Probabilistic AI

Lecture 1: Introduction to variational inference and the ELBO

Helge Langseth*

Jan. 2021

* Slides made together with Thomas D. Nielsen, Aalborg University.

Motivation for these lectures

Desiderata

We seek to build models that:

- Reflect human understanding of a domain with transparent and explicit modelling assumptions.
- Sound semantics – both wrt. modelling language and interpretation of the generated results.
- Ability to capture fine structure in data
  - . . . yet robust towards noisy inputs, out-of-distribution queries, and adversarial attacks.
- Efficient inference algorithms
  - . . . giving results that are useful for making decisions under uncertainty.
- Supported by a useful “programming language” for simple(-ish) implementation.

Is a *Deep Neural Network* the solution?

Limits on the scope of deep learning*

Deep learning thus far [in 2018] . . .

- . . . is data hungry
- . . . is not sufficiently transparent
- . . . has not been well integrated with prior knowledge
- . . . presumes a largely stable world, in ways that may be problematic
- . . . works well as an approximation, but **its answers often cannot be fully trusted**

* Gary Marcus: *Deep Learning: A Critical Appraisal*. arXiv:1801.00631 [cs.AI]

Is a *Deep Neural Network* the solution?

Limits on the scope of deep learning*

Deep learning thus far [in 2018] . . .

- . . . is data hungry
- . . . is not sufficiently transparent
- . . . has not been well integrated with prior knowledge
- . . . presumes a largely stable world, in ways that may be problematic
- . . . works well as an approximation, but **its answers often cannot be fully trusted**

* Gary Marcus: *Deep Learning: A Critical Appraisal*. arXiv:1801.00631 [cs.AI]

Probabilistic AI = Deep Learning + Probabilistic thinking

A marriage of probabilistic thinking and deep learning is a framework that . . .

- . . . allows explicit modelling.
- . . . has a sound probabilistic foundation.
- . . . balances expert knowledge and information from data.
- . . . avoids restrictive assumptions about modelling families.
- . . . supports efficient inference.

Probabilistic AI: A step towards **trustworthy AI**

Plan for these lectures

Day 1: Introduction to variational inference and the ELBO

Dive into the mathematical details of Probabilistic AI, understand the foundation, and investigate the effects of some of the "shortcuts" being made.

- **Approximate inference** via the KL divergence, a.k.a. **Variational Bayes**
- The **mean-field** approach to Variational Bayes
- **Black Box variational inference**

Day 2: Disentanglement in the variational auto encoder

Devise flexible models for representation learning, and consider their transparency.

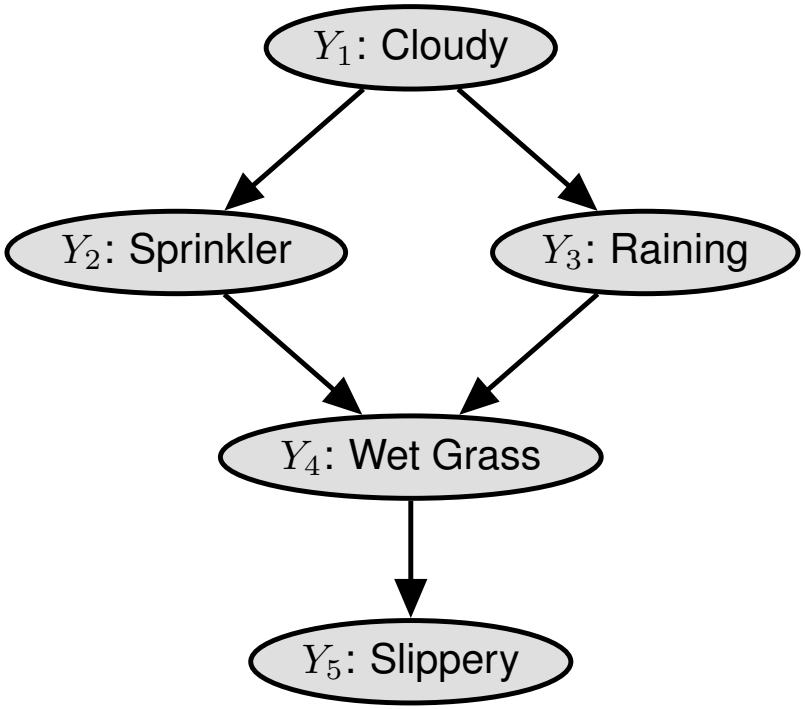
- **Variational Auto Encoders**
- **Disentanglement:** What, why, how?
- **Probabilistic Programming Languages**

If you want to learn more about these things:

Nordic Probabilistic AI School,
June 14th – 18th, 2021
`https://probabilistic.ai`

PGM Refresher

A simple example: “Wet grass”



- Each node is a random variable
- Edges indicate “influence” (Math-def: Graph encodes cond.indep. statements)
- For each variable Y_k, we must define $p(y_k \mid \mathrm{pa}(y_k))$.
- The full model is defined as $p(\mathbf{y}) = p(y_1, \ldots, y_n) = \prod_{i=1}^{n} p(y_i \mid \mathrm{pa}(y_i))$.
- Markov properties $\Leftrightarrow$ Factorisation property.

Summary – PGMs (or more precisely: Bayesian networks)

Bayesian networks represent (high-dim) distributions over random variables.

- **Simple syntax:** Nodes, links, DAG, conditional distributions.
- **Clear semantics:** Nodes + links = Markov properties; Joint distribution.

Inference: Find $p(\mathbf{z} \mid \mathbf{x})$, where $\mathbf{z}$ are variables of interest, $\mathbf{x}$ are the observed variables, $\mathbf{X} \cup \mathbf{Z} \subseteq \mathbf{Y}$. (We will assume $\mathbf{X} \cup \mathbf{Z} = \mathbf{Y}$ throughout.)

Note! Evaluating $p(\mathbf{y}) = p(\mathbf{z}, \mathbf{x})$ is simple: just use the definition of the model.

