PAC-Bayes & Variational Inference

Badr-Eddine Chérief-Abdellatif

badr-eddine.cherief-abdellatif@cnrs.fr

Master 2, Sorbonne Université
Paris, Spring 2024

Overview of the course

The course will be divided in 5 lectures :

- Lecture 1 : Introduction & Motivation
- Lecture 2 : Basics of PAC-Bayes Theory
- Lecture 3 : Advances in PAC-Bayes Theory
- Lecture 4 : Basics of Variational Inference
- Lecture 5 : Advances in Variational Inference

Lecture 4 : Basics of Variational Inference

Outline

Outline

- Variational inference (VI) : compute an approximate posterior distribution by maximizing the ELBO

Outline

- Variational inference (VI) : compute an approximate posterior distribution by maximizing the ELBO
- VI uses Stochastic Gradient Descent (SGD) algorithms to solve the optimization program

Outline

- Variational inference (VI) : compute an approximate posterior distribution by maximizing the ELBO
- VI uses Stochastic Gradient Descent (SGD) algorithms to solve the optimization program
- Some difficulties : compute the gradients of the ELBO

Outline

- Variational inference (VI) : compute an approximate posterior distribution by maximizing the ELBO
- VI uses Stochastic Gradient Descent (SGD) algorithms to solve the optimization program
- Some difficulties : compute the gradients of the ELBO
- Design faster and simpler methods by incorporating natural gradients

Introduction to Variational Inference

A short story of variational inference

- 1760's & 1770's : Bayes' and Laplace's early works on the concept of 'inverse probability'.
- 1950's & 1960's : Bayesian inference is impossible.
- 1970's & 1980's : Early work on approximate Bayesian inference (Metropolis-Hastings, importance sampling).
- 1990's : Gibbs sampling & better computation. Early work on variational inference.
- 2000's : VI in practice. Bayesian models can be fit faster.
- 2010's : VI is scalable and general. Large classes of models & large datasets can be studied.
- 2020's : ?

Notations

Assume that we observe $\mathcal{S}_1$, ..., $\mathcal{S}_n$ i.i.d from P^*. We denote the collection of r.v. $\mathcal{S} = (\mathcal{S}_1, ..., \mathcal{S}_n)$. We consider a model $\{P_\theta, \theta \in \Theta\}$, a prior π on Θ.

Notations

Assume that we observe $\mathcal{S}_1$, ..., $\mathcal{S}_n$ i.i.d from P^*. We denote the collection of r.v. $\mathcal{S} = (\mathcal{S}_1, ..., \mathcal{S}_n)$. We consider a model $\{P_\theta, \theta \in \Theta\}$, a prior π on Θ.

The likelihood

$$p_\theta(\mathcal{S}) = \prod_{i=1}^{n} p_\theta(\mathcal{S}_i)$$

Notations

Assume that we observe $\mathcal{S}_1, \ldots, \mathcal{S}_n$ i.i.d from P^*. We denote the collection of r.v. $\mathcal{S} = (\mathcal{S}_1, ..., \mathcal{S}_n)$. We consider a model $\{P_\theta, \theta \in \Theta\}$, a prior π on Θ.

The likelihood

$$p_\theta(\mathcal{S}) = \prod_{i=1}^{n} p_\theta(\mathcal{S}_i)$$

The posterior

$$\pi_n(\mathrm{d}\theta) \propto p_\theta(\mathcal{S})\pi(\mathrm{d}\theta)$$

Notations

Assume that we observe $\mathcal{S}_1, \ldots, \mathcal{S}_n$ i.i.d from P^*. We denote the collection of r.v. $\mathcal{S} = (\mathcal{S}_1, ..., \mathcal{S}_n)$. We consider a model $\{P_\theta, \theta \in \Theta\}$, a prior π on Θ.

The likelihood

$$p_\theta(\mathcal{S}) = \prod_{i=1}^{n} p_\theta(\mathcal{S}_i)$$

The posterior

$$\pi_n(\mathrm{d}\theta) \propto p_\theta(\mathcal{S})\pi(\mathrm{d}\theta)$$

Remark : the prior is sometimes simply written as $p(\theta)$ and the posterior $p(\theta|\mathcal{S})$.

Frequentist Inference

A typical example of frequentist estimator :

Frequentist Inference

A typical example of frequentist estimator :

Maximum Likelihood Estimation

$$\hat{\theta} = \arg\max_{\theta \in \Theta} \log p_{\theta}(\mathcal{S})$$

Frequentist Inference

A typical example of frequentist estimator :

Maximum Likelihood Estimation

$$\hat{\theta} = \arg\max_{\theta \in \Theta} \log p_\theta(\mathcal{S})$$

which can be computed using

Stochastic Gradient Descent

$$\theta_{t+1} = \theta_t + \eta_t \hat{\nabla}_\theta \log p_{\theta_t}(\mathcal{S})$$

where t is the iteration number, η_t is the step size, and $\hat{\nabla}_\theta \log p_{\theta_t}(\mathcal{S})$ is a stochastic estimate of the gradient of $\theta \mapsto \log p_\theta(\mathcal{S})$ at $\theta = \theta_t$.

Bayesian Inference

Major Difficulty

The exact posterior $\pi_n(d\theta) \propto \prod_{i=1}^n p_\theta(\mathcal{S}_i)\pi(d\theta)$ is often difficult to compute in complex models.

Bayesian Inference

Major Difficulty

The exact posterior $\pi_n(d\theta) \propto \prod_{i=1}^{n} p_\theta(\mathcal{S}_i)\pi(d\theta)$ is often difficult to compute in complex models.

Example : Bayesian Neural Networks

Data $\mathcal{S}_i$ contains input $X_i \in \mathbb{R}^d$ and a scalar output $y_i \in \mathbb{R}$. θ is the vector of network weights. The likelihood $p_\theta(\mathcal{S}_i)$ is a Gaussian distribution $p(y_i|f_\theta(X_i))$ whose parameter $f_\theta(\cdot)$ is a neural network parameterized by θ. The prior is usually taken as $\pi = \mathcal{N}(0, I)$.

Bayesian Inference

Major Difficulty

The exact posterior $\pi_n(d\theta) \propto \prod_{i=1}^n p_\theta(\mathcal{S}_i)\pi(d\theta)$ is often difficult to compute in complex models.

Example : Bayesian Neural Networks

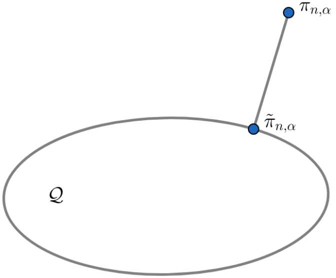
Data $\mathcal{S}_i$ contains input $X_i \in \mathbb{R}^d$ and a scalar output $y_i \in \mathbb{R}$. θ is the vector of network weights. The likelihood $p_\theta(\mathcal{S}_i)$ is a Gaussian distribution $p(y_i|f_\theta(X_i))$ whose parameter $f_\theta(\cdot)$ is a neural network parameterized by θ. The prior is usually taken as $\pi = \mathcal{N}(0, I)$.

How can we compute this posterior ?

Variational Inference

Idea of VI : choose a family $\mathcal{Q}$ of probability distributions on Θ and approximate π_n by a distribution in $\mathcal{Q}$:

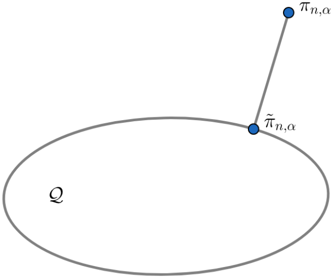
$$\min_{q \in \mathcal{Q}} \mathrm{KL}\left(q \| \pi_n\right).$$



Variational Inference

Idea of VI : choose a family $\mathcal{Q}$ of probability distributions on Θ and approximate π_n by a distribution in $\mathcal{Q}$:

$$\min_{q \in \mathcal{Q}} \mathrm{KL}\left(q \| \pi_n\right).$$



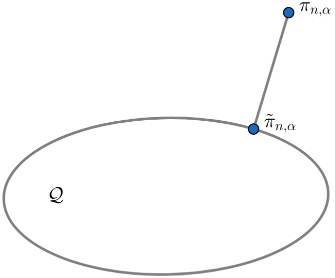
- parametric ($\Theta \subset \mathbb{R}^d$) :

$$\left\{\mathcal{N}(\mu, \Sigma) : \mu \in \mathbb{R}^d, \Sigma \in \mathcal{S}_d^+\right\}.$$

Variational Inference

Idea of VI : choose a family $\mathcal{Q}$ of probability distributions on Θ and approximate π_n by a distribution in $\mathcal{Q}$:

$$\min_{q \in \mathcal{Q}} \mathrm{KL}\left(q \| \pi_n\right).$$



- parametric ($\Theta \subset \mathbb{R}^d$) :

$$\left\{\mathcal{N}(\mu, \Sigma) : \mu \in \mathbb{R}^d, \Sigma \in \mathcal{S}_d^+\right\}.$$

- mean-field ($\Theta = \Theta_1 \times \Theta_2$) :

$$q(\mathrm{d}\theta) = q_1(\mathrm{d}\theta_1) \times q_2(\mathrm{d}\theta_2).$$

Shape of the variational approximation ?

