Fundamentals of Probabilistic Data Mining

Chapter VI - Variational Autoencoders (VAE)

Xavier Alameda-Pineda

Why does the EM work?

The main mathematical object in EM is $\mathcal{Q}$.
(The expected complete-data log-likelihood).

What is the relationship with the log-likelihood?
Let's take any distribution of $\boldsymbol{z}$: $q(\boldsymbol{z})$ and ignore $\boldsymbol{\Theta}$ for the time being.

$$\log p(\boldsymbol{x}) = \mathbb{E}_{q(\boldsymbol{z})}\Big\{ \log p(\boldsymbol{x}) \Big\} \tag{1}$$

$$= \mathbb{E}_{q(\boldsymbol{z})}\Big\{ \log p(\boldsymbol{x}) \frac{p(\boldsymbol{z}|\boldsymbol{x})q(\boldsymbol{z})}{p(\boldsymbol{z}|\boldsymbol{x})q(\boldsymbol{z})} \Big\} \tag{2}$$

$$= \mathbb{E}_{q(\boldsymbol{z})}\Big\{ \log \frac{p(\boldsymbol{x},\boldsymbol{z})}{q(\boldsymbol{z})} \Big\} + D_{\mathrm{KL}}\Big(q(\boldsymbol{z})\Big\|p(\boldsymbol{z}|\boldsymbol{x})\Big) \tag{3}$$

Why does the EM work? (II)

$$\log p(\boldsymbol{x};\boldsymbol{\Theta}) = \underbrace{\mathbb{E}_{q(\boldsymbol{z})}\Big\{\log\frac{p(\boldsymbol{x},\boldsymbol{z})}{q(\boldsymbol{z})}\Big\}}_{\text{M-step}} + \underbrace{D_{\mathrm{KL}}\Big(q(\boldsymbol{z})\Big\|p(\boldsymbol{z}|\boldsymbol{x})\Big)}_{\text{E-step}} \tag{4}$$

Another interpretation. Given $\bar{\boldsymbol{\Theta}}$:

1. Set $q(\boldsymbol{z}) = p(\boldsymbol{z}|\boldsymbol{x};\bar{\boldsymbol{\Theta}})$.
2. Optimise w.r.t. $\boldsymbol{\Theta}$:

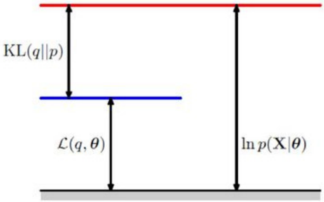
$$\mathbf{E}_{q(\boldsymbol{z})}\Big\{\log\frac{p(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\Theta})}{q(\boldsymbol{z})}\Big\} \tag{5}$$

Why?

E-step: reduce the distance between log-likelihood and $\mathcal{Q}$.
M-step: push $\mathcal{Q}$ and therefore push the log-likelihood.

Why does the EM work? (III)

$$\log p(\boldsymbol{x};\boldsymbol{\Theta}) = \underbrace{\mathbb{E}_{q(\boldsymbol{z})}\Big\{\log \frac{p(\boldsymbol{x},\boldsymbol{z})}{q(\boldsymbol{z})}\Big\}}_{\mathcal{L}(q,\boldsymbol{\Theta})} + \underbrace{D_{\mathrm{KL}}\Big(q(\boldsymbol{z})\Big\|p(\boldsymbol{z}|\boldsymbol{x})\Big)}_{\mathrm{KL}(q\|p)} \tag{6}$$



Crucial point in “Exact EM”

We **need** the exact a posteriori distribution: $p(\boldsymbol{z}|\boldsymbol{x};\bar{\boldsymbol{\Theta}})$.

What happens if we cannot use the exact posterior? **Approximate it**.

Two big families:

- $p(\boldsymbol{z}|\boldsymbol{x};\bar{\boldsymbol{\Theta}})$ has an analytic expression, but computationally too heavy.
- $p(\boldsymbol{z}|\boldsymbol{x};\bar{\boldsymbol{\Theta}})$ does not have an analytic expression.

We will focus in the second case, and in a model called **Variational Autoencoders (VAE)**.

