Variational Auto-Encoders

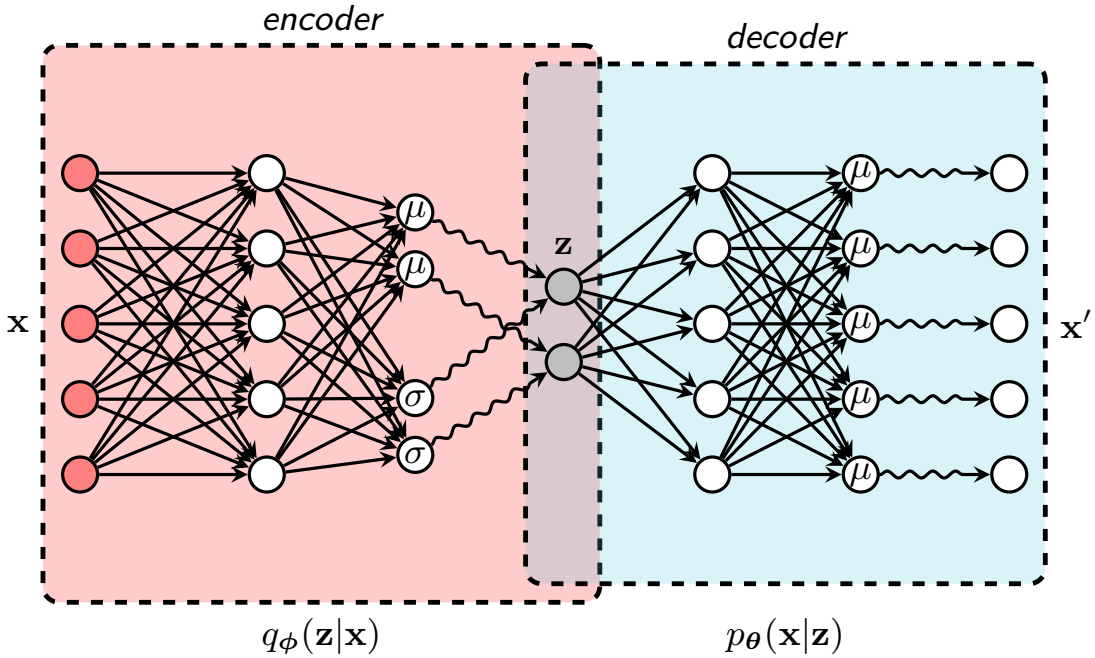
A probabilistic introduction

Alberto Lumbreras

a.lumbreras@criteo.com

April 2020

Variational Autoencoder overview



$$\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})] - D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})|p(\mathbf{z}))$$

$$\approx \sum_j \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}^{(j)})] - \log q_{\boldsymbol{\phi}}(\mathbf{z}^{(j)}|\mathbf{x}) - \log p_{\boldsymbol{\theta}}(\mathbf{z}^{(j)}), \qquad \mathbf{z}^{(j)} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$$

$$\approx \sum_j \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}^{(j)})] - \log q_{\boldsymbol{\phi}}(\mathbf{z}^{(j)}|\mathbf{x}) - \log p_{\boldsymbol{\theta}}(\mathbf{z}^{(j)}), \qquad \mathbf{z}^{(j)} = g_\phi(\boldsymbol{\epsilon}^{(\mathbf{j})}, \mathbf{x}),$$

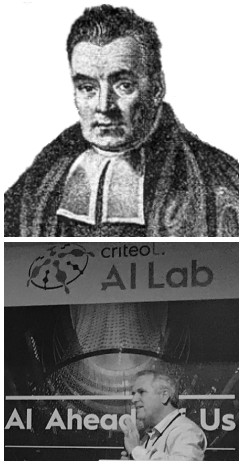
$$\epsilon^{(j)} \sim p(\boldsymbol{\epsilon})$$

criteo

Two schools of thought

- Connectionist school:
  - An auto-encoder that uses Variational Inference.
  - Scalable method for data generation.
  - Competes with Generative Adversarial Networks (GANs).
  - Mathematical justification at the end.

- Probabilistic school:
  - A Variational Inference that looks like an auto-encoder.
  - Scalable method for density estimation.
  - Competes with of Mean-Field Variational Bayes (MF-VB).
  - Mathematical justification from the beginning.