Evaluating $p(\mathbf{x})$ for $\mathbf{X} \subsetneq \mathbf{Y}$ (and thus $p(\mathbf{z} \mid \mathbf{x})$) is in general NP hard:

- **Exact inference:** Clever methods are available in some cases.
- **Approximate inference by sampling**: Markov Chain Monte Carlo is a common approximate solution.
- **Other approximate techniques**: This will be our approach!

PGMs in these lectures

Today we will look at the general inference problem, i.e., approximating $p(\mathbf{z} \mid \mathbf{x})$.
With that in place, we are ready to consider how to use PGMs for cool stuff tomorrow.

Variational Bayes: Approximate inference by optimization

Approximate inference through optimization

- The general goal is to somehow approximate $p(\mathbf{z} \mid \mathbf{x})$ without too costly computational operations.
- We will call the approximation $q(\cdot)$, hence hopefully "$q(\mathbf{z} \mid \mathbf{x}) \approx p(\mathbf{z} \mid \mathbf{x})$".
  - Often the conditioning part is dropped in $q(\cdot)$, hence $q(\mathbf{z})$ is a short-hand for $q(\mathbf{z} \mid \mathbf{x})$.

Formalization of approximate inference:

Given a family of tractable distributions $\mathcal{Q}$ and a distance measure between distributions Δ, choose

$$\hat{q}(\mathbf{z}) = \arg\min_{q \in \mathcal{Q}} \Delta(q(\mathbf{z}); p(\mathbf{z} \mid \mathbf{x})).$$

Decisions to be made:

1. How to define Δ so that we end up with a high-quality solution from $\mathcal{Q}$.
   - How to work with $\Delta(q(\mathbf{z}); p(\mathbf{z} \mid \mathbf{x}))$ when we don't even know what $p(\mathbf{z} \mid \mathbf{x})$ is.
2. How to define a family of distributions $\mathcal{Q}$ that is both flexible enough to generate good approximations and restrictive enough to support efficient calculations?

Desiderata

To use Δ to measure the distance from an object f to an object g it would be relevant to require that Δ has the following properties:

Positivity: $\Delta(f; g) \geq 0$ and $\Delta(f; g) = 0$ if and only if $f = g$.

Symmetry: $\Delta(f; g) = \Delta(g; f)$

Triangle: For objects f, g, and h we have that $\Delta(f; g) \leq \Delta(f; h) + \Delta(h; g)$.

Standard choice when working with probability distributions

It has become standard to choose the **Kullback-Leibler divergence** as the distance measure, where

$$\mathrm{KL}\left(f||g\right) = \mathbb{E}_{\mathbf{z}\sim f}\left[\log\left(\frac{f(\mathbf{Z})}{g(\mathbf{Z})}\right)\right] = \int_{\mathbf{z}} f(\mathbf{z})\,\log\left(\frac{f(\mathbf{z})}{g(\mathbf{z})}\right)\mathrm{d}\mathbf{z}.$$

Notice that while $\mathrm{KL}\left(f||g\right)$ obeys the positivity criterion, it satisfies neither symmetry nor the triangle inequality. It is thus **not a proper distance measure**.

Two alternative KL definitions

Moment-projection

- Minimizes $\mathrm{KL}\left(p||q\right) = -\mathbb{E}_{\mathbf{z}\sim p}[\log q(\mathbf{z})] - \mathcal{H}_p$.
- Preference given to q that has:
  1. High q-probability allocated to p-probable regions.
  2. $q(\mathbf{z}) > 0$ in any region where p is non-negligible.
     "$p(\mathbf{z}) > 0 \implies q(\mathbf{z}) > 0$"
  3. No explicit focus of entropy

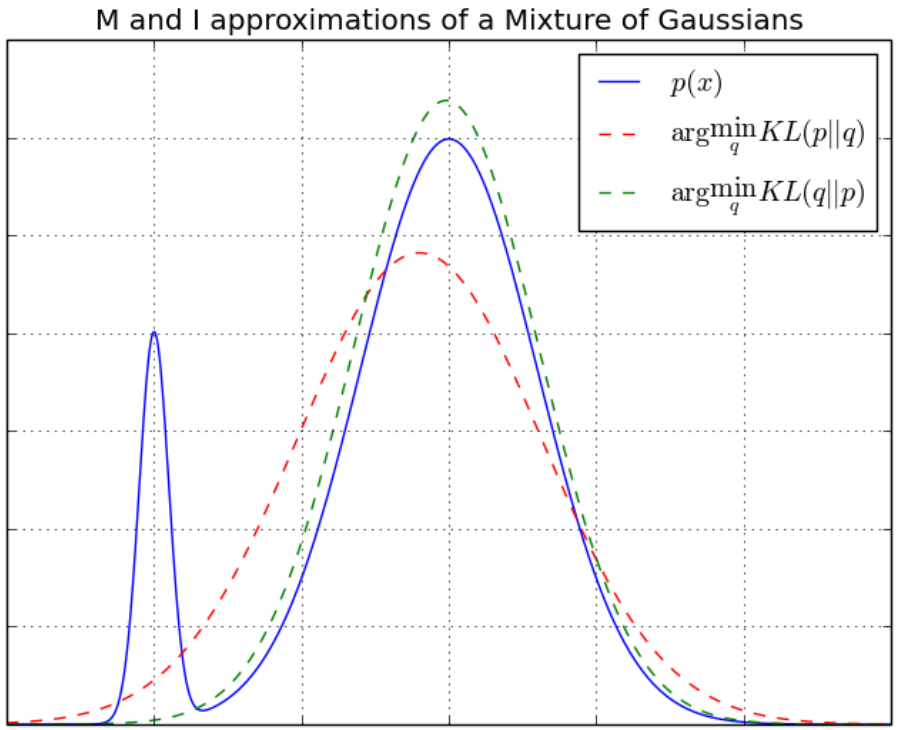
Information-projection

- Minimizes $\mathrm{KL}\left(q||p\right) = -\mathbb{E}_{\mathbf{z}\sim q}[\log p(\mathbf{z})] - \mathcal{H}_q$.
- Preference given to q that has:
  1. High q-probability allocated to p-probable regions.
  2. Very small q-probability given to any region where p is small.
     "$p(\mathbf{z}) = 0 \implies q(\mathbf{z}) = 0$".
  3. High entropy ("large variance")

