Variational inference

ELG 5218 - Uncertainty Evaluation in Engineering Measurements and Machine Learning

Miodrag Bolic

University of Ottawa

February 12, 2026

Outline

1. Gradients
   - Gradient Descent / SGD
   - Stochastic Optimization (SGD)
   - Automatic Differentiation (Autodiff)
2. KL
3. Intro to variational distribution
4. Mean-field approximation
5. Stochastic (Black-box) VI
6. Pathwise gradients of ELBO
7. Appendix
8. Reparameterization trick

Notebook: From_entropy_to_ADVI.ipynb

Gradients

- Gradients: definition, geometry, and why VI needs them
- Gradient descent / ascent and stochastic gradients
- Expectations inside objectives (Monte Carlo estimation)
- Automatic differentiation intuition (computation graphs)
- Mini-VI example: optimizing an ELBO for a Gaussian variational posterior

Gradients: the basic object

A **gradient** of a scalar function $f(\theta)$ with respect to parameters $\theta \in \mathbb{R}^d$ is

$$\nabla_\theta f(\theta) = \begin{bmatrix} \partial f / \partial \theta_1 \\ \vdots \\ \partial f / \partial \theta_d \end{bmatrix}.$$

Key geometry:

- ∇f points in the direction of *steepest increase*
- $-\nabla f$ points in the direction of *steepest decrease*

Quick example

Let $f(\theta) = (\theta - 5)^2$. Then

$$\frac{d}{d\theta} f(\theta) = 2(\theta - 5).$$

- If $\theta < 5$, the gradient is negative $\Rightarrow$ increase θ
- If $\theta > 5$, the gradient is positive $\Rightarrow$ decrease θ

Gradient descent vs ascent

Minimize $f(\theta)$ (gradient descent):

$$\theta \leftarrow \theta - \eta \nabla_\theta f(\theta)$$

Maximize $\mathcal{L}(\theta)$ (gradient ascent):

$$\theta \leftarrow \theta + \eta \nabla_\theta \mathcal{L}(\theta)$$

In VI we usually **maximize the ELBO**, so we do **ascent** (or minimize the negative ELBO).

Example: gradient descent trajectory

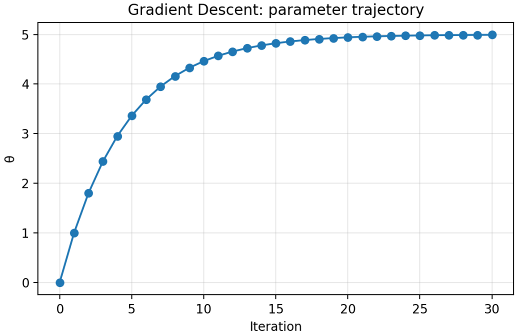
We minimize $f(\theta) = (\theta - 5)^2$.

- **Gradient descent update:**

$$\theta_{k+1} = \theta_k - \eta \nabla f(\theta_k) = \theta_k - \eta\, 2(\theta_k - 5).$$

- Figure shows the iterates θ_k converging to the minimizer:

$$\theta^\star = 5.$$



Definition: full gradient vs. stochastic gradient

Let the empirical risk be

$$f(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell(\theta; x_i, y_i).$$

Full (batch) gradient:

$$\nabla f(\theta) = \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta \ell(\theta; x_i, y_i).$$

Stochastic gradient (single sample): pick i_t and use

$$g_t = \nabla_\theta \ell(\theta_t; x_{i_t}, y_{i_t}), \qquad \theta_{t+1} = \theta_t - \eta_t \, g_t.$$

Key property (typical assumption): g_t is an *unbiased* estimator:

$$\mathbb{E}[g_t \mid \theta_t] = \nabla f(\theta_t) \quad \text{(true under uniform sampling with replacement).}$$

Mini-batch stochastic gradient (batch size k)

Pick a mini-batch $B_t \subset \{1, \dots, n\}$ with $|B_t| = k$ and use

$$g_t = \frac{1}{k} \sum_{i \in B_t} \nabla_\theta \ell(\theta_t; x_i, y_i), \qquad \theta_{t+1} = \theta_t - \eta_t \, g_t.$$

- **Variance reduction:** gradient noise typically decreases as $\approx 1/k$ (up to finite-population corrections).
- **Compute tradeoff:** each step costs k examples, but can be faster in wall-clock time on GPUs/TPUs.
- **Two common sampling modes:**
  - *With replacement*: indices i.i.d. uniform each step (clean analysis).
  - *Without replacement within an epoch*: shuffle once per epoch, then take consecutive slices of the shuffled order (common in ML code).

What parameters do you choose in SGD?

Core knobs

- **Batch size** k (or B): e.g., 1, 32, 128, 1024.
- **Learning rate** η and **schedule**: constant, step decay, cosine, warmup, etc.
- **Number of epochs** E (passes through data) or total steps T.
- **Shuffling / sampling policy**: shuffle each epoch; with/without replacement; sequential windows for time series.

Common practical knobs (often essential in deep learning)

- **Momentum** / **Nesterov** (SGD variants).
- **Weight decay** (L2 regularization), dropout (model-level).
- **Gradient clipping** (stability), especially with RNNs/transformers.
- **Early stopping** based on validation performance.

Algorithm: shuffle each epoch + mini-batches (no replacement within epoch)

This is the standard training loop used in most ML frameworks when `shuffle=True`.

Algorithm 1 Mini-batch SGD with epoch shuffling (typical in ML)

```
Dataset $\{(x_i, y_i)\}_{i=1}^{n}$, batch size $k$, epochs $E$, learning rates $\{\eta_t\}$
Initialize parameters $\theta$
for $e = 1$ to $E$ do
    Sample a random permutation $\pi$ of $\{1, \ldots, n\}$          ▷ shuffle indices
    for $b = 1$ to $\lceil n/k \rceil$ do
        $B \leftarrow \{\pi[(b-1)k+1], \ldots, \pi[\min(bk, n)]\}$          ▷ take next slice
        $g \leftarrow \frac{1}{|B|} \sum_{i \in B} \nabla_\theta \ell(\theta; x_i, y_i)$
        $\theta \leftarrow \theta - \eta_t \, g$
        $t \leftarrow t + 1$
    end for
end for
```

Algorithm: mini-batch sampling with replacement

Algorithm 2 Mini-batch SGD with replacement (clean unbiased estimator)

```
Dataset size $n$, batch size $k$, steps $T$, learning rates $\{\eta_t\}$
Initialize parameters $\theta$
for $t = 0$ to $T - 1$ do
    Sample indices $i_{t,1}, \dots, i_{t,k} \overset{i.i.d.}{\sim} \mathrm{Unif}(\{1, \dots, n\})$
    $g \leftarrow \frac{1}{k} \sum_{j=1}^{k} \nabla_\theta \ell(\theta; x_{i_{t,j}}, y_{i_{t,j}})$
    $\theta \leftarrow \theta - \eta_t \, g$
end for
```

Batch size $k = 1$

Set $k = 1$ in either algorithm to obtain "single-sample" SGD.

Automatic Differentiation (autodiff) and its relation to SGD

Definition (autodiff). Automatic differentiation is a set of techniques that compute *exact derivatives* (of a function implemented as code) by applying the chain rule to the program's computational graph. Unlike:

- *symbolic differentiation* (manipulates formulas), and
- *numerical differentiation* (finite differences, approximate),

autodiff produces derivatives accurate up to floating-point rounding.