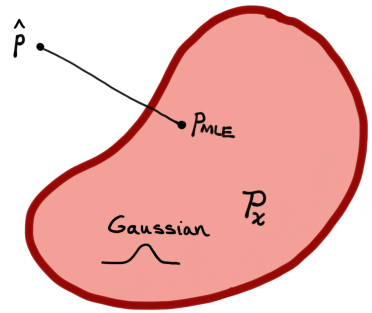


Maximum Likelihood Estimation (MLE)

- We observe $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N$, i.i.d samples from some (unknown) distribution $\hat{p}$,

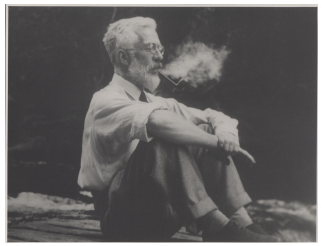
$$\mathbf{x}_n \sim \hat{p}(\mathbf{x})$$

- We want to find the distribution $p_{\boldsymbol{\theta}} \in \mathcal{P}_x$ that is closest to $\hat{p}$.



- The Maximum Likelihood Estimator minimises the Kullback-Leibler divergence:

$$\begin{aligned} D_{\mathrm{KL}}(\hat{p}(\mathbf{x})|p_{\boldsymbol{\theta}}(\mathbf{x})) &= \mathbb{E}_{\hat{p}(\mathbf{x})}\left[\log \frac{\hat{p}(\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{x})}\right] \\ &= -\mathbb{E}_{\hat{p}(\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x})] + \text{const.} \\ &\approx -\sum_{n=1}^{N} \log p_{\boldsymbol{\theta}}(\mathbf{x}_n) + \text{const.} \end{aligned}$$

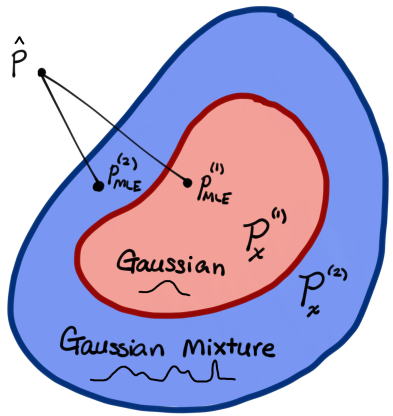


Maximum Likelihood Estimation with Latent Variables

- We can assume each observation depends on a latent (non-observed) variable $\mathbf{z}$:

$$\mathbf{z}_n \sim \hat{p}(\mathbf{z})$$
$$\mathbf{x}_n \sim \hat{p}(\mathbf{x}|\mathbf{z}_n)$$

- This increases expressiveness,



- But complicates Maximum Likelihood Estimation:

$$\arg\max_{\boldsymbol{\theta}} \sum_{n=1}^{N} \log p_{\boldsymbol{\theta}}(\mathbf{x}_n) = \arg\max_{\boldsymbol{\theta}} \sum_{n=1}^{N} \log \int p_{\boldsymbol{\theta}}(\mathbf{x}_n, \mathbf{z}_n) \mathrm{d}\mathbf{z}_n$$

The intractable integral and the Evidence Lower Bound

1. The integral is often intractable. We can re-express it as:

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) = \log \int p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) \mathrm{d}\mathbf{z} = \log \int \frac{p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z})} q_{\boldsymbol{\phi}}(\mathbf{z}) \mathrm{d}\mathbf{z} = \log \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})} \left[\frac{p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z})} \right]$$

   where $q_{\boldsymbol{\phi}}(\mathbf{z})$ can be **any** density[1].

2. Use Jensen ($\log \mathbb{E}_{q(z)}[f(z)] \geq \mathbb{E}_{q(z)}[\log f(z)]$) to find a tractable lower bound:

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) \geq \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})} \left[\log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z})} \right] = \mathcal{L}_{\boldsymbol{\theta}, \boldsymbol{\phi}}(\mathbf{x}) \tag{1}$$

Posterior and density estimation

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) = \mathcal{L}_{\boldsymbol{\theta}, \boldsymbol{\phi}}(\mathbf{x}) \iff q_{\boldsymbol{\phi}}(\mathbf{z}) = p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$$

[1] For those familiar with Importance Sampling: it is the same expression.

Maximizing the ELBO

Strategies for posteriors in close form, samples or approximations.