VAE Motivation: back to PPCA

Recall the definition of PPCA:

- $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$,
- $\boldsymbol{x}|\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{x}; \boldsymbol{A}\boldsymbol{z} + \boldsymbol{b}, \sigma^2 \boldsymbol{I})$, $\sigma > 0$.

Important limitations:

- The dependency of the mean with $\boldsymbol{z}$ is **afinne**.
- The covariance does not depend on $\boldsymbol{z}$.

Non-linear generative model:

- $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$,
- $\boldsymbol{x}|\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_{\boldsymbol{\Theta}}(\boldsymbol{z}), \boldsymbol{\Sigma}_{\boldsymbol{\Theta}}(\boldsymbol{z}))$,

where $\boldsymbol{\mu}_{\boldsymbol{\Theta}}(\boldsymbol{z})$ and $\boldsymbol{\Sigma}_{\boldsymbol{\Theta}}(\boldsymbol{z})$ are (non-linear) functions parametrised by $\boldsymbol{\Theta}$.

Formalising the generative model (I)

The generative model:

- $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$,
- $\boldsymbol{x}|\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_{\boldsymbol{\Theta}}(\boldsymbol{z}), \boldsymbol{\Sigma}_{\boldsymbol{\Theta}}(\boldsymbol{z}))$,

where $\boldsymbol{\mu}_{\boldsymbol{\Theta}}(\boldsymbol{z})$ and $\boldsymbol{\Sigma}_{\boldsymbol{\Theta}}(\boldsymbol{z})$ will be implemented **by deep neural networks** parametrised by $\boldsymbol{\Theta}$ with input $\boldsymbol{z}$.

Formalising the generative model (I)

The generative model:

- $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$,
- $\boldsymbol{x}|\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_{\boldsymbol{\Theta}}(\boldsymbol{z}), \boldsymbol{\Sigma}_{\boldsymbol{\Theta}}(\boldsymbol{z}))$,

where $\boldsymbol{\mu}_{\boldsymbol{\Theta}}(\boldsymbol{z})$ and $\boldsymbol{\Sigma}_{\boldsymbol{\Theta}}(\boldsymbol{z})$ will be implemented **by deep neural networks** parametrised by $\boldsymbol{\Theta}$ with input $\boldsymbol{z}$.

A few comments:

1. The optimal parameters $\boldsymbol{\Theta}^*$ need to maximise the log-likelihood.
2. $\boldsymbol{\Theta}$ cannot be estimated in closed-form.
3. $\boldsymbol{\mu}_{\boldsymbol{\Theta}}(\boldsymbol{z})$ and $\boldsymbol{\Sigma}_{\boldsymbol{\Theta}}(\boldsymbol{z})$ are differentiable w.r.t. $\boldsymbol{\Theta}$, and $\boldsymbol{z}$.
4. $\boldsymbol{\Sigma}_{\boldsymbol{\Theta}}(\boldsymbol{z})$ needs to be a covariance matrix.

Formalising the generative model (II)

How can we ensure that $\mathbf{\Sigma}_{\mathbf{\Theta}}(\boldsymbol{z})$ is a covariance matrix?

- The covariance matrix is assumed to be diagonal:

$$\mathbf{\Sigma}_{\mathbf{\Theta}}(\boldsymbol{z}) = \begin{pmatrix} \nu_{\mathbf{\Theta}}^{(1)}(\boldsymbol{z}) & 0 & \cdots & 0 \\ 0 & \nu_{\mathbf{\Theta}}^{(2)}(\boldsymbol{z}) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \nu_{\mathbf{\Theta}}^{(D)}(\boldsymbol{z}) \end{pmatrix} \tag{7}$$

  Reduces complexity and memory, but also expressivity.

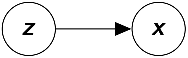
- We estimate the log-variance: $\eta_{\mathbf{\Theta}}^{(d)}(\boldsymbol{z}) = \log \nu_{\mathbf{\Theta}}^{(d)}(\boldsymbol{z})$:

$$\mathbf{\Sigma}_{\mathbf{\Theta}}(\boldsymbol{z}) = \mathrm{diag}_d \left(\exp \left(\eta_{\mathbf{\Theta}}^{(d)}(\boldsymbol{z}) \right) \right) \tag{8}$$

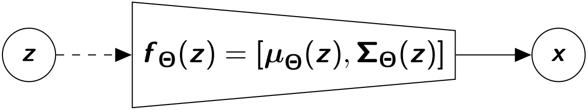
  The values of $\eta_{\mathbf{\Theta}}^{(d)}(\boldsymbol{z})$ can be positive or negative.

