Nordic probabilistic AI school

Variational Inference and Optimization

Helge Langseth, Andrés Masegosa, and Thomas Dyhre Nielsen

June 17, 2025

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".

"Standard" gradient ascent is not enough for ELBO optimization

We won't be able to calculate $\nabla_{\boldsymbol{\lambda}} \mathcal{L}\left(q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})\right)$ 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

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

Black Box Variational Inference

BBVI - Vanilla version

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

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

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

- We would typically approximate an expectation by a sample average:

$$\mathbb{E}_{\boldsymbol{\theta} \sim q_{\boldsymbol{\lambda}}} [f(\boldsymbol{\theta}, \boldsymbol{\lambda})] \approx \frac{1}{M} \sum_{j=1}^{M} f(\boldsymbol{\theta}_j, \boldsymbol{\lambda}), \text{ with } \{\boldsymbol{\theta}_1, \ldots \boldsymbol{\theta}_M\} \text{ sampled from } q_{\boldsymbol{\lambda}}(\boldsymbol{\theta} \mid \boldsymbol{\lambda}).$$

- This doesn't work when taking a gradient related to the sampling distribution.

BBVI - calculating the gradient

The algorithm requires that we can find

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

Solution: 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. $\mathbb{E}_q \left[\nabla_{\boldsymbol{\lambda}} \log q(\boldsymbol{\theta} \mid \boldsymbol{\lambda}) \right] = 0$ for any density function $q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})$.

Now it follows that

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

This is the so-called **score-function gradient**.

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 $p(\boldsymbol{\theta} \mid \mathcal{D})$.

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

Calculating the gradient – Things to notice

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

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

- $q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})$ 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".

- 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 $\nabla_{\boldsymbol{\lambda}_i} \log q(\theta_i \mid \boldsymbol{\lambda}_i)$, 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 $q(\theta_i \mid \boldsymbol{\lambda}_i)$.

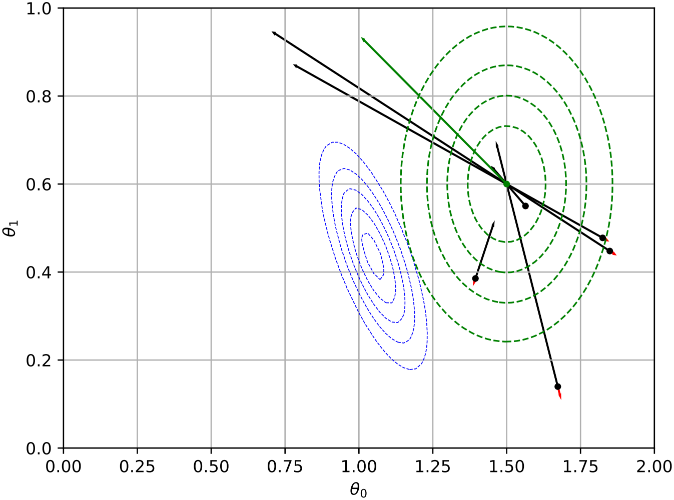
Calculating the gradient – in summary

We have observed the data $\mathcal{D}$, 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 fairly small.

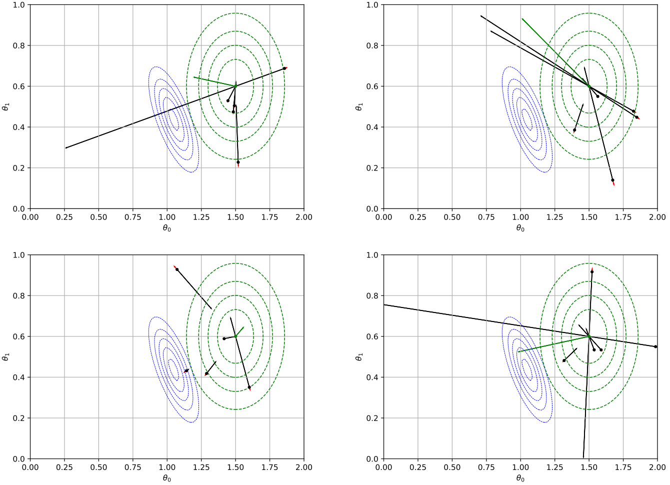
Does it work?



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

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

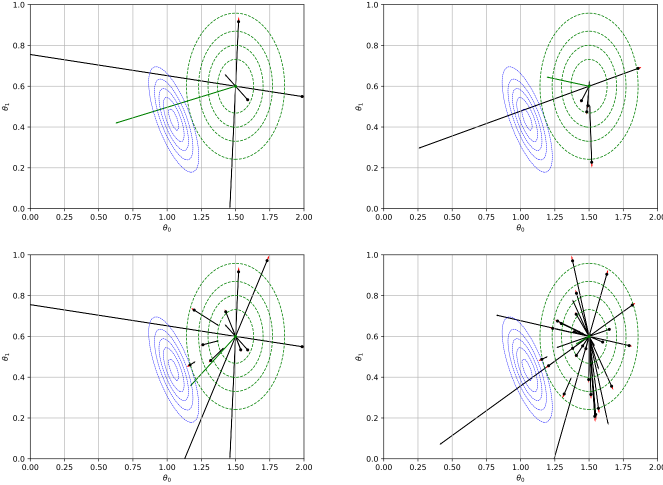
Does it work?



