Variational Autoencoders (VAEs)

STATS 305C: Applied Statistics

Scott Linderman

May 2, 2022

PCA as a linear autoencoder

Recall from Lecture 5 that PCA could be motivated as a linear autoencoder trained to minimize reconstruction error subject to having orthogonal weights.

Deep autoencoders

Why restrict ourselves to **linear** autoencoders? The neural network community has used **deep autoencoders** (a.k.a. autoassociative networks) for nonlinear dimensionality reduction [LeCun, 1987, Bourlard and Kamp, 1988, Hinton and Zemel, 1993, Hinton and Salakhutdinov, 2006, Vincent et al., 2010]. See also, Goodfellow et al. [2016, Ch. 14].

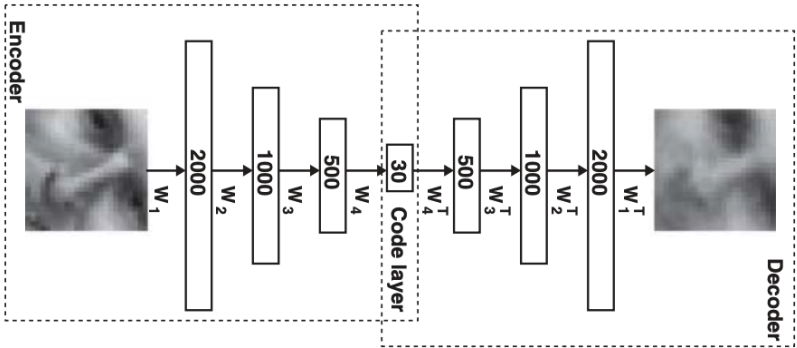
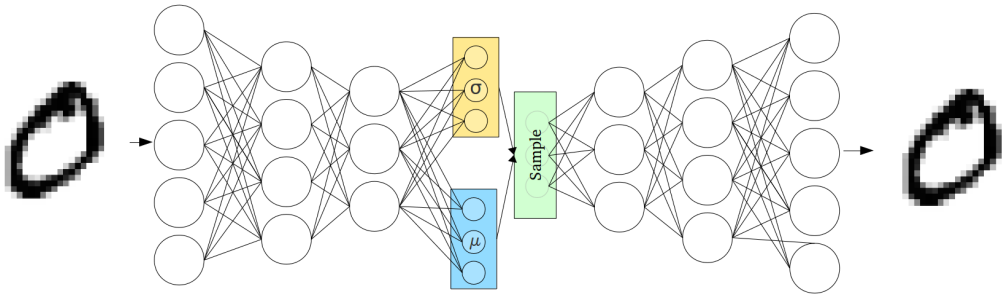


Figure: Figure from Hinton and Salakhutdinov [2006]

Variational autoencoders as deep, stochastic, regularized autoencoders

Kingma and Welling [2014] and Rezende et al. [2014] concurrently developed what we now call **variational autoencoders**. The idea is to treat the hidden codes as random variables. As we will see, VAEs can be viewed as **deep generative models** combined with **amortized variational inference**.



$$\mathscr{L}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathbb{E}_{q(\boldsymbol{z}_n; \boldsymbol{x}_n, \boldsymbol{\phi})}[\log p(\boldsymbol{x}_n \mid \boldsymbol{z}_n; \boldsymbol{\theta})] - D_{\mathrm{KL}}\left(q(\boldsymbol{z}_n; \boldsymbol{x}_n, \boldsymbol{\phi}) \,\|\, p(\boldsymbol{z}_n)\right) \tag{1}$$

Outline

- The generative model
- Learning via variational expectation maximization
- Stochastic gradient ascent
- Unbiased gradient estimators
- Amortized inference

The generative model

VAEs start with a "deep" but conceptually simple generative model,

$$\boldsymbol{z}_n \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}) \tag{2}$$

$$\boldsymbol{x}_n \sim \mathcal{N}(g(\boldsymbol{z}_n; \boldsymbol{\theta}), \boldsymbol{I}) \tag{3}$$

where $g : \mathbb{R}^H \to \mathbb{R}^D$ is a nonlinear mapping from $\boldsymbol{z}_n \in \mathbb{R}^H$ to $\mathbb{E}[\boldsymbol{x}_n] \in \mathbb{R}^D$, parameterized by $\boldsymbol{\theta}$.

We will assume g is a simple **feedforward neural network** (a.k.a. multilayer perceptron) of the form,

$$g(\boldsymbol{z}; \boldsymbol{\theta}) = g_L(g_{L-1}(\cdots g_1(\boldsymbol{z}) \cdots)) \tag{4}$$

where each **layer** is a cascade of a linear mapping followed by an element-wise nonlinearity (except for the last layer, perhaps). For example,

$$g_\ell(\boldsymbol{u}_\ell) = \mathrm{relu}(\boldsymbol{W}_\ell \boldsymbol{u}_\ell + \boldsymbol{b}_\ell); \qquad \mathrm{relu}(a) = \max(0, a). \tag{5}$$

The generative parameters consist of the weights and biases, $\boldsymbol{\theta} = \{\boldsymbol{W}_\ell, \boldsymbol{b}_\ell\}_{\ell=1}^L$.

