



# Designing variational autoencoders

Dr. Luca Ambrogioni





# Part I: VAEs in practice

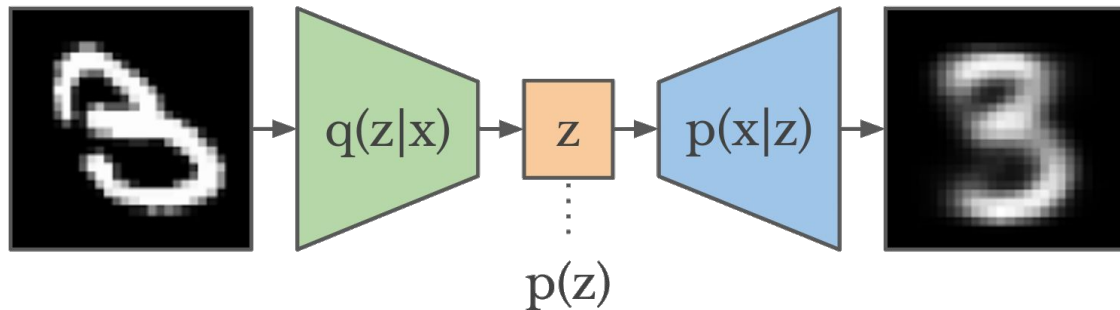
Dr. Luca Ambrogioni



# VAE architectures

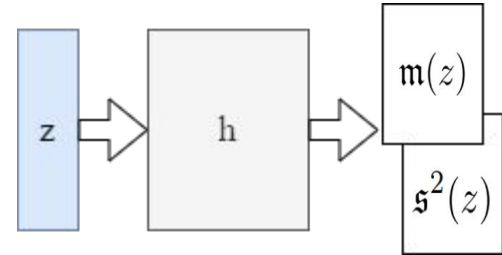
Two probabilistic neural architectures:

- The decoder  $p(x|z)$ : It maps a latent code  $z$  into a data point  $x$  (e.g. an image)
- The encoder  $q(z|x)$ : It encodes a data point  $x$  into a latent code  $z$ .



## A simple architecture: Fully connected decoders I

We assume that the probability of the data given the latent (i.e. the likelihood) is given by a spherical normal distribution with mean and variance given by the decoder architecture.



$$h(z) = \text{ReLU}(Wz + a)$$

$$\mathbf{m}(z) = Vh(z) + b, \quad \mathbf{s}^2(z) = \text{SoftPlus}(Kh(z) + c)$$

$$p(x | z) = \mathcal{N}(x; \mathbf{m}(z), \text{Diag}(\mathbf{s}^2(z) + \delta))$$