Different samples, each with $M = 5$.

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

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

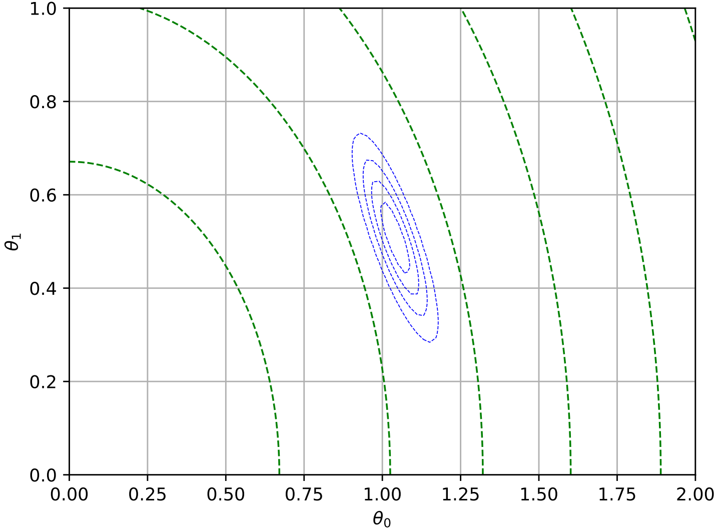


Different values of M ($M = 3$, 5, 10, and 25)

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

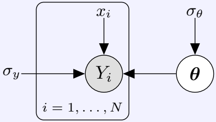
Length of gradients increased for visibility. Graphics inspired by Arto Klami @ ProbAI2021.

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 `AfterLunch/students_BBVI.ipynb`.
- We need $\nabla_{\boldsymbol{\lambda}} \log q(\theta \mid \boldsymbol{\lambda}) = \left[\frac{\partial}{\partial \mu} \log \mathcal{N}(\theta|\mu, \sigma^2), \frac{\partial}{\partial \sigma} \log \mathcal{N}(\theta|\mu, \sigma^2)\right]^\mathsf{T}$:
  You must find $\frac{\partial}{\partial \mu_j} \log \mathcal{N}\left(\theta_j|\mu_j, \sigma_j^2\right)$, but are given $\frac{\partial}{\partial \sigma_j} \log \mathcal{N}\left(\theta_j|\mu_j, \sigma_j^2\right)$.
- Implement your results in the function `score_function_gradient`.

The Reparametrization Trick

Goal: Find a more robust estimator for the gradient

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

The Reparametrization Trick

Goal: Find a more robust estimator for the gradient

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

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

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

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

The Reparametrization Trick

Goal: Find a more robust estimator for the gradient

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

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

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

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

Example: The univariate Gaussian distribution

Assume $q(\theta|\boldsymbol{\lambda}) = \mathcal{N}(\theta \mid \mu, \sigma^2)$, with $\boldsymbol{\lambda} = \{\mu, \sigma\}$. $q(\theta|\boldsymbol{\lambda})$ can be reparametrized by

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

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

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

Now we can calculate the gradient:

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

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

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

Now we can calculate the gradient:

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

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

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

Now we can calculate the gradient:

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

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

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

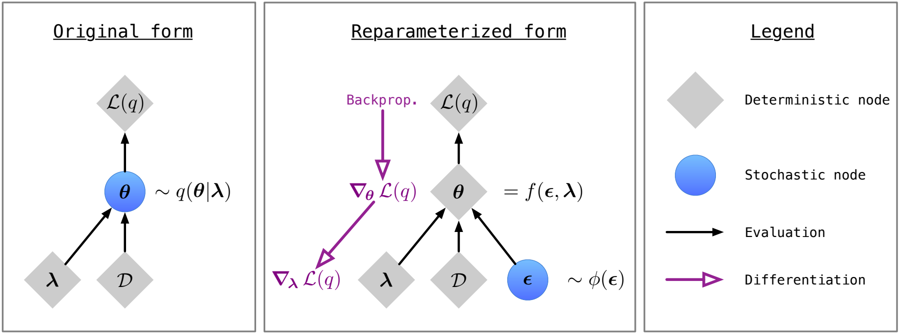
Now we can calculate the gradient:

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

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

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

Now we can calculate the gradient:



Monte-Carlo Estimation:

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

The Reparametrization Trick

Monte-Carlo Estimation:

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

Monte-Carlo Estimation:

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

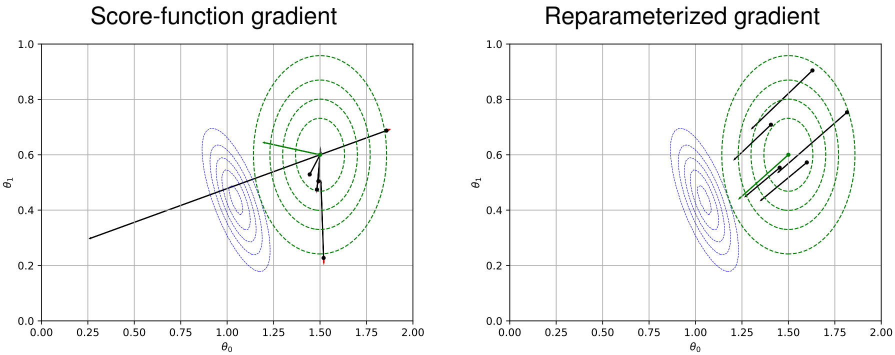
Monte-Carlo Estimation:

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

This gradient estimator...