Cheat-sheet:

- **KL-divergence:** $\mathrm{KL}\left(f||g\right) = \int_{\mathbf{z}} f(\mathbf{z}) \log\left(\frac{f(\mathbf{z})}{g(\mathbf{z})}\right) \mathrm{d}\mathbf{z} = \mathbb{E}_f\left[\log\left(\frac{f(\mathbf{z})}{g(\mathbf{z})}\right)\right]$.
- **Entropy:** $\mathcal{H}_f = -\int_{\mathbf{z}} f(\mathbf{z}) \log\left(f(\mathbf{z})\right) \mathrm{d}\mathbf{z} = -\mathbb{E}_f\left[\log\left(f(\mathbf{z})\right)\right]$.
- **Intuition:** Cheat a bit (measure-zero, convergence rates, etc.) and think *"If* $g(\mathbf{z}_0) \approx 0$*, then* $-\mathbb{E}_{\mathbf{z}\sim f}[\log g(\mathbf{z})]$ *becomes 'huge' unless* $f(\mathbf{z}_0) \approx 0$*"*.

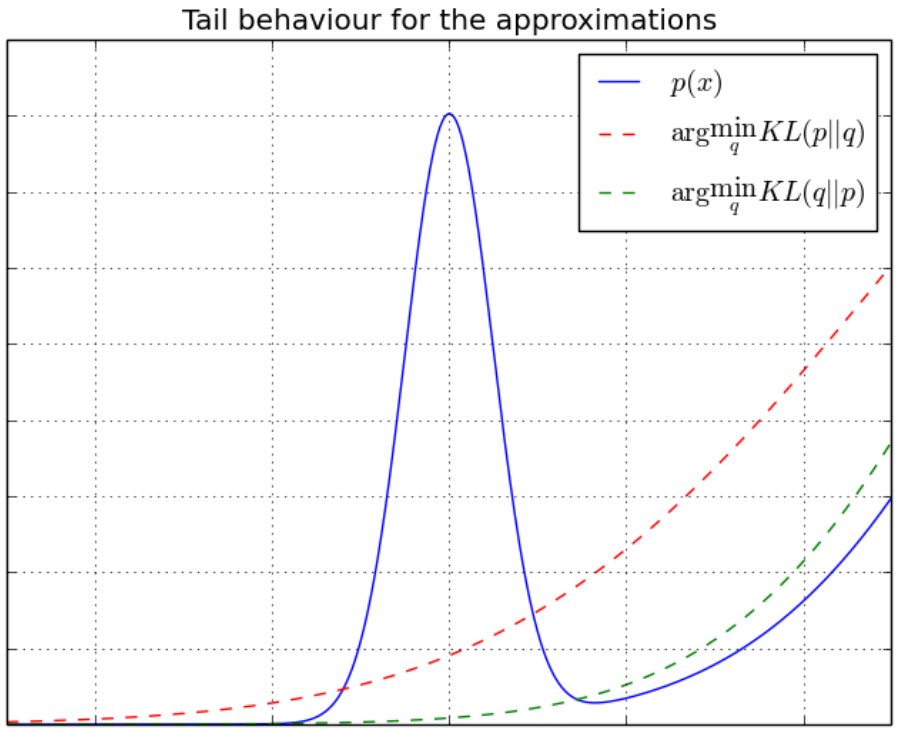
Moment and Information projection – main difference



Example: Approximating a Mix-of-Gaussians by a single Gaussian

- Moment projection – optimizing $\mathrm{KL}\,(p||q)$ – has slightly larger variance.
- Similar mean values, but Information projection – optimizing $\mathrm{KL}\,(q||p)$ – focuses mainly on the most prominent mode.

Moment and Information projection – main difference



Example: Approximating a Mix-of-Gaussians by a single Gaussian

- Moment projection – optimizing $\mathrm{KL}\,(p||q)$ – has slightly larger variance.
- Similar mean values, but Information projection – optimizing $\mathrm{KL}\,(q||p)$ – focuses mainly on the most prominent mode.
- M-projection is **zero-avoiding**, while I-projection is **zero-forcing**.

Variational Bayes w/ Mean Field

Variational Bayes setup

VB uses information projections:

Variational Bayes relies on **information projections**, i.e., approximates $p(\mathbf{z} \mid \mathbf{x})$ by

$$\hat{q}(\mathbf{z}) = \arg\min_{q \in \mathcal{Q}} \mathrm{KL}\left(q(\mathbf{z}) || p(\mathbf{z} \mid \mathbf{x})\right)$$

- **Positives:**
  - Very efficient inference when combined with cleverly chosen $\mathcal{Q}$.
  - Clever interpretation when used for (Bayesian) learning.
- **Negatives:**
  - As we have seen, this may result in *zero-forcing* behaviour.
    - The typical choice of $\mathcal{Q}$ can make this issue even more prominent.

ELBO: Evidence Lower-BOund

Notice how we can rearrange the KL divergence as follows:

$$\begin{aligned}\mathrm{KL}\left(q(\mathbf{z})||p(\mathbf{z}\,|\,\mathbf{x})\right) &= \mathbb{E}_{\mathbf{z}\sim q_\lambda}\left[\log\frac{q(\mathbf{z})}{p(\mathbf{z}\,|\,\mathbf{x})}\right] = \mathbb{E}_{\mathbf{z}\sim q_\lambda}\left[\log\frac{q(\mathbf{z})\cdot p(\mathbf{x})}{p(\mathbf{z}\,|\,\mathbf{x})\cdot p(\mathbf{x})}\right] \\ &= \log p(\mathbf{x}) - -\mathbb{E}_{\mathbf{z}\sim q_\lambda}\left[\log\frac{q(\mathbf{z})}{p(\mathbf{z},\mathbf{x})}\right] = \log p(\mathbf{x}) - \mathcal{L}\left(q\right)\end{aligned}$$

The Evidence Lower Bound (ELBO) is $\mathcal{L}\left(q\right) = -\mathbb{E}_q\left[\log\frac{q(\mathbf{z})}{p(\mathbf{z},\mathbf{x})}\right] = \mathbb{E}_q\left[\log\frac{p(\mathbf{z},\mathbf{x})}{q(\mathbf{z})}\right]$.