## A simple architecture: Fully connected decoders II

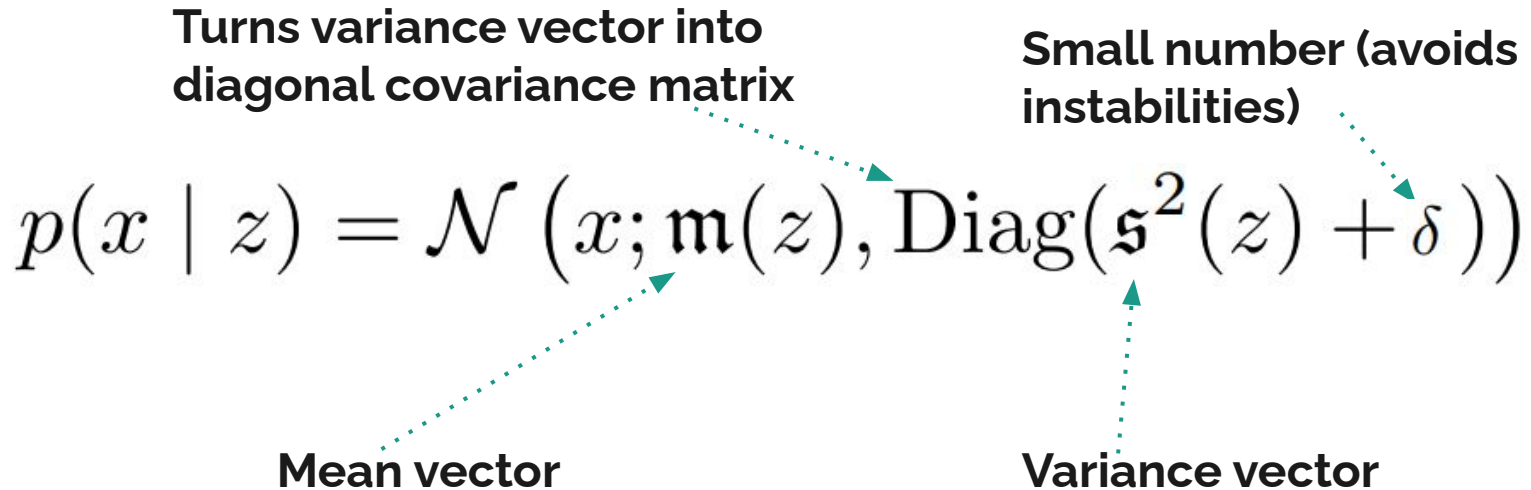
Turns variance vector into diagonal covariance matrix

Small number (avoids instabilities)

$$p(x \mid z) = \mathcal{N}(x; \mathbf{m}(z), \text{Diag}(\mathbf{s}^2(z) + \delta))$$

Mean vector

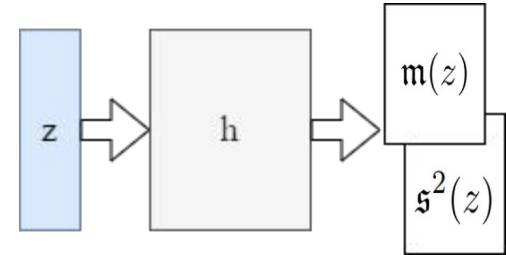
Variance vector



## A simple architecture: Fully connected decoders III

Learnable weight matrices

Learnable biases



$$h(z) = \text{ReLU}(Wz + a)$$

$$m(z) = Vh(z) + b, \quad s^2(z) = \text{SoftPlus}(Kh(z) + c)$$



## Interlude: The mean-field approximation

The encoder  $q(z | x)$  is an approximation of the posterior distribution  $p(z | x)$ .

- A mean-field encoder uses the simplifying assumption that all latent variables are conditionally independent given the data:

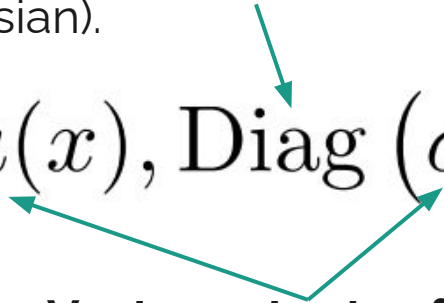
$$q(z | x) = \prod_{i=1}^M q_i(z_i | x)$$

- Under this approximation, the encoder network needs to output the individual parameters of each latent variable.



## Interlude: The Gaussian mean-field approximation

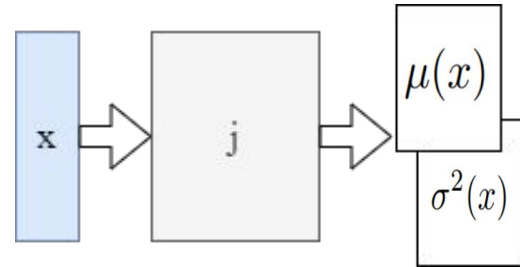
The marginal probability densities of the approximate posterior are often assumed to be Gaussian. The mean-field approximation results in a diagonal covariance matrix (Spherical Gaussian).