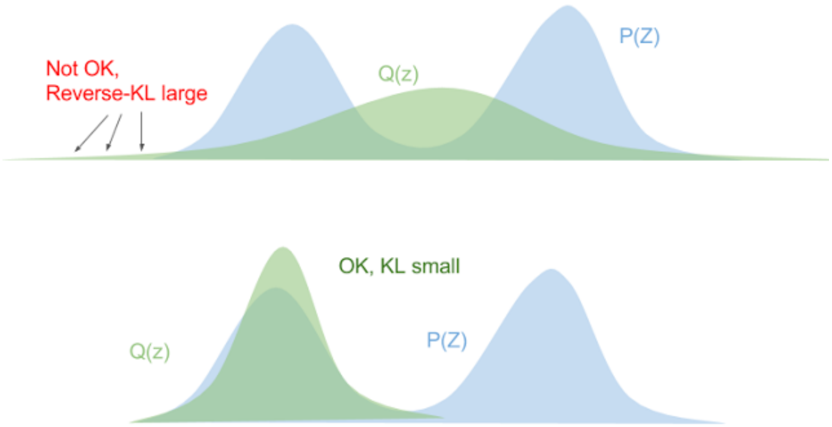
For a bimodal distribution $P(Z)$, what is the shape of its ‘best‘ Gaussian approximation ?

$$\min_{Q \text{ Gaussian}} \mathrm{KL}\left(Q(Z) \| P(Z)\right)?$$

Shape of the variational approximation ?

For a bimodal distribution $P(Z)$, what is the shape of its ‘best‘ Gaussian approximation ?

$$\min_{Q \text{ Gaussian}} \mathrm{KL}\left(Q(Z) \| P(Z)\right)?$$



Evidence lower bound (ELBO)

Define the joint likelihood with $\mathcal{S} = (\mathcal{S}_1, ..., \mathcal{S}_n)$:

$$p(\mathcal{S}, \theta) := \prod_{i=1}^{n} p_\theta(\mathcal{S}_i)\pi(\theta)$$

Evidence lower bound (ELBO)

Define the joint likelihood with $\mathcal{S} = (\mathcal{S}_1, ..., \mathcal{S}_n)$:

$$p(\mathcal{S}, \theta) := \prod_{i=1}^{n} p_\theta(\mathcal{S}_i)\pi(\theta)$$

and the evidence (or marginal likelihood) :

$$p(\mathcal{S}) := \int p(\mathcal{S}, \theta) d\theta.$$

Evidence lower bound (ELBO)

Define the joint likelihood with $\mathcal{S} = (\mathcal{S}_1, ..., \mathcal{S}_n)$:

$$p(\mathcal{S}, \theta) := \prod_{i=1}^{n} p_\theta(\mathcal{S}_i)\pi(\theta)$$

and the evidence (or marginal likelihood) :

$$p(\mathcal{S}) := \int p(\mathcal{S}, \theta) d\theta.$$

Then we have :

$$\log p(\mathcal{S}) = \mathrm{KL}\left(q \| \pi_n\right) + \underbrace{\mathbb{E}_{\theta \sim q}\left[\log\left(\frac{p(\mathcal{S}, \theta)}{q(\theta)}\right)\right]}_{\mathrm{ELBO}(q)}$$

Evidence lower bound (ELBO)

Define the joint likelihood with $\mathcal{S} = (\mathcal{S}_1, ..., \mathcal{S}_n)$:

$$p(\mathcal{S}, \theta) := \prod_{i=1}^{n} p_\theta(\mathcal{S}_i)\pi(\theta)$$

and the evidence (or marginal likelihood) :

$$p(\mathcal{S}) := \int p(\mathcal{S}, \theta) d\theta.$$

Then we have :

$$\log p(\mathcal{S}) = \mathrm{KL}\left(q \| \pi_n\right) + \underbrace{\mathbb{E}_{\theta \sim q}\left[\log\left(\frac{p(\mathcal{S}, \theta)}{q(\theta)}\right)\right]}_{\mathrm{ELBO}(q)}$$

We always have $\mathrm{ELBO}(q) \leq \log p(\mathcal{S})$ (hence the name).

Evidence lower bound (ELBO)

$$\log p(\mathcal{S}) = \mathrm{KL}\left(q \| \pi_n\right) + \underbrace{\mathbb{E}_{\theta\sim q}\left[\log\left(\frac{p(\mathcal{S},\theta)}{q(\theta)}\right)\right]}_{\mathrm{ELBO}(q)}$$

Evidence lower bound (ELBO)

$$\log p(\mathcal{S}) = \mathrm{KL}\left(q \| \pi_n\right) + \underbrace{\mathbb{E}_{\theta \sim q}\left[\log\left(\frac{p(\mathcal{S}, \theta)}{q(\theta)}\right)\right]}_{\mathrm{ELBO}(q)}$$

So we have the equivalent (and tractable) definition of VI :

$$\max_{q \in \mathcal{Q}} \mathrm{ELBO}(q)$$

Evidence lower bound (ELBO)

$$\log p(\mathcal{S}) = \mathrm{KL}\left(q \| \pi_n\right) + \underbrace{\mathbb{E}_{\theta \sim q}\left[\log\left(\frac{p(\mathcal{S},\theta)}{q(\theta)}\right)\right]}_{\mathrm{ELBO}(q)}$$

So we have the equivalent (and tractable) definition of VI :

$$\max_{q \in \mathcal{Q}} \mathrm{ELBO}(q)$$

This raises the question : how to optimize the ELBO ?

Evidence lower bound (ELBO)

$$\log p(\mathcal{S}) = \mathrm{KL}\left(q \| \pi_n\right) + \underbrace{\mathbb{E}_{\theta \sim q}\left[\log\left(\frac{p(\mathcal{S},\theta)}{q(\theta)}\right)\right]}_{\mathrm{ELBO}(q)}$$

So we have the equivalent (and tractable) definition of VI :

$$\max_{q \in \mathcal{Q}} \mathrm{ELBO}(q)$$

This raises the question : how to optimize the ELBO ?

Answer : choose a parametric family and run SGD !

Evidence lower bound (ELBO)

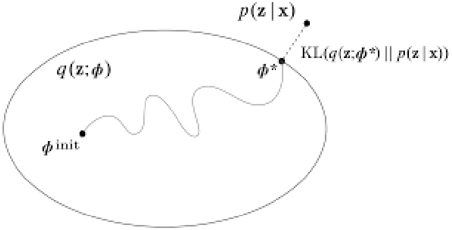
$$\log p(\mathcal{S}) = \mathrm{KL}\left(q \| \pi_n\right) + \underbrace{\mathbb{E}_{\theta\sim q}\left[\log\left(\frac{p(\mathcal{S},\theta)}{q(\theta)}\right)\right]}_{\mathrm{ELBO}(q)}$$

So we have the equivalent (and tractable) definition of VI :

$$\max_{q\in\mathcal{Q}} \mathrm{ELBO}(q)$$

This raises the question : how to optimize the ELBO ?

Answer : choose a parametric family and run SGD !



$p(\mathbf{z}\,|\,\mathbf{x})$
$\mathrm{KL}(q(\mathbf{z};\phi^*)\,||\,p(\mathbf{z}\,|\,\mathbf{x}))$
$q(\mathbf{z};\phi)$
ϕ^*
ϕ^{init}

Parametric VI

We introduce a parametric family $\mathcal{Q} = \{q_\lambda : \lambda \in \Lambda\}$, and perform SGD on the ELBO :

Parametric VI

We introduce a parametric family $\mathcal{Q} = \{q_\lambda : \lambda \in \Lambda\}$, and perform SGD on the ELBO :

$$\lambda_{t+1} = \lambda_t + \eta \hat{\nabla}_\lambda \mathrm{ELBO}(q_{\lambda_t})$$

where t is the iteration number, η is the step size, and $\hat{\nabla}_\lambda \mathrm{ELBO}(q_{\lambda_t})$ is a stochastic estimate of the gradient of $\lambda \mapsto \mathrm{ELBO}(q_\lambda)$ at $\lambda = \lambda_t$.