Two goals

The **learning goal** is to find the parameters that **maximize the marginal probability of the data**,

$$\boldsymbol{\theta}^\star = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{X}; \boldsymbol{\theta}) \tag{6}$$

$$= \arg\max_{\boldsymbol{\theta}} \prod_{n=1}^{N} \int p(\boldsymbol{x}_n \mid \boldsymbol{z}_n; \boldsymbol{\theta})\, p(\boldsymbol{z}_n; \boldsymbol{\theta})\, \mathrm{d}\boldsymbol{z}_n \tag{7}$$

The **inference goal** is to find the **posterior distribution of latent variables**,

$$p(\boldsymbol{z}_n \mid \boldsymbol{x}_n; \boldsymbol{\theta}) = \frac{p(\boldsymbol{x}_n \mid \boldsymbol{z}_n; \boldsymbol{\theta})\, p(\boldsymbol{z}_n; \boldsymbol{\theta})}{\int p(\boldsymbol{x}_n \mid \boldsymbol{z}'_n; \boldsymbol{\theta})\, p(\boldsymbol{z}'_n; \boldsymbol{\theta})\, \mathrm{d}\boldsymbol{z}'_n} \tag{8}$$

Both goals require an integral over $\boldsymbol{z}_n$, but that is intractable for deep generative models.

The evidence lower bound (ELBO)

Idea: Use the ELBO to get a bound on the marginal probability and maximize that instead.

$$\log p(\boldsymbol{X}; \boldsymbol{\theta}) = \sum_{n=1}^{N} \log p(\boldsymbol{x}_n; \boldsymbol{\theta}) \tag{9}$$

$$\geq \sum_{n=1}^{N} \log p(\boldsymbol{x}_n; \boldsymbol{\theta}) - D_{\mathrm{KL}}\left(q_n(\boldsymbol{z}_n) \,||\, p(\boldsymbol{z}_n \mid \boldsymbol{x}_n; \boldsymbol{\theta})\right) \tag{10}$$

$$= \sum_{n=1}^{N} \underbrace{\mathbb{E}_{q_n(\boldsymbol{z}_n)}\left[\log p(\boldsymbol{x}_n, \boldsymbol{z}_n; \boldsymbol{\theta}) - \log q_n(\boldsymbol{z}_n)\right]}_{\text{“local ELBO”}} \tag{11}$$

$$\triangleq \sum_{n=1}^{N} \mathscr{L}_n(q_n, \boldsymbol{\theta}) \tag{12}$$

$$= \mathscr{L}(q, \boldsymbol{\theta}) \tag{13}$$

where $q = \{q_n\}_{n=1}^{N}$. Here, I've written the ELBO as a sum of "local ELBOs" $\mathscr{L}_n(q_n, \boldsymbol{\theta})$.

Optimal variational posterior

The ELBO is still maximized (and the bound is tight) when each q_n is equal to the true posterior,

$$q_n(\boldsymbol{z}_n) = p(\boldsymbol{z}_n \mid \boldsymbol{x}_n, \boldsymbol{\theta}). \tag{14}$$

Question: The deep generative model above has a Gaussian prior on $\boldsymbol{z}_n$ and a Gaussian likelihood for $\boldsymbol{x}_n$ given $\boldsymbol{z}_n$. Why isn't the posterior Gaussian?

Fixed form variational inference

Nevertheless, we can still constrain q_n to be Gaussian and seek the best Gaussian approximation to the posterior. This is sometimes called **fixed-form variational inference**.

For example, let,

$$\mathcal{Q} = \left\{ q : q(\boldsymbol{z}) = \mathcal{N}\left(\boldsymbol{z} \mid \boldsymbol{\mu}, \operatorname{diag}(\boldsymbol{\sigma}^2)\right) \text{ for some } \boldsymbol{\mu} \in \mathbb{R}^H, \boldsymbol{\sigma}^2 \in \mathbb{R}_+^H \right\} \tag{15}$$

Then, for fixed parameters $\boldsymbol{\theta}$, the best q_n in this **variational family** is,

$$q_n^\star = \arg\max_{q_n \in \mathcal{Q}} \mathcal{L}_n(q_n, \boldsymbol{\theta}) \tag{16}$$

$$= \arg\min_{q_n \in \mathcal{Q}} D_{\mathrm{KL}}\left(q_n(\boldsymbol{z}_n) \,\|\, p(\boldsymbol{z}_n \mid \boldsymbol{x}_n; \boldsymbol{\theta})\right). \tag{17}$$

Variational expectation-maximization (vEM)

Now we can introduce a new algorithm: **variational expectation maximization**.

Repeat until either the ELBO or the parameters converges:

1. **M-step:** Set $\boldsymbol{\theta} \leftarrow \arg\max_{\boldsymbol{\theta}} \mathscr{L}(q, \boldsymbol{\theta})$
2. **E-step:** For $n = 1, \ldots, N$
   - Set $q_n \leftarrow \arg\max_{q_n \in \mathscr{Q}} \mathscr{L}_n(q_n, \boldsymbol{\theta})$
3. Compute the ELBO $\mathscr{L}(q, \boldsymbol{\theta})$.

Unfortunately, none of these steps will have closed form solutions, so we'll have to use approximations.

Generic M-step with gradient ascent

For exponential family mixture models and simple factor analysis, the M-steps had closed form.