- Expectation-Maximization (EM) (Dempster et al. 1977):
  (posterior: analytically available)

$$\mathcal{L}_{\boldsymbol{\theta}}(\mathbf{X}) = \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{Z}|\mathbf{X})}[\log p_{\boldsymbol{\theta}}(\mathbf{X}, \mathbf{Z}) - \log p_{\boldsymbol{\theta}}(\mathbf{Z}|\mathbf{X})]$$

- Monte-Carlo EM (MC-EM) (Wei and Tanner 1990):
  (posterior: analytically not available)

$$\mathcal{L}_{\boldsymbol{\theta}}(\mathbf{X}) = \frac{1}{J}\sum_{j}\{\log p_{\boldsymbol{\theta}}(\mathbf{X}, \mathbf{Z}^{(j)}) - \log p_{\boldsymbol{\theta}}(\mathbf{Z}^{(j)}|\mathbf{X})\}, \qquad \mathbf{Z}^{(j)} \sim p_{\boldsymbol{\theta}}(\mathbf{Z}|\mathbf{X})$$

- Mean-Field Variational Bayes EM (MFVB-EM) (Jordan et al. 1999):
  (posterior: approximation analytically available)

$$\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{Z})}[\log p_{\boldsymbol{\theta}}(\mathbf{X}, \mathbf{Z}) - \log q_{\boldsymbol{\phi}}(\mathbf{Z})], \qquad q_{\boldsymbol{\phi}}(\mathbf{Z}) = \prod_{n} q_{\boldsymbol{\phi}}^{\{n\}}(\mathbf{z}_n)$$

None of the above methods can be used for complicated models

1. Two schools of thought

2. Maximum likelihood estimation
   - Basic MLE
   - MLE with Latent Variables

3. **Variational Autoencoders**
   - You cannot always push gradients inside expectations
   - The old and the new hack
   - Scaling with amortization
   - The trinity of the Evidence Lower Bound

4. VAE algorithms
   - Gaussian VAE
   - Pseudo-code to train
   - Pseudo-code to generate synthetic samples
   - Pseudo-code to generate latent representations

5. FAQ

Maximizing the ELBO without assumptions

$\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}[\log_{\boldsymbol{\theta}} p(\mathbf{x},\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z})]$

- The posterior is too complex to make simplistic independence assumptions (~~MFVB-EM~~)

A very generic strategy:

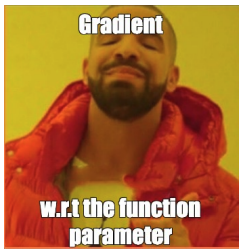
1. Push the gradient inside the expectation (careful: *here be dragons*)[2].
2. And **then** we approximate the expectation by Monte-Carlo sampling.

[2] See Mohamed et al. 2019 for an excellent review on this problem.

Pushing gradients inside expectations

$\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}[\log_{\boldsymbol{\theta}} p(\mathbf{x},\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z})]$

- The gradient w.r.t. $\boldsymbol{\theta}$ can be pushed-in:

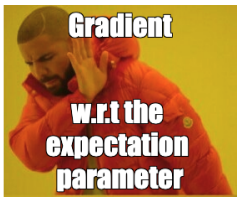


$$\nabla_{\boldsymbol{\theta}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}[f_{\boldsymbol{\theta}}(\mathbf{z})] = \nabla_{\boldsymbol{\theta}} \int f_{\boldsymbol{\theta}}(\mathbf{z}) q_{\boldsymbol{\phi}}(\mathbf{z}) \mathrm{d}\mathbf{z} \tag{2}$$

$$= \int \{\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{z})\} q_{\boldsymbol{\phi}}(\mathbf{z}) \mathrm{d}\mathbf{z} \tag{3}$$

$$= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}[\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{z})] \approx \frac{1}{J} \sum_{j} \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{z}^{(j)}) \tag{4}$$

- The gradient w.r.t. the posterior parameters $\boldsymbol{\phi}$ cannot be pushed-in:[3]



$$\nabla_{\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}[f_{\boldsymbol{\theta}}(\mathbf{z})] = \nabla_{\boldsymbol{\phi}} \int f_{\boldsymbol{\theta}}(\mathbf{z}) q_{\boldsymbol{\phi}}(\mathbf{z}) \mathrm{d}\mathbf{z} \tag{5}$$

$$= \int f_{\boldsymbol{\theta}}(\mathbf{z}) \nabla_{\boldsymbol{\phi}} q_{\boldsymbol{\phi}}(\mathbf{z}) \mathrm{d}\mathbf{z} \tag{6}$$

$$\neq \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}[\nabla_{\boldsymbol{\phi}} f_{\boldsymbol{\theta}}(\mathbf{z})] \approx \frac{1}{J} \sum_{j} \nabla_{\boldsymbol{\phi}} f_{\boldsymbol{\theta}}(\mathbf{z}^{(j)}) \tag{7}$$

[3] This problem did not appear in the era before Variational Inference, since we only had $\boldsymbol{\theta}$.

