsymbol{\lambda})$
2. $\nabla_{\boldsymbol{\lambda}} f(\boldsymbol{\theta}, \boldsymbol{\lambda}) = f(\boldsymbol{\theta}, \boldsymbol{\lambda}) \cdot \nabla_{\boldsymbol{\lambda}} \log f(\boldsymbol{\theta}, \boldsymbol{\lambda})$
3. $\mathbb{E}_q \left[\nabla_{\boldsymbol{\lambda}} \log q(\boldsymbol{\theta} \,|\, \boldsymbol{\lambda}) \right] = 0$ for any density function $q(\boldsymbol{\theta} \,|\, \boldsymbol{\lambda})$

Now it follows that

$$\nabla_{\boldsymbol{\lambda}} \mathcal{L}(q) = \mathbb{E}_{\boldsymbol{\theta} \sim q} \left[\log \frac{p(\boldsymbol{\theta}, \mathcal{D})}{q(\boldsymbol{\theta} \,|\, \boldsymbol{\lambda})} \cdot \nabla_{\boldsymbol{\lambda}} \log q(\boldsymbol{\theta} \,|\, \boldsymbol{\lambda}) \right].$$

Calculating the gradient – Things to notice

$$\nabla_{\boldsymbol{\lambda}} \mathcal{L}(q) = \mathbb{E}_{\boldsymbol{\theta} \sim q}\left[\log \frac{p(\boldsymbol{\theta}, \mathcal{D})}{q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})} \cdot \nabla_{\boldsymbol{\lambda}} \log q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})\right].$$

Calculating the gradient – Things to notice

- We still only need access to the joint distribution $p(\boldsymbol{\theta}, \mathcal{D})$ – not $p(\boldsymbol{\theta} \mid \mathcal{D})$.

$$\nabla_{\boldsymbol{\lambda}} \mathcal{L}\left(q\right) = \mathbb{E}_{\boldsymbol{\theta} \sim q}\left[\log \frac{p(\boldsymbol{\theta}, \mathcal{D})}{q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})} \cdot \nabla_{\boldsymbol{\lambda}} \log q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})\right].$$

Calculating the gradient – Things to notice

- We still only need access to the joint distribution $p(\boldsymbol{\theta}, \mathcal{D})$ – not $p(\boldsymbol{\theta} \mid \mathcal{D})$.

$$\nabla_{\boldsymbol{\lambda}} \mathcal{L}(q) = \mathbb{E}_{\boldsymbol{\theta} \sim q}\left[\log \frac{p(\boldsymbol{\theta}, \mathcal{D})}{q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})} \cdot \nabla_{\boldsymbol{\lambda}} \log q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})\right].$$

- $q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})$ factorizes under MF, s.t. we can optimize per variable: $q(\theta_i \mid \boldsymbol{\lambda}_i)$.

Calculating the gradient – Things to notice

- We still only need access to the joint distribution $p(\boldsymbol{\theta}, \mathcal{D})$ – not $p(\boldsymbol{\theta} \mid \mathcal{D})$.

$$\nabla_{\boldsymbol{\lambda}} \mathcal{L}(q) = \mathbb{E}_{\boldsymbol{\theta} \sim q}\left[\log \frac{p(\boldsymbol{\theta}, \mathcal{D})}{q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})} \cdot \nabla_{\boldsymbol{\lambda}} \log q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})\right].$$

- $q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})$ factorizes under MF, s.t. we can optimize per variable: $q(\theta_i \mid \boldsymbol{\lambda}_i)$.
- We must calculate $\nabla_{\boldsymbol{\lambda}_i} \log q(\theta_i \mid \boldsymbol{\lambda}_i)$, which is also known as the "score function".

Calculating the gradient – Things to notice

- We still only need access to the joint distribution $p(\boldsymbol{\theta}, \mathcal{D})$ – not $p(\boldsymbol{\theta} \mid \mathcal{D})$.

$$\nabla_{\boldsymbol{\lambda}} \mathcal{L}(q) = \mathbb{E}_{\boldsymbol{\theta} \sim q} \left[\log \frac{p(\boldsymbol{\theta}, \mathcal{D})}{q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})} \cdot \nabla_{\boldsymbol{\lambda}} \log q(\boldsymbol{\theta} \mid \boldsymbol{\lambda}) \right].$$

- $q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})$ factorizes under MF, s.t. we can optimize per variable: $q(\theta_i \mid \boldsymbol{\lambda}_i)$.
- We must calculate $\nabla_{\boldsymbol{\lambda}_i} \log q(\theta_i \mid \boldsymbol{\lambda}_i)$, which is also known as the "score function".
- The expectation will be approximated using a sample $\{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_M\}$ generated from $q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})$. Hence we require that we can **sample from** each $q(\theta_i \mid \boldsymbol{\lambda}_i)$.

Calculating the gradient – Things to notice

- We still only need access to the joint distribution $p(\boldsymbol{\theta}, \mathcal{D})$ – not $p(\boldsymbol{\theta} \mid \mathcal{D})$.

$$\nabla_{\boldsymbol{\lambda}} \mathcal{L}(q) = \mathbb{E}_{\boldsymbol{\theta} \sim q}\left[\log \frac{p(\boldsymbol{\theta}, \mathcal{D})}{q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})} \cdot \nabla_{\boldsymbol{\lambda}} \log q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})\right].$$

- $q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})$ factorizes under MF, s.t. we can optimize per variable: $q(\theta_i \mid \boldsymbol{\lambda}_i)$.
- We must calculate $\nabla_{\boldsymbol{\lambda}_i} \log q\left(\theta_i \mid \boldsymbol{\lambda}_i\right)$, which is also known as the "score function".
- The expectation will be approximated using a sample $\{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_M\}$ generated from $q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})$. Hence we require that we can **sample from** each $q(\theta_i \mid \boldsymbol{\lambda}_i)$.