For deep generative models, however, we need a more general approach.

If the parameters are unconstrained and the ELBO is differentiable wrt $\boldsymbol{\theta}$, we can use **gradient ascent**.

Repeat:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \nabla_{\boldsymbol{\theta}} \mathscr{L}(q, \boldsymbol{\theta}) = \boldsymbol{\theta} + \alpha \sum_{n=1}^{N} \mathbb{E}_{q(\boldsymbol{z}_n)} \left[\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{x}_n, \boldsymbol{z}_n; \boldsymbol{\theta}) \right] \tag{18}$$

with **step size** α. Typically, you decrease the step size over iterations so that $\alpha_1 \geq \alpha_2 \geq \ldots$

Generic M-step with *stochastic* gradient ascent (SGD)

- For more complex models, the expected gradient may be intractable as well. In those cases, we can use Monte Carlo to obtain an **unbiased estimate** of the gradient,

$$\nabla_{\boldsymbol{\theta}} \mathscr{L}(q, \boldsymbol{\theta}) \approx \widehat{\nabla}_{\boldsymbol{\theta}} \mathscr{L}(q, \boldsymbol{\theta}) = \sum_{n=1}^{N} \left[\frac{1}{M} \sum_{m=1}^{M} \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{x}_n, \boldsymbol{z}_n^{(m)}; \boldsymbol{\theta}) \right] \tag{19}$$

  where $\boldsymbol{z}_n^{(m)} \overset{\text{iid}}{\sim} q(\boldsymbol{z}_n)$ for $m = 1, \ldots, M$.

- While we're at it, we can use Monte Carlo to approximate the sum over data points as well,

$$\widehat{\nabla}_{\boldsymbol{\theta}} \mathscr{L}(q, \boldsymbol{\theta}) = N \mathbb{E}_{n \sim \text{Unif}(1,\ldots,N)} \left[\frac{1}{M} \sum_{m=1}^{M} \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{x}_n, \boldsymbol{z}_n^{(m)}; \boldsymbol{\theta}) \right] \tag{20}$$

$$\approx \frac{N}{M} \sum_{m=1}^{M} \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{x}_n, \boldsymbol{z}_n^{(m)}; \boldsymbol{\theta}) \tag{21}$$

  where $n \sim \text{Unif}(1, \ldots, N)$ and $\boldsymbol{z}_n^{(m)} \overset{\text{iid}}{\sim} q(\boldsymbol{z}_n)$ for $m = 1, \ldots, M$.

Generic M-step with *stochastic* gradient ascent (SGD)

- For more complex models, the expected gradient may be intractable as well. In those cases, we can use Monte Carlo to obtain an **unbiased estimate** of the gradient,

$$\nabla_{\boldsymbol{\theta}} \mathscr{L}(q, \boldsymbol{\theta}) \approx \widehat{\nabla}_{\boldsymbol{\theta}} \mathscr{L}(q, \boldsymbol{\theta}) = \sum_{n=1}^{N} \left[\frac{1}{M} \sum_{m=1}^{M} \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{x}_n, \boldsymbol{z}_n^{(m)}; \boldsymbol{\theta}) \right] \tag{19}$$

  where $\boldsymbol{z}_n^{(m)} \overset{\text{iid}}{\sim} q(\boldsymbol{z}_n)$ for $m = 1, \ldots, M$.

- While we're at it, we can use Monte Carlo to approximate the sum over data points as well,

$$\widehat{\nabla}_{\boldsymbol{\theta}} \mathscr{L}(q, \boldsymbol{\theta}) = N \mathbb{E}_{n \sim \text{Unif}(1,\ldots,N)} \left[\frac{1}{M} \sum_{m=1}^{M} \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{x}_n, \boldsymbol{z}_n^{(m)}; \boldsymbol{\theta}) \right] \tag{20}$$

$$\approx \frac{N}{M} \sum_{m=1}^{M} \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{x}_n, \boldsymbol{z}_n^{(m)}; \boldsymbol{\theta}) \tag{21}$$

  where $n \sim \text{Unif}(1, \ldots, N)$ and $\boldsymbol{z}_n^{(m)} \overset{\text{iid}}{\sim} q(\boldsymbol{z}_n)$ for $m = 1, \ldots, M$.

SGD convergence and extensions

When does SGD work? This is a well studied problem in stochastic optimization [Bottou et al., 1998, Robbins and Siegmund, 1971].

Under relatively mild conditions, SGD converges to a **local minimum** if the step sizes obey the **Robbins-Monro condtions**,

$$\sum_{i=0}^{\infty} \alpha_i = \infty \qquad \text{and} \qquad \sum_{i=0}^{\infty} \alpha_i^2 < \infty \tag{22}$$

There have been dozens of extensions to basic SGD including,

- SGD with momentum
- AdaGrad [Duchi et al., 2011]
- RMSProp
- Adam [Kingma and Ba, 2014]

Note: we still need to compute the gradient $\nabla_{\theta} \log p(\boldsymbol{x}_n, \boldsymbol{z}_n^{(m)}; \boldsymbol{\theta})$. We'll come back to this

Variational expectation-maximization (vEM)

Now we can introduce a new algorithm: **variational expectation maximization**.