The old hack

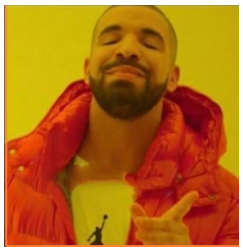
$\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}[\log_{\boldsymbol{\theta}} p(\mathbf{x},\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z})]$

- The score function:

$$\nabla_\theta \log p_\theta(x) = \frac{\nabla_\theta p_\theta(x)}{p_\theta(x)} \implies \nabla_\theta p_\theta(x) = p_\theta(x) \nabla_\theta \log p_\theta(x) \tag{8}$$

- The gradient w.r.t. the posterior parameters $\boldsymbol{\phi}$ ~~cannot be pushed in~~:

$$\begin{aligned}
\nabla_{\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}[f_{\boldsymbol{\theta}}(\mathbf{z})] &= \nabla_{\boldsymbol{\phi}} \int f_{\boldsymbol{\theta}}(\mathbf{z}) q_{\boldsymbol{\phi}}(\mathbf{z}) \mathrm{d}\mathbf{z} \\
&= \int f_{\boldsymbol{\theta}}(\mathbf{z}) \nabla_{\boldsymbol{\phi}} q_{\boldsymbol{\phi}}(\mathbf{z}) \mathrm{d}\mathbf{z} \\
&= \int f_{\boldsymbol{\theta}}(\mathbf{z}) q_{\boldsymbol{\phi}}(\mathbf{z}) \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{z}) \mathrm{d}\mathbf{z} \\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}[f_{\boldsymbol{\theta}}(\mathbf{z}) \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{z})] \\
&\approx \frac{1}{J} \sum_j f_{\boldsymbol{\theta}}(\mathbf{z}^{(j)}) \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{z}^{(j)})
\end{aligned}$$



- Needs variance reduction (Paisley et al. 2012).

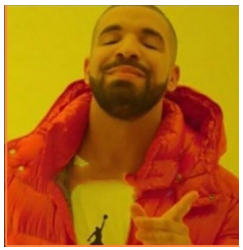
The new hack: Stochastic Gradient Variational Bayes estimator (SGVB)

- The Law Of the Unconscious Statistician (LOTUS):

  Given a random variable ϵ with a probability distribution $p(\epsilon)$ and a transformation $z = g(\epsilon)$:[4]

$$\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}[f(\mathbf{z})] = \mathbb{E}_{p(\boldsymbol{\epsilon})}[f(g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}))] \tag{9}$$

- The gradient w.r.t. the posterior parameters $\boldsymbol{\phi}$ ~~cannot be pushed in~~:



$$\begin{aligned}\nabla_{\boldsymbol{\phi}}\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}[f_{\boldsymbol{\theta}}(\mathbf{z})] &= \nabla_{\boldsymbol{\phi}}\mathbb{E}_{p(\boldsymbol{\epsilon})}[f_{\boldsymbol{\theta}}(g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}))] \\ &= \mathbb{E}_{p(\boldsymbol{\epsilon})}[\nabla_{\boldsymbol{\phi}} f_{\boldsymbol{\theta}}(g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}))] \\ &\approx \frac{1}{J}\sum_{j} \nabla_{\boldsymbol{\phi}} f_{\boldsymbol{\theta}}(g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^{(j)}))\end{aligned}$$

[4] For those familiar with Normalizing Flows: $g(\epsilon)$ can be a composition $g_k(g_{k-1}(\ldots g_1(\epsilon)))$

Stochastic Gradient Variational Bayes estimator (SGVB)

Using LOTUS and amortized inference

- Stochastic Gradient Variational Bayes estimator:
  (Kingma and Welling 2014)

$$\boldsymbol{\epsilon}^{(j)} \sim p(\boldsymbol{\epsilon})$$
$$\mathbf{z}^{(j)} = g_{\boldsymbol{\phi}}((\boldsymbol{\epsilon}^{(j)}, \mathbf{x}^{(j)}))$$
$$\nabla_{\boldsymbol{\phi}} \mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) \approx \frac{1}{J} \sum_{j} \nabla_{\boldsymbol{\phi}} \log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}^{(j)}) - \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{z}^{(j)}|\mathbf{x})$$

Samples are drawn from a distribution that is not affected by the gradient

Amortized Variational Inference

Making it scalable by reducing the number of parameters[5]

- Classic Variational Inference:
  - Posterior $q(\mathbf{z}_n; \boldsymbol{\phi}_n)$ depends on local parameters.
  - Variational parameters: $\mathcal{O}(N)$.
- Amortized Variational Inference:
  - Posterior $q(\mathbf{z}_n; f_{\boldsymbol{\phi}}(\mathbf{x}_n))$ depends on shared parameters (neural net), .
  - Variational parameters: $\mathcal{O}(1)$.