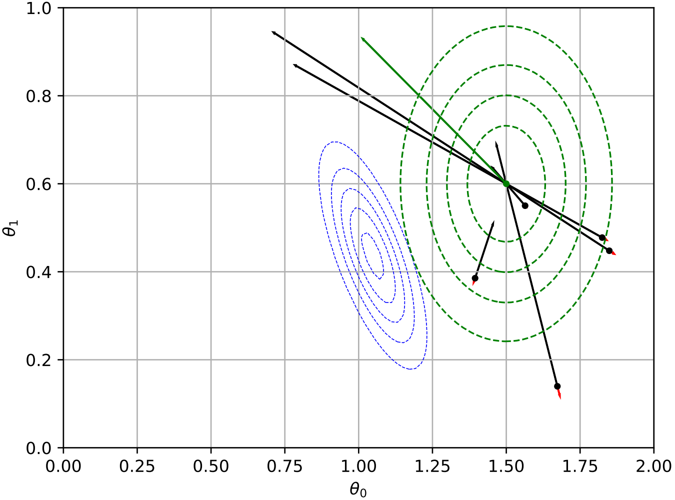
Calculating the gradient – in summary

We have observed the datapoint $\mathcal{D}$, and our current estimate for $\boldsymbol{\lambda}$ is $\hat{\boldsymbol{\lambda}}$. Then

$$\nabla_{\boldsymbol{\lambda}} \mathcal{L}(q)\big|_{\boldsymbol{\lambda}=\hat{\boldsymbol{\lambda}}} \approx \frac{1}{M} \sum_{j=1}^{M} \log \frac{p(\boldsymbol{\theta}_j, \mathcal{D})}{q(\boldsymbol{\theta}_j \mid \hat{\boldsymbol{\lambda}})} \cdot \nabla_{\boldsymbol{\lambda}} \log q(\boldsymbol{\theta}_j \mid \hat{\boldsymbol{\lambda}}).$$

where $\{\boldsymbol{\theta}_1, \ldots \boldsymbol{\theta}_M\}$ are samples from $q(\cdot \mid \hat{\boldsymbol{\lambda}})$. Typically M is small.

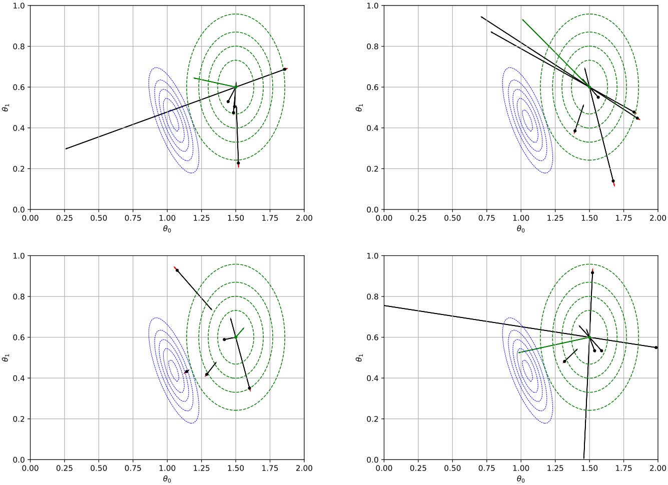
Does it work?



$\color{red}{\nabla_{\boldsymbol{\lambda}} \log q(\boldsymbol{\theta}_i \mid \boldsymbol{\lambda})}$; $\log \frac{p(\boldsymbol{\theta}_i, \mathcal{D})}{q(\boldsymbol{\theta}_i \mid \boldsymbol{\lambda})} \cdot \nabla_{\boldsymbol{\lambda}} \log q(\boldsymbol{\theta}_i \mid \boldsymbol{\lambda})$; $\color{green}{\frac{1}{M} \sum_{i=1}^{m} \log \frac{p(\boldsymbol{\theta}_i, \mathcal{D})}{q(\boldsymbol{\theta}_i \mid \boldsymbol{\lambda})} \cdot \nabla_{\boldsymbol{\lambda}} \log q(\boldsymbol{\theta}_i \mid \boldsymbol{\lambda})}$

Length of gradients increased for visibility. Graphics inspired by Arto Klami @ ProbAI2021.

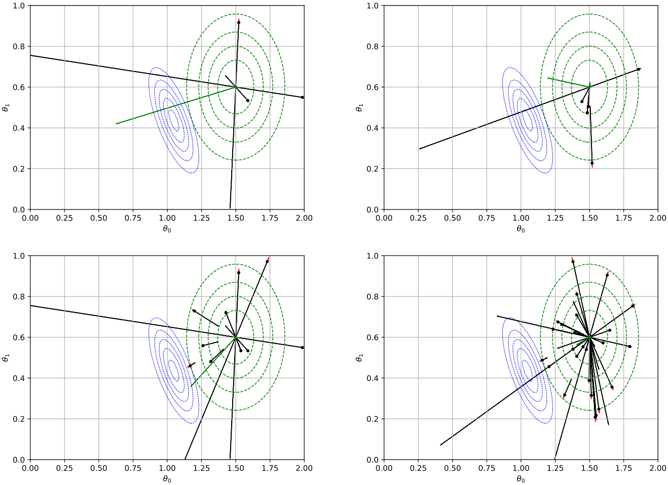
Does it work?



Different samples, each with $M = 5$.

$\nabla_{\boldsymbol{\lambda}} \log q(\boldsymbol{\theta}_i \mid \boldsymbol{\lambda})$; $\log \frac{p(\boldsymbol{\theta}_i, \mathcal{D})}{q(\boldsymbol{\theta}_i \mid \boldsymbol{\lambda})} \cdot \nabla_{\boldsymbol{\lambda}} \log q(\boldsymbol{\theta}_i \mid \boldsymbol{\lambda})$; $\frac{1}{M} \sum_{i=1}^{m} \log \frac{p(\boldsymbol{\theta}_i, \mathcal{D})}{q(\boldsymbol{\theta}_i \mid \boldsymbol{\lambda})} \cdot \nabla_{\boldsymbol{\lambda}} \log q(\boldsymbol{\theta}_i \mid \boldsymbol{\lambda})$

Length of gradients increased for visibility. Graphics inspired by Arto Klami @ ProbAI2021.

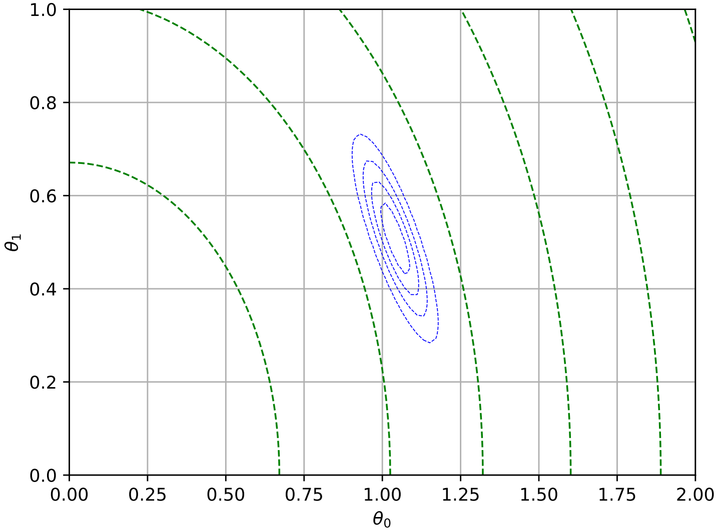


Different values of M ($M = 3$, 5, 10, and 25)