Parametric VI

We introduce a parametric family $\mathcal{Q} = \{q_\lambda : \lambda \in \Lambda\}$, and perform SGD on the ELBO :

$$\lambda_{t+1} = \lambda_t + \eta \hat{\nabla}_\lambda \mathrm{ELBO}(q_{\lambda_t})$$

where t is the iteration number, η is the step size, and $\hat{\nabla}_\lambda \mathrm{ELBO}(q_{\lambda_t})$ is a stochastic estimate of the gradient of $\lambda \mapsto \mathrm{ELBO}(q_\lambda)$ at $\lambda = \lambda_t$.

This is a very simple and powerful approach that applies to many models and scales to large data, but...

Parametric VI

We introduce a parametric family $\mathcal{Q} = \{q_\lambda : \lambda \in \Lambda\}$, and perform SGD on the ELBO :

$$\lambda_{t+1} = \lambda_t + \eta \hat{\nabla}_\lambda \mathrm{ELBO}(q_{\lambda_t})$$

where t is the iteration number, η is the step size, and $\hat{\nabla}_\lambda \mathrm{ELBO}(q_{\lambda_t})$ is a stochastic estimate of the gradient of $\lambda \mapsto \mathrm{ELBO}(q_\lambda)$ at $\lambda = \lambda_t$.

This is a very simple and powerful approach that applies to many models and scales to large data, but... how about the derivation of the gradients ?

Gradients derivation

Let us recall the expression of the ELBO :

$$\mathrm{ELBO}(q_\lambda) = \mathbb{E}_{\theta\sim q_\lambda}\left[\log\left(\frac{p(\mathcal{S},\theta)}{q_\lambda(\theta)}\right)\right].$$

Gradients derivation

Let us recall the expression of the ELBO :

$$\mathrm{ELBO}(q_\lambda) = \mathbb{E}_{\theta\sim q_\lambda}\left[\log\left(\frac{p(\mathcal{S},\theta)}{q_\lambda(\theta)}\right)\right].$$

The expectation hides the objective dependence on the variational parameters, which makes it hard to directly optimize...

Gradients derivation

Let us recall the expression of the ELBO :

$$\mathrm{ELBO}(q_\lambda) = \mathbb{E}_{\theta \sim q_\lambda}\left[\log\left(\frac{p(\mathcal{S}, \theta)}{q_\lambda(\theta)}\right)\right].$$

The expectation hides the objective dependence on the variational parameters, which makes it hard to directly optimize...

Goal : find a solution to deal with this for a wide class of models. Two solutions :

Gradients derivation

Let us recall the expression of the ELBO :

$$\mathrm{ELBO}(q_\lambda) = \mathbb{E}_{\theta \sim q_\lambda}\left[\log\left(\frac{p(\mathcal{S},\theta)}{q_\lambda(\theta)}\right)\right].$$

The expectation hides the objective dependence on the variational parameters, which makes it hard to directly optimize...

Goal : find a solution to deal with this for a wide class of models. Two solutions :

- Black-Box Variational Inference.
- The Reparameterization Trick.

Black-Box Variational Inference

Black-Box Variational Inference (BBVI)

$$\mathrm{ELBO}(q_\lambda) = \mathbb{E}_{\theta \sim q_\lambda}\left[\log\left(\frac{p(\mathcal{S},\theta)}{q_\lambda(\theta)}\right)\right].$$

Black-Box Variational Inference (BBVI)

$$\mathrm{ELBO}(q_\lambda) = \mathbb{E}_{\theta \sim q_\lambda} \left[\log \left(\frac{p(\mathcal{S}, \theta)}{q_\lambda(\theta)} \right) \right].$$

- First rewrite the gradient of that objective as the expectation of an easy-to-implement function of θ and $\mathcal{S}$, where the expectation is taken with respect to q_λ.

Black-Box Variational Inference (BBVI)

$$\mathrm{ELBO}(q_\lambda) = \mathbb{E}_{\theta\sim q_\lambda}\left[\log\left(\frac{p(\mathcal{S},\theta)}{q_\lambda(\theta)}\right)\right].$$

- First rewrite the gradient of that objective as the expectation of an easy-to-implement function of θ and $\mathcal{S}$, where the expectation is taken with respect to q_λ.
- Then optimize that objective by sampling from q_λ, evaluate the function, and form the corresponding Monte Carlo estimate of the gradient.

Black-Box Variational Inference (BBVI)

$$\mathrm{ELBO}(q_\lambda) = \mathbb{E}_{\theta \sim q_\lambda}\left[\log\left(\frac{p(\mathcal{S},\theta)}{q_\lambda(\theta)}\right)\right].$$

- First rewrite the gradient of that objective as the expectation of an easy-to-implement function of θ and $\mathcal{S}$, where the expectation is taken with respect to q_λ.
- Then optimize that objective by sampling from q_λ, evaluate the function, and form the corresponding Monte Carlo estimate of the gradient.
- Finally use these stochastic gradients in a stochastic optimization algorithm to update λ.

REINFORCE Gradients / Score Gradients

$$\mathrm{ELBO}(q_\lambda) = \mathbb{E}_{\theta\sim q_\lambda}[\underbrace{\log p(\mathcal{S},\theta) - \log q_\lambda(\theta)}_{g_{\mathcal{S}}(\lambda,\theta)}].$$

REINFORCE Gradients / Score Gradients

$$\mathrm{ELBO}(q_\lambda) = \mathbb{E}_{\theta \sim q_\lambda}[\underbrace{\log p(\mathcal{S}, \theta) - \log q_\lambda(\theta)}_{g_{\mathcal{S}}(\lambda,\theta)}].$$

$$\begin{aligned}
\nabla_\lambda \mathrm{ELBO}(q_\lambda) &= \nabla_\lambda \int g_{\mathcal{S}}(\lambda, \theta) q_\lambda(\theta) \mathrm{d}\theta \\
&= \int \left\{ \nabla_\lambda g_{\mathcal{S}}(\lambda, \theta) q_\lambda(\theta) + g_{\mathcal{S}}(\lambda, \theta) \nabla_\lambda q_\lambda(\theta) \right\} \mathrm{d}\theta \\
&= \int \left\{ \nabla_\lambda g_{\mathcal{S}}(\lambda, \theta) q_\lambda(\theta) + g_{\mathcal{S}}(\lambda, \theta) \nabla_\lambda \log q_\lambda(\theta) q_\lambda(\theta) \right\} \mathrm{d}\theta \\
&= \mathbb{E}_{\theta \sim q_\lambda}[\nabla_\lambda g_{\mathcal{S}}(\lambda, \theta) + g_{\mathcal{S}}(\lambda, \theta) \nabla_\lambda \log q_\lambda(\theta)].
\end{aligned}$$

REINFORCE Gradients (cont.)

$$g_{\mathcal{S}}(\lambda, \theta) = \log p(\mathcal{S}, \theta) - \log q_\lambda(\theta).$$

$$\nabla_\lambda \mathrm{ELBO}(q_\lambda) = \mathbb{E}_{\theta \sim q_\lambda}[\nabla_\lambda g_{\mathcal{S}}(\lambda, \theta) + g_{\mathcal{S}}(\lambda, \theta) \nabla_\lambda \log q_\lambda(\theta)].$$

REINFORCE Gradients (cont.)

$$g_{\mathcal{S}}(\lambda, \theta) = \log p(\mathcal{S}, \theta) - \log q_\lambda(\theta).$$

$$\nabla_\lambda \mathrm{ELBO}(q_\lambda) = \mathbb{E}_{\theta \sim q_\lambda}[\nabla_\lambda g_{\mathcal{S}}(\lambda, \theta) + g_{\mathcal{S}}(\lambda, \theta) \nabla_\lambda \log q_\lambda(\theta)].$$

Let's compute the first term of the gradient :

$$\begin{aligned}
\mathbb{E}_{\theta \sim q_\lambda}[\nabla_\lambda g_{\mathcal{S}}(\lambda, \theta)] &= -\mathbb{E}_{\theta \sim q_\lambda}[\nabla_\lambda \log q_\lambda(\theta)] \\
&= -\int \nabla_\lambda \log q_\lambda(\theta) q_\lambda(\theta) d\theta \\
&= -\int \nabla_\lambda q_\lambda(\theta) d\theta \\
&= -\nabla_\lambda \int q_\lambda(\theta) d\theta = -\nabla_\lambda 1 = 0.
\end{aligned}$$

BBVI algorithm

Final value of the gradient $\nabla_\lambda \mathrm{ELBO}(q_\lambda)$:

$$\mathbb{E}_{\theta\sim q_\lambda}[\{\log p(\mathcal{S},\theta) - \log q_\lambda(\theta)\}\nabla_\lambda \log q_\lambda(\theta)]$$

BBVI algorithm

Final value of the gradient $\nabla_\lambda \mathrm{ELBO}(q_\lambda)$:

$$\mathbb{E}_{\theta\sim q_\lambda}[\{\log p(\mathcal{S},\theta) - \log q_\lambda(\theta)\}\nabla_\lambda \log q_\lambda(\theta)]$$

To compute the noisy gradient of the ELBO we need :

- Sampling from q_λ.
- Evaluating $\nabla_\lambda \log q_\lambda$.
- Evaluating $\log q_\lambda$ and $\log p(\mathcal{S},\theta)$.