Repeat until either the ELBO or the parameters converge:

1. **M-step:** Set $\boldsymbol{\theta} \leftarrow \arg\max_{\boldsymbol{\theta}} \mathscr{L}(q, \boldsymbol{\theta})$
2. **E-step:** For $n = 1, \ldots, N$
   - Set $q_n \leftarrow \arg\max_{q_n \in \mathscr{Q}} \mathscr{L}_n(q_n, \boldsymbol{\theta})$
3. Compute the ELBO $\mathscr{L}(q, \boldsymbol{\theta})$.

Unfortunately, none of these steps will have closed form solutions, so we'll have to use approximations.

The variational E-step

As above, assume $\mathcal{Q}$ is the family of Gaussian distributions with diagonal covariance.

Then $q_n(\boldsymbol{z}_n) = \mathcal{N}(\boldsymbol{z}_n \mid \boldsymbol{\mu}_n, \mathrm{diag}(\boldsymbol{\sigma}_n^2))$, with **variational parameters** $\boldsymbol{\mu}_n \in \mathbb{R}^H$ and $\boldsymbol{\sigma}_n^2 \in \mathbb{R}_+^H$.

We know the optimal q_n is,

$$q_n(\boldsymbol{z}_n) = \arg\max_{q \in \mathcal{Q}} \mathcal{L}_n(q_n, \boldsymbol{\theta}) \tag{23}$$

$$= \arg\max_{q \in \mathcal{Q}} \mathbb{E}_{q_n(\boldsymbol{z}_n)} \left[\log p(\boldsymbol{x}_n, \boldsymbol{z}_n; \boldsymbol{\theta}) - \log q_n(\boldsymbol{z}_n)\right] \tag{24}$$

but unfortunately there is no closed form solution.

Stochastic gradient ascent on the local ELBO

- **Idea:** Write the objective in terms of the unconstrained variational parameters

$$\boldsymbol{\lambda}_n \triangleq (\boldsymbol{\mu}_n, \log \boldsymbol{\sigma}_n^2) \in \mathbb{R}^{2H} \tag{25}$$

  and then perform stochastic gradient ascent.

- **Note:** Now we will write $q_n(\boldsymbol{z}_n; \boldsymbol{\lambda}_n)$ and $\mathscr{L}_n(\boldsymbol{\lambda}_n, \boldsymbol{\theta}_n)$ to emphasize that the variational posterior and hence the local ELBO are both parameterized by $\boldsymbol{\lambda}_n$.

- To perform SGD, we need an unbiased estimate of the gradient of the local ELBO,

$$\nabla_{\boldsymbol{\lambda}_n} \mathscr{L}_n(\boldsymbol{\lambda}_n, \boldsymbol{\theta}) = \nabla_{\boldsymbol{\lambda}_n} \mathbb{E}_{q(\boldsymbol{z}_n; \boldsymbol{\lambda}_n)} \left[\log p(\boldsymbol{x}_n, \boldsymbol{z}_n; \boldsymbol{\theta}) - \log q(\boldsymbol{z}_n)\right] \tag{26}$$

$$\neq \mathbb{E}_{q(\boldsymbol{z}_n; \boldsymbol{\lambda}_n)} \left[\nabla_{\boldsymbol{\lambda}_n} \left(\log p(\boldsymbol{x}_n, \boldsymbol{z}_n; \boldsymbol{\theta}) - \log q(\boldsymbol{z}_n)\right)\right]. \tag{27}$$

- **Question:** Why can't we simply bring the gradient inside the expectation like we did for the M-step?

Stochastic gradient ascent on the local ELBO

- **Idea:** Write the objective in terms of the unconstrained variational parameters

$$\boldsymbol{\lambda}_n \triangleq (\boldsymbol{\mu}_n, \log \boldsymbol{\sigma}_n^2) \in \mathbb{R}^{2H} \tag{25}$$

  and then perform stochastic gradient ascent.

- **Note:** Now we will write $q_n(\boldsymbol{z}_n; \boldsymbol{\lambda}_n)$ and $\mathscr{L}_n(\boldsymbol{\lambda}_n, \boldsymbol{\theta}_n)$ to emphasize that the variational posterior and hence the local ELBO are both parameterized by $\boldsymbol{\lambda}_n$.

- To perform SGD, we need an unbiased estimate of the gradient of the local ELBO,

$$\nabla_{\boldsymbol{\lambda}_n} \mathscr{L}_n(\boldsymbol{\lambda}_n, \boldsymbol{\theta}) = \nabla_{\boldsymbol{\lambda}_n} \mathbb{E}_{q(\boldsymbol{z}_n; \boldsymbol{\lambda}_n)} \left[\log p(\boldsymbol{x}_n, \boldsymbol{z}_n; \boldsymbol{\theta}) - \log q(\boldsymbol{z}_n)\right] \tag{26}$$

$$\neq \mathbb{E}_{q(\boldsymbol{z}_n; \boldsymbol{\lambda}_n)} \left[\nabla_{\boldsymbol{\lambda}_n} \left(\log p(\boldsymbol{x}_n, \boldsymbol{z}_n; \boldsymbol{\theta}) - \log q(\boldsymbol{z}_n)\right)\right]. \tag{27}$$

- **Question:** Why can't we simply bring the gradient inside the expectation like we did for the M-step?