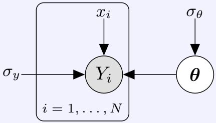
$\nabla_{\boldsymbol{\lambda}} \log q(\boldsymbol{\theta}_i \mid \boldsymbol{\lambda})$; $\log \frac{p(\boldsymbol{\theta}_i, \mathcal{D})}{q(\boldsymbol{\theta}_i \mid \boldsymbol{\lambda})} \cdot \nabla_{\boldsymbol{\lambda}} \log q(\boldsymbol{\theta}_i \mid \boldsymbol{\lambda})$; $\frac{1}{M} \sum_{i=1}^{m} \log \frac{p(\boldsymbol{\theta}_i, \mathcal{D})}{q(\boldsymbol{\theta}_i \mid \boldsymbol{\lambda})} \cdot \nabla_{\boldsymbol{\lambda}} \log q(\boldsymbol{\theta}_i \mid \boldsymbol{\lambda})$

Length of gradients increased for visibility. Graphics inspired by Arto Klami @ ProbAI2021.

Does it work?


1.0
0.8
0.6
0.4
0.2
0.0
θ_1
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
θ_0

Code Task: Score-function gradient for linear regression



- $\boldsymbol{\theta} = \{w_0, w_1\}$, $\boldsymbol{\theta} \sim \mathcal{N}(\mathbf{0}, \sigma_\theta \cdot \mathbf{I}_{2\times 2})$
- $Y_i \mid \{\boldsymbol{\theta}, x_i, \sigma_y\} \sim \mathcal{N}(w_0 + w_1 \cdot x_i, \sigma_y^2)$
- We choose $q_j(\theta_j \mid \boldsymbol{\lambda}_j) = \mathcal{N}(\theta_j \mid \mu_j, \sigma_j^2)$, so $\boldsymbol{\lambda}_j = \{\mu_j, \sigma_j\}$

In this task you will implement the score-function gradient:

$$\nabla_{\boldsymbol{\lambda}} \mathcal{L}(q) = \mathbb{E}_{\boldsymbol{\theta} \sim q}\left[\log \frac{p(\boldsymbol{\theta}, \mathcal{D})}{q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})} \cdot \nabla_{\boldsymbol{\lambda}} \log q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})\right].$$

- Look at `Exercise 1` in the notebook `Day2-AfterLunch/students_BBVI.ipynb`.
- Calculate $\nabla_{\boldsymbol{\lambda}} \log q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})$, i.e., $\frac{\partial}{\partial \mu} \log \mathcal{N}(\mu, \sigma^2)$ and $\frac{\partial}{\partial \sigma} \log \mathcal{N}(\mu, \sigma^2)$ by hand.
- Implement your results in the function `score_function_gradient`.

Let's try to find another trick to compute:

$$\nabla_{\boldsymbol{\lambda}} \mathcal{L}(q) = \nabla_{\boldsymbol{\lambda}} \mathbb{E}_{\boldsymbol{\theta} \sim q}\left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{\theta}, \mathcal{D})}{q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})}\right].$$

The Reparametrization Trick

Let's try to find another trick to compute:

$$\nabla_{\boldsymbol{\lambda}} \mathcal{L}(q) = \nabla_{\boldsymbol{\lambda}} \, \mathbb{E}_{\boldsymbol{\theta} \sim q} \left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{\theta}, \mathcal{D})}{q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})} \right].$$

Let's assume $q(\boldsymbol{\theta}|\boldsymbol{\lambda})$ can be *reparametrized*:

$$\begin{aligned} \boldsymbol{\epsilon} &\sim \phi(\boldsymbol{\epsilon}) \\ \boldsymbol{\theta} &= f(\boldsymbol{\epsilon}, \boldsymbol{\lambda}) \end{aligned}$$

where $\phi(\boldsymbol{\epsilon})$ is some simple distribution that does not depend on $\boldsymbol{\lambda}$ and $f(\boldsymbol{\epsilon}, \boldsymbol{\lambda})$ is a **deterministic transformation**.

The Reparametrization Trick

Let's try to find another trick to compute:

$$\nabla_{\boldsymbol{\lambda}} \mathcal{L}(q) = \nabla_{\boldsymbol{\lambda}} \, \mathbb{E}_{\boldsymbol{\theta} \sim q} \left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{\theta}, \mathcal{D})}{q(\boldsymbol{\theta} \,|\, \boldsymbol{\lambda})} \right].$$

Let's assume $q(\boldsymbol{\theta}|\boldsymbol{\lambda})$ can be *reparametrized*:

$$\begin{aligned} \boldsymbol{\epsilon} &\sim \phi(\boldsymbol{\epsilon}) \\ \boldsymbol{\theta} &= f(\boldsymbol{\epsilon}, \boldsymbol{\lambda}) \end{aligned}$$

where $\phi(\boldsymbol{\epsilon})$ is some simple distribution that does not depend on $\boldsymbol{\lambda}$ and $f(\boldsymbol{\epsilon}, \boldsymbol{\lambda})$ is a **deterministic transformation**.

The common example is $q(\boldsymbol{\theta}|\boldsymbol{\lambda}) = \mathcal{N}(\mu, \sigma)$ *reparametrized* using

$$\begin{aligned} \boldsymbol{\epsilon} &\sim \mathcal{N}(0, 1) \\ \boldsymbol{\theta} &= \mu + \sigma \boldsymbol{\epsilon} \end{aligned}$$

If $q(\boldsymbol{\theta}|\boldsymbol{\lambda})$ can be *reparametrized*:

$$\begin{aligned} \boldsymbol{\epsilon} &\sim \phi(\boldsymbol{\epsilon}) \\ \boldsymbol{\theta} &= f(\boldsymbol{\epsilon}, \boldsymbol{\lambda}) \end{aligned}$$

The Reparametrization Trick

If $q(\boldsymbol{\theta}|\boldsymbol{\lambda})$ can be *reparametrized*:

$$\begin{aligned}\boldsymbol{\epsilon} &\sim \phi(\boldsymbol{\epsilon})\\ \boldsymbol{\theta} &= f(\boldsymbol{\epsilon}, \boldsymbol{\lambda})\end{aligned}$$

Now we can do something different:

$$\nabla_{\boldsymbol{\lambda}} \mathcal{L}\left(q\right) = \nabla_{\boldsymbol{\lambda}} \mathbb{E}_{\boldsymbol{\theta}\sim q}\left[\log \frac{p(\boldsymbol{\theta}, \mathcal{D})}{q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})}\right]$$

