Inference in latent variable models

Variational Bayes, expectation maximisation, and the variational auto-encoder

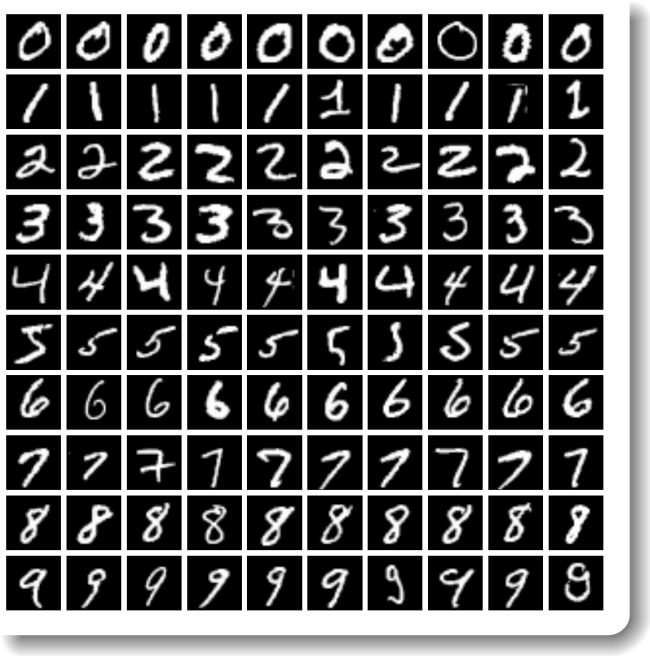
Maximilian Soelch

Technische Universität München

Recommended/Further Readings:

- Bishop, *PRML*, Chapters 9, 10.1
- Blei, David M., Alp Kucukelbir, and Jon D. McAuliffe. "Variational inference: A review for statisticians." arXiv preprint arXiv:1601.00670 (2016).
- Doersch, Carl. "Tutorial on variational autoencoders." arXiv preprint arXiv:1606.05908 (2016).
- This blog (for visuals).
- The papers mentioned throughout these slides.

Consider the following data:



MNIST handwritten digits
`http://deeplearning.net/data/mnist/`

We would like to do maximum likelihood:

$$\theta_{\mathrm{MLE}} = \arg\max_{\theta} p(\mathcal{D} \mid \theta)$$

What might be the underlying distribution $p(\mathbf{x} \mid \theta)$?

Our standard set of distributions is not expressive enough: 

There are *latent* (hidden) factors present.

They are not directly observable (for a machine):

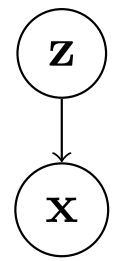
$$\begin{pmatrix}
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.184 & 0.117 & 0.992 & 0.051 & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0.988 & 0.988 & 0.629 & 0.988 & 0.992 & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0.543 & 0.871 & 0. & 0.141 & 0.992 & 0.414 & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0.027 & 0.988 & 0.312 & 0.988 & 0.426 & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0.988 & 0.988 & 0.699 & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0.668 & 0.988 & 0.988 & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0.363 & 0.992 & 0.051 & 0.988 & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0.992 & 0. & 0.422 & 0.992 & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0.988 & 0.75 & 0.988 & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0.461 & 0.746 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.
\end{pmatrix}$$

If we knew the latent factors, our lives would get easier:

$p(\mathbf{x} \mid \mathbf{x} \text{ shows a } 7)$ is easier!

This leads to the *graphical model* perspective.

$$
\begin{aligned}
p(\mathbf{x} \mid \theta) &= \int p(\mathbf{x}, \mathbf{z} \mid \theta) \, \mathrm{d}\mathbf{z} \\
&= \int p(\mathbf{x} \mid \theta) p(\mathbf{z} \mid \mathbf{x}, \theta) \, \mathrm{d}\mathbf{z} \\
&= \int p(\mathbf{z} \mid \theta) p(\mathbf{x} \mid \mathbf{z}, \theta) \, \mathrm{d}\mathbf{z}
\end{aligned}
$$



Interpretation: We do MLE on a model where some data is missing (e. g., the abstract digit).

A very simple model: $\mathbf{z} \in \{0, \ldots, 9\}$ indicates the digit.