$$q(z \mid x) = \mathcal{N}(z; \mu(x), \text{Diag}(\sigma^2(x) + \delta))$$

**Vector outputs of the encoder network**



## A simple architecture: Fully connected mean-field encoders



$$j(x) = \text{ReLu}(Hx + d)$$

$$\mu(x) = Jj(x) + v, \quad \sigma^2(x) = \text{SoftPlus}(Lj(x) + w)$$

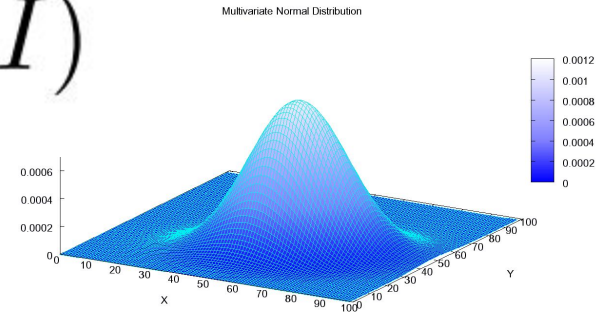
$$p(x | z) = \mathcal{N}(x; \mu(x), \text{Diag}(\sigma^2(x) + \epsilon))$$



## A simple architecture: Standard Gaussian prior

The standard Gaussian prior “pushes” the latent variables toward the origin.

$$p(z) = \mathcal{N}(z; 0, I)$$





## The model parameters

We can collect all the parameters (weights and biases) into model and posterior parameters.

$$\Phi = \underbrace{(W, V, K, a, b, c)}$$

Model parameters (Decoder)

$$\Psi = \underbrace{(H, J, L, d, v, w)}$$

Approximate posterior parameters (Encoder)



## The ELBO: a biased estimator of the marginal likelihood

The model parameters should in theory be trained by maximizing the log marginal likelihood. Since the marginalization is intractable, we maximize ELBO instead.

$$\log p(D \mid \Theta) = \sum_{j=1}^J \log p(x_j; \Theta) \geq \sum_{j=1}^J \text{ELBO} (q(z \mid x_j; \Psi), p(x_j, z; \Phi)) = -\mathcal{L}(\Psi, \Phi)$$

The ELBO becomes equal to the log marginal likelihood when the encoder  $q(z|x)$  is equal to the true posterior  $p(z|x)$ .



## The ELBO is proportional to the KL divergence

The parameters of the variational posterior can be trained by maximizing the ELBO as well since it is equal to the KL divergence between variational approximation (encoder) and true posterior up to an additive proportionality constant.

$$\mathcal{L}(\Psi, \Phi) = - \sum_{j=1}^J \text{ELBO} (q(z | x_j; \Psi), p(x_j, z; \Phi)) = \sum_{j=1}^J D_{\text{KL}}(q(z | x_j), p(z | x_j)) + C$$

This is a form of *amortized inference*, we are jointly training a set of many posteriors, one for each data point.

Amortization post: <http://ruishu.io/2017/11/07/amortized-optimization/>

# The ELBO in details

Number data-points

Sum over data dimensions (e.g. pixels)

$$\mathcal{L}(\Psi, \Phi) = J \mathbb{E}_{x \sim \text{dataset}} \left[ \left( \mathbb{E}_{z(x; \Psi) \sim q(z|x; \Psi)} \left[ \sum_{i=1}^M \frac{1}{2 \mathfrak{s}_i^2(z(x; \Psi); \Phi)} (\mathfrak{m}_i(z(x; \Psi); \Phi) - x_i)^2 \right] + \frac{1}{2} \log \mathfrak{s}_i(z(x; \Psi); \Phi) \right) \right] + J \mathbb{E}_{x \sim \text{dataset}} [D_{\text{KL}}(q(z | x; \Psi), p(z))]$$