Forward and reverse mode.

- **Forward-mode:** efficient when #inputs is small.
- **Reverse-mode (backpropagation):** efficient when #outputs is small (e.g., scalar loss).

How autodiff connects to batch vs. stochastic gradients. Let the empirical risk be

$$f(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell(\theta; x_i, y_i).$$

Autodiff is the engine that computes $\nabla_\theta \ell(\theta; x_i, y_i)$ *for whatever data you feed in.*

- **Full-batch GD:** feed all n points, autodiff returns $\nabla f(\theta)$.
- **Mini-batch SGD:** feed a batch B of size k, autodiff returns

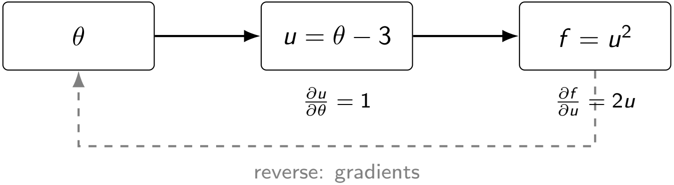
$$\nabla_\theta \left(\frac{1}{k} \sum_{i \in B} \ell(\theta; x_i, y_i) \right) = \frac{1}{k} \sum_{i \in B} \nabla_\theta \ell(\theta; x_i, y_i).$$

Key takeaway: *Autodiff computes gradients; batching controls* **which** *data points contribute to the gradient estimate.*

Autodiff (reverse-mode): forward values + reverse gradients on a computation graph

What we want. Compute $\nabla_\theta f(\theta)$ automatically.
Idea. Autodiff builds a *computation graph* of elementary operations.

- **Forward pass:** compute and store intermediate values.
- **Reverse pass (backprop):** propagate sensitivities using the chain rule.


θ
$u = \theta - 3$
$f = u^2$
$\frac{\partial u}{\partial \theta} = 1$
$\frac{\partial f}{\partial u} = 2u$
reverse: gradients

Example. $f(\theta) = (\theta - 3)^2$.
Forward: $u = \theta - 3$, then $f = u^2$.
Reverse:

$$\frac{df}{d\theta} = \frac{df}{du}\frac{du}{d\theta} = (2u) \cdot 1 = 2(\theta - 3).$$

Takeaway. Reverse-mode autodiff computes gradients efficiently by reusing stored forward values and applying the chain rule backward.

Shannon Entropy: Definition

Discrete random variable

Let X be a discrete random variable taking values $x_1, \ldots, x_n$ with probabilities $p_i = \mathbb{P}(X = x_i)$ and $\sum_{i=1}^{n} p_i = 1$.

The **Shannon entropy** of X (in base b) is

$$H_b(X) = -\sum_{i=1}^{n} p_i \log_b p_i .$$

- If we use $\log_2$, the units are **bits**.
- If X is deterministic (one outcome has probability 1), then $H(X) = 0$.
- Entropy measures the **average uncertainty** about X.

Entropy as Uncertainty

- High entropy ⇒ outcomes are hard to predict.
- Low entropy ⇒ outcomes are more predictable.

Example: coin toss

- Fair coin: $p(\mathrm{H}) = p(\mathrm{T}) = 0.5$.

$$H(X) = -\left[0.5 \log_2 0.5 + 0.5 \log_2 0.5\right] = 1 \text{ bit}.$$

- Biased coin: $p(\mathrm{H}) = 0.9$, $p(\mathrm{T}) = 0.1$.

$$H(X) = -\left[0.9 \log_2 0.9 + 0.1 \log_2 0.1\right] \approx 0.47 \text{ bits}.$$

The biased coin is more predictable, so its entropy is smaller.

Discrete Examples

Fair die and non-uniform distribution

- Fair 4-sided die: $p_i = 0.25$ for $i = 1, \dots, 4$.

$$H(X) = -4 \cdot 0.25 \log_2 0.25 = 2 \text{ bits.}$$

- Fair 6-sided die: $p_i = 1/6$.

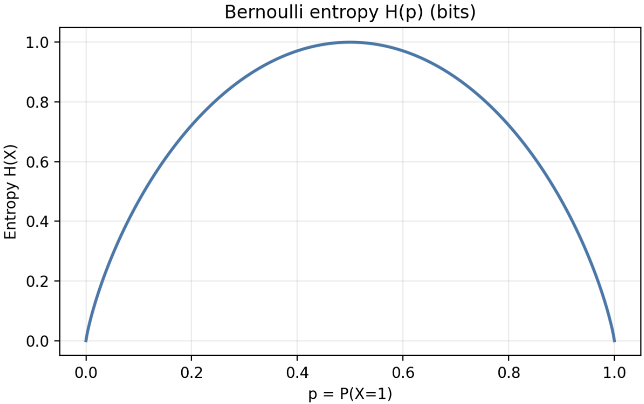
$$H(X) = -\sum_{i=1}^{6} \tfrac{1}{6} \log_2 \tfrac{1}{6} = \log_2 6 \approx 2.585 \text{ bits.}$$

- Non-uniform $(0.5, 0.25, 0.25)$:

$$H(X) = -\left[0.5 \log_2 0.5 + 0.25 \log_2 0.25 + 0.25 \log_2 0.25\right] = 1.5 \text{ bits.}$$

- More spread-out distributions (for fixed support) have higher entropy.
- More "peaked" distributions have lower entropy.

Bernoulli entropy example



Entropy is maximal at $p = 0.5$ (most uncertain).

Uniform Distribution and Maximum Entropy

Consider all discrete distributions on a finite set $\{x_1, \dots, x_n\}$:

$$\mathcal{P} = \left\{ (p_1, \dots, p_n) \, : \, p_i \geq 0, \, \sum_{i=1}^{n} p_i = 1 \right\}.$$

Maximum entropy principle (discrete case)

Among all $(p_1, \dots, p_n) \in \mathcal{P}$, entropy

$$H(X) = -\sum_{i=1}^{n} p_i \log p_i$$

is **maximized** when

$$p_i = \frac{1}{n}, \quad i = 1, \dots, n.$$

In this case, $H(X) = \log n$ (in the same log base).

- The uniform distribution represents **maximal uncertainty** when only the support size n is known.
- Any deviation from uniformity makes outcomes more predictable and reduces entropy.

Differential Entropy and the Normal Distribution

For a continuous random variable X with density $p(x)$, the **differential entropy** (in nats meaning that the $log = ln$) is

$$h(X) = -\int_{-\infty}^{\infty} p(x) \log p(x)\, dx.$$

Entropy of a normal distribution

If $X \sim \mathcal{N}(\mu, \sigma^2)$, then

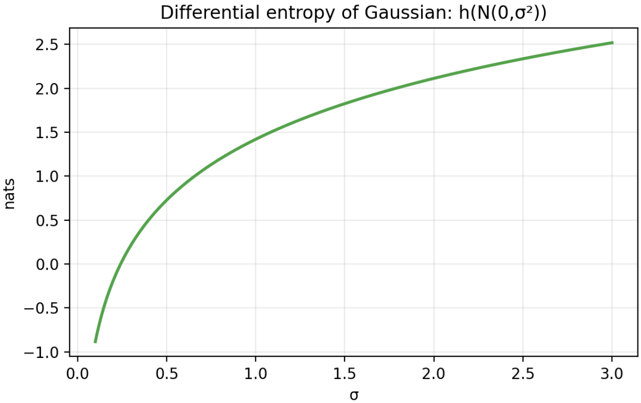
$$h(X) = \frac{1}{2} \log \left(2\pi e \sigma^2\right).$$

Maximum entropy principle (continuous case)

Among all real-valued random variables with a given mean μ and variance σ^2, the normal distribution $\mathcal{N}(\mu, \sigma^2)$ has the **largest** differential entropy.