- $p(\mathbf{z} = k) = \pi_k$, with $\pi_k \geq 0$, $\sum_k \pi_k = 1$
- $p(\mathbf{x} \mid \mathbf{z} = k) = \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$

In total, we get a Gaussian Mixture Model with 10 components:

$$p(\mathbf{x} \mid \theta) = \int p(\mathbf{x}, \mathbf{z} \mid \theta) \, \mathrm{d}\mathbf{z} = \sum_{k=0}^{9} \pi_k \, \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

$\theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k \mid k = 0, \ldots, 9\}$

But now:

$$\arg\max_{\theta} \ln p(\mathcal{D} \mid \theta) = \sum_{n=1}^{N} \ln \sum_{k=0}^{9} \pi_k \, \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

A sum (or an integral) *inside* the logarithm! No closed-form solution, no easy optimization.

Every component is a simple Gaussian distribution. If we knew $\mathbf{z}$, we could circumvent the inner integral and solve for each component. With $N_k = \sum_{n=1}^{N} \mathbb{1}(\mathbf{z}_n = k)$ (number of samples in class k):

$$\boldsymbol{\mu}_k^{\text{MLE}} = \frac{1}{N_k} \sum_{n=1}^{N} \mathbb{1}(\mathbf{z}_n = k)\mathbf{x}_n$$

$$\pi_k^{\text{MLE}} = \frac{N_k}{N}$$

The closest we can get is the posterior

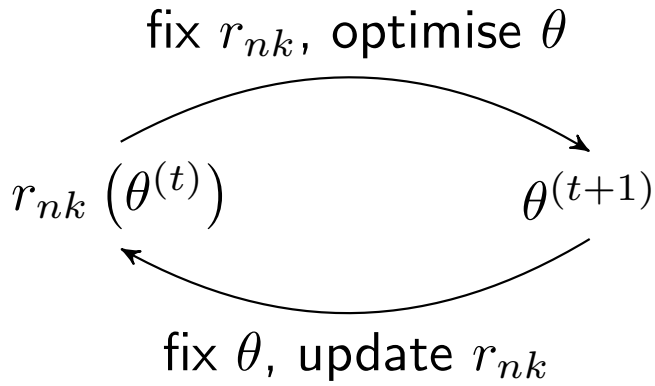
$$r_{nk}(\theta) \equiv p(\mathbf{z}_n = k \mid \mathbf{x}_n, \theta).$$

This closes a cycle:

- We need the r_{nk}'s to optimize for θ,
- and we need θ to compute the r_{nk}'s.

This makes it hard (if not impossible) to find closed-form solutions.

Idea: *Use* the cycle!



Problem: $r_{nk}\left(\theta^{(t)}\right)$ is not the MLE posterior value, which is $r_{nk}\left(\theta_{\mathrm{MLE}}\right)$.

Our hope would be: $r_{nk}\left(\theta^{(t)}\right) \xrightarrow{t\to\infty} r_{nk}\left(\theta_{\mathrm{MLE}}\right)$

To analyse this, it is beneficial to assume that we have an *oracle* $q(\mathbf{z})$. (In our case up there, the oracle has structure. This will be nice for implementation, but the subsequent analysis is more general.)

Something interesting happens:

$$\begin{aligned}\ln p(\mathbf{x} \mid \theta) &= \int q(\mathbf{z}) \ln \left(p(\mathbf{x} \mid \theta) \frac{q(\mathbf{z})}{q(\mathbf{z})} \right) \mathrm{d}\mathbf{z} \\ &= \underbrace{\int q(\mathbf{z}) \ln \frac{p(\mathbf{x}, \mathbf{z} \mid \theta)}{q(\mathbf{z})} \mathrm{d}\mathbf{z}}_{①} + \underbrace{\int q(\mathbf{z}) \ln \frac{q(\mathbf{z})}{p(\mathbf{z} \mid \mathbf{x}, \theta)} \mathrm{d}\mathbf{z}}_{②}\end{aligned}$$

② $= \mathrm{KL}(q(\mathbf{z}) \,||\, p(\mathbf{z} \mid \mathbf{x}, \theta)) \geq 0$ (and 0 *iff* $q(\mathbf{z}) \equiv p(\mathbf{z} \mid \mathbf{x}, \theta)$)

Since $\ln p(\mathbf{x} \mid \theta)$ is constant w. r. t. q, this implies ① $\leq \ln p(\mathbf{x} \mid \theta)$.

$$① = \int q(\mathbf{z}) \ln \frac{p(\mathbf{x}, \mathbf{z} \mid \theta)}{q(\mathbf{z})} \mathrm{d}\mathbf{z} = \mathbb{E}_{q(\mathbf{z})}[\ln p(\mathbf{x}, \mathbf{z} \mid \theta)] + \underbrace{\mathrm{H}(q)}_{\text{entropy}} = \mathcal{L}_{\mathrm{ELBO}}(q, \theta)$$

$\mathcal{L}_{\mathrm{ELBO}}(q, \theta)$ is the *(evidence) lower bound*, a.k.a. variational lower bound, a.k.a. (variational) free energy.