The encoded latent depends on the variational parameters

## The ELBO in more details (Analytic KL formula)

$$\mathcal{L}(\Psi, \Phi) = J \mathbb{E}_{x \sim \text{dataset}} \left[ \left( \mathbb{E}_{z(x; \Psi) \sim q(z|x; \Psi)} \left[ \sum_{i=1}^M \frac{1}{2 \mathfrak{s}_i^2(z(x; \Psi); \Phi)} (\mathfrak{m}_i(z(x; \Psi); \Phi) - x_i)^2 \right] + \frac{1}{2} \log \mathfrak{s}_i(z(x; \Psi); \Phi) \right) \right] \\ + JM \mathbb{E}_{x \sim \text{dataset}} \left[ -\log \sigma(x; \Psi) + \frac{1}{2} (\sigma^2(x; \Psi) + \mu^2(x; \Psi)) \right]$$

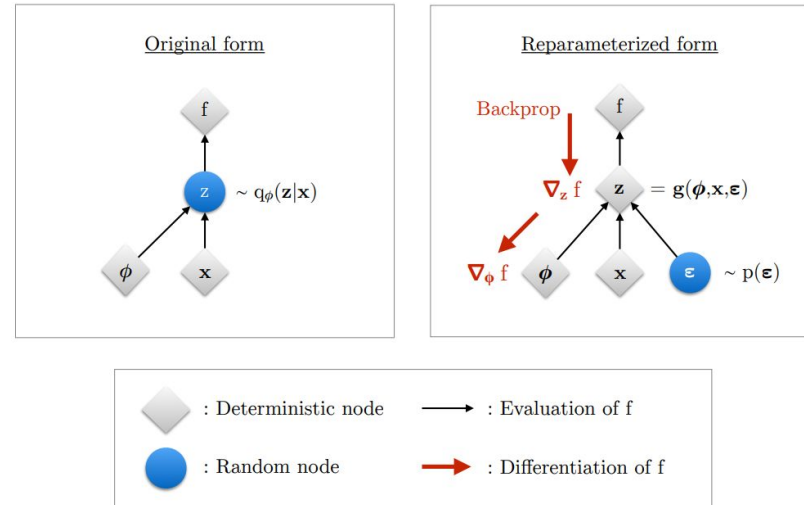
Number of latent dimensions

Penalizes high values of the encoded latent variable

# Reparameterization and stochastic backprop

Re-parameterization expresses the dependency of the sampled latent on the parameters as a function of both parameters and a fixed random variable.

$$z(x, \Psi) = \mu(x; \Psi) + \sigma(x; \Psi)\epsilon$$



Paper: <https://arxiv.org/pdf/1906.02691.pdf>

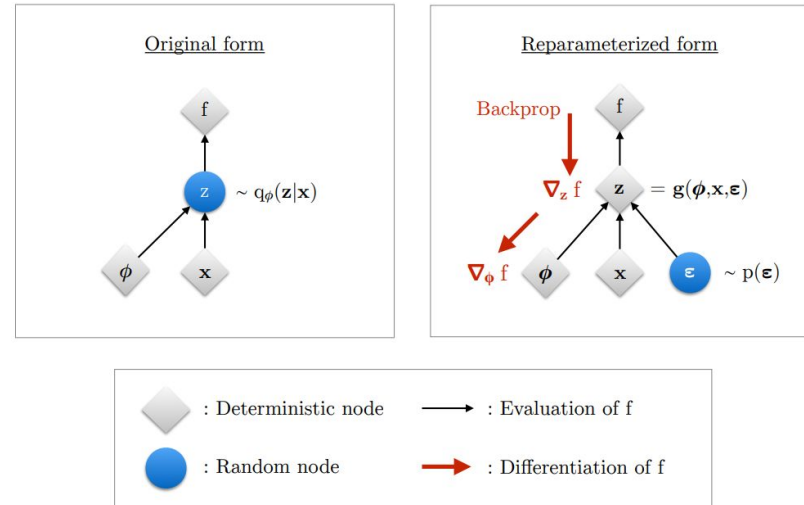


# Reparameterization and stochastic backprop

Using the formula, we can compute the gradients using regular automatic differentiation (back-prop).