BBVI algorithm

Final value of the gradient $\nabla_\lambda \mathrm{ELBO}(q_\lambda)$:

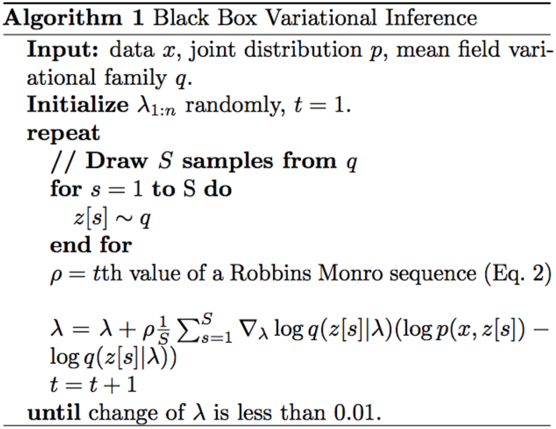
$$\mathbb{E}_{\theta \sim q_\lambda}[\{\log p(\mathcal{S}, \theta) - \log q_\lambda(\theta)\} \nabla_\lambda \log q_\lambda(\theta)]$$

To compute the noisy gradient of the ELBO we need :

- Sampling from q_λ.
- Evaluating $\nabla_\lambda \log q_\lambda$.
- Evaluating $\log q_\lambda$ and $\log p(\mathcal{S}, \theta)$.

The method is not model-specific : black-box criteria are satisfied.

BBVI algorithm



Algorithm 1 Black Box Variational Inference

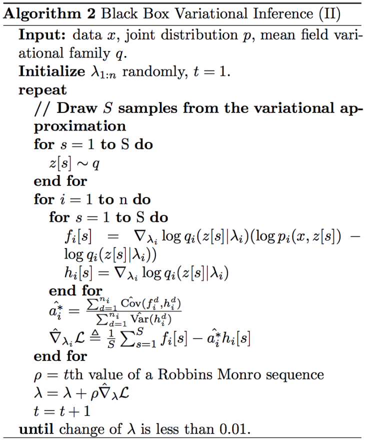
```
Input: data $x$, joint distribution $p$, mean field variational family $q$.
Initialize $\lambda_{1:n}$ randomly, $t = 1$.
repeat
    // Draw $S$ samples from $q$
    for $s = 1$ to S do
        $z[s] \sim q$
    end for
    $\rho = t$th value of a Robbins Monro sequence (Eq. 2)
    $\lambda = \lambda + \rho \frac{1}{S} \sum_{s=1}^{S} \nabla_\lambda \log q(z[s]||\lambda)(\log p(x, z[s]) - \log q(z[s]||\lambda))$
    $t = t + 1$
until change of $\lambda$ is less than 0.01.
```

Problem : BBVI has a large variance...

Can be fixed using Rao-Blackwellization and/or Control Variates.

Problem : BBVI has a large variance...

Can be fixed using Rao-Blackwellization and/or Control Variates.



Algorithm 2 Black Box Variational Inference (II)

```
Input: data $x$, joint distribution $p$, mean field variational family $q$.
Initialize $\lambda_{1:n}$ randomly, $t = 1$.
repeat
    // Draw $S$ samples from the variational approximation
    for $s = 1$ to S do
        $z[s] \sim q$
    end for
    for $i = 1$ to n do
        for $s = 1$ to S do
            $f_i[s] = \nabla_{\lambda_i} \log q_i(z[s]|\lambda_i)(\log p_i(x, z[s]) - \log q_i(z[s]|\lambda_i))$
            $h_i[s] = \nabla_{\lambda_i} \log q_i(z[s]|\lambda_i)$
        end for
        $\hat{a_i^*} = \frac{\sum_{d=1}^{n_i} \hat{\mathrm{Cov}}(f_i^d, h_i^d)}{\sum_{d=1}^{n_i} \hat{\mathrm{Var}}(h_i^d)}$
        $\hat{\nabla}_{\lambda_i} \mathcal{L} \triangleq \frac{1}{S} \sum_{s=1}^{S} f_i[s] - \hat{a_i^*} h_i[s]$
    end for
    $\rho = t$th value of a Robbins Monro sequence
    $\lambda = \lambda + \rho \hat{\nabla}_\lambda \mathcal{L}$
    $t = t + 1$
until change of $\lambda$ is less than 0.01.
```

The reparameterization trick

The reparameterization trick

The reparameterization trick

We assume that $\theta = t_\lambda(\varepsilon)$ for $\varepsilon \sim \nu$ implies $\theta \sim q_\lambda$.

The reparameterization trick

We assume that $\theta = t_\lambda(\varepsilon)$ for $\varepsilon \sim \nu$ implies $\theta \sim q_\lambda$.

Example :

$$\varepsilon \sim \mathcal{N}(0,1),$$

$$\theta = \epsilon\sigma + \mu,$$

$$\theta \sim \mathcal{N}(\mu, \sigma).$$

The reparameterization trick

We assume that $\theta = t_\lambda(\varepsilon)$ for $\varepsilon \sim \nu$ implies $\theta \sim q_\lambda$.

Example :

$$\varepsilon \sim \mathcal{N}(0,1),$$
$$\theta = \epsilon\sigma + \mu,$$
$$\theta \sim \mathcal{N}(\mu,\sigma).$$

The idea of the reparameterization trick is to write the expectation with respect to ν (which does not depend on λ) and then pull the gradient inside the expectation.

The reparameterization trick (cont.)

$$g_{\mathcal{S}}(\lambda, \theta) = \log p(\mathcal{S}, \theta) - \log q_{\lambda}(\theta).$$

The reparameterization trick (cont.)

$$g_{\mathcal{S}}(\lambda,\theta) = \log p(\mathcal{S},\theta) - \log q_\lambda(\theta).$$

$$\begin{aligned}
&\nabla_\lambda \mathrm{ELBO}(q_\lambda) \\
&\quad = \nabla_\lambda \mathbb{E}_{\theta\sim q_\lambda}[g_{\mathcal{S}}(\lambda,\theta)] \\
&\quad = \nabla_\lambda \mathbb{E}_{\varepsilon\sim\nu}[g_{\mathcal{S}}(\lambda,t_\lambda(\varepsilon))] \\
&\quad = \mathbb{E}_{\varepsilon\sim\nu}[\nabla_\lambda\{g_{\mathcal{S}}(\lambda,t_\lambda(\varepsilon))\}] \\
&\quad = \mathbb{E}_{\varepsilon\sim\nu}[\nabla_\lambda t_\lambda(\varepsilon)^T \nabla_\theta g_{\mathcal{S}}(\lambda,t_\lambda(\varepsilon)) + \nabla_\lambda g_{\mathcal{S}}(\lambda,t_\lambda(\varepsilon))] \\
&\quad = \mathbb{E}_{\varepsilon\sim\nu}[\nabla_\lambda t_\lambda(\varepsilon)^T \nabla_\theta\{\log p(\mathcal{S},t_\lambda(\varepsilon)) - \log q_\lambda(t_\lambda(\varepsilon))\} \\
&\qquad\qquad - \nabla_\lambda \log \underbrace{q_\lambda(t_\lambda(\varepsilon))}_{\nu(\varepsilon)}]
\end{aligned}$$

The reparameterization trick (cont.)