Stochastic gradient ascent on the local ELBO

- **Idea:** Write the objective in terms of the unconstrained variational parameters

$$\boldsymbol{\lambda}_n \triangleq (\boldsymbol{\mu}_n, \log \boldsymbol{\sigma}_n^2) \in \mathbb{R}^{2H} \tag{25}$$

  and then perform stochastic gradient ascent.

- **Note:** Now we will write $q_n(\boldsymbol{z}_n; \boldsymbol{\lambda}_n)$ and $\mathscr{L}_n(\boldsymbol{\lambda}_n, \boldsymbol{\theta}_n)$ to emphasize that the variational posterior and hence the local ELBO are both parameterized by $\boldsymbol{\lambda}_n$.

- To perform SGD, we need an unbiased estimate of the gradient of the local ELBO,

$$\nabla_{\boldsymbol{\lambda}_n} \mathscr{L}_n(\boldsymbol{\lambda}_n, \boldsymbol{\theta}) = \nabla_{\boldsymbol{\lambda}_n} \mathbb{E}_{q(\boldsymbol{z}_n; \boldsymbol{\lambda}_n)} \left[\log p(\boldsymbol{x}_n, \boldsymbol{z}_n; \boldsymbol{\theta}) - \log q(\boldsymbol{z}_n)\right] \tag{26}$$

$$\neq \mathbb{E}_{q(\boldsymbol{z}_n; \boldsymbol{\lambda}_n)} \left[\nabla_{\boldsymbol{\lambda}_n} \left(\log p(\boldsymbol{x}_n, \boldsymbol{z}_n; \boldsymbol{\theta}) - \log q(\boldsymbol{z}_n)\right)\right]. \tag{27}$$

- **Question:** Why can't we simply bring the gradient inside the expectation like we did for the M-step?

Stochastic gradient ascent on the local ELBO

- **Idea:** Write the objective in terms of the unconstrained variational parameters

$$\boldsymbol{\lambda}_n \triangleq (\boldsymbol{\mu}_n, \log \boldsymbol{\sigma}_n^2) \in \mathbb{R}^{2H} \tag{25}$$

  and then perform stochastic gradient ascent.

- **Note:** Now we will write $q_n(\boldsymbol{z}_n; \boldsymbol{\lambda}_n)$ and $\mathscr{L}_n(\boldsymbol{\lambda}_n, \boldsymbol{\theta}_n)$ to emphasize that the variational posterior and hence the local ELBO are both parameterized by $\boldsymbol{\lambda}_n$.

- To perform SGD, we need an unbiased estimate of the gradient of the local ELBO,

$$\nabla_{\boldsymbol{\lambda}_n} \mathscr{L}_n(\boldsymbol{\lambda}_n, \boldsymbol{\theta}) = \nabla_{\boldsymbol{\lambda}_n} \mathbb{E}_{q(\boldsymbol{z}_n; \boldsymbol{\lambda}_n)} \left[\log p(\boldsymbol{x}_n, \boldsymbol{z}_n; \boldsymbol{\theta}) - \log q(\boldsymbol{z}_n)\right] \tag{26}$$

$$\neq \mathbb{E}_{q(\boldsymbol{z}_n; \boldsymbol{\lambda}_n)} \left[\nabla_{\boldsymbol{\lambda}_n} \left(\log p(\boldsymbol{x}_n, \boldsymbol{z}_n; \boldsymbol{\theta}) - \log q(\boldsymbol{z}_n)\right)\right]. \tag{27}$$

- **Question:** Why can't we simply bring the gradient inside the expectation like we did for the M-step?

The "score function" gradient estimator I

The basic problem is that the variational parameters $\boldsymbol{\lambda}_n$ determine the distribution we are taking an expectation under. However, there are a few ways to obtain unbiased estimates of the gradient.

One approach is called the **score function gradient estimator** or the **REINFORCE estimator** [Williams, 1992]. It is based on the following identity,

$$\nabla_{\lambda} \log q(\boldsymbol{z}; \boldsymbol{\lambda}) = \frac{\nabla_{\lambda} q(\boldsymbol{z}; \boldsymbol{\lambda})}{q(\boldsymbol{z}; \boldsymbol{\lambda})} \tag{28}$$

where the l.h.s. is called the score function of distribution q.

The "score function" gradient estimator II

We can use this identity to obtain an unbiased estimate of the gradient of an expectation,

$$\nabla_{\boldsymbol{\lambda}} \mathbb{E}_{q(\boldsymbol{z};\boldsymbol{\lambda})}\left[h(\boldsymbol{z})\right] = \nabla_{\boldsymbol{\lambda}} \int q(\boldsymbol{z};\boldsymbol{\lambda})\, h(\boldsymbol{z})\, \mathrm{d}\boldsymbol{z} \tag{29}$$

$$= \int \left(\nabla_{\boldsymbol{\lambda}} q(\boldsymbol{z};\boldsymbol{\lambda})\right) h(\boldsymbol{z})\, \mathrm{d}\boldsymbol{z} \tag{30}$$

$$= \int \left(q(\boldsymbol{z};\boldsymbol{\lambda}) \nabla_{\boldsymbol{\lambda}} \log q(\boldsymbol{z};\boldsymbol{\lambda})\right) h(\boldsymbol{z})\, \mathrm{d}\boldsymbol{z} \tag{31}$$

$$= \mathbb{E}_{q(\boldsymbol{z};\boldsymbol{\lambda})}\left[\left(\nabla_{\boldsymbol{\lambda}} \log q(\boldsymbol{z};\boldsymbol{\lambda})\right) h(\boldsymbol{z})\right] \tag{32}$$