The Reparametrization Trick

If $q(\boldsymbol{\theta}|\boldsymbol{\lambda})$ can be *reparametrized*:

$$\begin{aligned}\boldsymbol{\epsilon} &\sim \phi(\boldsymbol{\epsilon})\\ \boldsymbol{\theta} &= f(\boldsymbol{\epsilon}, \boldsymbol{\lambda})\end{aligned}$$

Now we can do something different:

$$\begin{aligned}\nabla_{\boldsymbol{\lambda}}\mathcal{L}(q) &= \nabla_{\boldsymbol{\lambda}}\mathbb{E}_{\boldsymbol{\theta}\sim q}\left[\log\frac{p(\boldsymbol{\theta}, \mathcal{D})}{q(\boldsymbol{\theta}\,|\,\boldsymbol{\lambda})}\right]\\ &= \nabla_{\boldsymbol{\lambda}}\mathbb{E}_{\epsilon\sim\phi}\left[\log\frac{p(f(\boldsymbol{\epsilon}, \boldsymbol{\lambda}), \mathcal{D})}{q(f(\boldsymbol{\epsilon}, \boldsymbol{\lambda})\,|\,\boldsymbol{\lambda})}\right]\end{aligned}$$

The Reparametrization Trick

If $q(\boldsymbol{\theta}|\boldsymbol{\lambda})$ can be *reparametrized*:

$$\begin{aligned} \boldsymbol{\epsilon} &\sim \phi(\boldsymbol{\epsilon}) \\ \boldsymbol{\theta} &= f(\boldsymbol{\epsilon}, \boldsymbol{\lambda}) \end{aligned}$$

Now we can do something different:

$$\begin{aligned} \nabla_{\boldsymbol{\lambda}} \mathcal{L}(q) &= \nabla_{\boldsymbol{\lambda}} \mathbb{E}_{\boldsymbol{\theta} \sim q} \left[\log \frac{p(\boldsymbol{\theta}, \mathcal{D})}{q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})} \right] \\ &= {\color{red}\nabla_{\boldsymbol{\lambda}}} \mathbb{E}_{\epsilon \sim \phi} \left[\log \frac{p(f(\boldsymbol{\epsilon}, \boldsymbol{\lambda}), \mathcal{D})}{q(f(\boldsymbol{\epsilon}, \boldsymbol{\lambda}) \mid \boldsymbol{\lambda})} \right] \\ &= \mathbb{E}_{\epsilon \sim \phi} \left[{\color{red}\nabla_{\boldsymbol{\lambda}}} \log \frac{p(f(\boldsymbol{\epsilon}, \boldsymbol{\lambda}), \mathcal{D})}{q(f(\boldsymbol{\epsilon}, \boldsymbol{\lambda}) \mid \boldsymbol{\lambda})} \right] \end{aligned}$$

The Reparametrization Trick

If $q(\boldsymbol{\theta}|\boldsymbol{\lambda})$ can be *reparametrized*:

$$\begin{aligned}\boldsymbol{\epsilon} &\sim \phi(\boldsymbol{\epsilon}) \\ \boldsymbol{\theta} &= f(\boldsymbol{\epsilon}, \boldsymbol{\lambda})\end{aligned}$$

Now we can do something different:

$$\begin{aligned}\nabla_{\boldsymbol{\lambda}} \mathcal{L}(q) &= \nabla_{\boldsymbol{\lambda}} \mathbb{E}_{\boldsymbol{\theta} \sim q} \left[\log \frac{p(\boldsymbol{\theta}, \mathcal{D})}{q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})} \right] \\ &= \nabla_{\boldsymbol{\lambda}} \mathbb{E}_{\epsilon \sim \phi} \left[\log \frac{p(f(\boldsymbol{\epsilon}, \boldsymbol{\lambda}), \mathcal{D})}{q(f(\boldsymbol{\epsilon}, \boldsymbol{\lambda}) \mid \boldsymbol{\lambda})} \right] \\ &= \mathbb{E}_{\epsilon \sim \phi} \left[\nabla_{\boldsymbol{\lambda}} \log \frac{p(f(\boldsymbol{\epsilon}, \boldsymbol{\lambda}), \mathcal{D})}{q(f(\boldsymbol{\epsilon}, \boldsymbol{\lambda}) \mid \boldsymbol{\lambda})} \right] \\ &= \mathbb{E}_{\epsilon \sim \phi} \left[\nabla_{\boldsymbol{\theta}} \log \frac{p(\boldsymbol{\theta}, \mathcal{D})}{q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})} \nabla_{\boldsymbol{\lambda}} f(\boldsymbol{\epsilon}, \boldsymbol{\lambda}) + \nabla_{\boldsymbol{\lambda}} \log q(\boldsymbol{\theta}|\boldsymbol{\lambda}) \right]\end{aligned}$$

If $q(\boldsymbol{\theta}|\boldsymbol{\lambda})$ can be *reparametrized*:

$$\begin{aligned} \boldsymbol{\epsilon} &\sim \phi(\boldsymbol{\epsilon}) \\ \boldsymbol{\theta} &= f(\boldsymbol{\epsilon}, \boldsymbol{\lambda}) \end{aligned}$$

Now we can do something different:

$$\begin{aligned} \nabla_{\boldsymbol{\lambda}} \mathcal{L}(q) &= \nabla_{\boldsymbol{\lambda}} \mathbb{E}_{\boldsymbol{\theta} \sim q} \left[\log \frac{p(\boldsymbol{\theta}, \mathcal{D})}{q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})} \right] \\ &= \nabla_{\boldsymbol{\lambda}} \mathbb{E}_{\epsilon \sim \phi} \left[\log \frac{p(f(\boldsymbol{\epsilon}, \boldsymbol{\lambda}), \mathcal{D})}{q(f(\boldsymbol{\epsilon}, \boldsymbol{\lambda}) \mid \boldsymbol{\lambda})} \right] \\ &= \mathbb{E}_{\epsilon \sim \phi} \left[\nabla_{\boldsymbol{\lambda}} \log \frac{p(f(\boldsymbol{\epsilon}, \boldsymbol{\lambda}), \mathcal{D})}{q(f(\boldsymbol{\epsilon}, \boldsymbol{\lambda}) \mid \boldsymbol{\lambda})} \right] \\ &= \mathbb{E}_{\epsilon \sim \phi} \left[\nabla_{\boldsymbol{\theta}} \log \frac{p(\boldsymbol{\theta}, \mathcal{D})}{q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})} \nabla_{\boldsymbol{\lambda}} f(\boldsymbol{\epsilon}, \boldsymbol{\lambda}) + \nabla_{\boldsymbol{\lambda}} \log q(\boldsymbol{\theta}|\boldsymbol{\lambda}) \right] \\ &= \mathbb{E}_{\epsilon \sim \phi} \left[\nabla_{\boldsymbol{\theta}} \log \frac{p(\boldsymbol{\theta}, \mathcal{D})}{q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})} \nabla_{\boldsymbol{\lambda}} f(\boldsymbol{\epsilon}, \boldsymbol{\lambda}) \right] \end{aligned}$$