$$g_{\mathcal{S}}(\lambda, \theta) = \log p(\mathcal{S}, \theta) - \log q_\lambda(\theta).$$

$$\begin{aligned}
&\nabla_\lambda \mathrm{ELBO}(q_\lambda) \\
&\quad = \nabla_\lambda \mathbb{E}_{\theta \sim q_\lambda}[g_{\mathcal{S}}(\lambda, \theta)] \\
&\quad = \nabla_\lambda \mathbb{E}_{\varepsilon \sim \nu}[g_{\mathcal{S}}(\lambda, t_\lambda(\varepsilon))] \\
&\quad = \mathbb{E}_{\varepsilon \sim \nu}[\nabla_\lambda \{g_{\mathcal{S}}(\lambda, t_\lambda(\varepsilon))\}] \\
&\quad = \mathbb{E}_{\varepsilon \sim \nu}[\nabla_\lambda t_\lambda(\varepsilon)^T \nabla_\theta g_{\mathcal{S}}(\lambda, t_\lambda(\varepsilon)) + \nabla_\lambda g_{\mathcal{S}}(\lambda, t_\lambda(\varepsilon))] \\
&\quad = \mathbb{E}_{\varepsilon \sim \nu}[\nabla_\lambda t_\lambda(\varepsilon)^T \nabla_\theta \{\log p(\mathcal{S}, t_\lambda(\varepsilon)) - \log q_\lambda(t_\lambda(\varepsilon))\} \\
&\qquad\qquad\qquad\qquad - \nabla_\lambda \log \underbrace{q_\lambda(t_\lambda(\varepsilon))}_{\nu(\varepsilon)}]
\end{aligned}$$

and the gradient $\nabla_\lambda \mathrm{ELBO}(q_\lambda)$ is :

$$\mathbb{E}_{\varepsilon \sim \nu}[\nabla_\lambda t_\lambda(\varepsilon)^T \nabla_\theta \{\log p(\mathcal{S}, t_\lambda(\varepsilon)) - \log q_\lambda(t_\lambda(\varepsilon))\}].$$

REINFORCE vs Reparameterization Trick

$$\mathbb{E}_{\theta \sim q_\lambda}[\{\log p(\mathcal{S}, \theta) - \log q_\lambda(\theta)\}\nabla_\lambda \log q_\lambda(\theta)]$$

- Differentiates the density $q_\lambda(\theta)$.
- Works for both discrete and continuous models.
- Works for a large class of variational families.
- But the variance can be a big problem.

REINFORCE vs Reparameterization Trick

$$\mathbb{E}_{\theta\sim q_\lambda}[\{\log p(\mathcal{S},\theta)-\log q_\lambda(\theta)\}\nabla_\lambda \log q_\lambda(\theta)]$$

- Differentiates the density $q_\lambda(\theta)$.
- Works for both discrete and continuous models.
- Works for a large class of variational families.
- But the variance can be a big problem.

$$\mathbb{E}_{\varepsilon\sim\nu}[\nabla_\lambda t_\lambda(\varepsilon)^T\nabla_\theta\{\log p(\mathcal{S},t_\lambda(\varepsilon))-\log q_\lambda(t_\lambda(\varepsilon))\}]$$

- Differentiates the densities $p(\mathcal{S},\theta)$, $q_\lambda(\theta)$.
- Works for differentiable models only.
- Works for reparameterized variational families only.
- But the variance behaves much better.

Natural Gradient Variational Inference

SGD and Information Geometry

The standard SGD algorithm

$$\lambda_{t+1} = \lambda_t + \eta \hat{\nabla}_\lambda \mathrm{ELBO}(q_{\lambda_t})$$

SGD and Information Geometry

The standard SGD algorithm

$$\lambda_{t+1} = \lambda_t + \eta \hat{\nabla}_\lambda \mathrm{ELBO}(q_{\lambda_t})$$

can be rewritten

$$\lambda_{t+1} = \arg\max_\lambda \left\{ \lambda^T \hat{\nabla}_{\lambda} \mathrm{ELBO}(q_{\lambda_t}) - \frac{\|\lambda - \lambda_t\|_2^2}{2\eta} \right\}.$$

SGD and Information Geometry

The standard SGD algorithm

$$\lambda_{t+1} = \lambda_t + \eta \hat{\nabla}_\lambda \mathrm{ELBO}(q_{\lambda_t})$$

can be rewritten

$$\lambda_{t+1} = \arg\max_{\lambda} \left\{ \lambda^T \hat{\nabla}_{\lambda} \mathrm{ELBO}(q_{\lambda_t}) - \frac{\|\lambda - \lambda_t\|_2^2}{2\eta} \right\}.$$

Gradient descent can be seen as steepest descent which tries to prevent the update from moving too far (in Euclidean distance).

SGD and Information Geometry

The standard SGD algorithm

$$\lambda_{t+1} = \lambda_t + \eta \hat{\nabla}_\lambda \mathrm{ELBO}(q_{\lambda_t})$$

can be rewritten

$$\lambda_{t+1} = \arg\max_{\lambda} \left\{ \lambda^T \hat{\nabla}_\lambda \mathrm{ELBO}(q_{\lambda_t}) - \frac{\|\lambda - \lambda_t\|_2^2}{2\eta} \right\}.$$

Gradient descent can be seen as steepest descent which tries to prevent the update from moving too far (in Euclidean distance).

- Is there a 'good' parameterization ?

SGD and Information Geometry

The standard SGD algorithm

$$\lambda_{t+1} = \lambda_t + \eta \hat{\nabla}_\lambda \mathrm{ELBO}(q_{\lambda_t})$$

can be rewritten

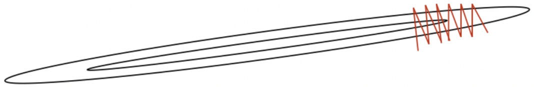
$$\lambda_{t+1} = \arg\max_{\lambda} \left\{ \lambda^T \hat{\nabla}_{\lambda} \mathrm{ELBO}(q_{\lambda_t}) - \frac{\|\lambda - \lambda_t\|_2^2}{2\eta} \right\}.$$

Gradient descent can be seen as steepest descent which tries to prevent the update from moving too far (in Euclidean distance).

- Is there a 'good' parameterization ?
- Is there a 'good' distance ?

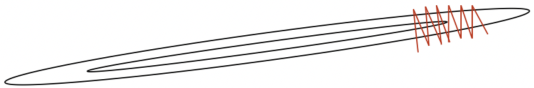
Which choice of the parameterization in SGD ?

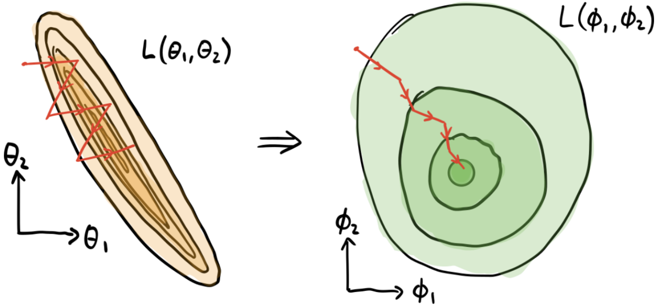
SGD bounces around in high curvature directions and makes slow progress in low curvature directions.



Which choice of the parameterization in SGD ?

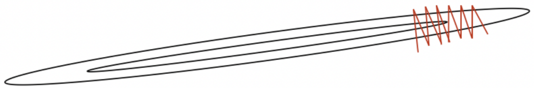
SGD bounces around in high curvature directions and makes slow progress in low curvature directions.

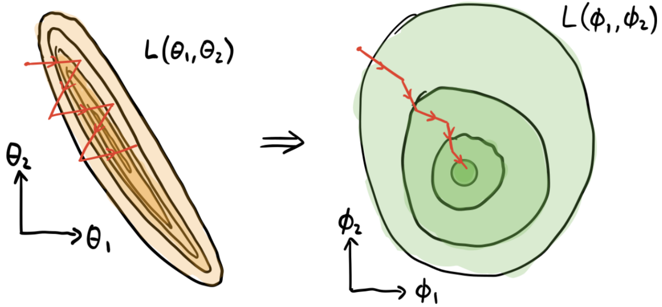




Which choice of the parameterization in SGD ?

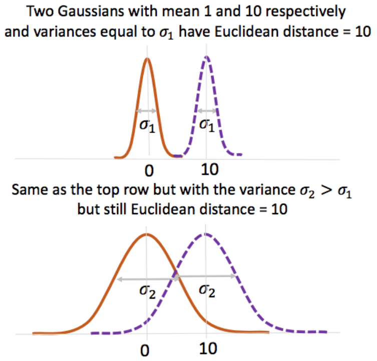
SGD bounces around in high curvature directions and makes slow progress in low curvature directions.





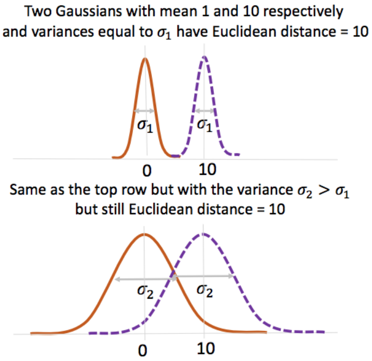
Toward a 'parameterization-invariant' paramaterization in SGD ?