- Uses the *model's* gradients (not so for the score-function gradient).
- Requires $q(\boldsymbol{\theta}|\boldsymbol{\lambda})$ to be *reparametrizable* and *differentiable* – this time wrt. $\boldsymbol{\theta}$.
- Requires $\log p(\boldsymbol{\theta}, \mathcal{D})$ to be differentiable wrt. $\boldsymbol{\theta}$.

Does it work?



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 high density region of the exact posterior $p(\boldsymbol{\theta}|\mathcal{D})$.

Comparison

Score function gradients:

- Gradients point towards the mode of $q(\boldsymbol{\theta}|\boldsymbol{\lambda})$, while $p(\mathcal{D}, \boldsymbol{\theta})$ only affects the *weights*. We need a *"large" number of samples*, typically in the order of tens to a hundred.
- **Requires** $\ln q(\boldsymbol{\theta}|\boldsymbol{\lambda})$ to be *differentiable wrt.* $\boldsymbol{\lambda}$.
- **Requires** $\ln p(\mathcal{D}, \boldsymbol{\theta})$ to be *computable*.

Reparametrization gradients:

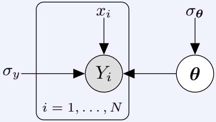
- Gradients utilize the model definition via the term $\nabla_{\boldsymbol{\theta}} \ln p(\mathcal{D}, \boldsymbol{\theta})$. Fairly robust, so we only need a *few samples*, typically only a single one!
- **Requires** $q(\boldsymbol{\theta}|\boldsymbol{\lambda})$ to be *reparametrizable*.
- **Requires** $\ln q(\boldsymbol{\theta}|\boldsymbol{\lambda})$ to be *differentiable wrt.* $\boldsymbol{\theta}$.
- **Requires** $\ln p(\mathcal{D}, \boldsymbol{\theta})$ to be *differentiable wrt.* $\boldsymbol{\theta}$.

Conclusion

The "Score function" approach is more general, but "Reparametrization" will usually provide better results quicker when applicable.

Code-task: Reparameterization-gradient for linear regression

Code Task: Reparameterization-gradient for linear regression



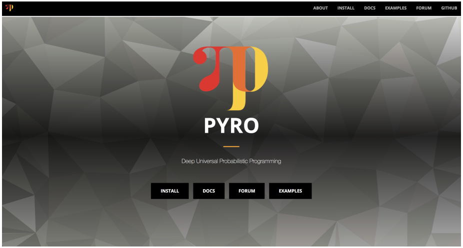
- $\boldsymbol{\theta} = \{w_0, w_1\}$, $\boldsymbol{\theta} \sim \mathcal{N}(\mathbf{0}, \sigma_{\boldsymbol{\theta}} \cdot \mathbf{I}_{2\times 2})$
- $Y_i \mid \{\boldsymbol{\theta}, x_i, \sigma_y\} \sim \mathcal{N}(w_0 + w_1 \cdot x_i, \sigma_y^2)$

In this task you will play with the reparameterization gradient:

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

- We provide $\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}, \mathcal{D})$, $\nabla_{\boldsymbol{\theta}} \log q(\boldsymbol{\theta} \mid \boldsymbol{\lambda})$ and $\nabla_{\boldsymbol{\lambda}} f(\boldsymbol{\epsilon}, \boldsymbol{\lambda})$ for this model.
- Go to `Exercise 2` in
  `AfterLunch/students_BBVI.ipynb`.
- Experiment with the number of Monte-Carlo samples M per iteration, the learning-rate, and the number of iterations. Compare with the output of the Score Function Gradient.

Probabilistic programming: Variational inference in Pyro



Pyro's main features (`www.pyro.ai`) :

- Initially developed by UBER (the car riding company).
- Community of contributors and a dedicated team at Broad Institute (US).
- Rely on Pytorch (Deep Learning Framework).
- Enable GPU accelaration and distributed learning.

https://github.com/PGM-Lab/2025-ProbAI

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.

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.

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

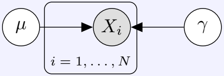
```python
#The observatons
obs = {'sensor': torch.tensor(18.0)}

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'])
```

```python
#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))
```

Exercise: Pyro implementation for a simple Gaussian model

`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).

  `BeforeLunch/student_simple_model.ipynb`