Formalising the generative model (III)

In terms of probabilistic dependencies, they are the same as PPCA:



But I would like to draw also the non-leariry:



The dependency of the parameters w.r.t. $\boldsymbol{z}$ is **deterministic**.

Denoted by $\boldsymbol{f}_{\boldsymbol{\Theta}}(\boldsymbol{z}) : \mathbb{R}^{d_z} \rightarrow \mathbb{R}^{2d_x}$, this non-linearity is implemented with a deep network, with parameters (weights and biases) $\boldsymbol{\Theta}$.

The posterior distribution

For any EM-like procedure, we would need the posterior distribution:

$$p(\boldsymbol{z}|\boldsymbol{x}) \overset{(\boldsymbol{z})}{\propto} p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z}) \tag{9}$$

$$\overset{(\boldsymbol{z})}{\propto} \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_{\boldsymbol{\Theta}}(\boldsymbol{z}), \boldsymbol{\Sigma}_{\boldsymbol{\Theta}}(\boldsymbol{z}))\mathcal{N}(\boldsymbol{z}; \mathbf{0}, \boldsymbol{I}) \tag{10}$$

$$\overset{(\boldsymbol{z})}{\propto} \frac{1}{|\boldsymbol{\Sigma}_{\boldsymbol{\Theta}}(\boldsymbol{z})|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_{\boldsymbol{\Theta}}(\boldsymbol{z}))^\top \boldsymbol{\Sigma}_{\boldsymbol{\Theta}}^{-1}(\boldsymbol{z})(\boldsymbol{x} - \boldsymbol{\mu}_{\boldsymbol{\Theta}}(\boldsymbol{z})) - \frac{1}{2}\|\boldsymbol{z}\|^2\right) \tag{11}$$

The posterior distribution

For any EM-like procedure, we would need the posterior distribution:

$$p(\boldsymbol{z}|\boldsymbol{x}) \overset{(\boldsymbol{z})}{\propto} p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z}) \tag{9}$$

$$\overset{(\boldsymbol{z})}{\propto} \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_{\boldsymbol{\Theta}}(\boldsymbol{z}), \boldsymbol{\Sigma}_{\boldsymbol{\Theta}}(\boldsymbol{z}))\mathcal{N}(\boldsymbol{z}; \boldsymbol{0}, \boldsymbol{I}) \tag{10}$$

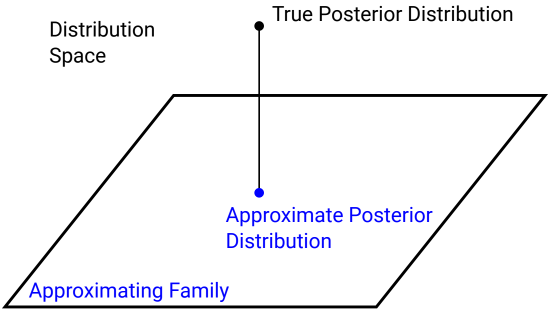
$$\overset{(\boldsymbol{z})}{\propto} \frac{1}{|\boldsymbol{\Sigma}_{\boldsymbol{\Theta}}(\boldsymbol{z})|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_{\boldsymbol{\Theta}}(\boldsymbol{z}))^\top \boldsymbol{\Sigma}_{\boldsymbol{\Theta}}^{-1}(\boldsymbol{z})(\boldsymbol{x} - \boldsymbol{\mu}_{\boldsymbol{\Theta}}(\boldsymbol{z})) - \frac{1}{2}\|\boldsymbol{z}\|^2\right) \tag{11}$$

We cannot go our “standard” way, because we cannot identify a distribution on $\boldsymbol{z}$.

The posterior distribution cannot be computed analytically!!!

Approximating the posterior distribution (I)

The posterior distribution needs to be approximated. We will propose a family of distributions, and find the **best candidate within this family**.



