

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

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

① Initialize $\lambda^{(0)}$ randomly.

② For $t = 1, \dots$:

$$\lambda^{(t)} \leftarrow \lambda^{(t-1)} + \rho \cdot \nabla_{\lambda} f \left(\lambda^{(t-1)} \right)$$

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

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

Why do we talk about this?

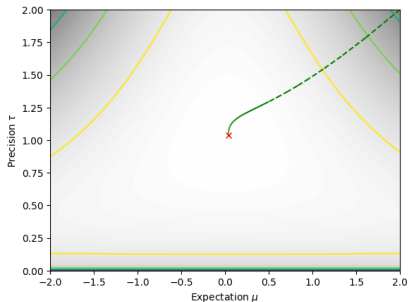
We want a way to optimize ELBO using gradient methods. If we can do Bayesian inference as optimization it will play well with, e.g., deep learning frameworks.

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 $\lambda = [\mu, \tau]^T$.

$$f(\lambda) = \sum_{i=1}^N \log p(x_i | \lambda) = \frac{N}{2} \log \tau - \frac{N}{2} \log(2\pi) - \frac{\tau}{2} \sum_{i=1}^N (x_i - \mu)^2$$

$$\nabla_{\lambda} f(\lambda) = \begin{bmatrix} -N\tau\mu + \tau \sum_{i=1}^N x_i \\ \frac{N}{2\tau} - \frac{1}{2} \sum_{i=1}^N (x_i - \mu)^2 \end{bmatrix} \quad \text{Cost of calculation per step: } O(N)$$



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

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

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

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

λ_t converges to a (local) optimum of $f(\cdot)$ if:

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

Relevance for ELBO optimization

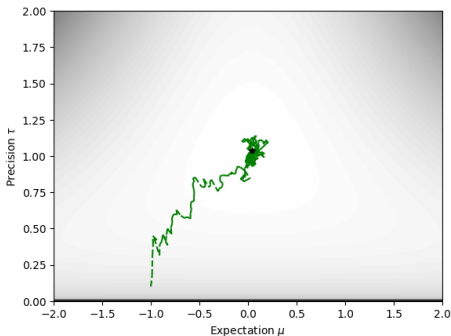
We won't be able to calculate $\nabla_{\lambda} \mathcal{L}(q(\theta | \lambda))$ exactly for (at least) two reasons:

- 1 We may have to resort to mini-batching (gradient from “random subset”)
- 2 We may not be able to calculate the gradient exactly even for the mini-batch

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 :

$$\mathbf{g}(\boldsymbol{\lambda} | x_t) = N \cdot \begin{bmatrix} -\tau\mu + \tau x_t \\ \frac{1}{2\tau} - \frac{1}{2} (x_t - \mu)^2 \end{bmatrix} \quad \text{Cost of calculation per step: } O(1)$$



Black Box Variational Inference

Main idea: Cast inference as an optimization problem

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

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

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

Important issue:

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

The algorithm requires that we can find

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

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

Use these properties to simplify the equation:

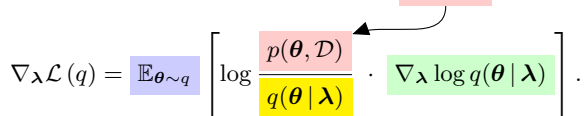
- ❶ $\nabla_{\lambda} (f(\theta, \lambda) \cdot g(\theta, \lambda)) = f(\theta, \lambda) \cdot \nabla_{\lambda} g(\theta, \lambda) + g(\theta, \lambda) \cdot \nabla_{\lambda} f(\theta, \lambda)$
- ❷ $\nabla_{\lambda} f(\theta, \lambda) = f(\theta, \lambda) \cdot \nabla_{\lambda} \log f(\theta, \lambda)$
- ❸ $\mathbb{E}_q [\nabla_{\lambda} \log q(\theta | \lambda)] = 0$ for any density function $q(\theta | \lambda)$