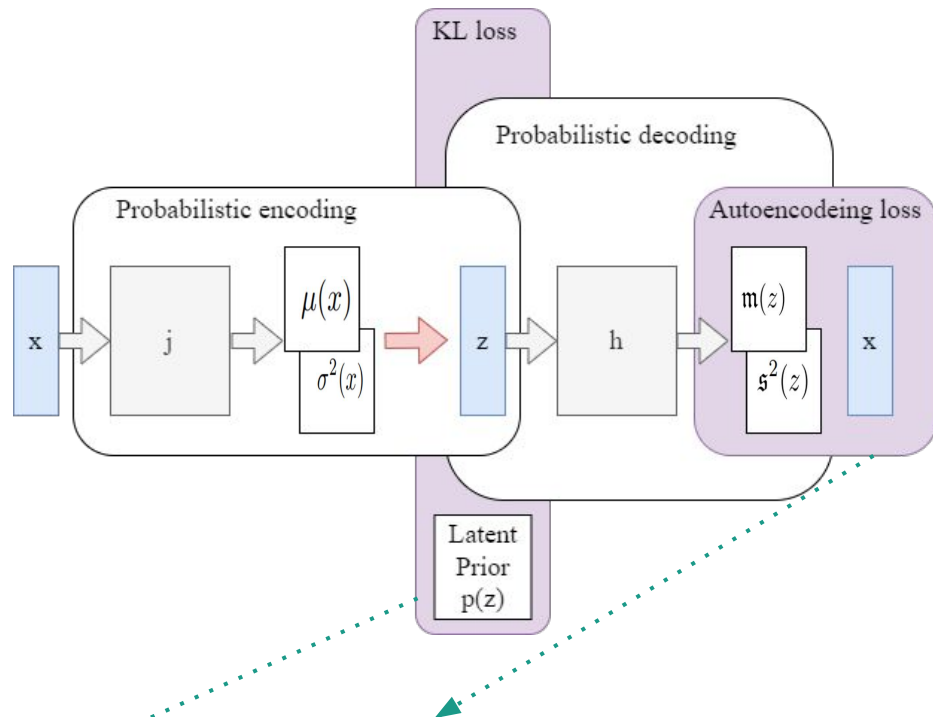
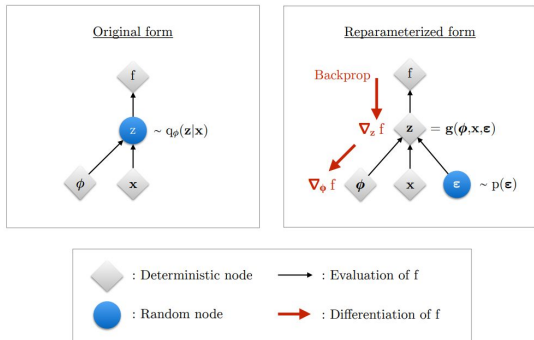
**Random input**

$$\nabla_{\Psi} z(x, \Psi) = \nabla_{\Psi} \mu(x; \Psi) + \nabla_{\Psi} \sigma(x; \Psi) \epsilon$$



Paper: <https://arxiv.org/pdf/1906.02691.pdf>

# Putting it all together



$$\mathcal{L}(\Psi, \Phi) = J\mathbb{E}_{x \sim \text{dataset}} \left[ \left( \mathbb{E}_{z(x; \Psi) \sim q(z|x; \Psi)} \left[ \sum_{i=1}^M \frac{1}{2\mathfrak{s}_i^2(z(x; \Psi); \Phi)} (m_i(z(x; \Psi); \Phi) - x_i)^2 \right] + \frac{1}{2} \log \mathfrak{s}_i(z(x; \Psi); \Phi) \right) \right] + J\mathbb{E}_{x \sim \text{dataset}} [D_{\text{KL}}(q(z | x; \Psi), p(z))]$$