Remember that $q(\mathbf{z})$ is supposed to mimic $p(\mathbf{z} \mid \mathbf{x}, \theta)$.
One way of formalising this would be finding a q that minimises

$$\mathrm{KL}(q(\mathbf{z}) \,||\, p(\mathbf{z} \mid \mathbf{x}, \theta)).$$

We are minimising an objective w. r. t. a function q. This is called a *variational approach*.

A very important observation:

$$\underbrace{\ln p(\mathbf{x} \mid \theta)}_{\text{const. wrt. } q} = \mathcal{L}_{\mathrm{ELBO}}(q, \theta) + \underbrace{\mathrm{KL}(q(\mathbf{z}) \,||\, p(\mathbf{z} \mid \mathbf{x}, \theta))}_{\geq 0}$$

$$\Rightarrow \arg\min_q \mathrm{KL}(q(\mathbf{z}) \,||\, p(\mathbf{z} \mid \mathbf{x}, \theta)) = \arg\max_q \mathcal{L}_{\mathrm{ELBO}}(q, \theta)$$

Nice theoretical insight, but why this detour?

Recall $r_{nk}\left(\theta^{(t)}\right) = p\big(\mathbf{z}_n = k \mid \mathbf{x}_n, \theta^{(t)}\big)$.

$r_{nk}\left(\theta^{(t)}\right)$ is the (trivial) optimal solution to

$$\arg\min_q \mathrm{KL}\Big(q(\mathbf{z}) \,||\, p\Big(\mathbf{z} \mid \mathbf{x}, \theta^{(t)}\Big)\Big) = \arg\max_q \mathcal{L}_{\mathrm{ELBO}}\left(q, \theta^{(t)}\right).$$

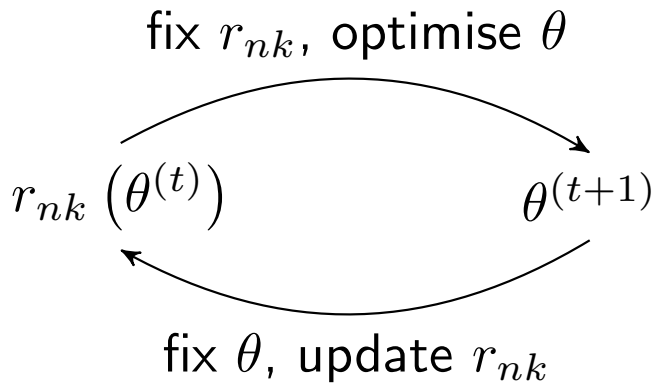
One of the steps in our cycle, the update of the r_{nk}'s, is exactly equivalent to *coordinate ascent* of $\mathcal{L}_{\mathrm{ELBO}}$ in q!

Idea: Maybe the second step is equivalent to coordinate ascent in the second component θ?

$$
\begin{aligned}
&\arg\max_{\theta} \mathcal{L}_{\mathrm{ELBO}}\left(q, \theta\right) \\
&= \arg\max_{\theta} \mathbb{E}_{q(\mathbf{z})}[\ln p(\mathbf{x}, \mathbf{z} \mid \theta)] + \underbrace{\mathrm{H}(q)}_{\text{const. wrt. } \theta} \\
&= \arg\max_{\theta} \mathbb{E}_{q(\mathbf{z})}[\ln p(\mathbf{x}, \mathbf{z} \mid \theta)]
\end{aligned}
$$

That is exactly the second step!

Our intuitive approach

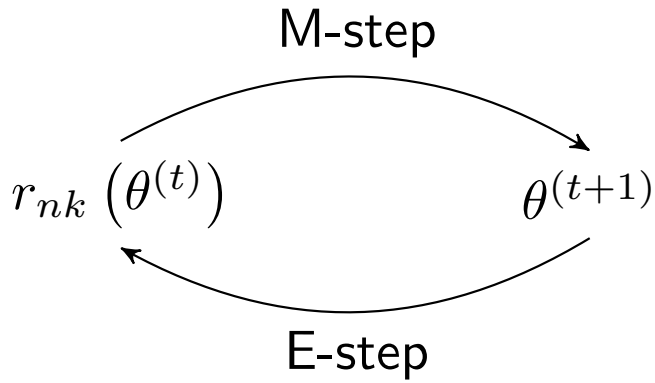


turns out to do alternating coordinate ascent on

$$\mathcal{L}_{\mathrm{ELBO}}(q, \theta) \qquad \leq p(\mathcal{D} \mid \theta) \leq p(\mathcal{D} \mid \theta_{\mathrm{MLE}}).$$

- Coordinate ascent guarantees that $\mathcal{L}_{\mathrm{ELBO}}\left(r_{nk}\left(\theta^{(t)}\right), \theta^{(t)}\right)$ increases monotonically with t. Since it is bounded by the constant $p(\mathcal{D} \mid \theta_{\mathrm{MLE}})$, our algorithm is *guaranteed to converge*.
- In the global optimum, $\theta^{(\infty)} = \theta_{\mathrm{MLE}}$.