Is the Euclidean distance a good one ?



(a) Gradient-based methods use the Euclidean distance which is a poor metric to measure distance between distributions. The bottom two distributions are almost identical while the top ones barely overlap, yet Euclidean distance is the same.

Is the Euclidean distance a good one ?



(a) Gradient-based methods use the Euclidean distance which is a poor metric to measure distance between distributions. The bottom two distributions are almost identical while the top ones barely overlap, yet Euclidean distance is the same.

Toward a 'distance' between probability distributions ?

Probability distributions & Information Geometry

'Shortest path' between two probability distributions ?

London
Frankfurt
Vancouver
New York
Miami
Cairo
Tel Aviv
Dubai
Bangkok
Singapore
Sao Paulo
Buenos Aires
Perth
Johannesburg

Mapping a manifold to a flat coordinate system distorts distances !

Exponential families

We define the exponential family of sufficient statistic T, natural parameter λ, carrier measure h and log-partition function A :

$$q_\lambda(\theta) = h(\theta) \exp\left(\lambda^T T(\theta) - A(\lambda)\right).$$

Exponential families

We define the exponential family of sufficient statistic T, natural parameter λ, carrier measure h and log-partition function A :

$$q_\lambda(\theta) = h(\theta) \exp\left(\lambda^T T(\theta) - A(\lambda)\right).$$

Example of Gaussian distributions :

$$T(\theta) = [\theta, \theta^2]^T$$

$$\lambda = (m/\sigma^2, -1/2\sigma^2)$$

$$h(\theta) = 1/\sqrt{2\pi}$$

$$A(\lambda) = -\lambda_1^2/4\lambda_2 - \log(-2\lambda_2)/2$$

Exponential families

We define the exponential family of sufficient statistic T, natural parameter λ, carrier measure h and log-partition function A :

$$q_\lambda(\theta) = h(\theta) \exp\left(\lambda^T T(\theta) - A(\lambda)\right).$$

Example of Gaussian distributions :

$$T(\theta) = [\theta, \theta^2]^T$$

$$\lambda = (m/\sigma^2, -1/2\sigma^2)$$

$$h(\theta) = 1/\sqrt{2\pi}$$

$$A(\lambda) = -\lambda_1^2/4\lambda_2 - \log(-2\lambda_2)/2$$

Why exponential families ? Because the 'natural parameterization' adresses both questions by defining a manifold !

The exponential family & Information Geometry

The exponential family induces a Riemannian manifold with a metric defined by the Fisher Information Matrix :

$$\|\lambda - \lambda_t\|^2_{F(\lambda_t)} = (\lambda - \lambda_t)^T F(\lambda_t)(\lambda - \lambda_t)$$

where $F(\lambda) = \mathbb{E}_{\theta \sim q_\lambda} \left[\nabla_\lambda \log q_\lambda(\theta) \nabla_\lambda \log q_\lambda(\theta)^T\right]$.

The exponential family & Information Geometry

The exponential family induces a Riemannian manifold with a metric defined by the Fisher Information Matrix :

$$\|\lambda - \lambda_t\|^2_{F(\lambda_t)} = (\lambda - \lambda_t)^T F(\lambda_t)(\lambda - \lambda_t)$$

where $F(\lambda) = \mathbb{E}_{\theta \sim q_\lambda}\left[\nabla_\lambda \log q_\lambda(\theta) \nabla_\lambda \log q_\lambda(\theta)^T\right]$.

The Fisher metric approximates the KL divergence :

$$\mathrm{KL}\left(q_\lambda \| q_{\lambda_t}\right) = (\lambda - \lambda_t)^T F(\lambda_t)(\lambda - \lambda_t) + \mathcal{O}((\lambda - \lambda_t)^3).$$

Natural parameters & Information Geometry

We use the natural parameterization and replace the Euclidean distance by the Riemannian metric :

$$\lambda_{t+1} = \arg\max_{\lambda} \left\{ \lambda^T \hat{\nabla}_{\lambda} \mathrm{ELBO}(q_{\lambda_t}) - \frac{\|\lambda - \lambda_t\|^2_{F(\lambda_t)}}{2\eta} \right\},$$

which can be rewritten as a gradient descent algorithm

Natural parameters & Information Geometry

We use the natural parameterization and replace the Euclidean distance by the Riemannian metric :

$$\lambda_{t+1} = \arg\max_{\lambda} \left\{ \lambda^T \hat{\nabla}_{\lambda} \mathrm{ELBO}(q_{\lambda_t}) - \frac{\|\lambda - \lambda_t\|^2_{F(\lambda_t)}}{2\eta} \right\},$$

which can be rewritten as a gradient descent algorithm

$$\lambda_{t+1} = \lambda_t + \eta \underbrace{F(\lambda_t)^{-1} \hat{\nabla}_{\lambda} \mathrm{ELBO}(q_{\lambda_t})}_{\tilde{\nabla}_{\lambda} \mathrm{ELBO}(q_{\lambda_t})}.$$

Natural parameters & Information Geometry

We use the natural parameterization and replace the Euclidean distance by the Riemannian metric :

$$\lambda_{t+1} = \arg\max_{\lambda} \left\{ \lambda^T \hat{\nabla}_{\lambda} \mathrm{ELBO}(q_{\lambda_t}) - \frac{\|\lambda - \lambda_t\|^2_{F(\lambda_t)}}{2\eta} \right\},$$

which can be rewritten as a gradient descent algorithm

$$\lambda_{t+1} = \lambda_t + \eta \underbrace{F(\lambda_t)^{-1} \hat{\nabla}_{\lambda} \mathrm{ELBO}(q_{\lambda_t})}_{\tilde{\nabla}_{\lambda} \mathrm{ELBO}(q_{\lambda_t})}.$$

Therefore, steepest descent in the Fisher metric (which approximates KL divergence) is invariant to parameterization, to the first order (hence the name "natural gradient").

Natural parameters & Information Geometry

We use the natural parameterization and replace the Euclidean distance by the Riemannian metric :

$$\lambda_{t+1} = \arg\max_{\lambda} \left\{ \lambda^T \hat{\nabla}_{\lambda} \mathrm{ELBO}(q_{\lambda_t}) - \frac{\|\lambda - \lambda_t\|^2_{F(\lambda_t)}}{2\eta} \right\},$$

which can be rewritten as a gradient descent algorithm

$$\lambda_{t+1} = \lambda_t + \eta \underbrace{F(\lambda_t)^{-1} \hat{\nabla}_{\lambda} \mathrm{ELBO}(q_{\lambda_t})}_{\tilde{\nabla}_{\lambda} \mathrm{ELBO}(q_{\lambda_t})}.$$

Therefore, steepest descent in the Fisher metric (which approximates KL divergence) is invariant to parameterization, to the first order (hence the name "natural gradient").

But the computation of the natural gradient $\tilde{\nabla}_{\lambda} \mathrm{ELBO}(q_{\lambda_t})$ can be really cumbersome due to $F(\lambda_t)^{-1}$...

Use the Mean Parameterization !

There is a one-to-one mapping between the natural parameterization and the mean parameterization :

$$\mu(\lambda) = \mathbb{E}_{\theta \sim q_\lambda}[T(\theta)].$$

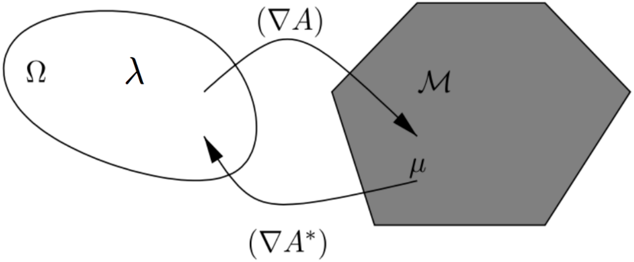
For a Gaussian : $\mu = (m, m^2 + \sigma^2)$.

Use the Mean Parameterization !

There is a one-to-one mapping between the natural parameterization and the mean parameterization :

$$\mu(\lambda) = \mathbb{E}_{\theta\sim q_\lambda}[T(\theta)].$$

For a Gaussian : $\mu = (m, m^2 + \sigma^2)$.

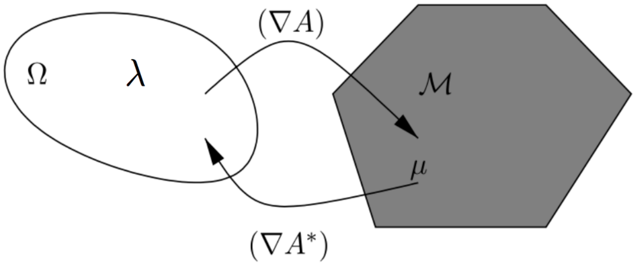


Use the Mean Parameterization !