ELBO: Evidence Lower-BOund

Notice how we can rearrange the KL divergence as follows:

$$\begin{aligned}\mathrm{KL}\left(q(\mathbf{z})||p(\mathbf{z}\,|\,\mathbf{x})\right) &= \mathbb{E}_{\mathbf{z}\sim q_\lambda}\left[\log\frac{q(\mathbf{z})}{p(\mathbf{z}\,|\,\mathbf{x})}\right] = \mathbb{E}_{\mathbf{z}\sim q_\lambda}\left[\log\frac{q(\mathbf{z})\cdot p(\mathbf{x})}{p(\mathbf{z}\,|\,\mathbf{x})\cdot p(\mathbf{x})}\right] \\ &= \log p(\mathbf{x}) - -\mathbb{E}_{\mathbf{z}\sim q_\lambda}\left[\log\frac{q(\mathbf{z})}{p(\mathbf{z},\mathbf{x})}\right] = \log p(\mathbf{x}) - \mathcal{L}\left(q\right)\end{aligned}$$

The Evidence Lower Bound (ELBO) is $\mathcal{L}\left(q\right) = -\mathbb{E}_q\left[\log\frac{q(\mathbf{z})}{p(\mathbf{z},\mathbf{x})}\right] = \mathbb{E}_q\left[\log\frac{p(\mathbf{z},\mathbf{x})}{q(\mathbf{z})}\right]$.

VB focuses on ELBO:

$$\log p(\mathbf{x}) = \mathcal{L}\left(q\right) + \mathrm{KL}\left(q(\mathbf{z})||p(\mathbf{z}\,|\,\mathbf{x})\right)$$

Since $\log p(\mathbf{x})$ is constant wrt. the distribution q it follows:

- We can minimize $\mathrm{KL}\left(q(\mathbf{z})||p(\mathbf{z}\,|\,\mathbf{x})\right)$ by maximizing $\mathcal{L}\left(q\right)$
- This is **computationally simpler** because it uses $p(\mathbf{z},\mathbf{x})$ and not $p(\mathbf{z}\,|\,\mathbf{x})$.
- $\mathcal{L}\left(q\right)$ is a **lower bound** of $\log p(\mathbf{x})$ because $\mathrm{KL}\left(q(\mathbf{z})||p(\mathbf{z}\,|\,\mathbf{x})\right) \geq 0$.

$\rightsquigarrow$ During inference, we will look for $\hat{q}(\mathbf{z}) = \arg\max_{q\in\mathcal{Q}} \mathcal{L}\left(q\right)$.

ELBO: Evidence Lower-BOund

Notice how we can rearrange the KL divergence as follows:

$$\begin{aligned}\mathrm{KL}\left(q(\mathbf{z})||p(\mathbf{z}\,|\,\mathbf{x})\right) &= \mathbb{E}_{\mathbf{z}\sim q_\lambda}\left[\log\frac{q(\mathbf{z})}{p(\mathbf{z}\,|\,\mathbf{x})}\right] = \mathbb{E}_{\mathbf{z}\sim q_\lambda}\left[\log\frac{q(\mathbf{z})\cdot p(\mathbf{x})}{p(\mathbf{z}\,|\,\mathbf{x})\cdot p(\mathbf{x})}\right] \\ &= \log p(\mathbf{x}) - -\mathbb{E}_{\mathbf{z}\sim q_\lambda}\left[\log\frac{q(\mathbf{z})}{p(\mathbf{z},\mathbf{x})}\right] = \log p(\mathbf{x}) - \mathcal{L}\left(q\right)\end{aligned}$$

The Evidence Lower Bound (ELBO) is $\mathcal{L}\left(q\right) = -\mathbb{E}_q\left[\log\frac{q(\mathbf{z})}{p(\mathbf{z},\mathbf{x})}\right] = \mathbb{E}_q\left[\log\frac{p(\mathbf{z},\mathbf{x})}{q(\mathbf{z})}\right]$.

Summary:

- We started out looking for the $q \in \mathcal{Q}$ closest to $p(\mathbf{z}\,|\,\mathbf{x})$ in terms of $\mathrm{KL}\left(q(\mathbf{z})||p(\mathbf{z}\,|\,\mathbf{x})\right)$
- An apparent problem is that we do not know what $p(\mathbf{z}\,|\,\mathbf{x})$ is, hence cannot calculate that distance.
- Still, we can find the optimal approximation by maximizing $\mathcal{L}\left(q\right)$:

$$\arg\max_{q\in\mathcal{Q}} \mathcal{L}\left(q\right) = \arg\min_{q\in\mathcal{Q}} \mathrm{KL}\left(q(\mathbf{z})||p(\mathbf{z}\,|\,\mathbf{x})\right)$$

The mean field assumption

What we have . . .

We now have the first building-block of the approximation:

$$\Delta(q; p) = \mathrm{KL}\left(q(\mathbf{z}) || p(\mathbf{z} \mid \mathbf{x})\right),$$

and avoided the issue with $p(\mathbf{z} \mid \mathbf{x})$ by focusing on $\mathcal{L}(q)$.

We still need the set $\mathcal{Q}$:

Very often you will see the **mean field assumption**, which states that $\mathcal{Q}$ consists of all distributions that **factorizes** according to the equation

$$q(\mathbf{z}) = \prod_i q_i(z_i)$$

Note! This may seem like a very restricive set. However, we can choose any $q(\mathbf{z}) \in \mathcal{Q}$, and this is how the magic ($\sim$ "absorbing information from $\mathbf{x}$") happens.