This alternating procedure is the *expectation-maximisation (EM) algorithm*.



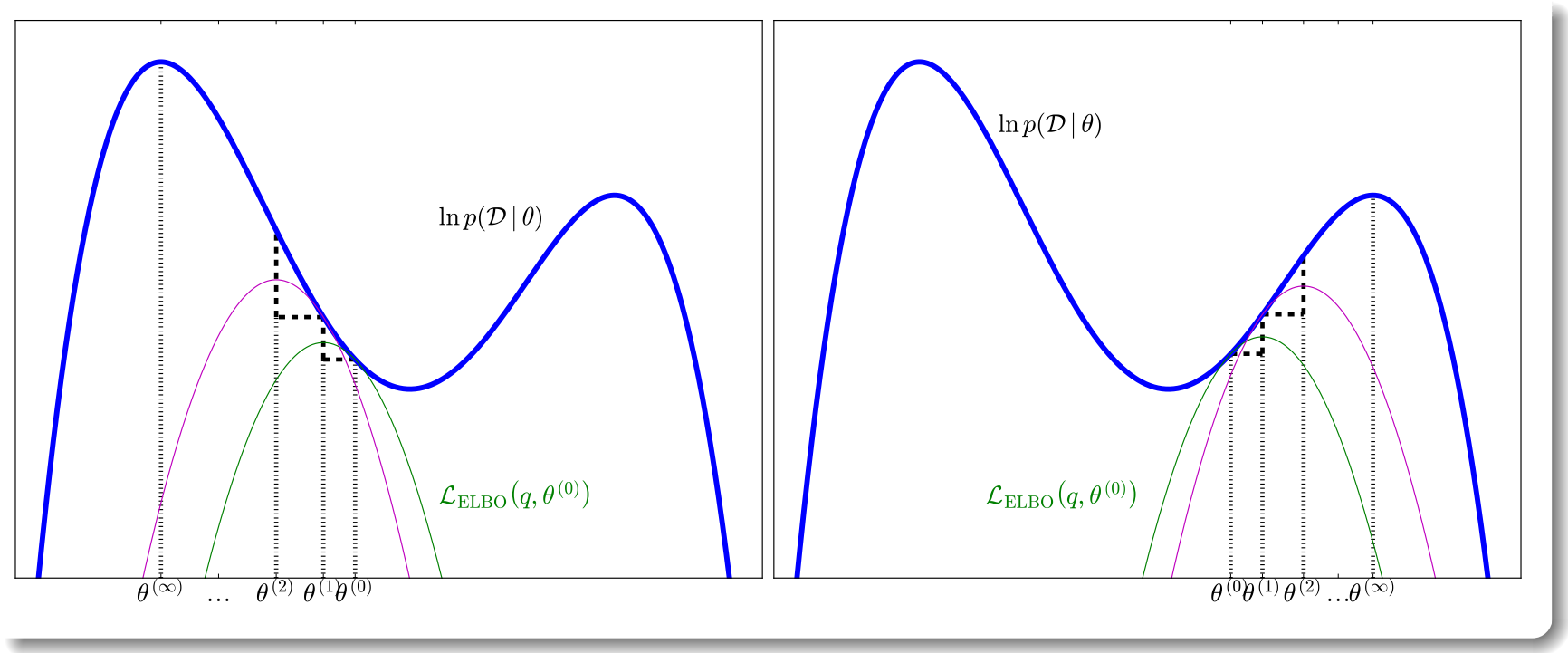
In the *maximisation step*, we do $\arg\max_\theta \mathbb{E}_{q(\mathbf{z})}[\ln p(\mathbf{x}, \mathbf{z} \mid \theta)]$.

Why *expectation step*, we are also doing a maximisation in this step?