Monte-Carlo Estimation:

$$\nabla_{\boldsymbol{\lambda}} \mathcal{L}\left(q\right) \quad = \quad \mathbb{E}_{\epsilon\sim\phi}\left[\nabla_{\boldsymbol{\theta}} \log \frac{p(\boldsymbol{\theta}, \mathcal{D})}{q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})} \nabla_{\boldsymbol{\lambda}} f(\boldsymbol{\epsilon}, \boldsymbol{\lambda})\right]$$

The Reparametrization Trick

Monte-Carlo Estimation:

$$\begin{aligned}\nabla_{\boldsymbol{\lambda}} \mathcal{L}(q) &= \mathbb{E}_{\epsilon \sim \phi}\left[\nabla_{\boldsymbol{\theta}} \log \frac{p(\boldsymbol{\theta}, \mathcal{D})}{q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})} \nabla_{\boldsymbol{\lambda}} f(\boldsymbol{\epsilon}, \boldsymbol{\lambda})\right] \\ &\approx \frac{1}{M} \sum_{j=1}^{M} \nabla_{\boldsymbol{\theta}} \log \frac{p(\boldsymbol{\theta}_j, \mathcal{D})}{q(\boldsymbol{\theta}_j \mid \boldsymbol{\lambda})} \nabla_{\boldsymbol{\lambda}} f(\boldsymbol{\epsilon}_j, \boldsymbol{\lambda}) \quad : \boldsymbol{\epsilon}_j \sim \phi(\boldsymbol{\epsilon}),\ \boldsymbol{\theta}_j = f(\boldsymbol{\epsilon}_j, \boldsymbol{\lambda})\end{aligned}$$

Monte-Carlo Estimation:

$$
\begin{aligned}
\nabla_{\boldsymbol{\lambda}} \mathcal{L}(q) &= \mathbb{E}_{\epsilon\sim\phi}\left[\nabla_{\boldsymbol{\theta}} \log \frac{p(\boldsymbol{\theta}, \mathcal{D})}{q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})} \nabla_{\boldsymbol{\lambda}} f(\boldsymbol{\epsilon}, \boldsymbol{\lambda})\right] \\
&\approx \frac{1}{M}\sum_{j=1}^{M} \nabla_{\boldsymbol{\theta}} \log \frac{p(\boldsymbol{\theta}_j, \mathcal{D})}{q(\boldsymbol{\theta}_j \mid \boldsymbol{\lambda})} \nabla_{\boldsymbol{\lambda}} f(\boldsymbol{\epsilon}_j, \boldsymbol{\lambda}) \quad : \boldsymbol{\epsilon}_j \sim \phi(\boldsymbol{\epsilon}),\ \boldsymbol{\theta}_j = f(\boldsymbol{\epsilon}_j, \boldsymbol{\lambda}) \\
&= \frac{1}{M}\sum_{j=1}^{M} \Big(\underbrace{\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}, \mathcal{D})}_{\text{Model's Gradient}} - \nabla_{\boldsymbol{\theta}} \log q(\boldsymbol{\theta} \mid \boldsymbol{\lambda}) \Big) \nabla_{\boldsymbol{\lambda}} f(\boldsymbol{\epsilon}, \boldsymbol{\lambda})
\end{aligned}
$$

Monte-Carlo Estimation:

$$
\begin{aligned}
\nabla_{\boldsymbol{\lambda}} \mathcal{L}(q) &= \mathbb{E}_{\epsilon\sim\phi}\left[\nabla_{\boldsymbol{\theta}} \log \frac{p(\boldsymbol{\theta}, \mathcal{D})}{q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})} \nabla_{\boldsymbol{\lambda}} f(\boldsymbol{\epsilon}, \boldsymbol{\lambda})\right] \\
&\approx \frac{1}{M}\sum_{j=1}^{M} \nabla_{\boldsymbol{\theta}} \log \frac{p(\boldsymbol{\theta}_j, \mathcal{D})}{q(\boldsymbol{\theta}_j \mid \boldsymbol{\lambda})} \nabla_{\boldsymbol{\lambda}} f(\boldsymbol{\epsilon}_j, \boldsymbol{\lambda}) \quad : \boldsymbol{\epsilon}_j \sim \phi(\boldsymbol{\epsilon}),\ \boldsymbol{\theta}_j = f(\boldsymbol{\epsilon}_j, \boldsymbol{\lambda}) \\
&= \frac{1}{M}\sum_{j=1}^{M} \Big(\underbrace{\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}, \mathcal{D})}_{\text{Model's Gradient}} - \nabla_{\boldsymbol{\theta}} \log q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})\Big) \nabla_{\boldsymbol{\lambda}} f(\boldsymbol{\epsilon}, \boldsymbol{\lambda})
\end{aligned}
$$

This gradient estimator directly uses **model's gradients**

The Reparametrization Trick

Monte-Carlo Estimation:

$$
\begin{aligned}
\nabla_{\boldsymbol{\lambda}} \mathcal{L}(q) &= \mathbb{E}_{\epsilon \sim \phi}\left[\nabla_{\boldsymbol{\theta}} \log \frac{p(\boldsymbol{\theta}, \mathcal{D})}{q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})} \nabla_{\boldsymbol{\lambda}} f(\boldsymbol{\epsilon}, \boldsymbol{\lambda})\right] \\
&\approx \frac{1}{M} \sum_{j=1}^{M} \nabla_{\boldsymbol{\theta}} \log \frac{p(\boldsymbol{\theta}_j, \mathcal{D})}{q(\boldsymbol{\theta}_j \mid \boldsymbol{\lambda})} \nabla_{\boldsymbol{\lambda}} f(\boldsymbol{\epsilon}_j, \boldsymbol{\lambda}) \quad : \boldsymbol{\epsilon}_j \sim \phi(\boldsymbol{\epsilon}),\ \boldsymbol{\theta}_j = f(\boldsymbol{\epsilon}_j, \boldsymbol{\lambda}) \\
&= \frac{1}{M} \sum_{j=1}^{M} \Big(\underbrace{\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}, \mathcal{D})}_{\text{Model's Gradient}} - \nabla_{\boldsymbol{\theta}} \log q(\boldsymbol{\theta} \mid \boldsymbol{\lambda}) \Big) \nabla_{\boldsymbol{\lambda}} f(\boldsymbol{\epsilon}, \boldsymbol{\lambda})
\end{aligned}
$$

This gradient estimator directly uses **model's gradients**

- While the **score function estimator** does not.