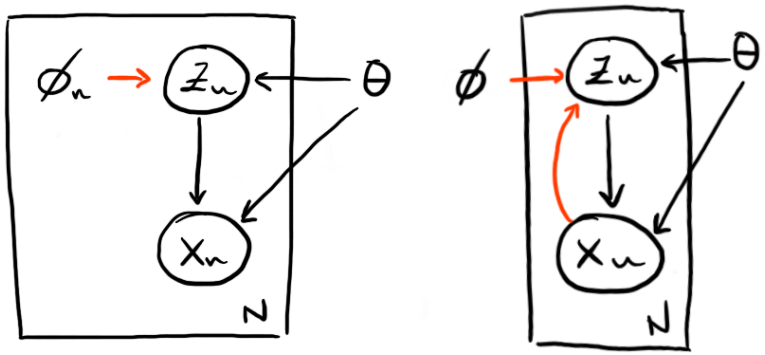


Figure: Non-amortized (left) and amortized variational models (right). Red lines represent the dependencies of the variational distribution of $\mathbf{z}_n$

[5] See Zhang et al. 2019 for more context.

The trinity of the ELBO

- The classic:
(expected joint log-likelihood plus entropy)

$$\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})] - \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})] \tag{10}$$

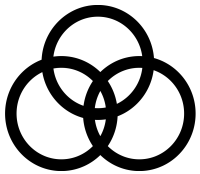
- The variational:
(evidence minus quality of the posterior approximation)

$$p(\mathbf{x}) - \mathrm{D}_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})|p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})) \tag{11}$$

- The auto-enconder:
(quality of reconstruction minus regularization term)

$$\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})] - D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})|p(\mathbf{z})) \tag{12}$$

The three are the same.



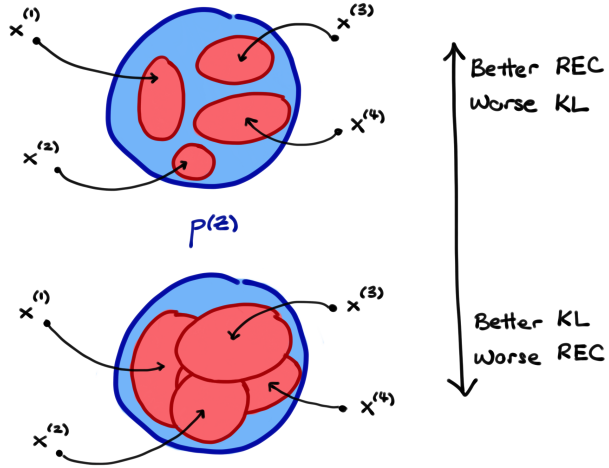
Why is it called Variational Autoencoder

- The third form of the ELBO trinity is:

$$\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})] - D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})|p(\mathbf{z}))$$

  where we identified a reconstruction and a regularizer term.

- This view makes explicit the trade-off between fitting the data (reconstruction) and fitting the prior (regularization):[6]



[6] The nice drawings are taken from http://ruishu.io/2018/03/14/vae/

1. Two schools of thought

2. Maximum likelihood estimation
   - Basic MLE
   - MLE with Latent Variables

3. Variational Autoencoders
   - You cannot always push gradients inside expectations
   - The old and the new hack
   - Scaling with amortization
   - The trinity of the Evidence Lower Bound

4. **VAE algorithms**
   - Gaussian VAE
   - Pseudo-code to train
   - Pseudo-code to generate synthetic samples
   - Pseudo-code to generate latent representations