- This maximisation can sometimes be solved in closed form.
- We already saw that $r_{nk}\left(\theta^{(t)}\right)$, the posterior at $\theta^{(t)}$, is optimal.
- The maximisation reduces to computing $\mathbb{E}_{q(\mathbf{z})}[\ln p(\mathbf{x}, \mathbf{z} \mid \theta)]$ for a known $q(\mathbf{z}) \rightsquigarrow$ E-step.

Since EM was historically developed for special cases where this holds, the somewhat misleading name has become standard.

A visualisation of iterative lower bound optimization.



Initialisation matters!

Some closing remarks on EM.

Nice features:

- We can do MLE on models where we cannot find a closed-form solution.
- Convergence to a *local* optimum is guaranteed.

But there are issues:

- Avoid bad local minima via multiple restarts.
- How to initialize?
- How to choose hyper-parameters such as the number of mixture models? (Overfitting, underfitting, ...)

For the rest of this lecture, we will deal with one issue:
What if the posterior is not known explicitly? E-step?

Latent variables $\mathbf{z}$ are often interpreted as the *quintessential information*. The posterior $p(\mathbf{z} \mid \mathbf{x}, \theta)$ then is the holy grail of many areas in machine learning.

A lot of supervised learning techniques implicitly try to learn this from *labeled* data.

We saw that EM can even learn it in an unsupervised fashion—but we explicitly needed the posterior for a given θ, otherwise the E-step does not work.

What is so hard about the posterior?

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

Typically, the generative model (the numerator) is easy to formulate, the normalizing constant is at best cumbersome to obtain.

In our digit example, marginalisation of $\mathbf{z}$ was easy—because we only incorporated the digit as *quintessential information*. What about rotation? Skewness? ...?

Or even better: What if we do not manually hard-code structure?
$\rightsquigarrow$ Why not use a neural network for $p(\mathbf{x} \mid \mathbf{z}, \theta)$?

We conclude that the posterior in general is hard to obtain.
Now remember:

$$\arg\min_q \mathrm{KL}(q(\mathbf{z}) \,||\, p(\mathbf{z} \mid \mathbf{x}, \theta)) = \arg\max_q \mathcal{L}_{\mathrm{ELBO}}(q, \theta)$$

The E-step explicitly did the lhs, and by equivalence implicitly the rhs.

$$\begin{aligned}\mathcal{L}_{\mathrm{ELBO}}(q, \theta) &= \int q(\mathbf{z}) \ln \frac{p(\mathbf{x}, \mathbf{z} \mid \theta)}{q(\mathbf{z})} \,\mathrm{d}\mathbf{z} = \int q(\mathbf{z}) \ln \frac{p(\mathbf{x} \mid \mathbf{z}, \theta) p(\mathbf{z})}{q(\mathbf{z})} \,\mathrm{d}\mathbf{z} \\ &= \mathbb{E}_{q(\mathbf{z})}[\ln p(\mathbf{x} \mid \mathbf{z}, \theta)] - \mathrm{KL}(q(\mathbf{z}) \,||\, p(\mathbf{z}))\end{aligned}$$

The ELBO does not explicitly contain the posterior, but implicitly approximates it *if the feasible region for q is large enough*!

Caveat: What is the posterior of $p(\mathbf{x} \mid \mathbf{z}, \theta)$ if it is implemented by a neural net?

We need a set of parametrized (learnable) functions that cover a wide range of functions.

Neural networks! Set $q(\mathbf{z}) \equiv q(\mathbf{z} \mid \mathbf{x}, \phi)$, where the latter is a neural network with parameters ϕ, input $\mathbf{x}$. The output will be distribution parameters of $\mathbf{z}$, e. g., $q(\mathbf{z} \mid \mathbf{x}, \phi) = \mathcal{N}(\mathbf{z} \mid \boldsymbol{\mu}_\phi(\mathbf{x}), \boldsymbol{\Sigma}_\phi(\mathbf{x}))$—Gaussian distributions with nonlinear dependence on the input.