There is a one-to-one mapping between the natural parameterization and the mean parameterization :

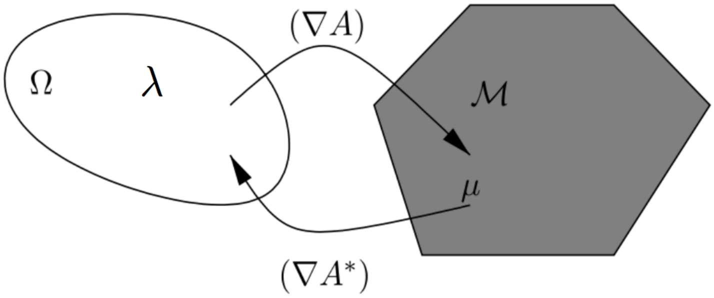
$$\mu(\lambda) = \mathbb{E}_{\theta \sim q_\lambda}[T(\theta)].$$

For a Gaussian : $\mu = (m, m^2 + \sigma^2)$.

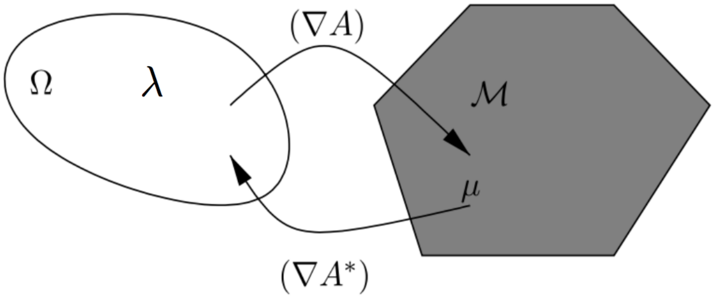


$$\tilde{\nabla}_\lambda \text{ELBO}(q_{\lambda_t}) = F(\lambda_t)^{-1} \hat{\nabla}_\lambda \text{ELBO}(q_{\lambda_t}) = \hat{\nabla}_\mu \text{ELBO}(q_{\mu_t})$$
$$\tilde{\nabla}_\mu \text{ELBO}(q_{\mu_t}) = F(\mu_t)^{-1} \hat{\nabla}_\mu \text{ELBO}(q_{\mu_t}) = \hat{\nabla}_\lambda \text{ELBO}(q_{\lambda_t})$$

Final learning rule

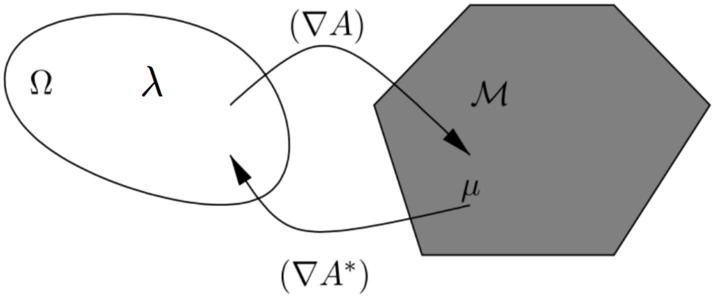


Final learning rule



Learning algorithm :

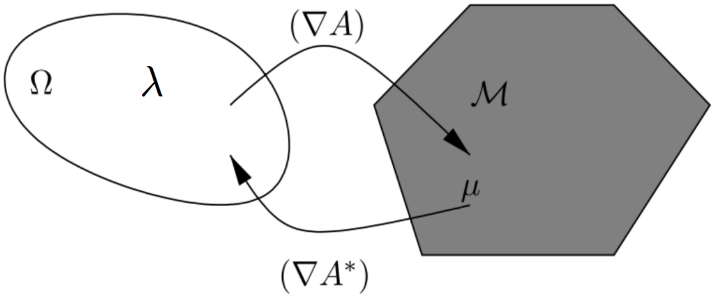
Final learning rule



Learning algorithm :

- Compute $\hat{\nabla}_\mu \mathrm{ELBO}(q_{\mu_t})$

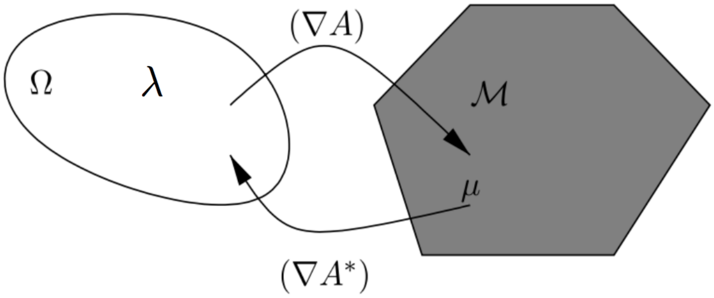
Final learning rule



Learning algorithm :

- Compute $\hat{\nabla}_\mu \mathrm{ELBO}(q_{\mu_t})$
- Update $\lambda_{t+1} = \lambda_t + \eta \hat{\nabla}_\mu \mathrm{ELBO}(q_{\mu_t})$

Final learning rule



Learning algorithm :

- Compute $\hat{\nabla}_\mu \mathrm{ELBO}(q_{\mu_t})$
- Update $\lambda_{t+1} = \lambda_t + \eta \hat{\nabla}_\mu \mathrm{ELBO}(q_{\mu_t})$
- Compute $\mu_{t+1} = \mu(\lambda_{t+1})$

The Gaussian example

For a Gaussian approximation $\mathcal{N}(m, \text{diag}(\sigma^2))$,

The Gaussian example

For a Gaussian approximation $\mathcal{N}(m, \mathrm{diag}(\sigma^2))$,

$$\sigma_{t+1}^{-2} = \sigma_t^{-2} - 2\eta \hat{\nabla}_{\sigma^2} \mathrm{ELBO}(q_{(m_t,\sigma_t^2)}),$$

$$m_{t+1} = m_t + \eta \sigma_{t+1}^2 \circ \hat{\nabla}_m \mathrm{ELBO}(q_{(m_t,\sigma_t^2)}).$$

The Gaussian example

For a Gaussian approximation $\mathcal{N}(m, \mathrm{diag}(\sigma^2))$,

$$\sigma_{t+1}^{-2} = \sigma_t^{-2} - 2\eta\hat{\nabla}_{\sigma^2}\mathrm{ELBO}(q_{(m_t,\sigma_t^2)}),$$

$$m_{t+1} = m_t + \eta\sigma_{t+1}^2 \circ \hat{\nabla}_m\mathrm{ELBO}(q_{(m_t,\sigma_t^2)}).$$

Question : how about the derivation of the gradients ?

Gradient Derivation

$$\mathrm{ELBO}(q_\lambda) = \mathbb{E}_{\theta \sim q_\lambda}[g_{\mathcal{S}}(\lambda, \theta)]$$

Gradient Derivation

$$\mathrm{ELBO}(q_\lambda) = \mathbb{E}_{\theta \sim q_\lambda}[g_{\mathcal{S}}(\lambda, \theta)]$$

Bonnet's and Price's formula :

$$\nabla_m \mathbb{E}_{\theta \sim q_\lambda}[g_{\mathcal{S}}(\lambda, \theta)] = \mathbb{E}_{\theta \sim q_\lambda}[\nabla_\theta g_{\mathcal{S}}(\lambda, \theta)],$$

$$\nabla_\Sigma \mathbb{E}_{\theta \sim q_\lambda}[g_{\mathcal{S}}(\lambda, \theta)] = \mathbb{E}_{\theta \sim q_\lambda}[\nabla^2_{\theta,\theta} g_{\mathcal{S}}(\lambda, \theta)].$$

Gradient Derivation

$$\mathrm{ELBO}(q_\lambda) = \mathbb{E}_{\theta\sim q_\lambda}[g_{\mathcal{S}}(\lambda,\theta)]$$

Bonnet's and Price's formula :

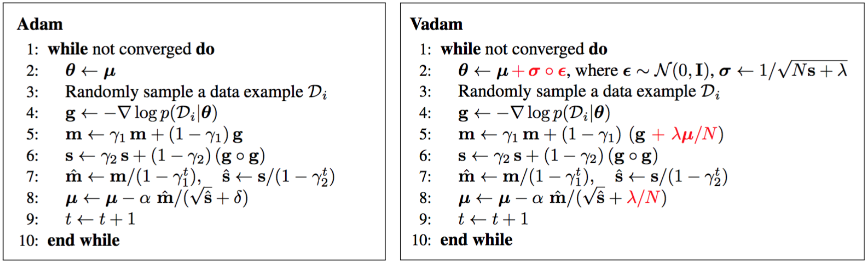
$$\nabla_m \mathbb{E}_{\theta\sim q_\lambda}[g_{\mathcal{S}}(\lambda,\theta)] = \mathbb{E}_{\theta\sim q_\lambda}[\nabla_\theta g_{\mathcal{S}}(\lambda,\theta)],$$

$$\nabla_\Sigma \mathbb{E}_{\theta\sim q_\lambda}[g_{\mathcal{S}}(\lambda,\theta)] = \mathbb{E}_{\theta\sim q_\lambda}[\nabla^2_{\theta,\theta} g_{\mathcal{S}}(\lambda,\theta)].$$