5. FAQ

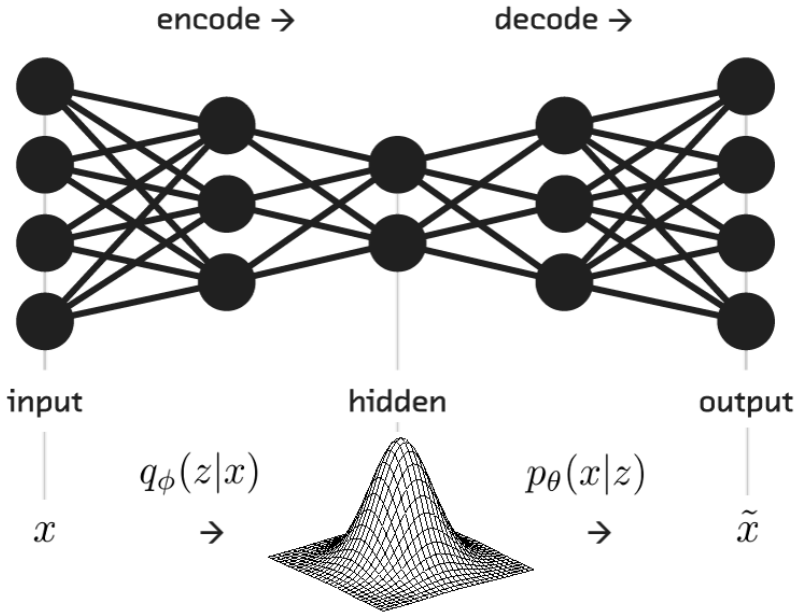
Variational Auto-Encoder

Gaussian posterior, Gaussian likelihood

- For the re-parameterization trick to be used, we need:
  1. A family distribution for the posterior $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$.
  2. A differentiable transformation $\mathbf{z} = g_{\phi}(\boldsymbol{\epsilon})$.
  3. An distribution over the auxiliary variable $p(\boldsymbol{\epsilon})$

- Gaussian posterior (encoder):
  1. $\mathbf{z}|\mathbf{x} \sim \mathcal{N}(\mathbf{z}|f_{\boldsymbol{\mu}}(\mathbf{x}), f_{\sigma}(\mathbf{x})\mathbf{I})$
  2. $\mathbf{z} = g_{\boldsymbol{\phi}}(\mathbf{x}, \boldsymbol{\epsilon}) = f_{\boldsymbol{\mu}}^{(e)}(\mathbf{x}) + f_{\sigma}^{(e)}(\mathbf{x})\mathbf{I}\boldsymbol{\epsilon}$
  3. $\boldsymbol{\epsilon} \sim \mathcal{N}(0, 1)$
- Gaussian likelihood (decoder):
  1. $\mathbf{x}|\mathbf{z} \sim \mathcal{N}(\mathbf{x}|f_{\boldsymbol{\mu}}(\mathbf{z}), f_{\sigma}(\mathbf{z})\mathbf{I})$
  2. $\mathbf{x} = g_{\boldsymbol{\theta}}(\mathbf{z}, \boldsymbol{\epsilon}) = f_{\boldsymbol{\mu}}^{(d)}(\mathbf{z}) + f_{\sigma}^{(d)}(\mathbf{z})\mathbf{I}\boldsymbol{\epsilon}$
  3. $\boldsymbol{\epsilon} \sim \mathcal{N}(0, 1)$

  where $f^{(e)}, f^{(d)}$ can be any differentiable function such as a neural network.



Training algorithm

Algorithm 1: Batch version of Auto-Encoding VB

Input: Observed data vectors $\mathbf{x}_1, ...\mathbf{x}_N$

Parameters: Base distribution $p(\boldsymbol{\epsilon})$, sampling path $\mathbf{z} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{x})$, prior distribution $p(\mathbf{z})$, likelihood $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$.

Output: Model parameters $\boldsymbol{\theta}$, variational parameters $\boldsymbol{\phi}$

Initialize: Random initialization of $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$

repeat

  Sample $\boldsymbol{\epsilon}_n \sim p(\boldsymbol{\epsilon})$

  $\mathbf{z}_n = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{x}_n)$

  Compute gradients:

  $\frac{1}{N} \sum_n \nabla_{\boldsymbol{\theta},\boldsymbol{\phi}} [\log p_{\boldsymbol{\theta}}(\mathbf{x}_n|\mathbf{z}_n)) - \{\log q_{\phi}(\mathbf{z}_n|\mathbf{x}_n) - \log p(\mathbf{z}_n)\}]$

  $(\boldsymbol{\theta}, \boldsymbol{\phi}) \leftarrow$ Update parameters using gradients (e.g.: SGD)

until *convergence of* $(\boldsymbol{\theta}, \boldsymbol{\phi})$;

- The more frequently used mini-batch version can be easily derived from this.
- The parameters of the encoder q_{ϕ} are the outputs of a neural net with inputs $\mathbf{x}$.
- The parameters of the decoder p_{θ} are the outputs of a neural net with inputs $\mathbf{z}$.