- The normal distribution is the "most spread out" shape compatible with the specified mean and variance.

Gaussian differential entropy



Larger σ means a more spread-out distribution and higher entropy.

Kullback–Leibler divergence Definition (Discrete Case)

Let P and Q be two discrete probability distributions on the same support $\{x_1, \ldots, x_n\}$ with

$$p_i = P(x_i), \qquad q_i = Q(x_i), \qquad \sum_{i=1}^{n} p_i = \sum_{i=1}^{n} q_i = 1.$$

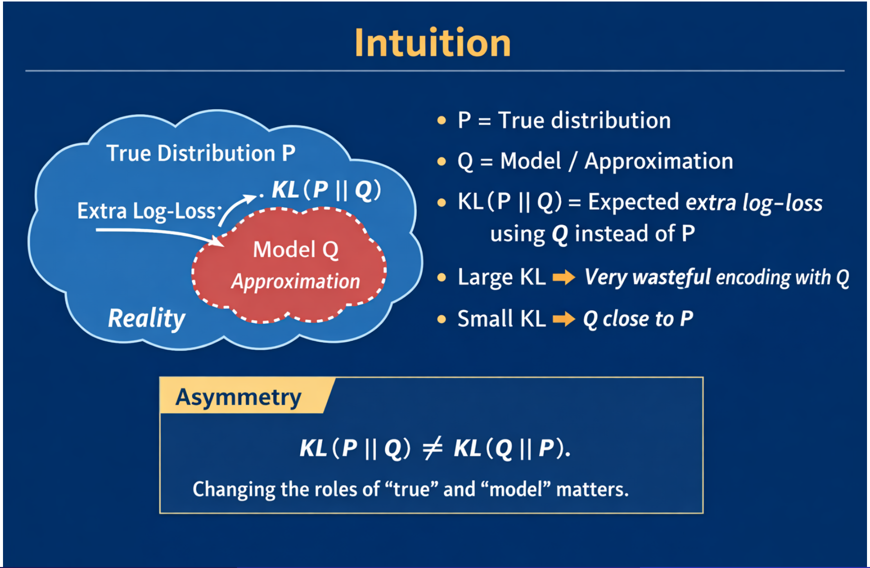
Kullback–Leibler divergence

The KL divergence from Q to P is

$$\mathrm{KL}(P\|Q) = \sum_{i=1}^{n} p_i \log \frac{p_i}{q_i}.$$

- Usually computed with $\log_2$ (bits) or natural log (nats).
- $\mathrm{KL}(P\|Q) \geq 0$ and equals 0 iff $P = Q$.
- Measures how different Q is from P when P is the *true* distribution.

Intuition



Intuition

- P = True distribution
- Q = Model / Approximation
- KL(P || Q) = Expected *extra log−loss* using ***Q*** instead of P
- Large KL ➡ ***Very wasteful** encoding with Q*
- Small KL ➡ ***Q close to P***

Asymmetry

$$KL(P \,||\, Q) \neq KL(Q \,||\, P).$$

Changing the roles of "true" and "model" matters.

Entropy, Cross-Entropy, and KL as *Expected Log-Loss*

Setup. Reality generates outcomes i from the true distribution P (probabilities p_i). We predict using a model Q (probabilities q_i). Consider the **log-loss** (negative log-likelihood) for outcome i: $\ell_Q(i) = -\log q_i$.

Interpretations

- **Entropy**:

$$H(P) = -\sum_i p_i \log p_i = \mathbb{E}_{i\sim P}[-\log p_i]$$

  is the *irreducible average uncertainty/surprise* in outcomes drawn from P.

- **Cross-entropy**:

$$H(P, Q) = -\sum_i p_i \log q_i = \mathbb{E}_{i\sim P}[-\log q_i]$$

  is the *average log-loss you incur* when using model Q on data from P.

Decomposition (mismatch penalty)

$$\mathrm{KL}(P\|Q) = \sum_i p_i \log \frac{p_i}{q_i} = H(P, Q) - H(P).$$

- $\mathrm{KL}(P\|Q)$: **extra expected log-loss** due to using the wrong probabilities Q.

Example: Bernoulli Distributions

Let $P = \mathrm{Ber}(p)$ and $Q = \mathrm{Ber}(q)$ on $\{0, 1\}$:

$$P(X = 1) = p, \quad P(X = 0) = 1 - p,$$
$$Q(X = 1) = q, \quad Q(X = 0) = 1 - q.$$

Closed form

$$\mathrm{KL}(P\|Q) = p \log \frac{p}{q} + (1 - p) \log \frac{1 - p}{1 - q}.$$

- If $p = q$ then $\mathrm{KL}(P\|Q) = 0$.
- If q is far from p (e.g. $p = 0.9$, $q = 0.1$), the divergence becomes large.
- If $q = 0$ but $p > 0$ (or vice versa), the divergence is *infinite*: Q assigns probability 0 to an event that actually occurs.

Discrete Examples

Two-outcome example

Let

$$P = (0.6,\ 0.4), \quad Q = (0.5,\ 0.5).$$

Then

$$\mathrm{KL}(P\|Q) = 0.6 \log \frac{0.6}{0.5} + 0.4 \log \frac{0.4}{0.5}.$$

Numerically (base 2): $\mathrm{KL}(P\|Q) \approx 0.029$ bits.

Three-outcome example

Let

$$P = (0.6,\ 0.3,\ 0.1), \quad Q = (0.5,\ 0.25,\ 0.25).$$

Then $\mathrm{KL}(P\|Q) \approx 0.066$ bits, while $\mathrm{KL}(Q\|P) \approx 0.084$ bits.

These values are small: Q is reasonably close to P, but not identical.

Continuous Case (Briefly)

For continuous distributions with densities $p(x)$ and $q(x)$, the KL divergence is

$$\mathrm{KL}(P\|Q) = \int p(x) \log \frac{p(x)}{q(x)}\, dx.$$

- The same basic interpretation holds: difference in log-likelihood between the true density p and the model density q.
- For example, KL between two normal distributions has a closed-form expression.
- In practice, continuous KL is often approximated via Monte Carlo.

Core properties

- **Non-negativity (Gibbs):** $\mathrm{KL}(p\|q) \geq 0$ and equals 0 iff $p = q$ (a.e.).
- **Asymmetry:** generally $\mathrm{KL}(p\|q) \neq \mathrm{KL}(q\|p)$.
- **Support sensitivity:** if $q(z) = 0$ where $p(z) > 0$ then $\mathrm{KL}(p\|q) = \infty$.