Monte-Carlo Estimation:

$$
\begin{aligned}
\nabla_{\boldsymbol{\lambda}} \mathcal{L}(q) &= \mathbb{E}_{\epsilon \sim \phi}\left[\nabla_{\boldsymbol{\theta}} \log \frac{p(\boldsymbol{\theta}, \mathcal{D})}{q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})} \nabla_{\boldsymbol{\lambda}} f(\boldsymbol{\epsilon}, \boldsymbol{\lambda})\right] \\
&\approx \frac{1}{M} \sum_{j=1}^{M} \nabla_{\boldsymbol{\theta}} \log \frac{p(\boldsymbol{\theta}_j, \mathcal{D})}{q(\boldsymbol{\theta}_j \mid \boldsymbol{\lambda})} \nabla_{\boldsymbol{\lambda}} f(\boldsymbol{\epsilon}_j, \boldsymbol{\lambda}) \quad : \boldsymbol{\epsilon}_j \sim \phi(\boldsymbol{\epsilon}),\ \boldsymbol{\theta}_j = f(\boldsymbol{\epsilon}_j, \boldsymbol{\lambda}) \\
&= \frac{1}{M} \sum_{j=1}^{M} \Big(\underbrace{\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}, \mathcal{D})}_{\text{Model's Gradient}} - \nabla_{\boldsymbol{\theta}} \log q(\boldsymbol{\theta} \mid \boldsymbol{\lambda}) \Big) \nabla_{\boldsymbol{\lambda}} f(\boldsymbol{\epsilon}, \boldsymbol{\lambda})
\end{aligned}
$$

This gradient estimator directly uses **model's gradients**

- While the **score function estimator** does not.
- $\log p(\boldsymbol{\theta}, \mathcal{D})$ needs to be differentiable wrt $\boldsymbol{\theta}$ (i.e. **no discrete variables**).
- $q(\boldsymbol{\theta}|\boldsymbol{\lambda})$ needs to be **differentiable** and **reparametrizable**

Reparameterization can be done for a **(growing) set of distributions**:

Target	$p(z;\theta)$	Base $p(\epsilon)$	One-liner $g(\epsilon;\theta)$
Exponential	$\exp(-x); x > 0$	$\epsilon \sim [0;1]$	$\ln(1/\epsilon)$
Cauchy	$\frac{1}{\pi(1+x^2)}$	$\epsilon \sim [0;1]$	$\tan(\pi\epsilon)$
Laplace	$\mathcal{L}(0;1) = \exp(-\|x\|)$	$\epsilon \sim [0;1]$	$\ln(\frac{\epsilon_1}{\epsilon_2})$
Laplace	$\mathcal{L}(\mu;b)$	$\epsilon \sim [0;1]$	$\mu - bsgn(\epsilon)\ln(1-2\|\epsilon\|)$
Std Gaussian	$\mathcal{N}(0;1)$	$\epsilon \sim [0;1]$	$\sqrt{\ln(\frac{1}{\epsilon_1})}\cos(2\pi\epsilon_2)$
Gaussian	$\mathcal{N}(\mu;RR^\top)$	$\epsilon \sim \mathcal{N}(0;1)$	$\mu + R\epsilon$
Rademacher	$Rad(\frac{1}{2})$	$\epsilon \sim Bern(\frac{1}{2})$	$2\epsilon - 1$
Log-Normal	$\ln\mathcal{N}(\mu;\sigma)$	$\epsilon \sim \mathcal{N}(\mu;\sigma^2)$	$\exp(\epsilon)$
Inv Gamma	$i\mathcal{G}(k;\theta)$	$\epsilon \sim \mathcal{G}(k;\theta^{-1})$	$\frac{1}{\epsilon}$

Table from http://blog.shakirm.com/2015/10/ machine-learning-trick-of-the-day-4-reparameterisation-tricks/

Reparametrizable Distributions

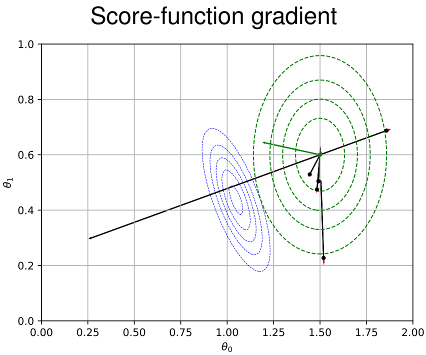
Reparameterization can be done for a **(growing) set of distributions**:

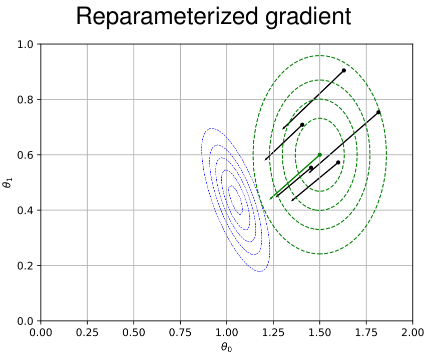
Target	$p(z;\theta)$	Base $p(\epsilon)$	One-liner $g(\epsilon;\theta)$
Exponential	$\exp(-x); x > 0$	$\epsilon \sim [0;1]$	$\ln(1/\epsilon)$
Cauchy	$\frac{1}{\pi(1+x^2)}$	$\epsilon \sim [0;1]$	$\tan(\pi\epsilon)$
Laplace	$\mathcal{L}(0;1) = \exp(-\lvert x\rvert)$	$\epsilon \sim [0;1]$	$\ln(\frac{\epsilon_1}{\epsilon_2})$
Laplace	$\mathcal{L}(\mu;b)$	$\epsilon \sim [0;1]$	$\mu - b sgn(\epsilon) \ln(1 - 2\lvert\epsilon\rvert)$
Std Gaussian	$\mathcal{N}(0;1)$	$\epsilon \sim [0;1]$	$\sqrt{\ln(\frac{1}{\epsilon_1})} \cos(2\pi\epsilon_2)$
Gaussian	$\mathcal{N}(\mu; RR^\top)$	$\epsilon \sim \mathcal{N}(0;1)$	$\mu + R\epsilon$
Rademacher	$Rad(\frac{1}{2})$	$\epsilon \sim Bern(\frac{1}{2})$	$2\epsilon - 1$
Log-Normal	$\ln \mathcal{N}(\mu;\sigma)$	$\epsilon \sim \mathcal{N}(\mu;\sigma^2)$	$\exp(\epsilon)$
Inv Gamma	$i\mathcal{G}(k;\theta)$	$\epsilon \sim \mathcal{G}(k;\theta^{-1})$	$\frac{1}{\epsilon}$

Table from http://blog.shakirm.com/2015/10/ machine-learning-trick-of-the-day-4-reparameterisation-tricks/

A nice survey with more recent developments (very active area of research)

Zhang, Cheng, et al. "Advances in variational inference." IEEE transactions on pattern analysis and machine intelligence 41.8 (2018): 2008-2026.