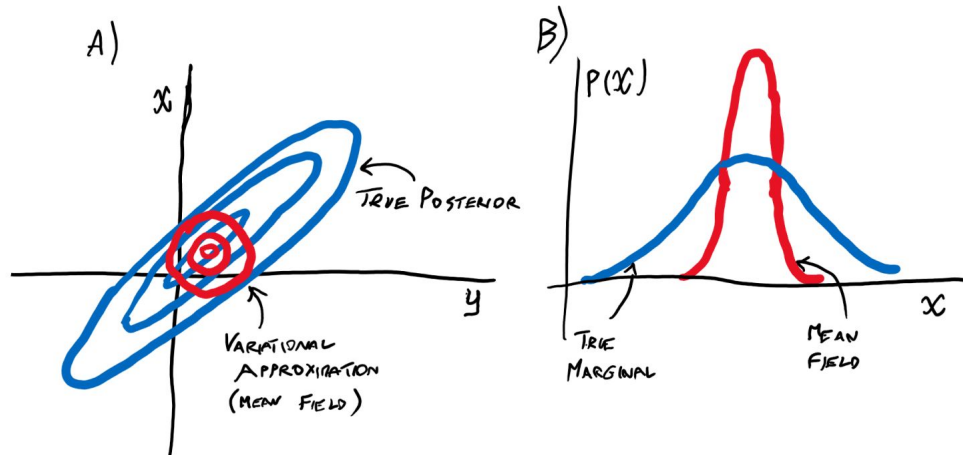
# Part II: Beyond mean-field

Dr. Luca Ambrogioni



## Problems with mean-field

The mean field approximation tends to underestimate the true variance of the posterior.



## Full covariance encoders

The mean field approximation can be a poor model of the posterior and therefore lead to a loose bound (high bias). We can improve performance using a non-spherical multivariate normal.

$$q(z \mid x) = \mathcal{N}\left(z; \mu(x), \underbrace{L(x)^T L(x)}_{\Sigma(x)}\right)$$

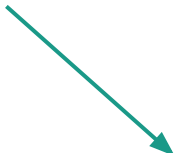
**Lower Triangular NxN matrix**

**Covariance matrix**



## Full covariance encoders: Multivariate reparameterization formula

Mean vector


$$z = \mu(x) + L(x)^T \epsilon$$



Lower Triangular NxN matrix



## A fully connected full covariance encoder

Converts vector to lower triangular matrix



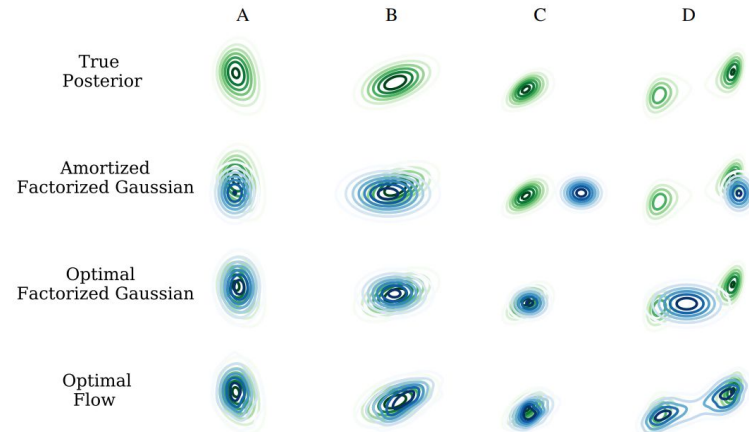
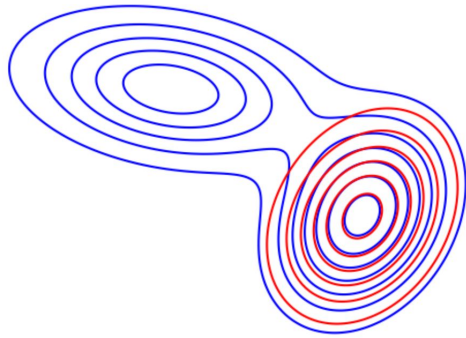
$$j(x) = \text{ReLu}(Hx + d)$$

$$\mu(x) = Jj(x) + v, \quad L(x) = \text{Trig}(Uj(z) + w)$$

$$p(x \mid z) = \mathcal{N}(x; \mu(x), L(x)^T L(x))$$

# Non-Gaussian encoders

The Gaussianity assumption is restrictive. We can have non-Gaussian encoders using advanced methods such as autoregressive nets and normalizing flows. We will discuss these approaches in future lectures!