Or using the reparameterization trick :

$$\begin{aligned}\nabla_{\sigma^2} \mathbb{E}_{\theta\sim q_\lambda}[g_{\mathcal{S}}(\lambda,\theta)] &= \nabla_{\sigma^2} \mathbb{E}_{\varepsilon\sim\mathcal{N}(0,I)}[g_{\mathcal{S}}(\lambda, m+\sigma\varepsilon)]\\ &= \mathbb{E}_{\varepsilon\sim\mathcal{N}(0,I)}[\nabla_{\sigma^2} g_{\mathcal{S}}(\lambda, m+\sigma\varepsilon)]\\ &= \mathbb{E}_{\varepsilon\sim\mathcal{N}(0,I)}[\nabla_\theta g_{\mathcal{S}}(\lambda, m+\sigma\varepsilon)\varepsilon/2\sigma].\end{aligned}$$

Adam-like algorithm



Adam

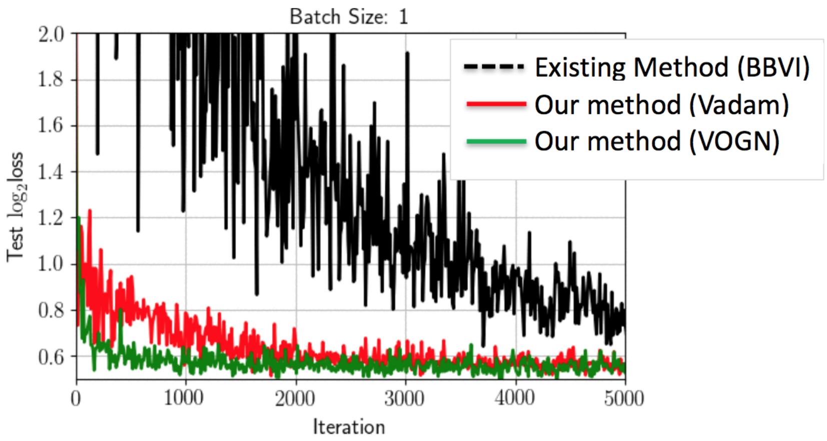
```
1: while not converged do
2:     $\boldsymbol{\theta} \leftarrow \boldsymbol{\mu}$
3:     Randomly sample a data example $\mathcal{D}_i$
4:     $\mathbf{g} \leftarrow -\nabla \log p(\mathcal{D}_i|\boldsymbol{\theta})$
5:     $\mathbf{m} \leftarrow \gamma_1 \mathbf{m} + (1-\gamma_1)\, \mathbf{g}$
6:     $\mathbf{s} \leftarrow \gamma_2 \mathbf{s} + (1-\gamma_2)\, (\mathbf{g} \circ \mathbf{g})$
7:     $\hat{\mathbf{m}} \leftarrow \mathbf{m}/(1-\gamma_1^t), \quad \hat{\mathbf{s}} \leftarrow \mathbf{s}/(1-\gamma_2^t)$
8:     $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} - \alpha\, \hat{\mathbf{m}}/(\sqrt{\hat{\mathbf{s}}} + \delta)$
9:     $t \leftarrow t + 1$
10: end while
```

Vadam

```
1: while not converged do
2:     $\boldsymbol{\theta} \leftarrow \boldsymbol{\mu} + \boldsymbol{\sigma} \circ \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$, $\boldsymbol{\sigma} \leftarrow 1/\sqrt{N\mathbf{s} + \lambda}$
3:     Randomly sample a data example $\mathcal{D}_i$
4:     $\mathbf{g} \leftarrow -\nabla \log p(\mathcal{D}_i|\boldsymbol{\theta})$
5:     $\mathbf{m} \leftarrow \gamma_1 \mathbf{m} + (1-\gamma_1)\, (\mathbf{g} + \lambda\boldsymbol{\mu}/N)$
6:     $\mathbf{s} \leftarrow \gamma_2 \mathbf{s} + (1-\gamma_2)\, (\mathbf{g} \circ \mathbf{g})$
7:     $\hat{\mathbf{m}} \leftarrow \mathbf{m}/(1-\gamma_1^t), \quad \hat{\mathbf{s}} \leftarrow \mathbf{s}/(1-\gamma_2^t)$
8:     $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} - \alpha\, \hat{\mathbf{m}}/(\sqrt{\hat{\mathbf{s}}} + \lambda/N)$
9:     $t \leftarrow t + 1$
10: end while
```

Figure 1. Comparison of Adam (left) and one of our proposed method Vadam (right). Adam performs maximum-likelihood estimation while Vadam performs variational inference, yet the two pseudocodes differ only slightly (differences highlighted in red). A major difference is in line 2 where, in Vadam, weights are perturbed during the gradient evaluations.

Fast Convergence



Performance

Table 1. Performance comparisons for BNN regression. The better method out of BBVI and Vadam is shown in boldface according to a paired t-test with p-value> 0.01. Both methods perform comparably but MC-Dropout outperforms them.

			Test RMSE			Test log-likelihood		
Dataset	**N**	**D**	**MC-Dropout**	**BBVI**	**Vadam**	**MC-Dropout**	**BBVI**	**Vadam**
Boston	506	13	2.97 ± 0.19	$\mathbf{3.58 \pm 0.21}$	3.93 ± 0.26	-2.46 ± 0.06	$\mathbf{-2.73 \pm 0.05}$	-2.85 ± 0.07
Concrete	1030	8	5.23 ± 0.12	$\mathbf{6.14 \pm 0.13}$	6.85 ± 0.09	-3.04 ± 0.02	$\mathbf{-3.24 \pm 0.02}$	-3.39 ± 0.02
Energy	768	8	1.66 ± 0.04	2.79 ± 0.06	$\mathbf{1.55 \pm 0.08}$	-1.99 ± 0.02	-2.47 ± 0.02	$\mathbf{-2.15 \pm 0.07}$
Kin8nm	8192	8	0.10 ± 0.00	$\mathbf{0.09 \pm 0.00}$	0.10 ± 0.00	0.95 ± 0.01	$\mathbf{0.95 \pm 0.01}$	0.76 ± 0.00
Naval	11934	16	0.01 ± 0.00	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	3.80 ± 0.01	$\mathbf{4.46 \pm 0.03}$	$\mathbf{4.72 \pm 0.22}$
Power	9568	4	4.02 ± 0.04	4.31 ± 0.03	$\mathbf{4.28 \pm 0.03}$	-2.80 ± 0.01	$\mathbf{-2.88 \pm 0.01}$	$\mathbf{-2.88 \pm 0.01}$
Wine	1599	11	0.62 ± 0.01	$\mathbf{0.65 \pm 0.01}$	0.66 ± 0.01	-0.93 ± 0.01	$\mathbf{-1.00 \pm 0.01}$	-1.01 ± 0.01
Yacht	308	6	1.11 ± 0.09	2.05 ± 0.06	$\mathbf{1.32 \pm 0.10}$	-1.55 ± 0.03	-2.41 ± 0.02	$\mathbf{-1.70 \pm 0.03}$

Conclusion

Take-home messages :

Conclusion

Take-home messages :

- It is possible to use SGD to perform (approximate) Bayesian inference.

Conclusion

Take-home messages :

- It is possible to use SGD to perform (approximate) Bayesian inference.
- The major difficulty is to compute the gradients.

Conclusion

Take-home messages :

- It is possible to use SGD to perform (approximate) Bayesian inference.
- The major difficulty is to compute the gradients.
- There exists many ways to deal with this derivation (e.g. REINFORCE & reparameterization trick).

Conclusion

Take-home messages :

- It is possible to use SGD to perform (approximate) Bayesian inference.
- The major difficulty is to compute the gradients.
- There exists many ways to deal with this derivation (e.g. REINFORCE & reparameterization trick).
- Convergence can be accelerated using the natural gradient.

Next lecture

Next time

We'll focus on 3 different topics :

- Learning Latent Variable Models.
- Theory of Variational Inference.
- Bayesian Model Averaging in Deep Learning.