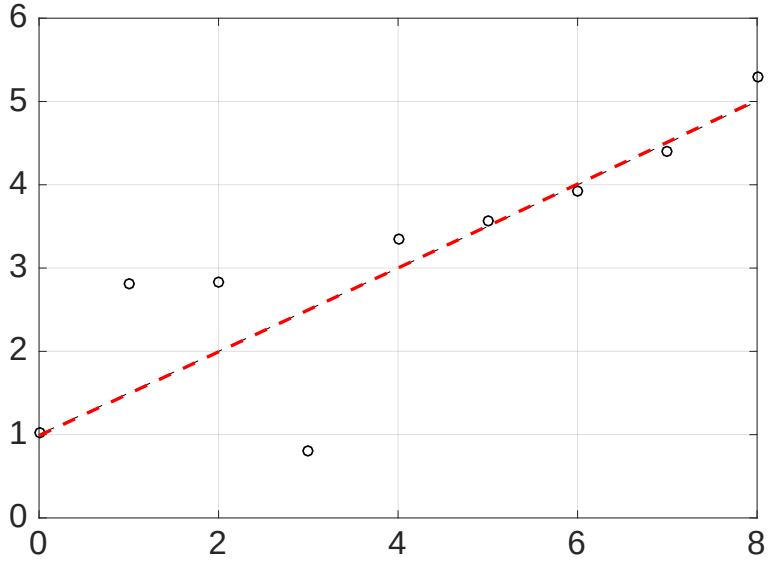
Simple example

Bayesian regression model:

$$Y_i \,|\, \{w_1, w_2, x_i\} = w_1 + w_2\, x_i + \epsilon_i.$$

Notation: Write $\mathbf{x}_i$ for $[1, x_i]^\mathsf{T}$. Then

$$\begin{aligned} Y_i \,|\, \{\mathbf{w}, \mathbf{x}_i\} &= \mathbf{w}^\mathsf{T}\mathbf{x}_i + \epsilon_i \\ \epsilon &\sim N(0, 1/\gamma); \gamma \text{ known} \\ \mathbf{w} &\sim \mathcal{N}\left(\boldsymbol{\mu}_0 = \mathbf{0}, \boldsymbol{\Sigma}_0 = \mathbf{I}_{2\times 2}\right) \end{aligned}$$



Data generated using $\mathbf{w} = [1, .5]^\mathsf{T}$.

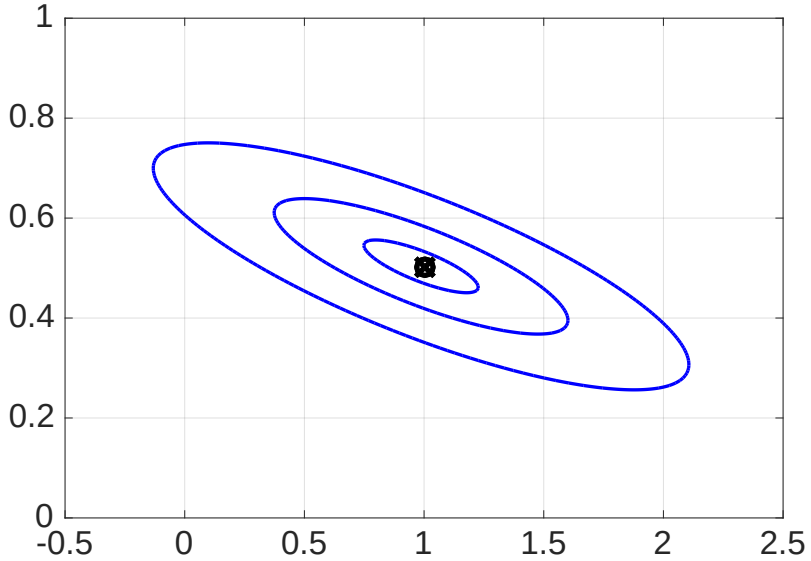
Simple example

Bayesian regression model:

$$Y_i \,|\, \{w_1, w_2, x_i\} = w_1 + w_2\, x_i + \epsilon_i.$$

Notation: Write $\mathbf{x}_i$ for $[1, x_i]^\mathsf{T}$. Then

$$\begin{aligned} Y_i \,|\, \{\mathbf{w}, \mathbf{x}_i\} &= \mathbf{w}^\mathsf{T}\mathbf{x}_i + \epsilon_i \\ \epsilon &\sim N(0, 1/\gamma); \gamma \text{ known} \\ \mathbf{w} &\sim \mathcal{N}\left(\boldsymbol{\mu}_0 = \mathbf{0}, \boldsymbol{\Sigma}_0 = \mathbf{I}_{2\times 2}\right) \end{aligned}$$



Exact: $\mathbb{E}[\mathbf{w}] \approx [1, .5]^\mathsf{T}$, $\rho_{w_1,w_2} \approx -.8$.
Contours: .10, .50, .90 prob.mass.

Exact Bayesian solution:

$$\mathbf{w} \,|\, \{\mathbf{x}, \mathbf{y}, \gamma, \boldsymbol{\mu}_0 = \mathbf{0}, \boldsymbol{\Sigma}_0 = \mathbf{I}_d\} \sim \mathcal{N}\left(\gamma(\mathbf{I}_d + \gamma\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{y}, (\mathbf{I}_d + \gamma\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\right).$$