From this identity, we can obtain an unbiased Monte Carlo estimate,

$$\widehat{\nabla}_{\boldsymbol{\lambda}} \mathbb{E}_{q(\boldsymbol{z};\boldsymbol{\lambda})}[h(\boldsymbol{z})] = \frac{1}{M} \sum_{m=1}^{M} \left[\nabla_{\boldsymbol{\lambda}} \log q(\boldsymbol{z}^{(m)};\boldsymbol{\lambda})\, h(\boldsymbol{z}^{(m)})\right]; \qquad \boldsymbol{z}^{(m)} \overset{\text{iid}}{\sim} q(\boldsymbol{z};\boldsymbol{\lambda}) \tag{33}$$

The "score function" gradient estimator III

Notes:

1. The exchange of the gradient and the integral is allowed as long as the dominated convergence theorem holds, and it usually does for ML applications.
2. The score function gradient estimator is broadly applicable; e.g. it works for discrete and continuous latent variables $\boldsymbol{z}$. We just need the log density to be continuously differentiable wrt $\boldsymbol{\lambda}$ and to be able to sample from q.
3. If h is a function of both $\boldsymbol{z}$ and $\boldsymbol{\lambda}$, you need to apply the product rule. This gives another term,

$$\nabla_{\boldsymbol{\lambda}} \mathbb{E}_{q(\boldsymbol{z};\boldsymbol{\lambda})} \left[h(\boldsymbol{z}, \boldsymbol{\lambda}) \right] = \mathbb{E}_{q(\boldsymbol{z};\boldsymbol{\lambda})} \left[\left(\nabla_{\boldsymbol{\lambda}} \log q(\boldsymbol{z}; \boldsymbol{\lambda}) \right) h(\boldsymbol{z}, \boldsymbol{\lambda}) \right] + \mathbb{E}_{q(\boldsymbol{z};\boldsymbol{\lambda})} \left[\nabla_{\boldsymbol{\lambda}} h(\boldsymbol{z}, \boldsymbol{\lambda}) \right] \tag{34}$$

Control variates

Though broadly applicable, the score function estimator is often too high variance to be useful. This problem can often be mitigated with **control variates**.

Recall that the expectation of the score is zero,

$$\mathbb{E}_{q(\boldsymbol{z};\boldsymbol{\lambda})}\left[\nabla_{\boldsymbol{\lambda}} \log q(\boldsymbol{z};\boldsymbol{\lambda})\right] = \int q(\boldsymbol{z};\boldsymbol{\lambda}) \nabla_{\boldsymbol{\lambda}} \log q(\boldsymbol{z};\boldsymbol{\lambda}) \,\mathrm{d}\boldsymbol{z} \tag{35}$$

$$= \int \nabla_{\boldsymbol{\lambda}} q(\boldsymbol{z};\boldsymbol{\lambda}) \,\mathrm{d}\boldsymbol{z} \tag{36}$$

$$= \nabla_{\boldsymbol{\lambda}} \int q(\boldsymbol{z};\boldsymbol{\lambda}) \,\mathrm{d}\boldsymbol{z} \tag{37}$$

$$= \nabla_{\boldsymbol{\lambda}} 1 = 0. \tag{38}$$

Thus, we can subtract off any **baseline** from the function of interest without changing the expectation,

$$\mathbb{E}_{q(\boldsymbol{z};\boldsymbol{\lambda})}\left[h(\boldsymbol{z}) \nabla_{\boldsymbol{\lambda}} \log q(\boldsymbol{z};\boldsymbol{\lambda})\right] = \mathbb{E}_{q(\boldsymbol{z};\boldsymbol{\lambda})}\left[(h(\boldsymbol{z}) - b) \nabla_{\boldsymbol{\lambda}} \log q(\boldsymbol{z};\boldsymbol{\lambda})\right]. \tag{39}$$

The pathwise gradient estimator

Suppose $q(\boldsymbol{z};\boldsymbol{\lambda}) = \mathcal{N}(\boldsymbol{z};\boldsymbol{\mu},\mathrm{diag}(\boldsymbol{\sigma}^2))$, as in our variational posteriors (again $\boldsymbol{\lambda} = (\boldsymbol{\mu},\log\boldsymbol{\sigma}^2)$). Then,

$$\boldsymbol{z} \sim q(\boldsymbol{z};\boldsymbol{\lambda}) \quad \Longleftrightarrow \quad \boldsymbol{z} = r(\boldsymbol{\lambda},\boldsymbol{\epsilon}) \tag{40}$$

$$\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0},\boldsymbol{I}) \tag{41}$$

where $r(\boldsymbol{\lambda},\boldsymbol{\epsilon}) = \boldsymbol{\mu} + \boldsymbol{\sigma}\boldsymbol{\epsilon}$ is a **reparameterization** of $\boldsymbol{z}$ in terms of parameters $\boldsymbol{\lambda}$ and noise $\boldsymbol{\epsilon}$.