Approximating the posterior distribution (II)

The posterior distribution will be approximated with **another** feed-forward network parametrised with $\boldsymbol{\Phi}$:

$$p(\boldsymbol{z}|\boldsymbol{x}) \approx q(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}; \tilde{\boldsymbol{\mu}}_{\boldsymbol{\Phi}}(\boldsymbol{x}), \tilde{\boldsymbol{\Sigma}}_{\boldsymbol{\Phi}}(\boldsymbol{x})) \tag{12}$$

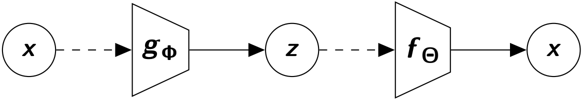
The approximating family is composed of all the distributions that can be expressed as above, for a certain value of $\boldsymbol{\Phi}$.

$$\mathcal{G} = \{\boldsymbol{g}_{\boldsymbol{\Phi}} : \mathbb{R}^{d_x} \rightarrow \mathbb{R}^{2d_z}; \boldsymbol{\Phi} \in \boldsymbol{\Phi}\}, \tag{13}$$

with $\boldsymbol{g}_{\boldsymbol{\Phi}}(\boldsymbol{x}) = [\tilde{\boldsymbol{\mu}}_{\boldsymbol{\Phi}}(\boldsymbol{x}), \tilde{\boldsymbol{\Sigma}}_{\boldsymbol{\Phi}}(\boldsymbol{x})]$.

Overall architecture

If we “chain” the posterior and the generative model:



- The generative model is also called the **decoder**.
- The inference or posterior is also called the **encoder**.

This is why we call these architectures **variational autoencoders** VAE.

But how do we optimise for the parameters $\boldsymbol{\Theta}$ and $\boldsymbol{\Phi}$?

Learning - ELBO

If we recall the formulation for the EM:

$$\log p(\boldsymbol{x}) = \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}\left\{ \log \frac{p(\boldsymbol{x}, \boldsymbol{z})}{q(\boldsymbol{z}|\boldsymbol{x})} \right\} + D_{\mathrm{KL}}\left(q(\boldsymbol{z}|\boldsymbol{x}) \middle\| p(\boldsymbol{z}|\boldsymbol{x}) \right) \tag{14}$$

Problem: the second term cannot be computed! But it's positive:

$$\log p(\boldsymbol{x}; \boldsymbol{\Theta}, \boldsymbol{\Phi}) \geq \mathbb{E}_{q_{\boldsymbol{\Phi}}(\boldsymbol{z}|\boldsymbol{x})}\left\{ \log \frac{p(\boldsymbol{x}, \boldsymbol{z})}{q_{\boldsymbol{\Phi}}(\boldsymbol{z}|\boldsymbol{x})} \right\} \tag{15}$$

$$\log p(\boldsymbol{x}; \boldsymbol{\Theta}, \boldsymbol{\Phi}) \geq \underbrace{\mathbb{E}_{q_{\boldsymbol{\Phi}}(\boldsymbol{z}|\boldsymbol{x})}\left\{ \log p_{\boldsymbol{\Theta}}(\boldsymbol{x}|\boldsymbol{z}) \right\}}_{\text{Reconstruction}} - \underbrace{D_{\mathrm{KL}}\left(q_{\boldsymbol{\Phi}}(\boldsymbol{z}|\boldsymbol{x}) \middle\| p(\boldsymbol{z}) \right)}_{\text{Regularisation}} \tag{16}$$

This is known as **Evidence Lower-BOund or ELBO**: $\mathcal{L}_{\mathrm{ELBO}}(\boldsymbol{\Theta}, \boldsymbol{\Phi})$.

Be VERY careful with these expressions.
They look alike, but they are NOT the same.

