Variational Inference

Justus Piater

Table of Contents

1. Calculus of Variations

1.1. Functionals

	Function $y(x)$	***Functional*** $J[y(x)]$
Maps	a value to a value	a function to a value
Study of Extrema	Calculus	Calculus of Variations
Minimum	$y(x) \leq y(x+\epsilon)$	$J[y(x)] \leq J[y(x)+\epsilon\eta(x)]$
	differential $\epsilon = \mathrm{d}x$	***variation*** $\epsilon\eta(x) = \delta y$
Extremum	*derivative* $\frac{\mathrm{d}y}{\mathrm{d}x} = 0$	***functional derivative*** $\frac{\delta J}{\delta y} = 0$

η is an (almost) *arbitrary* function (the details depend on J).

Examples of functionals:

- the length of a curve
- the entropy of a probability distribution

While *calculus* seeks *values* that extremize a *function*, *calculus of variations* seeks *functions* that extremize a *functional*.

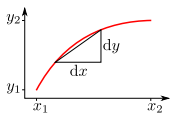
Notably, this can be done without limiting the search to a specific parametric family of functions.

1.2. Example: Shortest Curve Between Two Points

See also Wikipedia[1].

Problem: Find the function $y(x)$ that connects two points (x_1, y_1) and (x_2, y_2) that minimizes its arc length $A[y]$.

$$A[y] = \int_{x_1}^{x_2} L(y'(x))\,\mathrm{d}x = \int_{x_1}^{x_2} \sqrt{1+\left(\frac{\mathrm{d}y}{\mathrm{d}x}\right)^2}\,\mathrm{d}x$$



The integrand is Pythagoras on an infinitesimal triangle. Check the value of the integral if you set, e.g., $\mathrm{d}y = 0$ or $\mathrm{d}y = \mathrm{d}x$.

$\Phi(\epsilon) = A[f + \epsilon\eta]$ has a minimum at $\epsilon = 0$. Thus, with $y' = f' + \epsilon\eta'$:

$$\Phi'(0) = \left.\frac{\mathrm{d}\Phi}{\mathrm{d}\epsilon}\right|_{\epsilon=0} = \left.\frac{\mathrm{d}}{\mathrm{d}\epsilon}\int_{x_1}^{x_2} L\,\mathrm{d}x\right|_{\epsilon=0} = \int_{x_1}^{x_2} \left.\frac{\mathrm{d}L}{\mathrm{d}\epsilon}\right|_{\epsilon=0}\mathrm{d}x = 0$$

$$\frac{\mathrm{d}L}{\mathrm{d}\epsilon} = \frac{\mathrm{d}L}{\mathrm{d}y'}\frac{\mathrm{d}y'}{\mathrm{d}\epsilon} = \frac{\mathrm{d}L}{\mathrm{d}y'}\eta'; \quad y' = f' \text{ at } \epsilon = 0$$

$$\Phi'(0) = \int_{x_1}^{x_2} \frac{\mathrm{d}L}{\mathrm{d}f'}\eta'\,\mathrm{d}x = \frac{\mathrm{d}L}{\mathrm{d}f'}\eta\Big|_{x_1}^{x_2} - \int_{x_1}^{x_2} \eta\frac{\mathrm{d}}{\mathrm{d}x}\frac{\mathrm{d}L}{\mathrm{d}f'}\,\mathrm{d}x = 0$$

$$\frac{\mathrm{d}}{\mathrm{d}x}\frac{\mathrm{d}L}{\mathrm{d}f'} = 0 = \frac{\mathrm{d}}{\mathrm{d}x}\frac{f'}{\sqrt{1+f'^2}}$$

- The differentiation interchanges with the integral thanks to Leibnitz' Rule[2].
- η is an arbitrary differentiable function of x that vanishes at x_1 and x_2.
- The expression involving the highlighted term results from applying *integration by parts* to the expression on its left.

  The highlighted term equals zero because $\eta(x_1) = \eta(x_2) = 0$.

  One obvious way to get the value of the remaining integral in this expression to vanish is to set its integrand to zero; this implies $\frac{\mathrm{d}}{\mathrm{d}x}\frac{\mathrm{d}L}{\mathrm{d}f'} = 0$ because $\eta(x)$ can be anything. According to the *Fundamental Lemma of Calculus of Variations*, this is actually the *only* way.
- Incidentally, this derivation is a special case of the more general and widespread class of functionals $J[y] = \int_{x_1}^{x_2} L(x, y(x), y'(x))\,\mathrm{d}x$ whose final condition $\frac{\partial L}{\partial f} - \frac{\mathrm{d}}{\mathrm{d}x}\frac{\partial L}{\partial f'} = 0$ is known as the *Euler-Lagrange Equation*.

Since the derivative is zero the function must be constant:

[1] [2] https://en.wikipedia.org/wiki/Calculus_of_variations#Example
https://en.wikipedia.org/wiki/Leibniz_integral_rule

$$
\begin{aligned}
\frac{f'}{\sqrt{1+f'^2}} &= c \\
\frac{f'^2}{1+f'^2} &= c^2 \qquad 0 \le c^2 < 1 \\
f'^2 &= \frac{c^2}{1-c^2} \\
f' &= m \qquad m = \frac{c}{\sqrt{1-c^2}} \\
f(x) &= mx + b
\end{aligned}
$$

Since the derivative is constant the function must be linear – the shortest curve connecting two points is a straight line!