Why order matters (intuition)

Compare

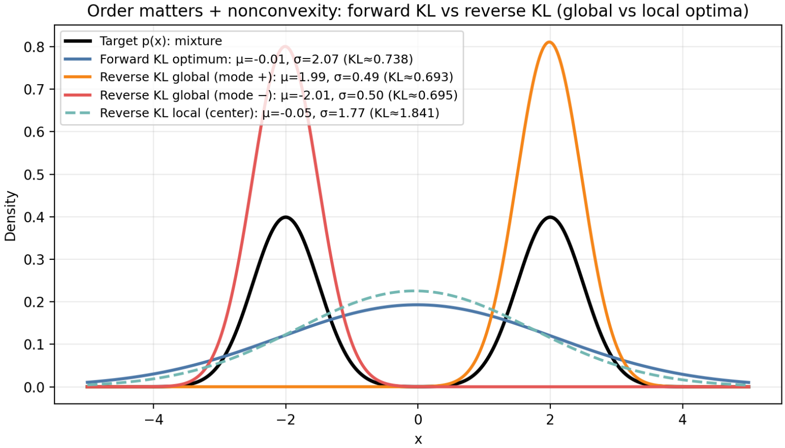
$$\mathrm{KL}(p\|q) = \mathbb{E}_p[\log p - \log q] \quad \text{vs} \quad \mathrm{KL}(q\|p) = \mathbb{E}_q[\log q - \log p].$$

- $\mathrm{KL}(p\|q)$: expectation under $p \Rightarrow$ penalizes q for missing any region where p has mass (**mass-covering**).
- $\mathrm{KL}(q\|p)$: expectation under $q \Rightarrow$ penalizes q for placing mass where p is tiny (**mode-seeking**).

Rule of thumb (global optima): forward KL tends to *cover modes*; reverse KL tends to *choose a mode*.

Example: Target: $p(x) = 0.5\,\mathcal{N}(-2, 0.5^2) + 0.5\,\mathcal{N}(2, 0.5^2)$. Approximation family: $q(x) = \mathcal{N}(\mu, \sigma^2)$. We'll optimize forward and reverse KL.

Bimodal example (corrected): global vs local optima



Reverse KL has two global minima (one per mode) and a higher-cost local optimum near the center.

Why VI uses reverse KL $\mathrm{KL}(q\|p(z \mid x))$

Variational inference chooses $q_\lambda(z)$ and minimizes

$$\mathrm{KL}(q_\lambda(z)\|p(z \mid x)).$$

This direction is convenient because the expectation is under q:

- we can sample from q and evaluate $\log p$ up to a constant,
- this yields the ELBO objective.

Consequence: mode-seeking behavior can cause *mode dropping* for multi-modal posteriors.

Cross-Entropy and NLL in Classification (Derivation)

Setup.

- True data distribution: $p_{\text{data}}(x, y)$.
- Model: conditional probabilities $p_\theta(y \mid x)$.
- Per-example negative log-likelihood (NLL):

$$\mathrm{NLL}(x, y) = -\log p_\theta(y \mid x).$$

Expected NLL under the true distribution:

$$\mathbb{E}_{(X,Y)\sim p_{\text{data}}}[\mathrm{NLL}(X, Y)] = \mathbb{E}_{p_{\text{data}}(x,y)}[-\log p_\theta(y \mid x)].$$

We can write this as a two-step expectation:

$$\begin{aligned}\mathbb{E}_{p_{\text{data}}(x,y)}[-\log p_\theta(y \mid x)] &= \mathbb{E}_{p_{\text{data}}(x)}\Big[\mathbb{E}_{p_{\text{data}}(y|x)}[-\log p_\theta(y \mid x)]\Big] \\ &= \mathbb{E}_{p_{\text{data}}(x)}\Big[H\big(p_{\text{data}}(\cdot \mid x), p_\theta(\cdot \mid x)\big)\Big],\end{aligned}$$

where for each fixed x we define the *cross-entropy*

$$H\big(p_{\text{data}}(\cdot \mid x), p_\theta(\cdot \mid x)\big) = -\sum_y p_{\text{data}}(y \mid x) \log p_\theta(y \mid x).$$

Cross-Entropy, Entropy, KL and NLL (General Case)

For each fixed x, we have the identity

$$H\big(p_{\text{data}}(\cdot \mid x), p_\theta(\cdot \mid x)\big) = H\big(p_{\text{data}}(\cdot \mid x)\big) + \mathrm{KL}\big(p_{\text{data}}(\cdot \mid x) \,\|\, p_\theta(\cdot \mid x)\big),$$

where

$$H\big(p_{\text{data}}(\cdot \mid x)\big) = -\sum_y p_{\text{data}}(y \mid x) \log p_{\text{data}}(y \mid x)$$

is the (conditional) entropy of the true labels given x.

Taking expectation over x:

$$\mathbb{E}_{p_{\text{data}}(x,y)}[-\log p_\theta(y \mid x)] = H_{p_{\text{data}}}(Y \mid X) + \mathbb{E}_{p_{\text{data}}(x)}\Big[\mathrm{KL}\big(p_{\text{data}}(\cdot \mid x) \,\|\, p_\theta(\cdot \mid x)\big)\Big],$$

where

$$H_{p_{\text{data}}}(Y \mid X) = \mathbb{E}_{p_{\text{data}}(x)}\Big[H\big(p_{\text{data}}(\cdot \mid x)\big)\Big].$$

Interpretation and generality

- $H_{p_{\text{data}}}(Y \mid X)$: intrinsic label uncertainty (Bayes error).
- The KL term: how far the model $p_\theta(y \mid x)$ is from the true conditional $p_{\text{data}}(y \mid x)$.
- This derivation is valid for *any discrete label space*: binary, multi-class, or any finite set.

Setup and Notation for Variational Inference

We consider a probabilistic model with:

- Latent variable z
- Observed data x
- Joint distribution $p(x, z)$
- Posterior distribution $p(z \mid x)$

Key quantities

- **Posterior:**
$$p(z \mid x) = \frac{p(x, z)}{p(x)}.$$
- **Evidence (marginal likelihood):**
$$p(x) = \int p(x, z)\, dz.$$

In many interesting models, $p(z \mid x)$ is intractable. **Variational inference (VI)** approximates $p(z \mid x)$ with a simpler distribution $q(z)$ and finds the best q by optimization.

Introducing a Variational Distribution

We want to approximate the posterior $p(z \mid x)$.

- Choose a variational family $\mathcal{Q} = \{q(z; \lambda)\}$, e.g. Gaussians with parameters λ.
- Goal: find $q^*(z) \in \mathcal{Q}$ that is as close as possible to the true posterior by finding the optimal value of λ: $q^*(z) = \arg\min_\lambda \mathrm{KL}(q_\lambda(z) \,\|\, p(z \mid x))$

Start from the marginal likelihood:

$$\log p(x) = \log \int p(x, z)\, dz.$$

Multiply and divide by any density $q(z)$:

$$\log p(x) = \log \int q(z) \frac{p(x, z)}{q(z)}\, dz = \log \mathbb{E}_{q(z)}\left[\frac{p(x, z)}{q(z)}\right].$$

Next: apply Jensen's inequality.