Learning - Sampling

But we still have one problem:

$$\mathcal{L}_{\text{ELBO}}(\boldsymbol{\Theta}, \boldsymbol{\Phi}) = \underbrace{\mathbb{E}_{q_{\boldsymbol{\Phi}}(\boldsymbol{z}|\boldsymbol{x})}\Big\{ \log p_{\boldsymbol{\Theta}}(\boldsymbol{x}|\boldsymbol{z}) \Big\}}_{\text{Reconstruction}} - \underbrace{D_{\text{KL}}\Big(q_{\boldsymbol{\Phi}}(\boldsymbol{z}|\boldsymbol{x}) \Big\| p(\boldsymbol{z}) \Big)}_{\text{Regularisation}} \tag{17}$$

To compute the "reconstruction" term we need to take the expectation w.r.t. $q_{\boldsymbol{\Phi}}(\boldsymbol{z}|\boldsymbol{x})$, but recall that:

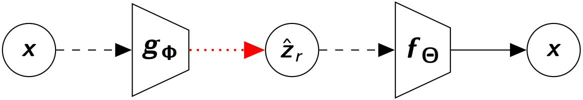
$$p_{\boldsymbol{\Theta}}(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_{\boldsymbol{\Theta}}(\boldsymbol{z}), \boldsymbol{\Sigma}_{\boldsymbol{\Theta}}(\boldsymbol{z})). \tag{18}$$

Due to the non-linearity, we cannot compute the reconstruction term in closed form $\rightarrow$ we sample R points $\hat{\boldsymbol{z}}_1, \ldots, \hat{\boldsymbol{z}}_R$ from $q_{\boldsymbol{\Phi}}$:

$$\mathcal{L}_{\text{ELBO}}(\boldsymbol{\Theta}, \boldsymbol{\Phi}) = \underbrace{\frac{1}{R} \sum_{r=1}^{R} \log p_{\boldsymbol{\Theta}}(\boldsymbol{x}|\hat{\boldsymbol{z}}_r)}_{\text{Reconstruction}} - \underbrace{D_{\text{KL}}\Big(q_{\boldsymbol{\Phi}}(\boldsymbol{z}|\boldsymbol{x}) \Big\| p(\boldsymbol{z}) \Big)}_{\text{Regularisation}} \tag{19}$$

Learning - Gradient ascent?

Let's go back to the architecture:



where dashed lines are deterministic, dotted lines are sampling (we will see later for the solid line).

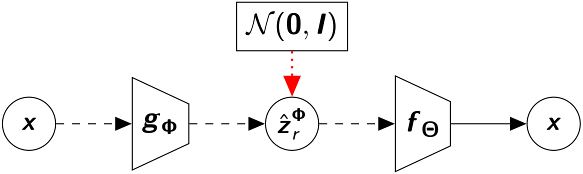
We agreed that there is no closed-form solution for the parameters (due to non-linearity). We will learn the parameters using stochastic gradient ascent (to maximise the ELBO).

We assume that $\boldsymbol{g}_{\boldsymbol{\Phi}}$ and $\boldsymbol{f}_{\boldsymbol{\Theta}}$ are differentiable (to compute the gradient).

The sampling operation from $q_{\boldsymbol{\Phi}}$ is NOT differentiable w.r.t. $\boldsymbol{\Phi}$.

Learning - Reparametrisation trick

We use the so-called **reparametrisation trick**:



Formally ($\hat{\boldsymbol{z}}_r^{\boldsymbol{\Phi}}$ denotes explicitly the dependency on $\boldsymbol{\Phi}$):

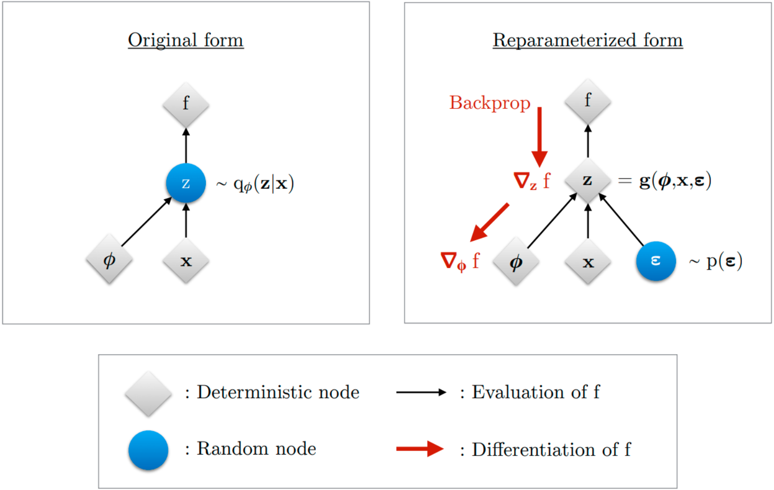
$$\hat{\boldsymbol{z}}_r^{\boldsymbol{\Phi}} = \tilde{\boldsymbol{\Sigma}}_{\boldsymbol{\Phi}}^{1/2} \hat{\boldsymbol{\epsilon}}_r + \tilde{\boldsymbol{\mu}}_{\boldsymbol{\Phi}} \quad \text{with} \quad \hat{\boldsymbol{\epsilon}}_r \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}) \tag{20}$$