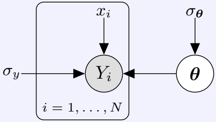


Length of gradients increased for visibility. Graphics inspired by Arto Klami @ ProbAI2021.

Notice the direction of each sample's gradient:

- **Score-function gradient:** Towards the mode of q
- **Reparameterization-gradient:** (Approximately) towards the mode of p

Code Task: Reparameterization-gradient for linear regression



- $\boldsymbol{\theta} = \{w_0, w_1\}$, $\boldsymbol{\theta} \sim \mathcal{N}(\mathbf{0}, \sigma_{\boldsymbol{\theta}} \cdot \mathbf{I}_{2\times 2})$
- $Y_i \mid \{\boldsymbol{\theta}, x_i, \sigma_y\} \sim \mathcal{N}(w_0 + w_1 \cdot x_i, \sigma_y^2)$

In this task you will implement the score-function gradient:

$$\nabla_{\boldsymbol{\lambda}} \mathcal{L}(q) = \mathop{\mathbb{E}}_{\epsilon \sim \phi}\left[\left(\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}, \mathcal{D}) - \nabla_{\boldsymbol{\theta}} \log q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})\right) \nabla_{\boldsymbol{\lambda}} f(\boldsymbol{\epsilon}, \boldsymbol{\lambda})\right]$$

- Calculate $\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}, \mathcal{D})$, $\nabla_{\boldsymbol{\theta}} \log q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})$ and $\nabla_{\boldsymbol{\lambda}} f(\boldsymbol{\epsilon}, \boldsymbol{\lambda})$ for this model.
- Implement these gradients in the `Exercise 2` in
  `Day2-AfterLunch/students_BBVI.ipynb`.
- Experiment with the number of Monte-Carlo samples M per iteration, the learning-rate, and the number of iterations.

Summary

Reparametrization: Gradients align with model's gradient ($\nabla_{\boldsymbol{\theta}} \ln p(\mathcal{D}, \boldsymbol{\theta})$). But:

Summary

Reparametrization: Gradients align with model's gradient ($\nabla_{\boldsymbol{\theta}} \ln p(\mathcal{D}, \boldsymbol{\theta})$). But:

- Requires $q(\boldsymbol{\theta}|\boldsymbol{\lambda})$ to be **reparametrizable**.
- Requires $\ln p(\mathcal{D}, \boldsymbol{\theta})$ and $\ln q(\boldsymbol{\theta}|\boldsymbol{\lambda})$ be **differentiable** (i.e. no categorical variables).

Reparametrization: Gradients align with model's gradient ($\nabla_{\boldsymbol{\theta}} \ln p(\mathcal{D}, \boldsymbol{\theta})$). But:

- Requires $q(\boldsymbol{\theta}|\boldsymbol{\lambda})$ to be **reparametrizable**.
- Requires $\ln p(\mathcal{D}, \boldsymbol{\theta})$ and $\ln q(\boldsymbol{\theta}|\boldsymbol{\lambda})$ be **differentiable** (i.e. no categorical variables).

Score Function: Gradients point towards the **mode of the approximation**, and the **only way the model influences them** is through $\log p(\mathcal{D}, \boldsymbol{\theta})$ in the weights.

Reparametrization: Gradients align with model's gradient ($\nabla_{\boldsymbol{\theta}} \ln p(\mathcal{D}, \boldsymbol{\theta})$). But:

- Requires $q(\boldsymbol{\theta}|\boldsymbol{\lambda})$ to be **reparametrizable**.
- Requires $\ln p(\mathcal{D}, \boldsymbol{\theta})$ and $\ln q(\boldsymbol{\theta}|\boldsymbol{\lambda})$ be **differentiable** (i.e. no categorical variables).

Score Function: Gradients point towards the **mode of the approximation**, and the **only way the model influences them** is through $\log p(\mathcal{D}, \boldsymbol{\theta})$ in the weights.

- Only requires $\ln q(\boldsymbol{\theta}|\boldsymbol{\lambda})$ to be **differentiable**.
- No requirements for $\ln p(\mathcal{D}, \boldsymbol{\theta})$ (only to be computable).

Summary

Reparametrization: Gradients align with model's gradient ($\nabla_{\boldsymbol{\theta}} \ln p(\mathcal{D}, \boldsymbol{\theta})$). But:

- Requires $q(\boldsymbol{\theta}|\boldsymbol{\lambda})$ to be **reparametrizable**.
- Requires $\ln p(\mathcal{D}, \boldsymbol{\theta})$ and $\ln q(\boldsymbol{\theta}|\boldsymbol{\lambda})$ be **differentiable** (i.e. no categorical variables).

Score Function: Gradients point towards the **mode of the approximation**, and the **only way the model influences them** is through $\log p(\mathcal{D}, \boldsymbol{\theta})$ in the weights.

- Only requires $\ln q(\boldsymbol{\theta}|\boldsymbol{\lambda})$ to be **differentiable**.
- No requirements for $\ln p(\mathcal{D}, \boldsymbol{\theta})$ (only to be computable).

Takeaway Message

Score Function is more general, but Reparametrization is better if applicable.

Automatic Variational Inference in PPLs

1. (Manual) Define your data model and the prior.

$$p(\mathcal{D}, \boldsymbol{\theta}) = p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})$$

Automatic Variational Inference in PPLs

1. (Manual) Define your data model and the prior.

$$p(\mathcal{D}, \boldsymbol{\theta}) = p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})$$

2. (Manual/Automatic) Define the variational distribution

$$q(\boldsymbol{\theta}|\boldsymbol{\lambda})$$

1. (Manual) Define your data model and the prior.

$$p(\mathcal{D}, \boldsymbol{\theta}) = p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})$$

2. (Manual/Automatic) Define the variational distribution

$$q(\boldsymbol{\theta}|\boldsymbol{\lambda})$$

3. (Automatic) Optimize the ELBO:

$$\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}_t + \rho\nabla_{\boldsymbol{\lambda}}\mathcal{L}(\boldsymbol{\lambda}_t)$$

Automatic Variational Inference in PPLs

1. (Manual) Define your data model and the prior.

$$p(\mathcal{D}, \boldsymbol{\theta}) = p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})$$

2. (Manual/Automatic) Define the variational distribution

$$q(\boldsymbol{\theta}|\boldsymbol{\lambda})$$

3. (Automatic) Optimize the ELBO:

$$\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}_t + \rho\nabla_{\boldsymbol{\lambda}}\mathcal{L}(\boldsymbol{\lambda}_t)$$

   - Using either score-funtion or reparametrization gradients.