Algorithm to generate synthetic samples

- We want to sample from

$$p_{\boldsymbol{\theta}}(\mathbf{x}) = \int p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p(\mathbf{z})\mathrm{d}\mathbf{z}. \tag{13}$$

Algorithm 2: VAE sample generator

Parameters: Prior distribution $p(\mathbf{z})$, likelihood $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$, decoder parameters $\boldsymbol{\theta}$
Output: Synthetic data points $\mathbf{x}$
Sample $\mathbf{z} \sim p(\mathbf{z})$
Sample $\mathbf{x} \sim p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$

Algorithm to generate latent representation

- We want to sample from

$$p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) \tag{14}$$

Algorithm 3: VAE code generator

Input: Observed data vector $\mathbf{x}$
Parameters: Posterior distribution $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$, encoder parameters $\boldsymbol{\phi}$
Output: Latent vector $\mathbf{z}$
Sample $\mathbf{z} \sim p_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$

Frequenly Asked Questions

- **Do we loose expressiveness when we do Amortized Inference?**
  Yes. This is known as the amortization error (Cremer et al. 2018). In some cases (Sønderby et al. 2016), this gap is reasonably small. In general, the more layers your neural nets have, the shorter this gap can be. Note also that this gap is the only thing that prevents the ELBO from reaching $p(\mathbf{x})$. In other words, without this gap, the KL divergence between the variational and the true posterior could be zero.
- **Can we use discrete likelihoods for discrete data?**
  Of course. For instance, for binary data you can use a Bernoulli likelihood.
- **Can we use discrete latent variables?**
  Not straightforwardly, but yes, with some hack called the Gumbel-Max trick (Jang et al. 2017).
- **What should I use as a prior $p(\mathbf{z})$?**
  People often use a standard normal distribution. You can use whatever you want. Of course there will be better and worse priors for each problem. As there are better and worse regularization terms in any other model.
- **Can I sample from the approximate posterior $q(\mathbf{z}|\mathbf{x})$?**
  Yes. Just put an $\mathbf{x}$ in the encoder and it will output the parameters of the posterior (e.g.: the mean and variance of a Gaussian, if we choose our posterior to be Gaussian) to sample from.

References I

Cremer, Chris et al. (2018). "Inference Suboptimality in Variational Autoencoders". In: *Proceedings of the 35th International Conference on Machine Learning (ICML '18)*. Vol. 80, pp. 1078–1086.

Dempster, A. P. et al. (1977). "Maximum Likelihood from Incomplete Data via the EM Algorithm". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 39.1, pp. 1–38.

Jang, Eric et al. (2017). "Categorical Reparametrization with Gumbel-Softmax". In: *Proceedings International Conference on Learning Representations (ICLR '17)*.

Jordan, Michael I. et al. (1999). "An Introduction to Variational Methods for Graphical Models". In: *Machine Learning* 37.2, pp. 183–233.

Kingma, Diederik P. and Max Welling (2014). "Auto-Encoding Variational Bayes". In: *2nd International Conference on Learning Representations, (ICLR '14)*.

Mohamed, Shakir et al. (2019). *Monte Carlo Gradient Estimation in Machine Learning*. arXiv: 1906.10652 [stat.ML].

Paisley, John et al. (2012). "Variational Bayesian Inference with Stochastic Search". In: *Proceedings of the 29th International Coference on International Conference on Machine Learning (ICML '12)*, pp. 1363–1370.

References II

Sønderby, Casper Kaae et al. (2016). "Ladder Variational Autoencoders". In: *Advances in Neural Information Processing Systems 29*, pp. 3738–3746.

Wei, Greg C. G. and Martin A. Tanner (1990). "A Monte Carlo Implementation of the EM Algorithm and the Poor Man's Data Augmentation Algorithms". In: *Journal of the American Statistical Association* 85.411, pp. 699–704.

Zhang, C. et al. (2019). "Advances in Variational Inference". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41.08, pp. 2008–2026.