Deriving the ELBO via Jensen's Inequality

Recall Jensen's inequality for a concave function f:

$$f(\mathbb{E}[Y]) \ \geq \ \mathbb{E}[f(Y)].$$

For $f = \log$ (concave) and

$$Y = \frac{p(x,z)}{q(z)},$$

we obtain

$$\log \mathbb{E}_{q(z)}\left[\frac{p(x,z)}{q(z)}\right] \ \geq \ \mathbb{E}_{q(z)}\left[\log \frac{p(x,z)}{q(z)}\right].$$

Therefore

$$\begin{aligned} \log p(x) &= \log \mathbb{E}_{q(z)}\left[\frac{p(x,z)}{q(z)}\right] \\ &\geq \mathbb{E}_{q(z)}[\log p(x,z) - \log q(z)]\,. \end{aligned}$$

Evidence Lower Bound (ELBO)

$$\mathcal{L}(q) := \mathbb{E}_{q(z)}[\log p(x,z)] - \mathbb{E}_{q(z)}[\log q(z)],$$

so that

$$\log p(x) \ \geq \ \mathcal{L}(q).$$

LogNormal example: Jensen quantities as functions of the parameter s

Distribution. Let $X \sim \mathrm{LogNormal}(m, s^2)$, i.e.

$$\log X \sim \mathcal{N}(m, s^2), \qquad X > 0.$$

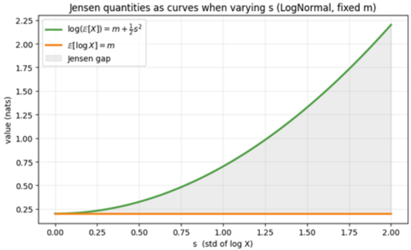
Closed-form expectations (fix m).

- $\mathbb{E}[\log X] = m$ *(flat line in s)*
- $\mathbb{E}[X] = \exp\left(m + \frac{1}{2}s^2\right)$
- $\log(\mathbb{E}[X]) = m + \frac{1}{2}s^2$ *(curves upward in s)*

Jensen gap.

$$\log(\mathbb{E}[X]) - \mathbb{E}[\log X] = \tfrac{1}{2}s^2 \;\geq\; 0,$$

with equality only when $s = 0$ (degenerate case).



Jensen geometry for log: points at $x = \mathbb{E}[X]$ (as x varies)

For any positive random variable X (with finite expectations),

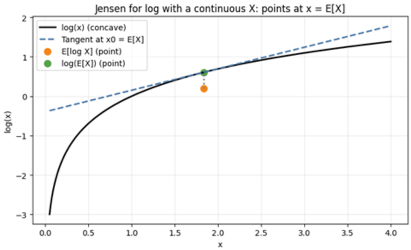
$$\log(\mathbb{E}[X]) \;\geq\; \mathbb{E}[\log X].$$

Geometric proof idea (concavity). Because log is concave, the tangent line at $x_0 = \mathbb{E}[X]$ lies above the curve:

$$\log(x) \;\leq\; \log(x_0) + \frac{1}{x_0}(x - x_0), \qquad x_0 = \mathbb{E}[X].$$

Taking expectations of both sides gives Jensen's inequality. **How to read the plot.**

- Both quantities are **single numbers** once the distribution is fixed.
- They are shown as two **points** at the same $x = \mathbb{E}[X]$: $(\mathbb{E}[X], \mathbb{E}[\log X])$ and $(\mathbb{E}[X], \log(\mathbb{E}[X]))$.
- The vertical gap is $\log(\mathbb{E}[X]) - \mathbb{E}[\log X] \geq 0$.



- E[X] = 1.83
- E[log X] = 0.2
- log(E[X]) = 0.61
- gap = 0.41

Gap to evidence: $\log p(x) = \mathcal{L}(q) + \mathrm{KL}(q(z)\|p(z \mid x))$

Derivation

Start from the KL divergence to the true posterior:

$$\mathrm{KL}(q(z) \,\|\, p(z \mid x)) = \mathbb{E}_q\left[\log \frac{q(z)}{p(z \mid x)}\right] = \mathbb{E}_q[\log q(z)] - \mathbb{E}_q[\log p(z \mid x)].$$

Use Bayes' rule $p(z \mid x) = \frac{p(x,z)}{p(x)}$, so $\log p(z \mid x) = \log p(x,z) - \log p(x)$. Plug in:

$$\mathrm{KL}(q\|p(\cdot \mid x)) = \mathbb{E}_q[\log q(z)] - \mathbb{E}_q[\log p(x,z) - \log p(x)] = \mathbb{E}_q[\log q(z)] - \mathbb{E}_q[\log p(x,z)] + \log p(x).$$

Rearrange:

$$\log p(x) = \underbrace{\mathbb{E}_q[\log p(x,z)] - \mathbb{E}_q[\log q(z)]}_{\mathcal{L}(q)} + \mathrm{KL}(q(z) \,\|\, p(z \mid x)) \,.$$

Key takeaway

Since $\mathrm{KL}(\cdot) \geq 0$, we have $\log p(x) \geq \mathcal{L}(q)$, and maximizing the ELBO is equivalent to minimizing $\mathrm{KL}(q(z)\|p(z \mid x))$.

Alternative Form of the ELBO

Decompose the joint:

$$\log p(x, z) = \log p(x \mid z) + \log p(z).$$

Plug into the ELBO:

$$\begin{aligned}\mathcal{L}(q) &= \mathbb{E}_{q(z)}[\log p(x, z)] - \mathbb{E}_{q(z)}[\log q(z)] \\ &= \mathbb{E}_{q(z)}[\log p(x \mid z)] + \mathbb{E}_{q(z)}[\log p(z)] - \mathbb{E}_{q(z)}[\log q(z)] \\ &= \mathbb{E}_{q(z)}[\log p(x \mid z)] - \mathrm{KL}(q(z) \| p(z)).\end{aligned}$$

Interpretation

- Minimizing L(q) will give a q that explains data well ($\mathbb{E}_{q(z)}[\log p(x \mid z)]$) and is close to the prior ($\mathrm{KL}(q(z) \| p(z))$).
- $\mathbb{E}_{q(z)}[\log p(x \mid z)]$ is the expected log-likelihood under the variational posterior.
- $\mathrm{KL}(q(z) \| p(z))$ penalizes deviation from the prior.
- VI trades off data fit vs. regularization toward the prior.

Mean-field approximation (independence)

Mean-field assumes independence among latent components:

$$q(z) = \prod_{i=1}^{m} q_i(z_i).$$

Why it matters:

- Makes expectations under q easier (breaks high-dimensional integrals).
- Enables **coordinate ascent** updates with closed forms in conjugate models.
- Scales to large latent spaces.

Cost: ignores posterior dependencies/correlations $\Rightarrow$ can underestimate uncertainty.

Optimal update for factor q_i:

$$\boxed{\log q_i^*(z_i) = \mathbb{E}_{q_{-i}}[\log p(x,z)] + \text{const}}$$

This is a *closed-form* solution: it directly gives the best q_i given the current values of q_{-i}. vspace0.7em Intuition: mean field approximates the influence of all other variables by their expected value $\mathbb{E}_{q_{-i}}[\log p(x,z)]$

Mean-Field VI: Coordinate Ascent Update

Why this form? Mean-field VI solves

$$\min_q \mathrm{KL}\big(q(z) \,\|\, p(z \mid x)\big) \qquad \text{s.t.} \qquad q(z) = \prod_i q_i(z_i).$$

Taking the functional derivative of the ELBO w.r.t. q_i yields:

$$q_i^*(z_i) \;\propto\; \exp\big(\mathbb{E}_{q_{-i}}[\log p(x,z)]\big)\,.$$

Meaning: q_i mimics the conditional $p(z_i \mid x, z_{-i})$ but with z_{-i} replaced by its variational expectation under q_{-i}.

Coordinate Ascent Algorithm:

$$q_1^{(t+1)}(z_1) \propto \exp\left(\mathbb{E}_{q_{-1}^{(t)}}[\log p(x,z)]\right),$$

$$q_2^{(t+1)}(z_2) \propto \exp\left(\mathbb{E}_{q_{-2}^{(t)}}[\log p(x,z)]\right), \quad \dots$$

Repeat cyclically until convergence.