So we sample from a standard Gaussian, and use the parameters $\tilde{\boldsymbol{\mu}}_{\boldsymbol{\Phi}}$ and $\tilde{\boldsymbol{\Sigma}}_{\boldsymbol{\Phi}}$ in **differentiable operations** (multiplication and addition).

If the last arrow is differentiable, then we can use gradient ascent.

Learning - Reparametrisation trick (II)

Another way to see the reparametrisation trick (from "An Introduction to Variational Autoencoders" by Diederik P. Kingma, Max Welling, `chamilo`):



Learning - The loss

We are now ready to write the loss:

$$\mathcal{L}_{\text{ELBO}}(\mathbf{\Theta}, \mathbf{\Phi}) = \underbrace{\mathbb{E}_{q_{\mathbf{\Phi}}(\boldsymbol{z}|\boldsymbol{x})}\Big\{ \log p_{\mathbf{\Theta}}(\boldsymbol{x}|\boldsymbol{z}) \Big\}}_{\text{Reconstruction}} - \underbrace{D_{\text{KL}}\Big(q_{\mathbf{\Phi}}(\boldsymbol{z}|\boldsymbol{x}) \Big\| p(\boldsymbol{z})\Big)}_{\text{Regularisation}} \tag{21}$$

$$= \underbrace{\sum_{r=1}^{R} \log p_{\mathbf{\Theta}}(\boldsymbol{x}|\hat{\boldsymbol{z}}_r^{\mathbf{\Phi}})}_{\text{Reconstruction}} - \underbrace{D_{\text{KL}}\Big(q_{\mathbf{\Phi}}(\boldsymbol{z}|\boldsymbol{x}) \Big\| p(\boldsymbol{z})\Big)}_{\text{Regularisation}} \tag{22}$$

$$\overset{(\mathbf{\Theta},\mathbf{\Phi})}{=} -\frac{1}{2}\Bigg[\frac{1}{R}\sum_{r=1}^{R}\Big(\log|\mathbf{\Sigma}_{\mathbf{\Theta}}(\hat{\boldsymbol{z}}_r^{\mathbf{\Phi}})| + \|\boldsymbol{x} - \boldsymbol{\mu}_{\mathbf{\Theta}}(\hat{\boldsymbol{z}}_r^{\mathbf{\Phi}})\|^2_{\mathbf{\Sigma}_{\Theta}(\hat{\boldsymbol{z}}_r^{\mathbf{\Phi}})}\Big) \tag{23}$$

$$+ \operatorname{Tr}(\mathbf{\Sigma}_{\mathbf{\Phi}}(\boldsymbol{x})) + \|\boldsymbol{\mu}_{\mathbf{\Phi}}(\boldsymbol{x})\|^2 - \log|\mathbf{\Sigma}_{\mathbf{\Phi}}(\boldsymbol{x})|\Bigg] \tag{24}$$

Where (23) and (24) are the reconstruction and regularisation terms resp.
Homework: use the definition of the terms above to prove that.