Automatic Variational Inference in PPLs

1. (Manual) Define your data model and the prior.

$$p(\mathcal{D}, \boldsymbol{\theta}) = p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})$$

2. (Manual/Automatic) Define the variational distribution

$$q(\boldsymbol{\theta}|\boldsymbol{\lambda})$$

3. (Automatic) Optimize the ELBO:

$$\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}_t + \rho\nabla_{\boldsymbol{\lambda}}\mathcal{L}(\boldsymbol{\lambda}_t)$$

   - Using either score-funtion or reparametrization gradients.
   - **Automatic-Differentiation engines** take care of gradients.

Automatic Variational Inference in PPLs

1. (Manual) Define your data model and the prior.

$$p(\mathcal{D}, \boldsymbol{\theta}) = p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})$$

2. (Manual/Automatic) Define the variational distribution

$$q(\boldsymbol{\theta}|\boldsymbol{\lambda})$$

3. (Automatic) Optimize the ELBO:

$$\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}_t + \rho\nabla_{\boldsymbol{\lambda}}\mathcal{L}(\boldsymbol{\lambda}_t)$$

   - Using either score-funtion or reparametrization gradients.
   - **Automatic-Differentiation engines** take care of gradients.

4. (Automatic) Approximate inference result

$$q(\boldsymbol{\theta}|\boldsymbol{\lambda}^{\star}) = \arg\min_q \mathrm{KL}\left(q(\boldsymbol{\theta}|\boldsymbol{\lambda})||p(\boldsymbol{\theta}|\mathcal{D})\right)$$

Probabilistic programming: Variational inference in Pyro

`Pyro`

Pyro (`pyro.ai`) is a Python library for probabilistic modeling, inference, and criticism, integrated with PyTorch.

Modeling:
- Directed graphical models
- Neural networks (via `nn.Module`)
- ...

Inference:
- Variational inference – including BBVI, SVI
- Monte Carlo – including Importance sampling and Hamiltonian Monte Carlo
- ...

Criticism:
- Point-based evaluations
- Posterior predictive checks
- ...

... and there are also many other possibilities

`Tensorflow` is integrating probabilistic thinking into its core, `InferPy` is a local alternative, etc.

Simple example

$$\begin{aligned} \text{temp} &\sim \mathcal{N}(15, 2) \\ \text{sensor} &\sim \mathcal{N}(\text{temp}, 1) \end{aligned}$$

$$p(\text{sensor} = 18, \text{temp})$$

Simple example

$$\begin{aligned} \text{temp} &\sim \mathcal{N}(15, 2) \\ \text{sensor} &\sim \mathcal{N}(\text{temp}, 1) \end{aligned}$$

$$p(\text{sensor} = 18, \text{temp})$$

Pyro models:

- random variables $\Leftrightarrow$ `pyro.sample`
- observations $\Leftrightarrow$ `pyro.sample` with the `obs` argument

Simple example

$$\begin{aligned} \text{temp} &\sim \mathcal{N}(15, 2) \\ \text{sensor} &\sim \mathcal{N}(\text{temp}, 1) \end{aligned}$$

$$p(\text{sensor} = 18, \text{temp})$$

Pyro models:

- random variables ⇔ `pyro.sample`
- observations ⇔ `pyro.sample` with the `obs` argument

```
#The observatons
obs = {'sensor': torch.tensor(18.0)}

def model(obs):
    temp = pyro.sample('temp', dist.Normal(15.0, 2.0))
    sensor = pyro.sample('sensor', dist.Normal(temp, 1.0), obs=obs['sensor'])
```

Inference Problem

$$p(\text{temp}|\text{sensor} = 18)$$

Inference Problem

$$p(\text{temp}|\text{sensor} = 18)$$

Variational Solution

$$\min_{q} \mathrm{KL}\left(q(\text{temp})||p(\text{temp}|\text{sensor} = 18)\right)$$

Inference Problem

$$p(\text{temp}|\text{sensor} = 18)$$

Variational Solution

$$\min_q \mathrm{KL}\left(q(\text{temp})||p(\text{temp}|\text{sensor} = 18)\right)$$

Pyro Guides:

- Define the q **distributions** in variational settings.

Pyro guides

Inference Problem

$$p(\text{temp}|\text{sensor} = 18)$$

Variational Solution

$$\min_{q} \mathrm{KL}\left(q(\text{temp})||p(\text{temp}|\text{sensor} = 18)\right)$$

Pyro Guides:

- Define the q **distributions** in variational settings.
- Build **proposal distributions** in importance sampling, MCMC.
- ...

Pyro Guides:

- Guides are **arbitrary stochastic functions**.
- Guides produces samples for those variables of the model which are **not observed**.

Pyro Guides:

- Guides are **arbitrary stochastic functions**.
- Guides produces samples for those variables of the model which are **not observed**.

Guide requirements

1. the guide has the same input signature as the model
2. all unobserved sample statements that appear in the model appear in the guide.

Example

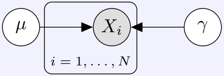
```
#The observatons
obs = {'sensor': torch.tensor(18.0)}

def model(obs):
    temp = pyro.sample('temp', dist.Normal(15.0, 2.0))
    sensor = pyro.sample('sensor', dist.Normal(temp, 1.0), obs=obs['sensor'])
```

```
#The guide
def guide(obs):
    a = pyro.param("mean", torch.tensor(0.0))
    b = pyro.param("scale", torch.tensor(1.), constraint=constraints.positive)
    temp = pyro.sample('temp', dist.Normal(a, b))
```

Code-task: VB for a simple Gaussian model

Exercise: Pyro implementation for a simple Gaussian model

`Day2-AfterLunch/student_simple_gaussian_model_pyro.ipynb`



- $X_i \mid \{\mu, \gamma\} \sim \mathcal{N}(\mu, 1/\gamma)$
- $\mu \sim \mathcal{N}(0, \tau)$
- $\gamma \sim \text{Gamma}(\alpha, \beta)$

- Implement a pyro **guide** for the graphical model above.
- Specify suitable **variational approximation** in the form of a Pyro guide.

$$q(\mu, \gamma) = ...$$

- **Check** the differences with the following notebook (no Pyro implementation).

  `Day2-BeforeLunch/student_simple_model.ipynb`