Intuition: Each factor updates by matching the model's conditional structure, averaged over the remaining latent variables according to $q(z)$.

Mean-Field VI: A Simple Gaussian Example

Model:

$$x_n \sim \mathcal{N}(\mu, \tau^{-1}), \qquad p(\mu \mid \tau) = \mathcal{N}(\mu_0, (\lambda_0 \tau)^{-1}), \qquad p(\tau) = \mathrm{Gamma}(a_0, b_0).$$

Goal: Illustrate coordinate ascent VI (even though the true posterior is conjugate).

Mean-field assumption:

$$q(\mu, \tau) = q_\mu(\mu)\, q_\tau(\tau).$$

Coordinate ascent updates:

$$\boxed{\log q_\mu^*(\mu) = \mathbb{E}_{q_\tau}[\log p(X, \mu, \tau)] + \text{const}}$$

$$\boxed{\log q_\tau^*(\tau) = \mathbb{E}_{q_\mu}[\log p(X, \mu, \tau)] + \text{const}}$$

Idea: Each factor matches the model's conditional distribution, averaged over the other variational factors.

Updating $q_\mu(\mu)$

Log-joint terms involving μ:

$$\log p(X, \mu, \tau) = -\frac{\tau}{2} \sum_{n=1}^{N} (x_n - \mu)^2 - \frac{\lambda_0 \tau}{2} (\mu - \mu_0)^2 + C.$$

Apply MF update:

$$\log q_\mu^*(\mu) = -\frac{\mathbb{E}_{q_\tau}[\tau]}{2} \left[\sum_{n=1}^{N} (x_n - \mu)^2 + \lambda_0 (\mu - \mu_0)^2 \right] + \text{const.}$$

Recognize a Gaussian:

$$q_\mu^*(\mu) = \mathcal{N}(\mu_N, \tau_N^{-1}),$$

$$\mu_N = \frac{\lambda_0 \mu_0 + N\bar{x}}{\lambda_0 + N}, \qquad \tau_N = (\lambda_0 + N)\, \mathbb{E}_{q_\tau}[\tau].$$

Note: q_μ depends on $\mathbb{E}[\tau]$, so we alternate updates.

Updating $q_\tau(\tau)$ and Summary

Log-joint terms involving τ:

$$\log p(X,\mu,\tau) = \left(a_0 - 1 + \frac{N+1}{2}\right)\log\tau - \tau\left[b_0 + \frac{1}{2}\left(\sum_{n=1}^{N}(x_n-\mu)^2 + \lambda_0(\mu-\mu_0)^2\right)\right] + C.$$

Apply MF update:

$$q_\tau^*(\tau) = \mathrm{Gamma}(a_N, b_N),$$

$$a_N = a_0 + \frac{N+1}{2}, \qquad b_N = b_0 + \frac{1}{2}\,\mathbb{E}_{q_\mu}\left[\sum_{n=1}^{N}(x_n-\mu)^2 + \lambda_0(\mu-\mu_0)^2\right].$$

Summary:

- q_μ and q_τ have closed-form Gaussian and Gamma updates.
- Updates depend on each other ⇒ run *cyclic* coordinate ascent.
- This example shows the classic MFVI pattern: exponentiate expected log-joint and normalize.

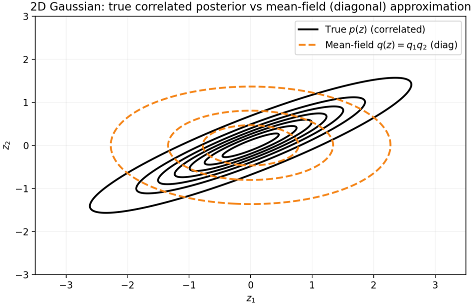
Correlated posterior vs mean-field: geometry of uncertainty (contours)

Setup. True posterior: $p(z) = \mathcal{N}(0, \Sigma)$ with correlation. Mean-field Gaussian: $q(z) = \mathcal{N}(0, \Sigma_q)$ with Σ_q diagonal (equivalently $q(z) = q_1(z_1)q_2(z_2)$).

What we expect. Correlation rotates the principal axes of uncertainty:

- correlated Gaussian ⇒ **tilted ellipses** (off-axis principal components),
- diagonal covariance ⇒ **axis-aligned ellipses** (no tilt).

- **Observed:** black contours (true p) are tilted.
- **Observed:** dashed contours (mean-field q) are axis-aligned.
- **Meaning:** q cannot "rotate" to match p because independence forbids correlation.



Takeaway. Mean-field captures some spread but misses joint structure ⇒ biased uncertainty and wrong dependence.

Fixed-form variational inference (beyond mean-field)

- The independence assumption in *mean-field* VI can be restrictive.
- **Fixed-form VI:** choose a parametric family $q_\lambda(z)$ with variational parameters λ.
- Optimize the approximation by solving

$$\lambda^\star \in \arg\min_{\lambda} \; \mathrm{KL}(q_\lambda(z) \,\|\, p(z \mid x))\,.$$

- In practice we maximize a tractable lower bound (the ELBO) instead of working directly with $\log p(x)$.

Examples of fixed-form families $q_\lambda(z)$

- **Full Gaussian:** $q_\lambda(z) = \mathcal{N}(z \mid \mu, \Sigma)$, $\quad \lambda = (\mu, L)$ with $\Sigma = LL^\top$ (Cholesky).
- **Low-rank + diagonal:** $q_\lambda(z) = \mathcal{N}(z \mid \mu, aa^\top + D)$, $\quad \lambda = (\mu, a, d)$, $D = \mathrm{Diag}(d)$.
- **Richer:** mixtures of Gaussians, flows, etc.

Fixed-form VI: gradient-based optimization of the ELBO

Objective (ELBO). Define

$$\mathcal{L}(\lambda) \equiv \mathcal{L}(q_\lambda) := \mathbb{E}_{q_\lambda(z)}\big[\log p(x,z)\big] - \mathbb{E}_{q_\lambda(z)}\big[\log q_\lambda(z)\big] = \mathbb{E}_{q_\lambda(z)}\left[\log \frac{p(x,z)}{q_\lambda(z)}\right].$$

Gradient ascent

For $t = 0, 1, 2, \ldots$ until convergence:

$$\lambda^{(t+1)} = \lambda^{(t)} + a_t \, \nabla_\lambda \mathcal{L}\big(\lambda^{(t)}\big), \qquad a_t > 0.$$

Stop e.g. when the change in the objective is small:

$$\left|\mathcal{L}(\lambda^{(t+1)}) - \mathcal{L}(\lambda^{(t)})\right| < \varepsilon.$$

Link to your "gap" slide: $\log p(x) = \mathcal{L}(q_\lambda) + \mathrm{KL}(q_\lambda(z) \| p(z \mid x))$, so maximizing $\mathcal{L}$ is equivalent to minimizing the KL gap.