Now it follows that

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

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

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

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


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

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

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

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

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

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

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

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

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

Calculating the gradient – Things to notice

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

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

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

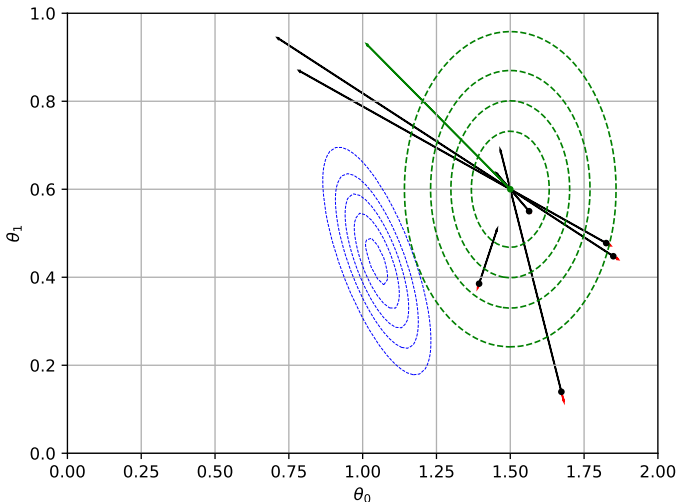
Calculating the gradient – in summary

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

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

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

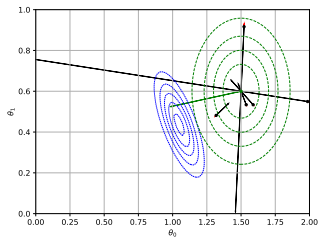
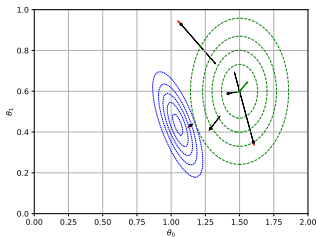
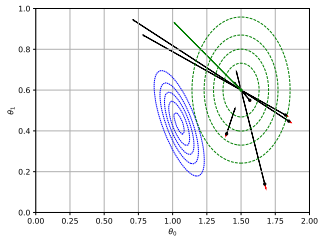
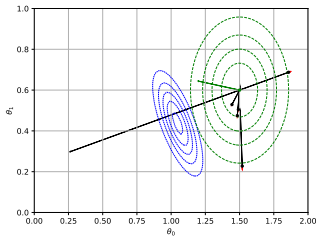
Does it work?



$$\nabla_{\lambda} \log q(\theta_i | \lambda); \quad \log \frac{p(\theta_i, \mathcal{D})}{q(\theta_i | \lambda)} \cdot \nabla_{\lambda} \log q(\theta_i | \lambda); \quad \frac{1}{M} \sum_{i=1}^m \log \frac{p(\theta_i, \mathcal{D})}{q(\theta_i | \lambda)} \cdot \nabla_{\lambda} \log q(\theta_i | \lambda)$$

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

Does it work?

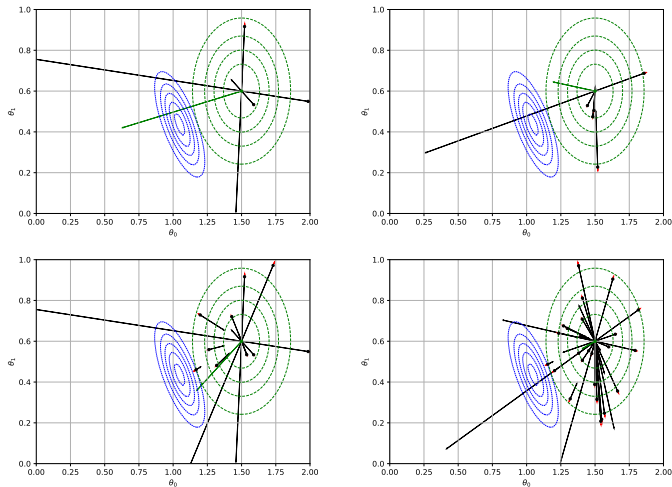


Different samples, each with $M = 5$.

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

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

Does it work?



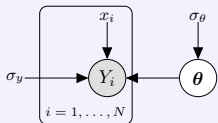
Different values of M ($M = 3, 5, 10$, and 25)

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

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

Does it work?

Code Task: Score-function gradient for linear regression



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

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

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

- Look at `Exercise 1` in the notebook

`Day2-AfterLunch/students_BBVI.ipynb`.

- Calculate $\nabla_{\lambda} \log q(\theta \mid \lambda)$, 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:

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

Let's try to find another trick to compute:

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

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

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

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

Let's try to find another trick to compute:

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

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

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

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

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

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

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

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

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:

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

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:

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

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:

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

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:

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

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:

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

Monte-Carlo Estimation:

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

Monte-Carlo Estimation:

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

Monte-Carlo Estimation:

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

Monte-Carlo Estimation:

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

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

Monte-Carlo Estimation:

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

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

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

Monte-Carlo Estimation:

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

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

- While the **score function estimator** does not.
- $\log p(\theta, \mathcal{D})$ needs to be differentiable wrt θ (i.e. **no discrete variables**).
- $q(\theta | \lambda)$ 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	$\mathcal{L}(\mu; b)$	$\epsilon \sim [0; 1]$	$\mu - b \operatorname{sgn}(\epsilon) \ln(1 - 2 \epsilon)$
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	$\operatorname{Rad}(\frac{1}{2})$	$\epsilon \sim \operatorname{Bern}(\frac{1}{2})$	$2\epsilon - 1$
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/>