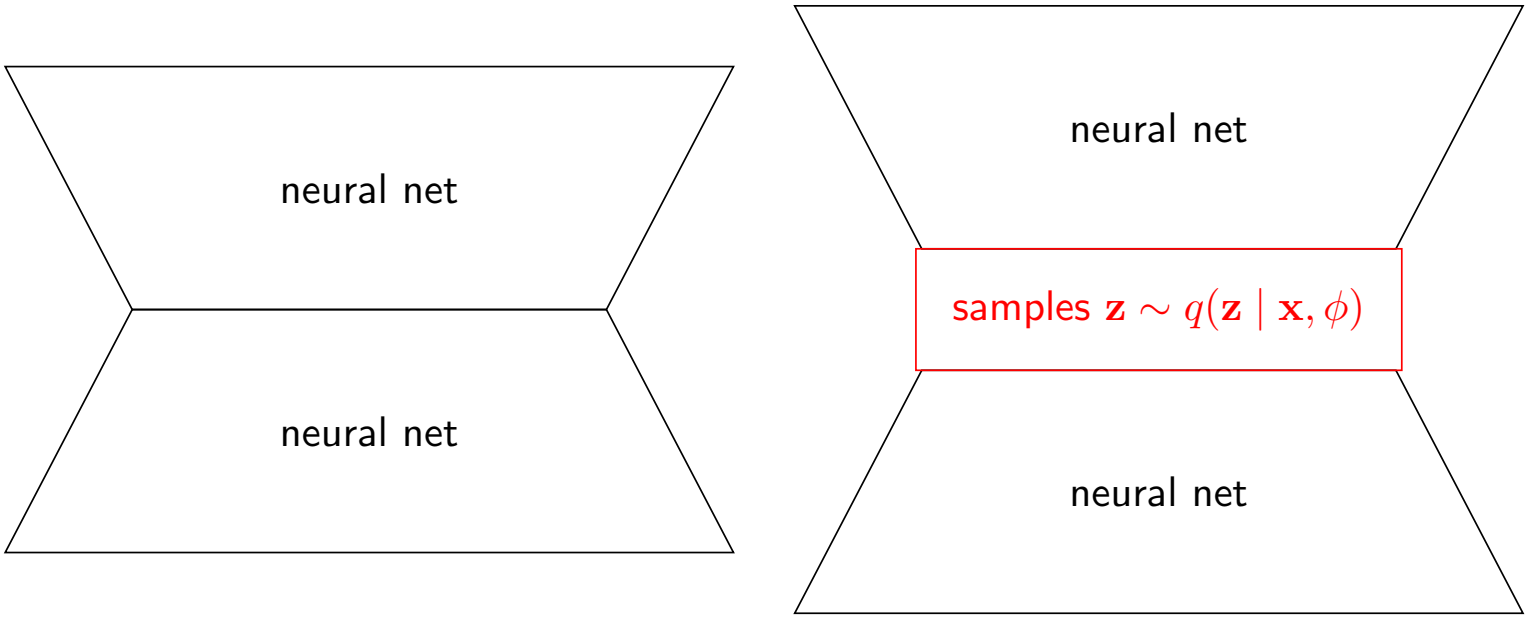
Let us put together the parts. We have:

- a neural network mapping $\mathbf{z}$ onto distributions over $\mathbf{x}$ with parameters θ—the *generative model* $p(\mathbf{x} \mid \mathbf{z}, \theta)$,
- a neural network mapping $\mathbf{x}$ onto distributions over $\mathbf{z}$ with parameters ϕ—the *recognition model* $q(\mathbf{z} \mid \mathbf{x}, \phi)$,
- a loss function $\mathcal{L}_{\text{ELBO}}(q, \theta) \equiv \mathcal{L}_{\text{ELBO}}(\phi, \theta)$ that captures both parameter sets and has intriguing theoretical properties.

These three parts amount to the *variational auto-encoder (VAE)*.

- Kingma, Diederik P., and Max Welling. "Auto-encoding variational bayes." arXiv preprint arXiv:1312.6114 (2013).
- Rezende, Danilo Jimenez, Shakir Mohamed, and Daan Wierstra. "Stochastic backpropagation and approximate inference in deep generative models." arXiv preprint arXiv:1401.4082 (2014).

Deterministic auto-encoder vs. variational auto-encoder:



The bottleneck now comes from stochasticity through sampling rather than dimensionality reduction.

$$\mathcal{L}_{\text{ELBO}}(q, \theta) = \mathbb{E}_{q(\mathbf{z})}[\ln p(\mathbf{x} \mid \mathbf{z}, \theta)] - \text{KL}(q(\mathbf{z}) \mid\mid p(\mathbf{z}))$$