Paper: <https://arxiv.org/pdf/1801.03558.pdf>





# Part III: Learning the prior

Dr. Luca Ambrogioni





## The empirical prior

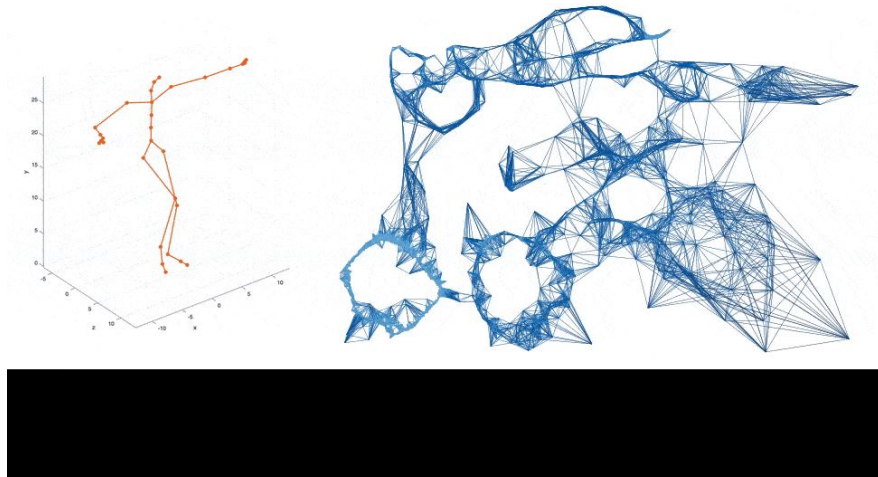
By mapping all data points to their (posterior) latent variables, we can get an idea of the (empirical) prior associated to a trained VAE.

$$p_{\text{empirical}}(z) = \mathbb{E}_{x \sim \text{dataset}} [q(z \mid x)]$$



## Exploring the empirical prior

The resulting empirical prior is usually very different from a simple Gaussian!



Source: <https://argmax.ai/blog/vhp-vae/>

## Learning the prior

Therefore, it makes sense to learn the structure of the prior from the data.  
We can do that by minimizing the negative ELBO as usual!

$$\mathcal{L}(\Psi, \Phi, \Lambda) = J\mathbb{E}_{x \sim \text{dataset}} \left[ \left( \mathbb{E}_{z(x; \Psi) \sim q(z|x; \Psi)} \left[ \sum_{i=1}^M \frac{1}{2\mathfrak{s}_i^2(z(x; \Psi); \Phi)} (\mathfrak{m}_i(z(x; \Psi); \Phi) - x_i)^2 \right] + \frac{1}{2} \log \mathfrak{s}_i(z(x; \Psi); \Phi) \right) \right. \\ \left. + J\mathbb{E}_{x \sim \text{dataset}} [D_{\text{KL}}(q(z | x; \Psi), p(z; \Lambda))] \right]$$

**Prior (hyper-)parameters**



## The empirical prior as optimized prior

It is easy to show that the ELBO is optimized by the empirical prior!

$$p_{\text{empirical}}(z) = \mathbb{E}_{x \sim \text{dataset}} [q(z \mid x)]$$

Source: <https://arxiv.org/pdf/1705.07120.pdf>



## Problems with the empirical prior

Using the empirical prior is not a good idea for two reasons:

- It tends to overfit the training set as it basically just reproduces the encoded data
- It is cumbersome to compute since it is averaged over all the training set.