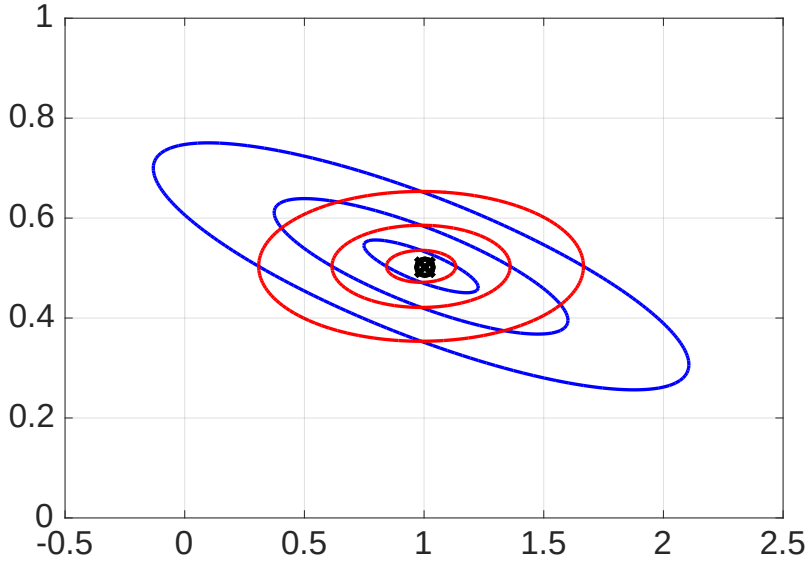
Simple example

Bayesian regression model:

$$Y_i \,|\, \{w_1, w_2, x_i\} = w_1 + w_2\, x_i + \epsilon_i.$$

Notation: Write $\mathbf{x}_i$ for $[1, x_i]^\mathsf{T}$. Then

$$\begin{aligned} Y_i \,|\, \{\mathbf{w}, \mathbf{x}_i\} &= \mathbf{w}^\mathsf{T}\mathbf{x}_i + \epsilon_i \\ \epsilon &\sim N(0, 1/\gamma); \gamma \text{ known} \\ \mathbf{w} &\sim \mathcal{N}\left(\boldsymbol{\mu}_0 = \mathbf{0}, \boldsymbol{\Sigma}_0 = \mathbf{I}_{2\times 2}\right) \end{aligned}$$



VB-MF: $\mathbb{E}[\mathbf{w}] \approx [1, .5]^\mathsf{T}$, $\rho_{w_1,w_2} \equiv 0$.
Contours: .10, .50, .90 prob.mass.

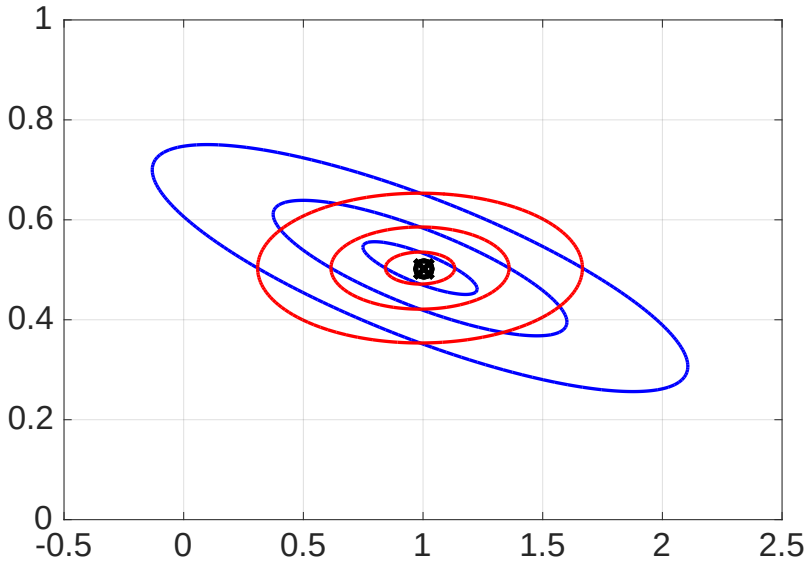
Exact Bayesian solution:

$$\mathbf{w} \,|\, \{\mathbf{x}, \mathbf{y}, \gamma, \boldsymbol{\mu}_0 = \mathbf{0}, \boldsymbol{\Sigma}_0 = \mathbf{I}_d\} \sim \mathcal{N}\left(\gamma(\mathbf{I}_d + \gamma\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{y}, (\mathbf{I}_d + \gamma\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\right).$$

Variational Bayesian solution w/ Mean Field:

Iterative approach; assumes factorized posterior (that is, a Gaussian with diagonal covariance matrix).

Important observations from the example



- The VB-MF solution approximates the mean of the true posterior well.
  - Not **always** the case – depends on the problem.
- VB-MF totally **disregards correlation** between variables.
- VB-MF **under-estimates the uncertainty** of the true posterior.
  - Evident for the full joint, as well as for each marginal.
  - In this example, the **underestimation of each marginal variance** is by a factor ~ 2.7.

Wrapping it all up: The VB algorithm under MF

Setup:

- We have observed $\mathbf{X} = \mathbf{x}$, and have **access to the full joint** $p(\mathbf{z}, \mathbf{x})$.
- The posterior approximation is assumed to factorize according to the **mean-field assumption**, and we use the $\mathrm{KL}\left(q(\mathbf{z}) || p(\mathbf{z} \mid \mathbf{x})\right)$ as our objective.
- We posit a **variational family** of distributions $q_j(\cdot \mid \boldsymbol{\lambda}_j)$, i.e., we choose the distributional form, while wanting to optimize the parameterization $\boldsymbol{\lambda}_j$.
- The optimal $\boldsymbol{\lambda}_j$ **will** depend on $\mathbf{x}$ – in fact $\boldsymbol{\lambda}_j$ encodes **all the information about the other variables** in the domain that Z_j is "aware of".

Algorithm:

Repeat until negligible improvement in terms of $\mathcal{L}\left(q\right)$:

1. For each j:
   - Somehow choose $\boldsymbol{\lambda}_j$ to maximize $\mathcal{L}\left(q\right)$, typically based on $\{\boldsymbol{\lambda}_i\}_{i \neq j}$. This can sometimes be done analytically, but today we will use a more general approach.
2. Calculate the new $\mathcal{L}\left(q\right)$.

Stochastic Gradient Ascent

A small side-step: Gradient Ascent

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".

Why do we talk about this?