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	$\mathcal{L}(\mu; b)$	$\epsilon \sim [0; 1]$	$\mu - b \operatorname{sgn}(\epsilon) \ln(1 - 2 \epsilon)$
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	$\operatorname{Rad}(\frac{1}{2})$	$\epsilon \sim \operatorname{Bern}(\frac{1}{2})$	$2\epsilon - 1$
Log-Normal	$\ln \mathcal{N}(\mu; \sigma)$	$\epsilon \sim \mathcal{N}(\mu; \sigma^2)$	$\exp(\epsilon)$
Inv Gamma	$i\mathcal{G}(k; \theta)$	$\epsilon \sim \mathcal{G}(k; \theta^{-1})$	$\frac{1}{\epsilon}$

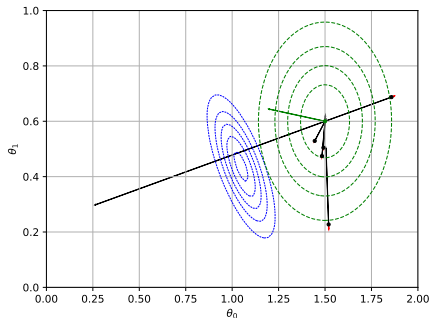
Table from <http://blog.shakirm.com/2015/10/machine-learning-trick-of-the-day-4-reparameterisation-tricks/>

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

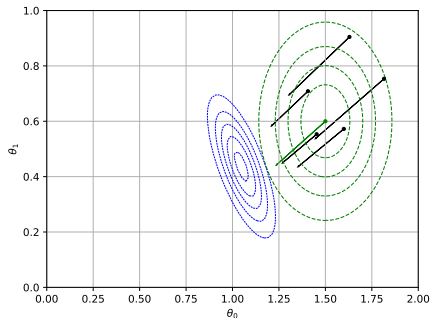
Zhang, Cheng, et al. "Advances in variational inference." IEEE transactions on pattern analysis and machine intelligence 41.8 (2018): 2008-2026.

Does it work?

Score-function gradient



Reparameterized gradient

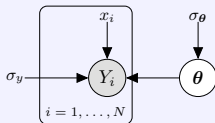


Length of gradients increased for visibility. Graphics inspired by Arto Klami @ ProbAI2021.

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:

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

- 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.

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

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

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

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

- Requires $q(\theta|\lambda)$ to be **reparametrizable**.
- Requires $\ln p(\mathcal{D}, \theta)$ and $\ln q(\theta|\lambda)$ 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 $\log p(\mathcal{D}, \theta)$ in the weights.

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

- Requires $q(\theta|\lambda)$ to be **reparametrizable**.
- Requires $\ln p(\mathcal{D}, \theta)$ and $\ln q(\theta|\lambda)$ 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 $\log p(\mathcal{D}, \theta)$ in the weights.

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

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

- Requires $q(\theta|\lambda)$ to be **reparametrizable**.
- Requires $\ln p(\mathcal{D}, \theta)$ and $\ln q(\theta|\lambda)$ 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 $\log p(\mathcal{D}, \theta)$ in the weights.

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

Takeaway Message

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

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

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

- 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:

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

- 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-function or reparametrization gradients.

- 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-function or reparametrization gradients.
- **Automatic-Differentiation engines** take care of gradients.

- 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-function or reparametrization gradients.
- **Automatic-Differentiation engines** take care of gradients.

- 4 (Automatic) Approximate inference result

$$q(\boldsymbol{\theta}|\boldsymbol{\lambda}^*) = \arg \min_q \text{KL} (q(\boldsymbol{\theta}|\boldsymbol{\lambda})||p(\boldsymbol{\theta}|\mathcal{D}))$$

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

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

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

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

```
1 #The observations
2 obs = {'sensor': torch.tensor(18.0)}
3
4 def model(obs):
5     temp = pyro.sample('temp', dist.Normal(15.0, 2.0))
6     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

$$\min_{\underset{\textcolor{red}{q}}{q}} \text{KL}(\textcolor{red}{q}(\text{temp}) || p(\text{temp}|\text{sensor} = 18))$$

Inference Problem

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

Variational Solution

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

Pyro Guides:

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

Inference Problem

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

Variational Solution

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

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.

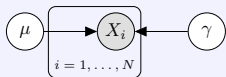
Example

```
1 #The observatons
2 obs = {'sensor': torch.tensor(18.0)}
3
4 def model(obs):
5     temp = pyro.sample('temp', dist.Normal(15.0, 2.0))
6     sensor = pyro.sample('sensor', dist.Normal(temp, 1.0), obs=obs['sensor'])
```

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

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

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

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

Day2-BeforeLunch/student_simple_model.ipynb