2. Variational Inference

2.1. Typical Setting

Given: a model $p(\mathbf{Z}, \mathbf{X})$ and observations $\mathbf{x} = \{x_1, \ldots, x_N\}$

Sought: the posterior $p(\mathbf{Z}|\mathbf{x})$ and the evidence $p(\mathbf{x})$

Examples are graphical models composed of latent and observed variables $\mathbf{Z}$ and $\mathbf{X}$, respectively. This includes HMM, but due to their tree structure these allow exact inference. For more general graphical models we have to resort to approximate inference.

Approach: Approximate the intractable posterior $p(\mathbf{Z}|\mathbf{x})$ with $q(\mathbf{Z}|\mathbf{x})$ from a tractable family.

Examples of tractable families include factored models, tree-structured models, and multivariate normal distributions.

Method: Use calculus of variations limited to the chosen family of probability distributions.

2.2. Example Application: Image Denoising

$\mathbf{x} \sim p(\mathbf{X}|\mathbf{z})$ $\quad$ $\underset{\mathbf{z}}{\operatorname{argmax}}\, p(\mathbf{z}|\mathbf{x})$ $\quad$ True $\mathbf{z}$

2.3. Approximating $p(\mathbf{Z}|\mathbf{x})$ by $q(\mathbf{Z}|\mathbf{x})$

[Murphy 2012 Sec. 21.2 (Intro)]

- Minimize $\mathrm{KL}(p\|q) = \sum_{\mathbf{z}} p(\mathbf{z}|\mathbf{x}) \log \frac{p(\mathbf{z}|\mathbf{x})}{q(\mathbf{z}|\mathbf{x})} = \mathbb{E}_{p(\mathbf{z}|\mathbf{x})}\left[\log \frac{p(\mathbf{z}|\mathbf{x})}{q(\mathbf{z}|\mathbf{x})}\right]$?

  No, since evaluating $p(\mathbf{z}|\mathbf{x})$ is intractable.

- Instead, minimize $\widetilde{\mathrm{KL}}(q\|p) = J(q)$:

$$\begin{aligned} J(q) &= \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \log \frac{q(\mathbf{z}|\mathbf{x})}{\tilde{p}(\mathbf{z}|\mathbf{x})} \qquad \tilde{p}(\mathbf{z}|\mathbf{x}) = p(\mathbf{z}|\mathbf{x})Z = p(\mathbf{z},\mathbf{x}) \\ &= \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \log \frac{q(\mathbf{z}|\mathbf{x})}{p(\mathbf{z}|\mathbf{x})} - \log Z \\ &= \mathrm{KL}(q\|p) - \log Z \end{aligned}$$

  $Z = p(\mathbf{x})$ is intractable to compute but constant (does not depend on q or $\mathbf{z}$), so maximizing J will force q to become close to p.

Equivalently to minimizing J we can *maximize* the ***Evidence Lower Bound (ELBO)***

$$L(q) = -J(q) = \log p(\mathbf{x}) - \mathrm{KL}(q(\mathbf{Z}|\mathbf{x}) \,\|\, p(\mathbf{Z}|\mathbf{x})).$$

2.4. Mean-Field Approximation

[Murphy 2012 Sec. 21.3.1]

Approximation: $q(\mathbf{Z}|\mathbf{x}) = \prod_n q_n(\mathbf{Z}_n|\mathbf{x})$ that factors into disjoint groups $\mathbf{Z}_n$

Note that we are not making any assumptions about the (parametric) form of the q_n.

Sought: $\min_{q_1,\ldots,q_N} \mathrm{KL}(q\|p)$

Approach: Coordinate descent, e.g.,

$$\mathbb{E}_{-q_2}[f(\mathbf{Z})] = \sum_{\mathbf{z}_1} \sum_{\mathbf{z}_3} q_1(\mathbf{z}_1|\mathbf{x}) q_3(\mathbf{z}_3|\mathbf{x}) f(\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3)$$

where $\mathbb{E}_{-q_n}[f(\mathbf{Z})]$ is the expectation of $f(\mathbf{Z})$ over all variables in $\mathbf{Z}$ except for $\mathbf{Z}_n$.

When updating the q_n one at a time we only need to consider $\mathbf{Z}_n$'s *Markov blanket*, i.e. those variables that share a factor with $\mathbf{Z}_n$ in $p(\mathbf{Z}, \mathbf{X})$; the other factors are absorbed into the constant term of $L(q)$.

> ***Note***
>
> The Mean-Field Approximation is a very general method that can be used with a wide variety of parametric forms of the q_n, discrete or continous.

2.5. Rewriting the Mean-Field ELBO

In the following we are omitting the dependencies on $\mathbf{x}$ for brevity.