We can use the **law of the unconscious statistician** to rewrite the expectations as,

$$\mathbb{E}_{q(\boldsymbol{z};\boldsymbol{\lambda})}\left[h(\boldsymbol{z},\boldsymbol{\lambda})\right] = \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{I})}\left[h(r(\boldsymbol{\lambda},\boldsymbol{\epsilon}),\boldsymbol{\lambda})\right] \tag{42}$$

The distribution that the expectation is taken under no longer depends on the parameters $\boldsymbol{\lambda}$, so we can simply take the gradient inside the expectation,

$$\nabla_{\boldsymbol{\lambda}}\mathbb{E}_{q(\boldsymbol{z};\boldsymbol{\lambda})}\left[h(\boldsymbol{z},\boldsymbol{\lambda})\right] = \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{I})}\left[\nabla_{\boldsymbol{\lambda}}h(\boldsymbol{\mu}+\boldsymbol{\sigma}\boldsymbol{\epsilon},\boldsymbol{\lambda})\right] \tag{43}$$

and **use Monte Carlo to obtain an unbiased estimate of the final expectation.**

Empirically comparing estimator variances

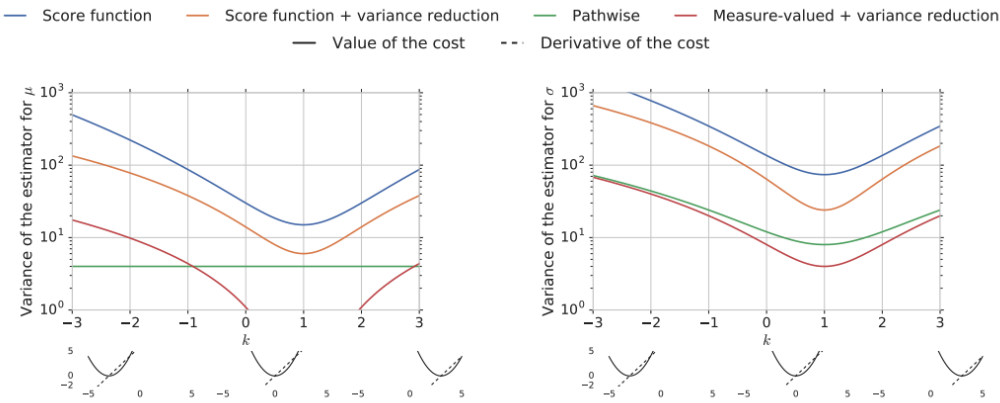


Figure 2: Variance of the stochastic estimates of $\nabla_\theta \mathbb{E}_{\mathcal{N}(x|\mu,\sigma^2)}\left[(x-k)^2\right]$ for $\mu = \sigma = 1$ as a function of k for three different classes of gradient estimators. Left: $\theta = \mu$; right: $\theta = \sigma$. The graphs in the bottom row show the function (solid) and its gradient (dashed) for $k \in \{-3, 0, 3\}$.

Empirical comparisons from Mohamed et al. [2020].

Variational expectation-maximization (vEM)

Now we can add some detail to our variational expectation maximization algorithm.

Repeat until either the ELBO or the parameters converges:

1. **M-step:** Set $\boldsymbol{\theta} \leftarrow \arg\max_{\boldsymbol{\theta}} \mathscr{L}(q, \boldsymbol{\theta})$ [with stochastic gradient ascent on the ELBO]
2. **E-step:** For $n = 1, \ldots, N$
   - ~~Set $q_n \leftarrow \arg\max_{q_n \in \mathscr{Q}} \mathscr{L}_n(q_n, \boldsymbol{\theta})$~~
   - Set $\boldsymbol{\lambda}_n \leftarrow \arg\max_{\boldsymbol{\lambda}_n} \mathscr{L}_n(\boldsymbol{\lambda}_n, \boldsymbol{\theta})$
     [with stochastic gradient ascent on the local ELBO using either the score function estimator or the pathwise gradient estimator]
3. Compute the ELBO $\mathscr{L}(q, \boldsymbol{\theta})$. [with Monte Carlo]

Amortized inference with recognition networks

- Note that vEM involves a costly E-step to find the variational parameters $\boldsymbol{\lambda}_n$ for each data point. This could involve many steps of stochastic gradient descent inside just the E-step!
- With a finite computational budget, we might be better off doing more gradient steps on $\boldsymbol{\theta}$ and fewer on the local variational parameters.
- Note that the optimal variational parameters are just a function of the data point and the model parameters,

$$\boldsymbol{\lambda}_n^\star = \arg\min_{\boldsymbol{\lambda}_n} D_{\mathrm{KL}}\left(q(\boldsymbol{z}_n; \boldsymbol{\lambda}_n) \,\|\, p(\boldsymbol{z}_n \mid \boldsymbol{x}_n, \boldsymbol{\theta})\right) \triangleq f^\star(\boldsymbol{x}_n, \boldsymbol{\theta}). \tag{44}$$

  for some implicit and generally nonlinear function $f^\star$.