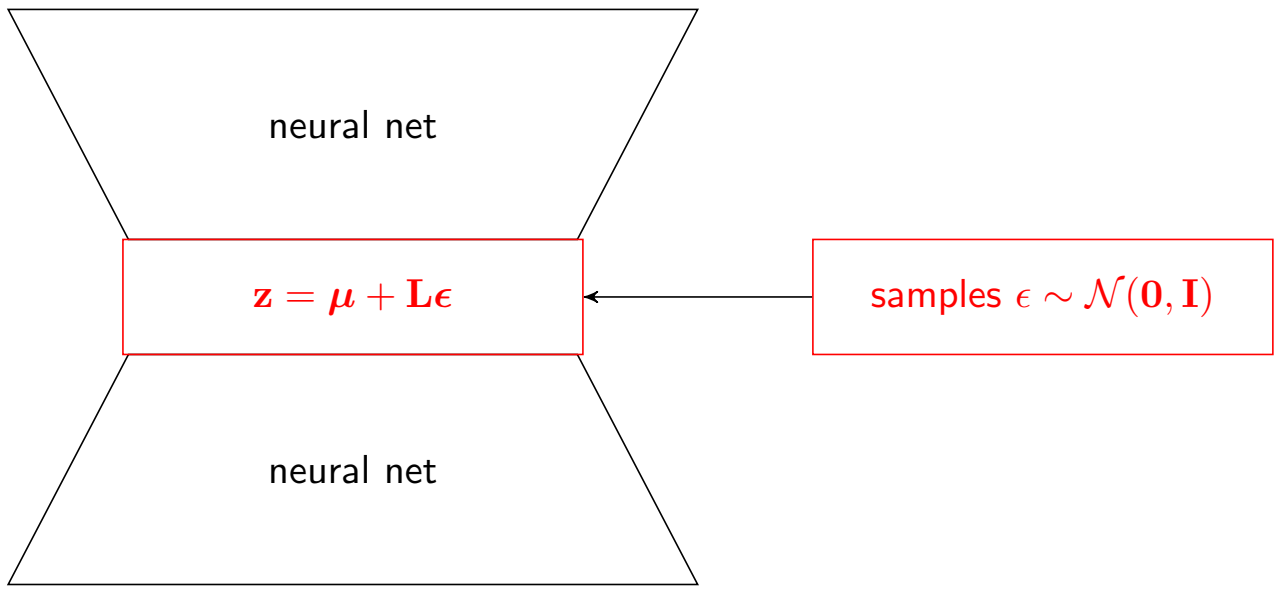
The ELBO rewards good reconstruction (first term), and comes with an inbuilt regulariser (the KL) that prevents collapsing to the deterministic autoencoder.

One more obstacle: Backpropagation through random sampling?

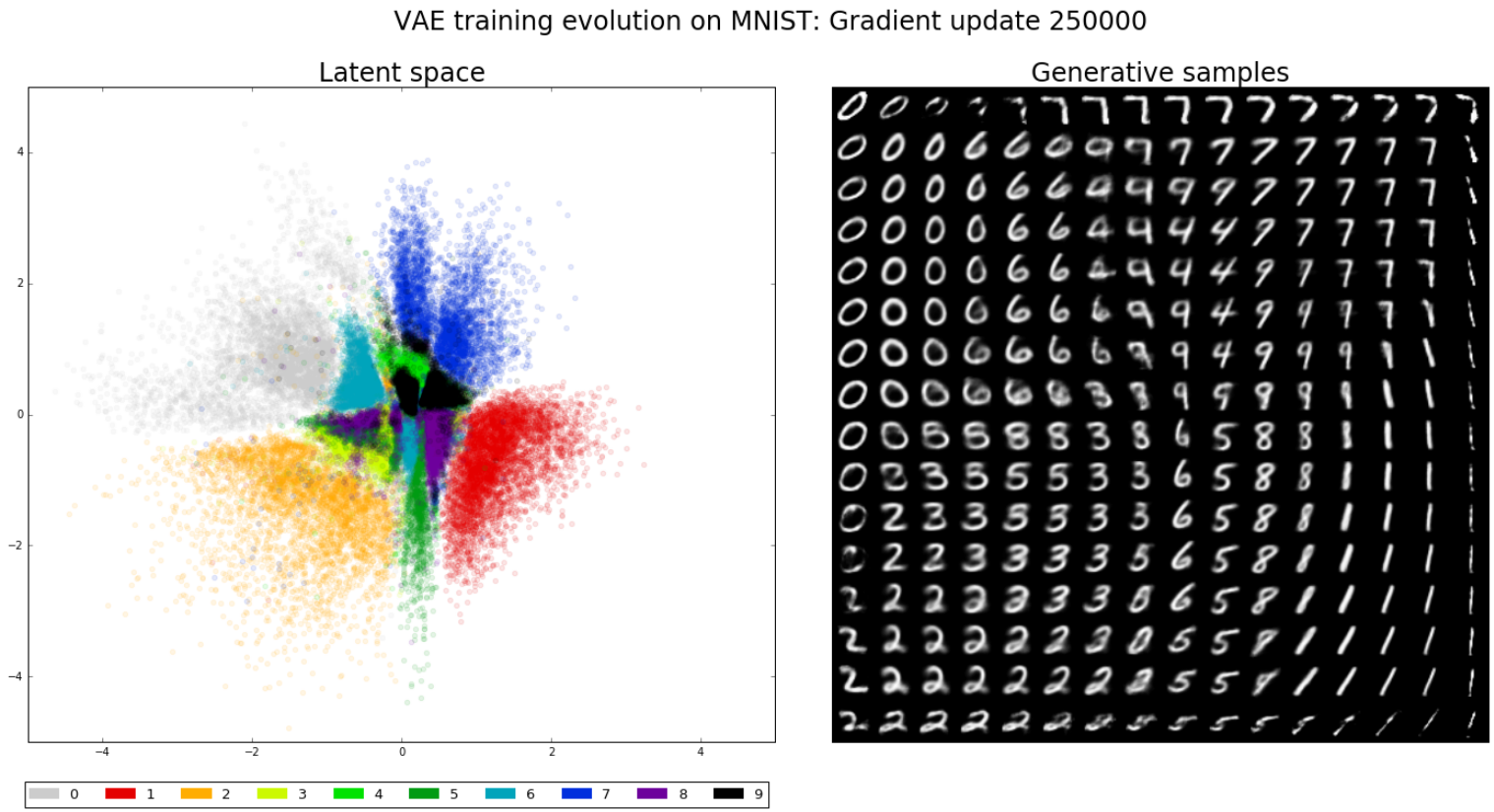
Solution: Reparametrisation.