Single out the terms that involve q_n and treat all other terms as constants:

$$
\begin{aligned}
L(q_n) &= \sum_{\mathbf{z}} \prod_i q_i(\mathbf{z}_i) \left[\log \tilde{p}(\mathbf{z}) - \sum_k \log q_k(\mathbf{z}_k) \right] \\
&= \sum_{\mathbf{z}_n} \sum_{\mathbf{z}_{-n}} q_n(\mathbf{z}_n) \prod_{i \neq n} q_i(\mathbf{z}_i) \left[\log \tilde{p}(\mathbf{z}) - \sum_k \log q_k(\mathbf{z}_k) \right] \\
&= \sum_{\mathbf{z}_n} q_n(\mathbf{z}_n) \sum_{\mathbf{z}_{-n}} \prod_{i \neq n} q_i(\mathbf{z}_i) \log \tilde{p}(\mathbf{z}) - \\
&\quad \sum_{\mathbf{z}_n} q_n(\mathbf{z}_n) \sum_{\mathbf{z}_{-n}} \prod_{i \neq n} q_i(\mathbf{z}_i) \left[\sum_{k \neq n} \log q_k(\mathbf{z}_k) + \log q_n(\mathbf{z}_n) \right] \\
&= \sum_{\mathbf{z}_n} q_n(\mathbf{z}_n) \log f_n(\mathbf{z}_n) - \sum_{\mathbf{z}_n} q_n(\mathbf{z}_n) \log q_n(\mathbf{z}_n) + \text{const} \\
\log f_n(\mathbf{z}_n) &= \sum_{\mathbf{z}_{-n}} \prod_{i \neq n} q_i(\mathbf{z}_i) \log \tilde{p}(\mathbf{z}) = \mathbb{E}_{-q_n}[\log \tilde{p}(\mathbf{z})]
\end{aligned}
$$

2.6. Mean-Field Update Equations

Disregarding the constant term:

$$
\begin{aligned}
L(q_n) &= \sum_{\mathbf{z}_n} q_n(\mathbf{z}_n) \log f_n(\mathbf{z}_n) - \sum_{\mathbf{z}_n} q_n(\mathbf{z}_n) \log q_n(\mathbf{z}_n) \\
&= -\sum_{\mathbf{z}_n} q_n(\mathbf{z}_n) \left(\log q_n(\mathbf{z}_n) - \log f_n(\mathbf{z}_n) \right) \\
&= -\sum_{\mathbf{z}_n} q_n(\mathbf{z}_n) \log \frac{q_n(\mathbf{z}_n)}{f_n(\mathbf{z}_n)} \\
&= -\mathrm{KL}(q_n(\mathbf{Z}_n) \| f_n(\mathbf{Z}_n))
\end{aligned}
$$

This is maximized by $q_n = f_n = \exp(\mathbb{E}_{-q_n}[\log \tilde{p}(\mathbf{z})])$, which – since we can compute $\tilde{p}(\mathbf{z})$ – we can achieve by setting

$$
\begin{aligned}
q_n(\mathbf{z}_n) &= \frac{1}{Z_n} \exp(\mathbb{E}_{-q_n}[\log \tilde{p}(\mathbf{z})]) \\
\log q_n(\mathbf{z}_n) &= \mathbb{E}_{-q_n}[\log \tilde{p}(\mathbf{z})] - Z_n
\end{aligned}
$$

where $Z_n = \sum_{\mathbf{z}_n} q_n(\mathbf{z}_n)$ can be computed because the q_n are designed to be tractable.

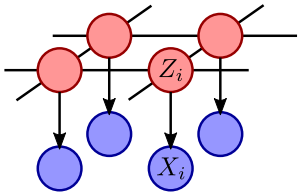
These update equations give the method its name: It assigns means to (log) factors.

This procedure is closely related to the MCMC *Gibbs sampling* method that passes samples rather than means. Passing means tends to be more efficient than passing samples because many samples can be represented by a single mean. On the other hand, samples can be sparse, potentially allowing Gibbs sampling to scale better.

3. Mean Field for the Ising Model

3.1. Ising Model

[Murphy 2012 Sec. 21.3.2]



The $z_i \in \{-1, +1\}$ are the hidden (pixel) values (of a clean image), and the x_i are their perturbed observations (in the noisy image).

$$
\begin{aligned}
p(\mathbf{z}) &= \frac{1}{Z_0}\exp(-E_0(\mathbf{z})) && \text{prior} \\
E_0(\mathbf{z}) &= -\sum_i \sum_{j \in \text{nbr}(i)} W_{ij} z_i z_j && \text{energy of the prior} \\
p(\mathbf{x}|\mathbf{z}) &= \prod_i \exp(\ell_i(z_i)) && \text{likelihood; } \ell_i = \log p(x_i|z_i) \\
p(\mathbf{z}|\mathbf{x}) &= \frac{1}{Z}\exp(-E(\mathbf{z})) && \text{posterior} \\
E(\mathbf{z}) &= E_0(\mathbf{x}) - \sum_i \ell_i(x_i) && \text{energy of the posterior}
\end{aligned}
$$

$j \in \text{nbr}(i)$ iff z_i and z_j are neighbors on the grid.