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

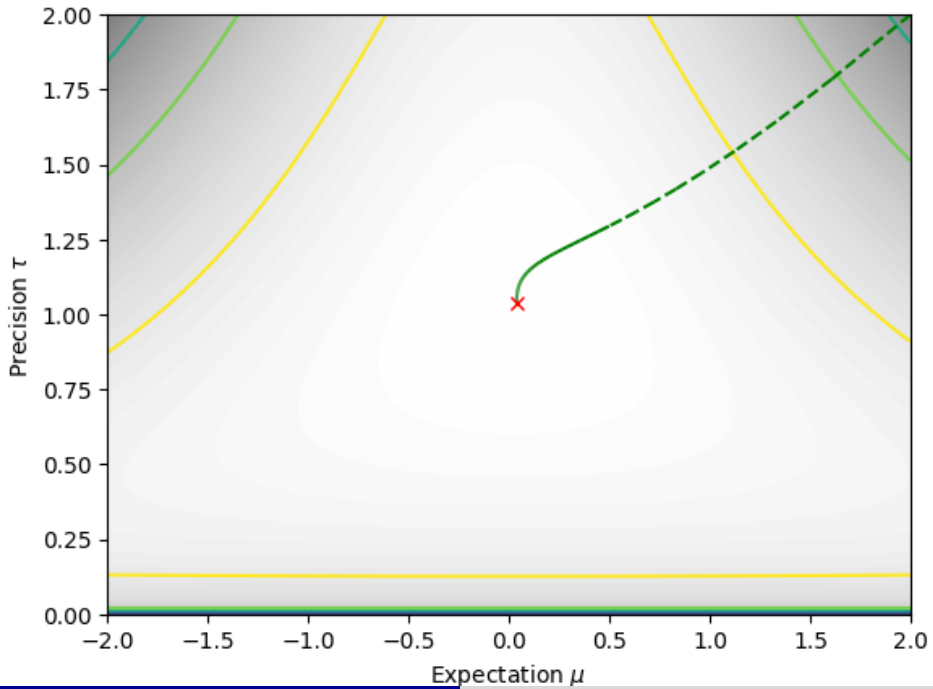
Example: Maximum log likelihood in a Gaussian model

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

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

$$\nabla_{\boldsymbol{\lambda}} f(\boldsymbol{\lambda}) = \begin{bmatrix} -N\tau\mu + \tau \sum_{i=1}^{N} x_i \\ \frac{N}{2\tau} - \frac{1}{2} \sum_{i=1}^{N} (x_i - \mu)^2 \end{bmatrix}$$

Cost of calculation: $O(N)$



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

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

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

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

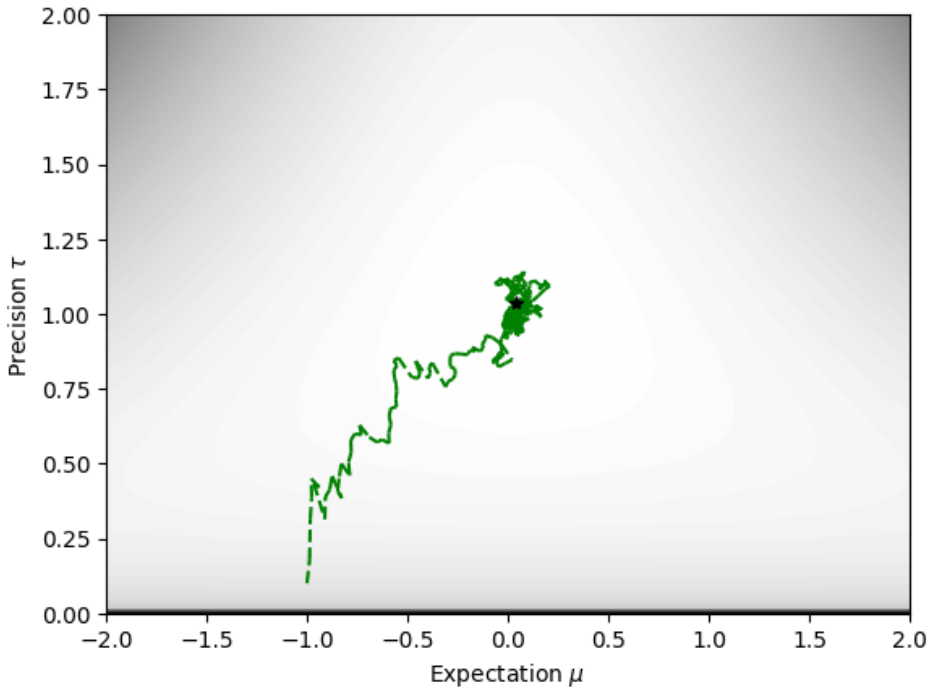
Example: Maximum log likelihood in a Gaussian model

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

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

Cost of calculation: $O(1)$

Randomness in $\mathbf{g}$ is a consequence of the random data selection process; multiplying by N ensures that $\mathbb{E}[\mathbf{g}(\boldsymbol{\lambda})] = \nabla_\lambda f(\boldsymbol{\lambda})$.



Black Box Variational Inference

Main idea: Cast inference as an optimization problem

Optimize the ELBO by stochastic gradient ascent over the parameters $\boldsymbol{\lambda}$.

Algorithm: Maximize $\mathcal{L}(q) = \mathbb{E}_{q_\lambda}\left[\log \frac{p_\theta(\mathbf{z},\mathbf{x})}{q_\lambda(\mathbf{z})}\right]$ by gradient ascent

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

Important issue:

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

BBVI - calculating the gradient

The algorithm requires that we can find

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

We can use these properties to simplify the equation:

1. $\nabla_\lambda\left(f(\mathbf{z},\boldsymbol{\lambda})\cdot g(\mathbf{z},\boldsymbol{\lambda})\right) = f(\mathbf{z},\boldsymbol{\lambda})\cdot\nabla_\lambda g(\mathbf{z},\boldsymbol{\lambda}) + g(\mathbf{z},\boldsymbol{\lambda})\nabla_\lambda f(\mathbf{z},\boldsymbol{\lambda})$
2. $\nabla_\lambda f(\mathbf{z},\boldsymbol{\lambda}) = f(\mathbf{z},\boldsymbol{\lambda})\nabla_\lambda \log f(\mathbf{z},\boldsymbol{\lambda})$
3. $\mathbb{E}_{q_\lambda}\left[\nabla_\lambda \log q_\lambda(\mathbf{z}\,|\,\boldsymbol{\lambda})\right] = 0$ for a density function $q_\lambda(\mathbf{z}\,|\,\boldsymbol{\lambda})$

Now it follows that

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

Calculating the gradient – Things to notice

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

Calculating the gradient – Things to notice

- We still only need access to the joint distribution $p_\theta(\mathbf{z}, \mathbf{x})$ – not $p_\theta(\mathbf{z} \mid \mathbf{x})$.