Source: <https://arxiv.org/pdf/1705.07120.pdf>



## The VampPrior

The VampPrior replicates the form of the empirical prior but it averages over a smaller number of learnable (pseudo-)data-points.

$$p_{\text{vamp}}(z; \Lambda) = \frac{1}{U} \sum_{u=1}^U q(z \mid x_u(\Lambda); \Psi)$$



**Learnable fake data-point**

Source: <https://arxiv.org/pdf/1705.07120.pdf>



# Part IV: Modified VAE loss functions

Dr. Luca Ambrogioni





## β-VAEs: Simple, useful and ugly

In practice, it is useful to add a parameter that regulates the relative contribution of likelihood and KL divergence in the loss. This can lead to better training and more disentangled (i.e. interpretable) latent codes.

$$\mathcal{L}_\beta(\Psi, \Phi) = J \mathbb{E}_{x \sim \text{dataset}} \left[ \left( \mathbb{E}_{z(x; \Psi) \sim q(z|x; \Psi)} \left[ \sum_{i=1}^M \frac{1}{2 \mathfrak{s}_i^2(z(x; \Psi); \Phi)} (\mathfrak{m}_i(z(x; \Psi); \Phi) - x_i)^2 \right] + \frac{1}{2} \log \mathfrak{s}_i(z(x; \Psi); \Phi) \right) \right] + \beta J \mathbb{E}_{x \sim \text{dataset}} [D_{\text{KL}}(q(z|x; \Psi), p(z))]$$

The parameter beta can be determined by cross-validation.

Paper: <https://openreview.net/pdf?id=Sy2fzU9gl>

## The variance/bias trade-off

In the previous lecture, we saw that the ELBO is a lower bound of the log-marginal likelihood, which is the true loss we would like to use to train the decoder.

**Importance sampling: Unbiased,  
high variance**

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

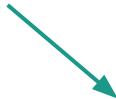
**ELBO: Biased, low variance**



## IWAEs: Importance weighted autoencoders

IWAEs use a loss that interpolates between importance sampling and the ELBO.

**Importance weighted ELBO: Less  
biased, higher variance**


$$\log p(x) \geq \mathbb{E}_{z^{(1)}, \dots, z^{(K)} \sim \text{iid } q(z|x)} \left[ \log \frac{1}{K} \sum_{k=1}^K \frac{p(x | z^{(k)}) p(z^{(k)})}{q(z^{(k)} | x)} \right] = -\mathcal{L}_K$$



## IWAEs: Importance weighted autoencoders II

- For  $K = 1$ , the importance weighted ELBO is just the regular ELBO.
- For  $K$  tending to infinity, the importance weighted ELBO converge to an unbiased importance sampling estimator.

$$\lim_{K \rightarrow \infty} \mathbb{E}_{z^{(1)}, \dots, z^{(K)} \sim \text{iid } q(z|x)} \left[ \log \frac{1}{K} \sum_{k=1}^K \frac{p(x | z^{(k)})p(z^{(k)})}{q(z^{(k)} | x)} \right] = \log \mathbb{E}_{z \sim q(z|x)} \left[ \frac{p(x | z)p(z)}{q(z | x)} \right]$$

Paper: <https://arxiv.org/pdf/1509.00519.pdf>

## IWAEs: Importance weighted autoencoders III

- In practice,  $K$  should not be too big in order to keep the variance low
- The IE-ELBO should only be used to train the decoder since for high values of  $K$  the loss becomes independent of the encoder.

$$\lim_{K \rightarrow \infty} \mathbb{E}_{z^{(1)}, \dots, z^{(K)} \sim \text{iid } q(z|x)} \left[ \log \frac{1}{K} \sum_{k=1}^K \frac{p(x | z^{(k)}) p(z^{(k)})}{q(z^{(k)} | x)} \right] = \log \mathbb{E}_{z \sim q(z|x)} \left[ \frac{p(x | z) p(z)}{q(z | x)} \right]$$

Paper: <https://arxiv.org/pdf/1509.00519.pdf>

**Does not depend on  $q$ !**



# Thank you.