In an image denoising application, W is typically the matrix of ones, and $\ell(z) = -\frac{1}{2\sigma^2}(x - z)^2$, the exponent of a Gaussian independent pixel noise model. (Its normalization factor is dropped but this is inconsequential because it will cancel later.)

Exercise: Using Bayes' Rule, show that the posterior has this form!

3.2. Mean Field Approximation of the Ising Model

Approximation: Approximate $p(\mathbf{z}|\mathbf{x})$ by a *fully-factorized* distribution $q(\mathbf{z}|\mathbf{x}) = \prod_i q(z_i|\mathbf{x}; \mu_i)$ paramterized by its means μ_i.

To derive the update rule for the variational parameter μ_i:

We neglect the normalization factor Z_i for now and normalize the q_i at the end.

$$
\begin{aligned}
\log \tilde{p}(\mathbf{z}) &= -E(\mathbf{z}) && \text{limit to } z_i\text{'s Markov blanket:} \\
&= z_i \sum_{j \in \text{nbr}(i)} W_{ij} z_j + \ell_i(z_i) + \text{const} && \text{drop terms without } z_i \\
\log q_i(z_i) &= \mathbb{E}_{-q_i}[\log \tilde{p}(\mathbf{z})] - Z_i && \text{Mean-Field update rule} \\
q_i(z_i) &\propto \exp\left(z_i \sum_{j \in \text{nbr}(i)} W_{ij} \mu_j + \ell_i(z_i) \right) && \Rightarrow \text{ Ising update rule for } q_i
\end{aligned}
$$

Some shorthand notation for the next step:

$$
\begin{aligned}
m_i &= \sum_{j \in \text{nbr}(i)} W_{ij} \mu_j && \text{mean field influence on } z_i \\
\ell_i^+ &= \ell_i(+1) \\
\ell_i^- &= \ell_i(-1)
\end{aligned}
$$

3.3. Ising Update Rule For μ_i

Now normalize to obtain the approxiate posterior probabilities:

$$
\begin{aligned}
q_i(z_i = +1) &= \frac{e^{m_i + \ell_i^+}}{e^{m_i + \ell_i^+} + e^{-m_i + \ell_i^-}} = \frac{1}{1 + e^{-2m_i + \ell_i^- - \ell_i^+}} = \sigma(2a_i) \\
a_i &= m_i + \frac{1}{2}(\ell_i^+ - \ell_i^-) \\
q_i(z_i = -1) &= \sigma(-2a_i) \\
\mu_i &= \mathbb{E}_{q_i}[z_i] = q_i(z_i = +1) \cdot (+1) + q_i(z_i = -1) \cdot (-1) \\
&= \frac{1}{1 + e^{-2a_i}} - \frac{1}{1 + e^{2a_i}} = \tanh a_i
\end{aligned}
$$

Exercise: Derive the solution for $q_i(z_i = -1)$!

In practice, it is often better to use ***damped updates*** using a damping factor $0 < \lambda \leq 1$:

$$\mu_i \leftarrow (1 - \lambda)\mu_i + \lambda \tanh a_i$$

This update rule is iterated to convergence.

3.4. Notes

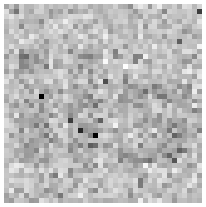
- Recall that the μ_i are the means of the distributions governing the states z_i. Thus, these are the values that are iteratively updated by this algorithm.
- Initialize the algorithm by setting $\mu_i = x_i$ for all i.
- The m_i must (and can easily) be recomputed at every iteration.
- The observation log likelihood functions $\ell_i(z_i)$ must be specified as model parameters.
- The values $\ell_i^+ = p(x_i|+1)$ and $\ell_i^- = p(x_i|-1)$ (and thus the term $\frac{1}{2}(\ell^+ - \ell^-)$) can be precomputed because there are only these two possible states, and the observations x_i are fixed.

  Note

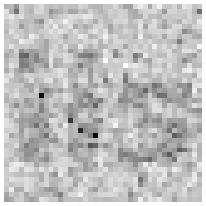
  After lots of math we arrived at a very simple, easily-implemented, and effective algorithm!

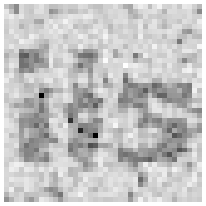
3.5. Example: Image With Gaussian Noise

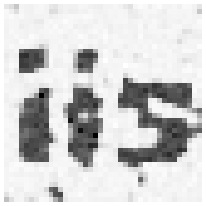
Noisy



Clean



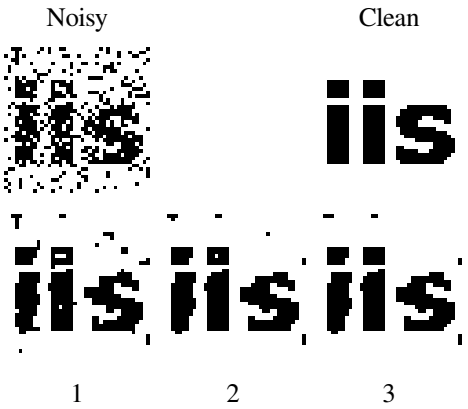




1 2 5 17

3.6. Example: Image With Flipped Pixels

3.7. Example: ICM



4. Variational Autoencoders

Reading:

- Tutorial on Variational Autoencoders [Doersch 2021]

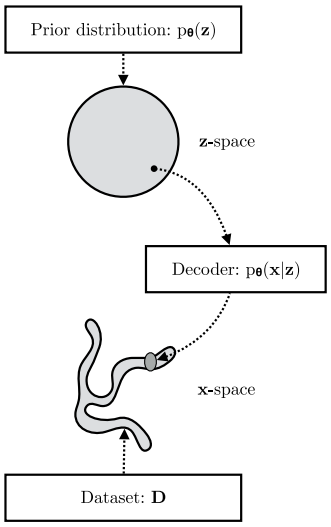
  Provides a relatively gentle introduction and intuitive motivation. Start here; read it through Section 2.3.

- An Introduction to Variational Autoencoders [Kingma and Welling 2019]

  Gives more background on methods that VAE build upon, and gives more details. Read it through Section 2.5 (Intro), plus Section 2.6.

These course notes largely follow the notation of Kingma and Welling (2019), with some adaptations for continuity with the preceding sections. They refer back to the general ELBO of variational inference, cutting out a sizable portion of the math of both tutorials.

4.1. Objective



[Figure adapted from Kingma and Welling 2019]

Objectives:

1. Learn a generative model of $p(\mathbf{X})$ that resembles $\boldsymbol{D}$ as much as possible.

   This allows us to generate new instances $\mathbf{x}$ that are plausible w.r.t. $\boldsymbol{D}$.

2. Learn a representation $p(\mathbf{Z}|\mathbf{X})$ for $\boldsymbol{D}$.

   This allows us to understand the internal structure of $\boldsymbol{D}$.

Example Representation:

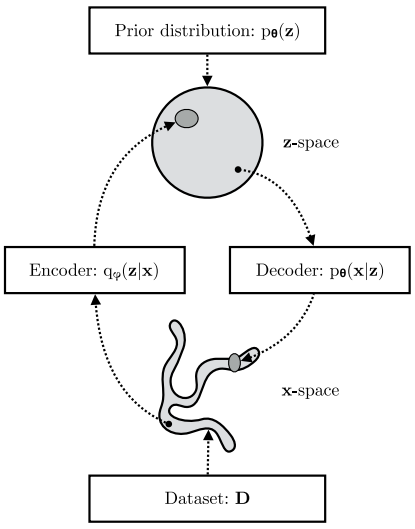
$$
\begin{aligned}
p(\mathbf{z}) &= \mathcal{N}(\mathbf{z}; 0, I) \\
p_\theta(\mathbf{x}|\mathbf{z}) &= \mathcal{N}(\mathbf{x}; \mathrm{dec}_\theta(\mathbf{z}), I) \\
p_\theta(\mathbf{X}) &= \int p_\theta(\mathbf{X}|\mathbf{z}) p(\mathbf{z}) \, d\mathbf{z}
\end{aligned}
$$

The two key distributions are both very simple; all the power lies in the ***decoder neural network*** $\mathrm{dec}_\theta(\mathbf{z})$ parameterized by $\boldsymbol{\theta}$.

Note

The VAE as presented here can be extended to the Conditional VAE. While a VAE can be used to generate any data resembling $\boldsymbol{D}$ (such as an image), a CVAE can be used to generate data resembling $\boldsymbol{D}$ that additionally obeys certain conditions (e.g. for image inpainting).

4.2. Variational Approach



[Figure from Kingma and Welling 2019]

How to learn $p_\theta(\mathbf{X}|\mathbf{Z})$? To obtain paired examples $(\mathbf{z}, \mathbf{x})$ we need $p(\mathbf{z}|\mathbf{x})$ to map D onto $p(\mathbf{Z})$. Approximate it by $q_\phi(\mathbf{z}|\mathbf{x})$, maximizing the ELBO

$$\begin{aligned} & L(\mathbf{x}) \\ = \; & \log p(\mathbf{x}) - \mathrm{KL}(q_\phi(\mathbf{Z}|\mathbf{x}) \| p_\theta(\mathbf{Z}|\mathbf{x})) \\ = \; & \mathbb{E}_{q_\phi(\mathbf{Z}|\mathbf{x})}[\log p_\theta(\mathbf{z}, \mathbf{x}) - \log q_\phi(\mathbf{z}|\mathbf{x})] \end{aligned}$$

$$\begin{aligned} p_\theta(\mathbf{z}, \mathbf{x}) &= p_\theta(\mathbf{x}|\mathbf{z}) p(\mathbf{z}) \\ q_\phi(\mathbf{z}|\mathbf{x}) &= \mathcal{N}(\mathbf{z}; \mathrm{enc}_\phi(\mathbf{x})) \end{aligned}$$

Again, the probability distribution is very simple; all the power lies in the *encoder neural network* $\mathrm{enc}_\phi(\mathbf{x})$ that transforms a data point $\mathbf{x}$ into a mean and (typically) diagonal variance.