Amortized inference with recognition networks II

- VAEs learn an approximation to $f^\star(\boldsymbol{x}_n, \boldsymbol{\theta})$ with an **inference network**, a.k.a. **recognition network** or **encoder**.
- The inference network is (yet another) neural network that takes in a data point $\boldsymbol{x}_n$ and outputs variational parameters $\boldsymbol{z}_n$,

$$\boldsymbol{\lambda}_n \approx f(\boldsymbol{x}_n, \boldsymbol{\phi}), \tag{45}$$

  where $\boldsymbol{\phi}$ are the weights of the network.
- The advantage is that the inference network is very fast; in the E-step, we simply need to pass a data point through the network to obtain the variational parameters.
- The disadvantage is the output will not minimize the KL divergence. However, in practice we might tolerate a worse variational posterior and a weaker lower bound if it buys us more updates of $\boldsymbol{\theta}$.

Amortization and approximation gaps

Cremer et al. [2018] consider the relative effects of the **amortization gap** and the **approximation gap** on variational EM.

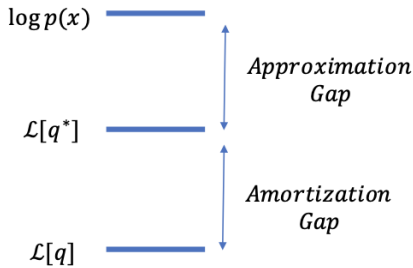


Figure 1. Gaps in Inference

Linear VAEs

Question: What does the optimal encoder network look like for a VAE with a linear generative model,

$$\boldsymbol{z}_n \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}) \tag{46}$$

$$\boldsymbol{x}_n \sim \mathcal{N}(\boldsymbol{W}\boldsymbol{z}_n + \boldsymbol{b}, \boldsymbol{I}) \tag{47}$$

Putting it all together

Logically, I find it helpful to distinguish between the E and M steps, but with recognition networks and stochastic gradient ascent, the line is blurred.

The final algorithm looks like this. Repeat until either the ELBO or the parameters converges:

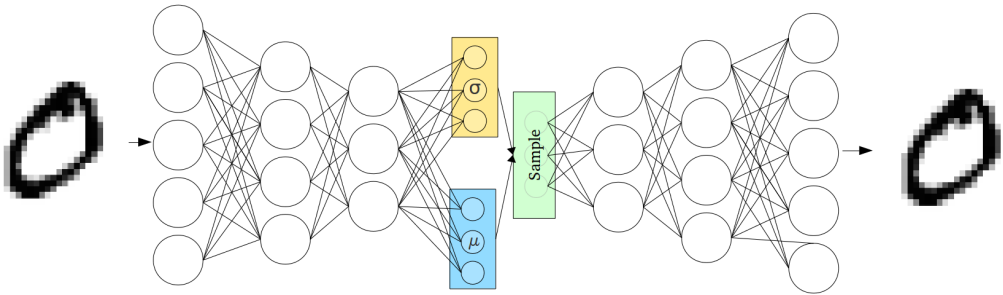
1. Sample data point $n \sim \text{Unif}(1, \ldots, N)$. [Or a minibatch of data points.]
2. Estimate the local ELBO $\mathscr{L}_n(\boldsymbol{\phi}, \boldsymbol{\theta})$ with Monte Carlo. [Note: it is a function of $\boldsymbol{\phi}$ instead of $\boldsymbol{\lambda}_n$.]
3. Compute unbiased Monte Carlo estimates of the gradients $\widehat{\nabla}_{\boldsymbol{\theta}} \mathscr{L}_n(\boldsymbol{\phi}, \boldsymbol{\theta})$ and $\widehat{\nabla}_{\boldsymbol{\phi}} \mathscr{L}_n(\boldsymbol{\phi}, \boldsymbol{\theta})$. [The latter requires the score function or pathwise gradient estimator.]
4. Set

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha_i \widehat{\nabla}_{\boldsymbol{\theta}} \mathscr{L}_n(\boldsymbol{\phi}, \boldsymbol{\theta}) \tag{48}$$

$$\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} + \alpha_i \widehat{\nabla}_{\boldsymbol{\phi}} \mathscr{L}_n(\boldsymbol{\phi}, \boldsymbol{\theta}) \tag{49}$$

with step size α_i decreasing over iterations i according to a valid schedule.

VAEs from an autoencoder perspective



From https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf

References I

Yann LeCun. *Modeles connexionnistes de lapprentissage*. PhD thesis, These de Doctorat, Universite Paris, 1987.

Hervé Bourlard and Yves Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*, 59(4):291–294, 1988.

Geoffrey E Hinton and Richard Zemel. Autoencoders, minimum description length and helmholtz free energy. *Advances in neural information processing systems*, 6, 1993.

G E Hinton and R R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, July 2006.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(12), 2010.

References II

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

D P Kingma and M Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2014.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. January 2014.

Léon Bottou et al. Online learning and stochastic approximations. *On-line learning in neural networks*, 17(9):142, 1998.

Herbert Robbins and David Siegmund. A convergence theorem for non negative almost supermartingales and some applications. In *Optimizing methods in statistics*, pages 233–257. Elsevier, 1971.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

References III

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Shakir Mohamed, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. Monte Carlo gradient estimation in machine learning. *Journal of Machine Learning Research*, 21(132):1–62, 2020.

Chris Cremer, Xuechen Li, and David Duvenaud. Inference suboptimality in variational autoencoders. In *International Conference on Machine Learning*, pages 1078–1086. PMLR, 2018.