Gradient of the ELBO is an expectation (score-function form)

Define

$$h_\lambda(z) := \log p(x,z) - \log q_\lambda(z) \quad \Longrightarrow \quad \mathcal{L}(\lambda) = \int q_\lambda(z)\, h_\lambda(z)\, dz.$$

Derivation sketch (log-derivative trick)

$$\begin{aligned}\nabla_\lambda \mathcal{L}(\lambda) &= \nabla_\lambda \int q_\lambda(z)\, h_\lambda(z)\, dz = \int \nabla_\lambda\big(q_\lambda(z)\, h_\lambda(z)\big)\, dz \\ &= \int \Big((\nabla_\lambda q_\lambda(z))\, h_\lambda(z) + q_\lambda(z)\, \nabla_\lambda h_\lambda(z) \Big)\, dz.\end{aligned}$$

Since $\nabla_\lambda h_\lambda(z) = -\nabla_\lambda \log q_\lambda(z)$ and $\nabla_\lambda q_\lambda(z) = q_\lambda(z) \nabla_\lambda \log q_\lambda(z)$, the second term cancels because

$$\int \nabla_\lambda q_\lambda(z)\, dz = \nabla_\lambda \int q_\lambda(z)\, dz = \nabla_\lambda 1 = 0.$$

Therefore,

$$\nabla_\lambda \mathcal{L}(\lambda) = \int q_\lambda(z)\, h_\lambda(z)\, \nabla_\lambda \log q_\lambda(z)\, dz = \mathbb{E}_{q_\lambda(z)}\big[h_\lambda(z)\, \nabla_\lambda \log q_\lambda(z)\big].$$

VI via stochastic gradient ascent (Monte Carlo estimate)

From the previous slide,

$$\nabla_\lambda \mathcal{L}(\lambda) = \mathbb{E}_{q_\lambda(z)}\big[h_\lambda(z)\,\nabla_\lambda \log q_\lambda(z)\big], \qquad h_\lambda(z) = \log p(x,z) - \log q_\lambda(z).$$

Monte Carlo estimator

Draw $z^{(1)},\ldots,z^{(S)} \sim q_\lambda(z)$ and estimate $\widehat{\nabla_\lambda \mathcal{L}}(\lambda) = \frac{1}{S}\sum_{s=1}^{S}\Big(h_\lambda(z^{(s)})\,\nabla_\lambda \log q_\lambda(z^{(s)})\Big)$.

Basic fixed-form VI algorithm (SGA)

1. Initialize $\lambda^{(0)}$, choose step sizes $\{a_t\}_{t\geq 0}$ and MC size S.
2. For $t = 0, 1, 2, \ldots$:
    1. Sample $z^{(1)},\ldots,z^{(S)} \sim q_{\lambda^{(t)}}(z)$.
    2. Compute $\widehat{\nabla_\lambda \mathcal{L}}(\lambda^{(t)})$ using the MC estimator above.
    3. Update $\lambda^{(t+1)} = \lambda^{(t)} + a_t\,\widehat{\nabla_\lambda \mathcal{L}}(\lambda^{(t)})$.
3. Stop when a convergence criterion is met (e.g., stabilized $\mathcal{L}$ or parameter changes).

The requirements for inference (score-function / black-box VI)

Noisy (Monte Carlo) gradient of the ELBO:

$$\widehat{\nabla_\lambda \mathcal{L}}(\lambda) = \frac{1}{S}\sum_{s=1}^{S} \nabla_\lambda \log q_\lambda(z^{(s)}) \left(\log p(x, z^{(s)}) - \log q_\lambda(z^{(s)})\right), \qquad z^{(s)} \sim q_\lambda(z).$$

To compute the noisy gradient of the ELBO we need:

- **Sampling from** $q_\lambda(z)$ (draw $z^{(s)}$).
- **Evaluating** $\nabla_\lambda \log q_\lambda(z)$ (score function of the variational family).
- **Evaluating** $\log p(x, z)$ and $\log q_\lambda(z)$ (joint density and variational density).

Black-box criterion

There is no model-specific inference work: as long as we can evaluate $\log p(x, z)$ (and sample / score q_λ), we can optimize $\mathcal{L}(q_\lambda)$ using stochastic gradients.

Pathwise (Reparameterization) Estimator: Assumptions

Goal: Compute $\nabla_\lambda \mathcal{L}(\lambda)$ for

$$\mathcal{L}(\lambda) \;=\; \mathbb{E}_{z\sim q_\lambda}\big[f(z)\big].$$

Assume we can reparameterize the variational sample:

$$z \;=\; g(\lambda,\epsilon), \qquad \epsilon \sim p(\epsilon) \text{ independent of } \lambda.$$

Example (Gaussian q_λ):

$$q_\lambda(z) = \mathcal{N}(z;\mu,\sigma^2), \quad \epsilon \sim \mathcal{N}(0,1), \quad z = g(\lambda,\epsilon) = \mu + \sigma\,\epsilon.$$

Smoothness: Assume $f(z)$, $\log p(x,z)$, and $\log q_\lambda(z)$ are differentiable in z so that gradients can flow through $z = g(\lambda,\epsilon)$.

Pathwise (Reparameterization) Estimator: Gradient Formula

Rewrite the expectation using the reparameterization:

$$\mathcal{L}(\lambda) = \mathbb{E}_{z \sim q_\lambda}[f(z)] = \mathbb{E}_{\epsilon \sim p(\epsilon)}\Big[f\big(g(\lambda, \epsilon)\big)\Big].$$

Differentiate under the expectation (valid by dominated convergence / smoothness):

$$\nabla_\lambda \mathcal{L}(\lambda) = \mathbb{E}_{\epsilon \sim p(\epsilon)}\Big[\nabla_\lambda f\big(g(\lambda, \epsilon)\big)\Big] = \mathbb{E}_\epsilon\Big[\nabla_z f(z)\big|_{z=g(\lambda,\epsilon)} \, \nabla_\lambda g(\lambda, \epsilon)\Big].$$

In VI (ELBO): If $f(z) = \log p(x, z) - \log q_\lambda(z)$, then

$$\nabla_\lambda \mathbb{E}_{q_\lambda}[\log p(x, z) - \log q_\lambda(z)] = \mathbb{E}_\epsilon\Big[\big(\nabla_z \log p(x, z) - \nabla_z \log q_\lambda(z)\big) \, \nabla_\lambda g(\lambda, \epsilon)\Big].$$

This is the reparameterization gradient / pathwise estimator.

From Formula to Algorithm

We want to maximize the ELBO:

$$\mathcal{L}(\lambda) \;=\; \mathbb{E}_{z\sim q_\lambda}\big[\log p(x,z) - \log q_\lambda(z)\big].$$

Stochastic gradient ascent step:

1. Sample noise $\epsilon^{(m)} \sim p(\epsilon)$ for $m = 1, \ldots, M$.
2. Reparameterize $z^{(m)} = g(\lambda, \epsilon^{(m)})$.
3. Compute

$$g_\lambda \;\approx\; \frac{1}{M}\sum_{m=1}^{M}\Big(\nabla_z \log p(x, z^{(m)}) - \nabla_z \log q_\lambda(z^{(m)})\Big)\;\nabla_\lambda g(\lambda, \epsilon^{(m)}).$$

4. Update $\lambda \leftarrow \lambda + \eta\, g_\lambda$.

Gaussian case:

$$z = \mu + \sigma\epsilon, \quad \nabla_\mu g = 1, \quad \nabla_\sigma g = \epsilon.$$

Thus

$$\nabla_\mu \mathcal{L} \approx \frac{1}{M}\sum_m \Big(\nabla_z \log p - \nabla_z \log q_\lambda\Big)\Big|_{z^{(m)}}, \qquad \nabla_\sigma \mathcal{L} \approx \frac{1}{M}\sum_m \Big(\nabla_z \log p - \nabla_z \log q_\lambda\Big)\Big|_{z^{(m)}}\epsilon^{(m)}.$$