In classical variational inference only the encoder is learned; there is no explicit decoder, and $p(\mathbf{X}|\mathbf{Z})$ is typically given as part of the model.

Exercise: Show that
$\log p(\mathbf{x}) - \mathrm{KL}(q_\phi(\mathbf{Z}|\mathbf{x}) \| p_\theta(\mathbf{Z}|\mathbf{x})) = \mathbb{E}_{q_\phi(\mathbf{Z}|\mathbf{x})}[\log p_\theta(\mathbf{z}, \mathbf{x}) - \log q_\phi(\mathbf{z}|\mathbf{x})]$.

4.3. Classical vs. Variational Autoencoders

Classical	**Variational**
Single neural network with bottleneck layer $\mathbf{Z}$	Marriage of graphical models with neural networks
Explicit objectives (dimensionality reduction, sparsity, …) and corresponding tunable parameters	–
Unknown distribution over $\mathbf{Z}$	Simple $p(\mathbf{Z})$ known by design
Most $\mathbf{z}$ will *not* generate useful output $\mathbf{x}$	All $\mathbf{z}$ will generate output $\mathbf{x}$ of utility $\sim p(\mathbf{z})$

4.4. Variational Inference vs. Variational Autoencoders

Variational Inference	**Variational Autoencoder**
ELBO formulation $\log p(\mathbf{x}) - \mathrm{KL}(q_\phi(\mathbf{Z}\|\mathbf{x}) \| p(\mathbf{Z}\|\mathbf{x}))$	ELBO formulation $\mathbb{E}_{q_\phi(\mathbf{Z}\|\mathbf{x})}[\log p_\theta(\mathbf{z}, \mathbf{x}) - \log q_\phi(\mathbf{z}\|\mathbf{x})]$
Approximates $p(\mathbf{Z}\|\mathbf{x})$ by $q_\phi(\mathbf{Z}\|\mathbf{x})$; $p(\mathbf{X}\|\mathbf{Z})$ is known	Approximates $p(\mathbf{Z}\|\mathbf{x})$ by $q_\phi(\mathbf{Z}\|\mathbf{x})$ while approximating $p(\mathbf{X})$ by $\mathbb{E}_{q_\phi(\mathbf{z}\|\mathbf{x})}\left[\frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}\|\mathbf{x})}\right]$
$q_\phi(\mathbf{Z}\|\mathbf{x})$ is a parametric probability distribution	$q_\phi(\mathbf{Z}\|\mathbf{x})$ and $p_\theta(\mathbf{Z}, \mathbf{x})$ are parametric probability distributions of random variables transformed by neural networks (*encoder* and *decoder*, respectively)

Note

The ELBO is identical in both cases; just the formulation is different.

4.5. Gradients for Training a VAE

Objective: Ascend the ELBO gradient $\nabla_{\theta,\phi} L(\mathbf{x})$

- Unbiased stochastic gradient estimator over θ:

$$\begin{aligned}
\nabla_\theta L(\mathbf{x}) &= \nabla_\theta \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{z},\mathbf{x}) - \log q_\phi(\mathbf{z}|\mathbf{x})] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\nabla_\theta(\log p_\theta(\mathbf{z},\mathbf{x}) - \log q_\phi(\mathbf{z}|\mathbf{x}))] \\
&\simeq \nabla_\theta(\log p_\theta(\mathbf{z},\mathbf{x}) - \log q_\phi(\mathbf{z}|\mathbf{x})) \qquad \mathbf{z} \sim q_\phi(\mathbf{Z}|\mathbf{x}) \\
&= \nabla_\theta \log p_\theta(\mathbf{z},\mathbf{x})
\end{aligned}$$

Here, $\simeq$ means that the right-hand side is an unbiased stochastic estimator of the left-hand side using samples $\mathbf{z} \sim q_\phi(\mathbf{Z}|\mathbf{x})$.

- An unbiased gradient estimator over the variational parameters ϕ is harder to obtain because the expectation is taken over $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$ which is a function of ϕ:

$$\begin{aligned}
\nabla_\phi L(\mathbf{x}) &= \nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{z},\mathbf{x}) - \log q_\phi(\mathbf{z}|\mathbf{x})] \\
&\neq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\nabla_\phi(\log p_\theta(\mathbf{z},\mathbf{x}) - \log q_\phi(\mathbf{z}|\mathbf{x}))]
\end{aligned}$$

We can solve this conundrum be reparameterizing $q_\phi(\mathbf{z}|\mathbf{x})$ to disentangle the randomness and the gradient.

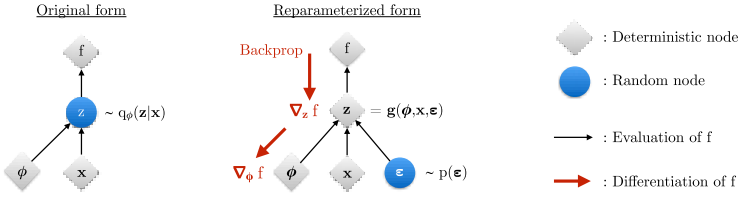
4.6. Reparameterizing $q_\phi(\mathbf{z}|\mathbf{x})$

Objective: Split $q_\phi(\mathbf{z}|\mathbf{x})$ into

- a deterministic function $\mathbf{g}$ through which we can backpropagate the gradient in ϕ, and
- a random variable ## not depending on ϕ over which we can take the expectation.