Appendix

Change of variable

- Let us consider density $q_\phi(\mathbf{z})$ and a change of variable $\mathbf{z} = g_\phi(\boldsymbol{\epsilon})$.
- The probability contained in a differential area must be invariant under change of variables.

$$p(\boldsymbol{\epsilon})\mathrm{d}\boldsymbol{\epsilon} = p(\mathbf{z})\mathrm{d}\mathbf{z} \tag{15}$$

$$p(\boldsymbol{\epsilon})\mathrm{d}\boldsymbol{\epsilon} = p(g(\boldsymbol{\epsilon}))\mathrm{d}g(\boldsymbol{\epsilon}) \tag{16}$$

$$p(\boldsymbol{\epsilon}) = p(g(\boldsymbol{\epsilon}))\left|\frac{\mathrm{d}g(\boldsymbol{\epsilon})}{\mathrm{d}\boldsymbol{\epsilon}}\right| \tag{17}$$

- Application to expectations (LOTUS):

$$\mathbb{E}_{q_\phi(\mathbf{z})}[f(\mathbf{z})] = \int f(\mathbf{z})q_\phi(\mathbf{z})\mathrm{d}\mathbf{z} = \int f(g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}))p(\boldsymbol{\epsilon})\mathrm{d}\boldsymbol{\epsilon} = \mathbb{E}_{p(\boldsymbol{\epsilon})}[f(g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}))] \tag{18}$$

Pushing gradients inside expectations

$\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}[\log_{\boldsymbol{\theta}} p(\mathbf{x},\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z})]$

- The gradient w.r.t. $\boldsymbol{\theta}$ can be pushed-in:[7]:

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) = \nabla_{\boldsymbol{\theta}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})] = \nabla_{\boldsymbol{\theta}} \int \log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}) q_{\boldsymbol{\phi}}(\mathbf{z}) \mathrm{d}\mathbf{z} \tag{19}$$

$$= \int \{\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})\} q_{\boldsymbol{\phi}}(\mathbf{z}) \mathrm{d}\mathbf{z} = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})] \tag{20}$$

- The gradient w.r.t. the posterior parameters $\boldsymbol{\phi}$ cannot be pushed-in:

$$\nabla_{\boldsymbol{\phi}} \mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) = \nabla_{\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}[\log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z})] \tag{21}$$

$$= \nabla_{\boldsymbol{\phi}} \int [\log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z})] q_{\boldsymbol{\phi}}(\mathbf{z}) \mathrm{d}\mathbf{z} \tag{22}$$

$$= \int \nabla_{\boldsymbol{\phi}} \{[\log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z})] q_{\boldsymbol{\phi}}(\mathbf{z})\} \mathrm{d}\mathbf{z} \tag{23}$$

$$\neq \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}[\nabla_{\boldsymbol{\phi}} \log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z})] \tag{24}$$

[7] For those familiar with Expectation Maximization: this gradient appears in the M-step. This push-in is used in MC-EM. No need to push-in if we can solve the expectation analytically.

Useful identities

- Score function:

$$\nabla_\theta \log p_\theta(x) = \frac{\nabla_\theta p_\theta(x)}{p_\theta(x)} \tag{25}$$

- The Law Of the Unconscious Statistician (LOTUS):

  Given a random variable ϵ with a probability distribution $p(\epsilon)$ and a transformation $z = g(\epsilon)$:[8]

$$\mathbb{E}_{p(z)}[z] = \mathbb{E}_{p(\epsilon)}[g(\epsilon)] \tag{26}$$

  which means that we can compute the expectation of a function of a random variable z without knowing its distribution, if we know its corresponding sampling path and base distribution.

[8] For those familiar with Normalizing Flows: $g(\epsilon)$ can be a composition $g_k(g_{k-1}(...g_1(\epsilon)))$

Naive Monte-Carlo gradient estimator

Using the Score Function

- Naïve Monte-Carlo gradient estimator (Paisley et al. 2012):
  Using the score function, we obtain:

$$\begin{aligned}\nabla_{\boldsymbol{\phi}}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) &= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}[\{\log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z})\}\nabla_{\boldsymbol{\phi}}\log q_{\boldsymbol{\phi}}(\mathbf{z})] \\ &\approx \frac{1}{J}\sum_{j}\{\log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}^{(j)}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}^{(j)})\}\nabla_{\boldsymbol{\phi}}\log q_{\boldsymbol{\phi}}(\mathbf{z}^{(j)}) \\ \text{where} \quad \mathbf{z}^{(j)} &\sim q_{\boldsymbol{\phi}}(\mathbf{z}) \end{aligned} \tag{27}$$

This estimator suffers from a high variance