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

Calculating the gradient – Things to notice

- We still only need access to the joint distribution $p_\theta(\mathbf{z}, \mathbf{x})$ – not $p_\theta(\mathbf{z} \mid \mathbf{x})$.

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

- $q_\lambda(\mathbf{z} \mid \boldsymbol{\lambda})$ factorizes under MF, s.t. we can optimize per variable: $q_{\lambda_i}(z_i \mid \boldsymbol{\lambda}_i)$.

Calculating the gradient – Things to notice

- We still only need access to the joint distribution $p_\theta(\mathbf{z}, \mathbf{x})$ – not $p_\theta(\mathbf{z} \mid \mathbf{x})$.

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

- $q_\lambda(\mathbf{z} \mid \boldsymbol{\lambda})$ factorizes under MF, s.t. we can optimize per variable: $q_{\lambda_i}(z_i \mid \boldsymbol{\lambda}_i)$.
- We must calculate $\nabla_{\lambda_i} \log q(z_i \mid \boldsymbol{\lambda}_i)$, which is also known as the "score function". This depends on the distributional family of $q(\cdot)$; can be precomputed for standard distributions and auto-diff'ed for more complex constructions.

Calculating the gradient – Things to notice

- We still only need access to the joint distribution $p_\theta(\mathbf{z}, \mathbf{x})$ – not $p_\theta(\mathbf{z} \mid \mathbf{x})$.

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

- $q_\lambda(\mathbf{z} \mid \boldsymbol{\lambda})$ factorizes under MF, s.t. we can optimize per variable: $q_{\lambda_i}(z_i \mid \boldsymbol{\lambda}_i)$.
- We must calculate $\nabla_{\lambda_i} \log q\left(z_i \mid \boldsymbol{\lambda}_i\right)$, which is also known as the "score function". This depends on the distributional family of $q(\cdot)$; can be precomputed for standard distributions and auto-diff'ed for more complex constructions.
- The expectation will be approximated using a sample $\{\mathbf{z}_1, \ldots, \mathbf{z}_M\}$ generated from $q(\mathbf{z} \mid \boldsymbol{\lambda})$. Hence we require that we can **sample from** $q_{\lambda_i}(\cdot)$.

Calculating the gradient – Things to notice

- We still only need access to the joint distribution $p_\theta(\mathbf{z}, \mathbf{x})$ – not $p_\theta(\mathbf{z} \mid \mathbf{x})$.

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

- $q_\lambda(\mathbf{z} \mid \boldsymbol{\lambda})$ factorizes under MF, s.t. we can optimize per variable: $q_{\lambda_i}(z_i \mid \boldsymbol{\lambda}_i)$.
- We must calculate $\nabla_{\lambda_i} \log q(z_i \mid \boldsymbol{\lambda}_i)$, which is also known as the "score function". This depends on the distributional family of $q(\cdot)$; can be precomputed for standard distributions and auto-diff'ed for more complex constructions.
- The expectation will be approximated using a sample $\{\mathbf{z}_1, \ldots, \mathbf{z}_M\}$ generated from $q(\mathbf{z} \mid \boldsymbol{\lambda})$. Hence we require that we can **sample from** $q_{\lambda_i}(\cdot)$.

Calculating the gradient – in summary

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

$$\nabla_{\lambda_i} \mathcal{L}(q)|_{\lambda = \hat{\lambda}_i} \approx \frac{1}{M} \sum_{j=1}^{M} \log \frac{p(\mathbf{z}_j, \mathbf{x})}{q_{\lambda_i}(z_{i,j} \mid \hat{\lambda}_i)} \cdot \nabla_{\lambda_i} \log q_{\lambda_i}(z_{i,j} \mid \hat{\lambda}_i).$$

where $\{z_{i,1}, \ldots z_{i,M}\}$ are samples from $q_{\lambda_i}(\cdot \mid \hat{\lambda}_i)$.

BBVI Summary

Black Box Variational Inference

Black box variational inference is a **general purpose** approach for VI, that can maximize $\mathcal{L}(q)$ if we are able to ...

- ... **sample** from $q_{\lambda_i}(z_i \mid \mathbf{x}, \boldsymbol{\lambda}_i)$;
- ... calculate the "**score function**" $\nabla_{\lambda_i} \log q_{\lambda_i}(z_i \mid \mathbf{x}, \boldsymbol{\lambda}_i)$.

Since $q_{\lambda_i}(z_i \mid \mathbf{x}, \boldsymbol{\lambda}_i)$ is under our control, this should be OK, e.g., by letting $q_{\lambda_i}(\cdot)$ be a standard distribution parameterized by a DNN (input $\mathbf{x}$; weights $\boldsymbol{\lambda}_i$).

Consequences

- Since probabilistic inference now is done by gradient methods, we can rely on **autodiff-tools** like Tensorflow and Pytorch to work with arbitrarily complex distributions.
- Probabilistic modelling can thus be **seamlessly integrated** with building-blocks from other machine learning approaches (like deep learning).
  - We can e.g. represent $q(\boldsymbol{\theta} \mid \mathcal{D})$ via a DNN, and iteratively tune the DNN's weights while calculating the posterior (given the weights).
- We will see an example of this tomorrow, in the **Variational Auto Encoder**.