In the following, $f(\mathbf{z})$ is shorthand for $\log p_\theta(\mathbf{z},\mathbf{x}) - \log q_\phi(\mathbf{z}|\mathbf{x})$, the argument of the expectation in question.

$$\begin{aligned}
\mathbf{z} &= \mathbf{g}(\epsilon,\phi,\mathbf{x}) && \text{Reparameterization} \\
\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[f(\mathbf{z})] &= \mathbb{E}_{p(\epsilon)}[f(\mathbf{z})] \\
\nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[f(\mathbf{z})] &= \nabla_\phi \mathbb{E}_{p(\epsilon)}[f(\mathbf{z})] = \mathbb{E}_{p(\epsilon)}[\nabla_\phi f(\mathbf{z})] && \text{Now this holds!} \\
&\simeq \nabla_\phi f(\mathbf{z}) && \epsilon \sim p(\epsilon)
\end{aligned}$$



[Figure adapted from Kingma and Welling 2019]

4.7. Computing $\log q_\phi(\mathbf{z}|\mathbf{x})$ as a function of ϵ

$$\log q_\phi(\mathbf{z}|\mathbf{x}) = \log p(\epsilon) - \log\left|\det \frac{\partial \mathbf{z}}{\partial \epsilon}\right| \quad \text{with } \mathbf{z}(\epsilon) = \mathbf{g}(\epsilon, \phi, \mathbf{x})$$

$$\frac{\partial \mathbf{z}}{\partial \epsilon} = \begin{bmatrix} \frac{\partial z_1}{\partial \epsilon_1} & \cdots & \frac{\partial z_1}{\partial \epsilon_k} \\ \vdots & \ddots & \vdots \\ \frac{\partial z_k}{\partial \epsilon_1} & \cdots & \frac{\partial z_k}{\partial \epsilon_k} \end{bmatrix} \quad \text{Jacobian of } \mathbf{z}(\epsilon)$$

Why is $q_\phi(\mathbf{g}(\epsilon, \phi, \mathbf{x})|\mathbf{x}) \neq p(\epsilon)$; why do we have to divide by the determinant of the Jacobian of $\mathbf{g}(\epsilon, \cdot, \cdot)$?

Consider a function $f_x(x)$ under a change of variables $x = g(y)$, which defines a new function $f_y(y) = f_x(g(y))$. Thus the infinitesimal interval $[x, x + \mathrm{d}x]$ corresponds to $[y, y + \mathrm{d}y]$ where $x + \mathrm{d}x = g(y + \mathrm{d}y)$ and generally $\mathrm{d}x \neq \mathrm{d}y$, which entails that the infinitesimal areas $f_x(x)\mathrm{d}x \neq f_y(y)\mathrm{d}y$.

For example, if $g(y) = cy$, then $g(y + \mathrm{d}y) = g(y) + g(\mathrm{d}y)$, which entails $\mathrm{d}x = c\mathrm{d}y$.

Now consider a PDF $p_x(x)$ under a change of variables $x = g(y)$ where the density w.r.t. y is $p_y(y)$. Here $[x, x + \mathrm{d}x]$ and $[y, y + \mathrm{d}y]$ contain the same samples and thus the *same probability mass*, which entails that $p_x(x)\mathrm{d}x = p_y(y)\mathrm{d}y$, i.e., $p_y(y) = p_x(x)\left|\frac{\mathrm{d}x}{\mathrm{d}y}\right|$ and $p_x(x) = p_y(y)\frac{1}{\left|\frac{\mathrm{d}x}{\mathrm{d}y}\right|}$.

The last equation is the one-dimensional case of our above situation with $\mathbf{z} \mathrel{\widehat{=}} x$ and $\epsilon \mathrel{\widehat{=}} y$.

[Bishop 2006 Sec. 1.2.1; solution[3] to Ex. 1.4]

4.8. Training a VAE by Stochastic Gradient Ascent

Maximize the ELBO $L(D) = \sum_{\mathbf{x} \in D} L(\mathbf{x})$ jointly over all parameters ϕ and θ by ***stochastic gradient ascent*** training of the encoder and decoder networks simultaneously:

$$\begin{aligned} \epsilon &\sim \mathcal{N}(0, I) \\ \mathbf{z} &= \mathbf{g}(\epsilon, \phi, \mathbf{x}) \\ L(\mathbf{x}) &\simeq \log p_\theta(\mathbf{z}, \mathbf{x}) - \log q_\phi(\mathbf{z}|\mathbf{x}) \end{aligned}$$

The gradient $\nabla_{\theta,\phi} L(\mathbf{x})$ is easily computed and used for minibatch stochastic gradient ascent using modern neural-network toolkits.