Learning - The loss (II)

$$\mathcal{L}_{\text{ELBO}}(\boldsymbol{\Theta}, \boldsymbol{\Phi}) \overset{(\boldsymbol{\Theta},\boldsymbol{\Phi})}{=} -\frac{1}{2}\left[\frac{1}{R}\sum_{r=1}^{R}\left(\log|\boldsymbol{\Sigma}_{\boldsymbol{\Theta}}(\hat{\boldsymbol{z}}_r^{\boldsymbol{\Phi}})| + \|\boldsymbol{x} - \boldsymbol{\mu}_{\boldsymbol{\Theta}}(\hat{\boldsymbol{z}}_r^{\boldsymbol{\Phi}})\|^2_{\boldsymbol{\Sigma}_{\boldsymbol{\Theta}}(\hat{\boldsymbol{z}}_r^{\boldsymbol{\Phi}})}\right)\right. \tag{25}$$

$$\left. + \operatorname{Tr}(\boldsymbol{\Sigma}_{\boldsymbol{\Phi}}(\boldsymbol{x})) + \|\boldsymbol{\mu}_{\boldsymbol{\Phi}}(\boldsymbol{x})\|^2 - \log|\boldsymbol{\Sigma}_{\boldsymbol{\Phi}}(\boldsymbol{x})|\right] \tag{26}$$

Comments:

- We recall that $\hat{\boldsymbol{z}}_r^{\boldsymbol{\Phi}} = \tilde{\boldsymbol{\Sigma}}_{\boldsymbol{\Phi}}^{1/2}\hat{\epsilon}_r + \tilde{\boldsymbol{\mu}}_{\boldsymbol{\Phi}}$ with $\hat{\boldsymbol{\epsilon}}_r \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$.
- We remark that all operators are differentiable w.r.t. $\boldsymbol{\Theta}$ and $\boldsymbol{\Phi}$.
- If we remove the "$-\frac{1}{2}$" we use gradient descent.
- The term in blue is the Mahalanobis distance and can be replaced...

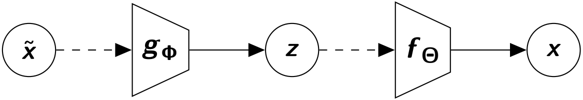
Other reconstruction possibilities

Often, we forget about the covariance matrix of the generative model $\boldsymbol{\Sigma_\Theta}$ and use other distances rather than Mahalanobis:

- The Euclidean distance (equivalent to set $\boldsymbol{\Sigma_\Theta} = \boldsymbol{I}$): $\|\boldsymbol{x} - \boldsymbol{\mu_\Theta}(\hat{\boldsymbol{z}}_r^{\boldsymbol{\Phi}})\|_2^2$
- The L_1 distance: $\|\boldsymbol{x} - \boldsymbol{\mu_\Theta}(\hat{\boldsymbol{z}}_r^{\boldsymbol{\Phi}})\|_1$
- ...

In that case $\boldsymbol{f_\Theta}(\boldsymbol{z}) : \mathbb{R}^{d_z} \to \mathbb{R}^{d_x}$ (instead of $\mathbb{R}^{2d_x}$), and this links to the deterministic autoencoders.

In addition, we can attempt to reconstruct $\boldsymbol{x}$ from another signal $\tilde{\boldsymbol{x}}$:



a clear example are denoising VAE ($\tilde{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{b}$, with $\boldsymbol{b}$ being noise).

Differences w.r.t. EM

[EM]: Start with $\bar{\boldsymbol{\Theta}}$:

- E-step: Compute $p(\boldsymbol{z}_n | \boldsymbol{x}_n; \bar{\boldsymbol{\Theta}})$, $\forall n$.
- M-step: Compute $\boldsymbol{\Theta}^*$, and set $\bar{\boldsymbol{\Theta}}$ to that.

(Until convergence)

[SGD]: Start with $\bar{\boldsymbol{\Theta}}$. Initialise also $\bar{\boldsymbol{\Phi}}$:

- Forward: Compute $\boldsymbol{g}_{\bar{\boldsymbol{\Phi}}}(\boldsymbol{x}_n)$, sample $\boldsymbol{z}_n$, compute $\boldsymbol{f}_{\bar{\boldsymbol{\Theta}}}(\boldsymbol{z}_n)$, $\forall n$ in batch.
- Backward: Compute $\mathcal{L}_{\text{ELBO}}$, $\nabla_{\bar{\boldsymbol{\Theta}}}\mathcal{L}_{\text{ELBO}}$, and $\nabla_{\bar{\boldsymbol{\Phi}}}\mathcal{L}_{\text{ELBO}}$.
- Update $\bar{\boldsymbol{\Theta}}$ and $\bar{\boldsymbol{\Phi}}$ with your preferred gradient update rule.

(Until convergence)