Worked Example: 1D Gaussian VI

Model:

$$p(z) = \mathcal{N}(0,1), \quad p(x \mid z) = \mathcal{N}(x; z, 1), \quad \text{observed } x.$$

Variational family: $q_\lambda(z) = \mathcal{N}(z; \mu, \sigma^2)$ with $\lambda = (\mu, \sigma)$ and

$$z = g(\lambda, \epsilon) = \mu + \sigma\epsilon, \quad \epsilon \sim \mathcal{N}(0,1).$$

ELBO integrand:

$$f(z) = \log p(x,z) - \log q_\lambda(z) = -\tfrac{1}{2}(x-z)^2 - \tfrac{1}{2}z^2 \; - \; \Big(- \tfrac{1}{2}\log(2\pi\sigma^2) - \tfrac{1}{2}\frac{(z-\mu)^2}{\sigma^2}\Big) + C.$$

Pathwise gradients:

$$\nabla_z \log p(x,z) = (x-z) - z = x - 2z, \quad \nabla_z \log q_\lambda(z) = -\frac{z-\mu}{\sigma^2}.$$

$$\nabla_\mu g = 1, \qquad \nabla_\sigma g = \epsilon.$$

SG estimator (single sample):

$$\widehat{\nabla_\mu \mathcal{L}} = \left[x - 2z + \tfrac{z-\mu}{\sigma^2}\right]\Big|_{z=\mu+\sigma\epsilon}, \quad \widehat{\nabla_\sigma \mathcal{L}} = \left[x - 2z + \tfrac{z-\mu}{\sigma^2}\right]\Big|_{z=\mu+\sigma\epsilon} \cdot \epsilon.$$

Update: $\mu \leftarrow \mu + \eta\,\widehat{\nabla_\mu \mathcal{L}},\ \sigma \leftarrow \sigma + \eta\,\widehat{\nabla_\sigma \mathcal{L}}.$

Score-function estimator vs. pathwise estimator (reparameterization)

Score-function (REINFORCE) estimator

- Differentiates the **variational density**:

$$\nabla_\lambda \log q_\lambda(z).$$

- Works for **discrete and continuous** latent variables.
- Applies to a **large class** of variational families q_λ (no reparameterization needed).
- **Downside:** variance can be large (often needs baselines/control variates).

Pathwise (reparameterization) estimator

- Rewrite sampling as

$$z = g(\epsilon; \lambda), \qquad \epsilon \sim p(\epsilon)$$

(e.g., $\epsilon \sim \mathcal{N}(0, I)$ for Gaussians).

- Differentiate the **sample path** (the integrand) through z:

$$\nabla_\lambda \Big[\log p\big(x, g(\epsilon; \lambda)\big) - \log q_\lambda\big(g(\epsilon; \lambda)\big) \Big].$$

- Requires **(reparameterizable) continuous** q_λ and a differentiable model.
- **Upside:** typically much lower-variance gradients.

References for Variational Inference and Reparameterization

Core Textbook References

- D. MacKay (2003), *Information Theory, Inference, and Learning Algorithms*, Chapter 33: “Variational Methods”, `https://www.inference.org.uk/itprnn/book.pdf`

Key Review Articles

- D. Blei, A. Kucukelbir, and J. McAuliffe (2017), “Variational Inference: A Review for Statisticians”, *JASA*, `https://arxiv.org/abs/1601.00670`.

Advanced Tutorials

- T. Broderick (2018), “Variational Bayes and Beyond: Bayesian Inference for Big Data”, ICML Tutorial `https://tamarabroderick.com/tutorial_2018_icml.html`.

Online Course

- P. Huijse et al., *Bayesian Learning and Neural Networks*, available at: `https://phuijse.github.io/BLNNbook/README.html`

Course Slide Reference

- M. Villani (2024), *Advanced Bayesian Learning, Lecture 6: Beyond Mean-Field VI*.

Acknowledgements/ Document preparation

This document was developed and converted into LaTeX slides and formatted with assistance from ChatGPT (OpenAI) and CoPilot (Microsoft). The instructor modified the source text and verified the final structure and wording.

Appendix

Problem: gradients of expectations

In VI and VAEs we optimize objectives of the form:

$$\mathcal{L}(\lambda) = \mathbb{E}_{z \sim q_\lambda}[f(z)].$$

We need $\nabla_\lambda \mathcal{L}(\lambda)$, but z is random.

Two main unbiased estimators:

- **Reparameterization (pathwise)**: differentiate through $z = g(\lambda, \epsilon)$
- **Score-function (REINFORCE)**: use $\nabla_\lambda \log q_\lambda(z)$

What it is

If we can write

$$z = g(\lambda, \epsilon), \quad \epsilon \sim p(\epsilon) \text{ independent of } \lambda,$$

then

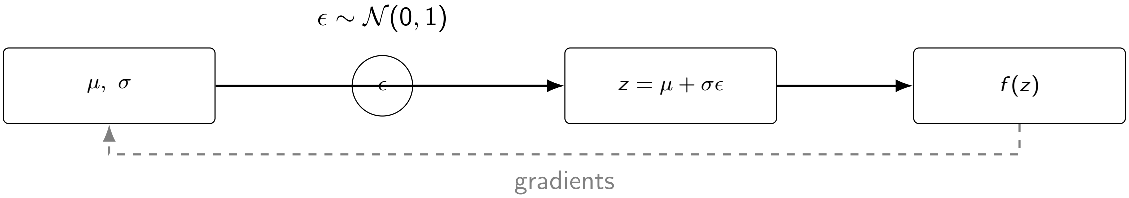
$$\nabla_\lambda \mathbb{E}_{q_\lambda}[f(z)] = \mathbb{E}_\epsilon\big[\nabla_\lambda f(g(\lambda, \epsilon))\big].$$

Gaussian case:

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

This lets gradients flow through the computation graph.

Reparameterization computation graph



Backprop flows through $z = \mu + \sigma\epsilon$; randomness is isolated in ϵ.

Intuition. Move randomness to ϵ and make z a deterministic function of (μ, σ, ϵ).
Why it helps. Backprop can compute $\partial f/\partial\mu$ and $\partial f/\partial\sigma$ through z.
Result. Typically much lower-variance gradients than REINFORCE.

Score-function estimator

Using the log-derivative identity:

$$\nabla_\lambda \mathbb{E}_{q_\lambda}[f(z)] = \mathbb{E}_{q_\lambda}[f(z)\nabla_\lambda \log q_\lambda(z)].$$

Monte Carlo:

$$\widehat{g} = \frac{1}{S}\sum_{s=1}^{S} f(z^{(s)})\nabla_\lambda \log q_\lambda(z^{(s)}), \quad z^{(s)} \sim q_\lambda.$$

Pros: works for discrete z. **Cons:** often high variance.

Control variates / baselines

Subtract a baseline b (independent of z):

$$\mathbb{E}[(f(z) - b)\nabla_\lambda \log q_\lambda(z)] = \mathbb{E}[f(z)\nabla_\lambda \log q_\lambda(z)].$$

So the estimator stays unbiased but can have much lower variance.

- Common choice: $b \approx \mathbb{E}[f(z)]$ (running mean)
- More advanced: learned baseline / critic (RL)