[3] https://www.microsoft.com/en-us/research/wp-content/uploads/2016/05/prml-web-sol-2009-09-08.pdf

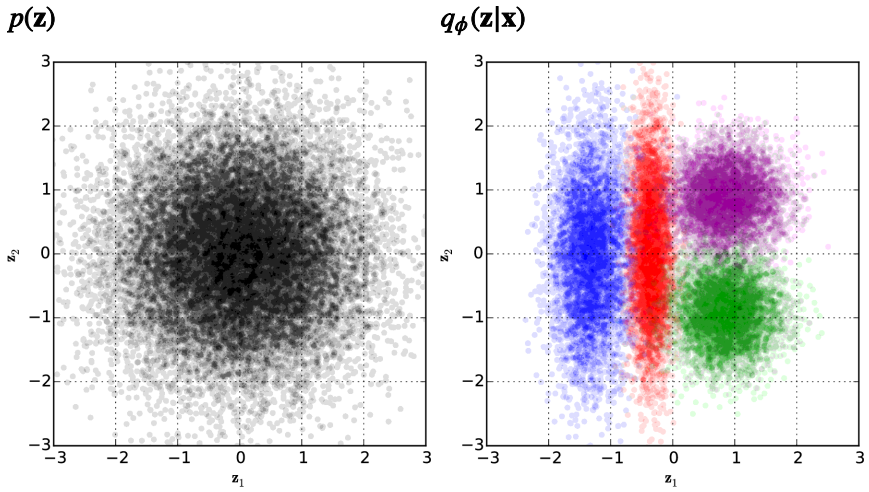
4.9. Example: $q_\phi(\mathbf{z}|\mathbf{x})$ is a Factorized Gaussian

Cf. Variational Approach (Section 4.2)

$$
\begin{aligned}
q_\phi(\mathbf{z}|\mathbf{x}) &= \prod_i q_\phi(z_i|\mathbf{x}) = \prod_i \mathcal{N}(z_i; \mu_i, \sigma_i^2) \\
(\boldsymbol{\mu}, \log \boldsymbol{\sigma}) &= \text{enc}_\phi(\mathbf{x}) \\
\boldsymbol{\epsilon} &\sim \mathcal{N}(0, I) \\
\mathbf{z} &= \mathbf{g}(\boldsymbol{\epsilon}, \phi, \mathbf{x}) = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon} \\
\frac{\partial \mathbf{z}}{\partial \boldsymbol{\epsilon}} &= \text{diag}(\boldsymbol{\sigma}) \\
\log \left| \det \frac{\partial \mathbf{z}}{\partial \boldsymbol{\epsilon}} \right| &= \sum_i \log \sigma_i \\
\log q_\phi(\mathbf{z}|\mathbf{x}) &= \log p(\boldsymbol{\epsilon}) - \log \left| \det \frac{\partial \mathbf{z}}{\partial \boldsymbol{\epsilon}} \right| \\
&= \sum_i \left(\log \mathcal{N}(\epsilon_i; 0, 1) - \log \sigma_i \right)
\end{aligned}
$$

The same end result holds if $q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is a full-covariance Gaussian.

4.10. Toy Example



[Figure adapted from Kingma and Welling 2019]

Kingma and Welling (2019) do not give much detail about this figure, but the filenames suggest it is an XOR problem, so I suspect that the result for $q_\phi(\mathbf{z}|\mathbf{x})$ is a poor local optimum.

4.11. Using a VAE

- To generate an instance $\mathbf{x}$:

  See the Example Representation:

  1. Draw $\mathbf{z} \sim p(\mathbf{z})$.
  2. Draw $\mathbf{x} \sim p(\mathbf{x}|\mathbf{z})$.

- To estimate the marginal likelihood of a given instance $\mathbf{x}$:

$$p(\mathbf{x}) = \int \frac{p(\mathbf{x},\mathbf{z})}{p(\mathbf{z}|\mathbf{x})}\,\mathrm{d}\mathbf{z}$$

$$p_\theta(\mathbf{x}) \simeq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right] \approx \frac{1}{K}\sum_{k=1}^{K}\frac{p_\theta(\mathbf{x},\mathbf{z}_k)}{q_\phi(\mathbf{z}_k|\mathbf{x})},\ \mathbf{z}_k \sim q_\phi(\mathbf{z}|\mathbf{x})$$

  This procedure follows the principle of so-called *importance sampling*.

  > ***Note***
  >
  > With $K = 1$ this equals our ELBO estimator.
  >
  > We could also use this with $K > 1$ for our ELBO estimator, improving its approximation. However, importance sampling scales poorly to high-dimensional spaces, diminishing its benefits.

5. References

5.1. References

C. Bishop, *Pattern Recognition and Machine Learning*[1], Springer 2006.

C. Doersch, *Tutorial on Variational Autoencoders*[2], 2021.

D. Kingma, M. Welling, "An Introduction to Variational Autoencoders"[3]. *Foundations and Trends® in Machine Learning* 12, pp. 307–392, 2019.

K. Murphy, *Machine Learning: a Probabilistic Perspective*[4], MIT Press 2012.

[1] https://www.microsoft.com/en-us/research/people/cmbishop/#!prml-book/
[2] https://arxiv.org/abs/1606.05908
[3] https://arxiv.org/abs/1906.02691
[4] https://probml.github.io/pml-book/book0.html