$$\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \quad \rightsquigarrow \quad \mathbf{z} = \boldsymbol{\mu} + \mathbf{L}\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad \boldsymbol{\Sigma} = \mathbf{L}\mathbf{L}^T$$

This allows taking partial derivatives w. r. t. $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$.



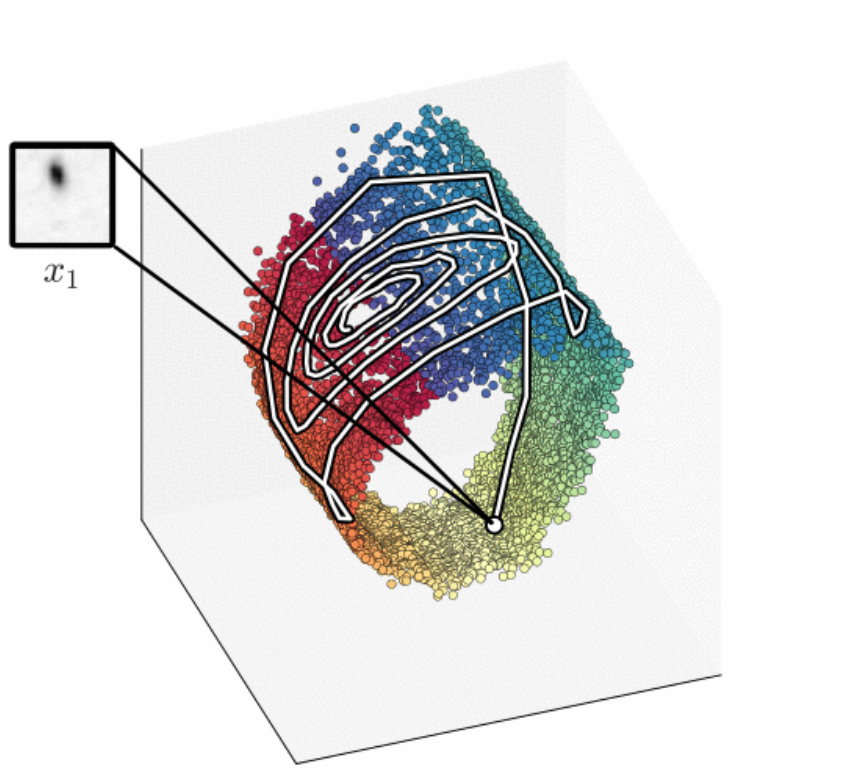
Results:



For visualization, the model only uses two latent variables. With a higher number, the model trains faster with even better results.

If this slide is not animated, you might want to try a different pdf reader.

Results from our lab:



Karl, Maximilian, Maximilian Soelch, Justin Bayer, and Patrick van der Smagt. “Deep Variational Bayes Filters: Unsupervised Learning of State Space Models from Raw Data.” arXiv preprint arXiv:1605.06432 (2016).

If this slide is not animated, you might want to try a different pdf reader.

What we learned

- Latent variables and graphical models.
- Gaussian mixture models.
- Variational techniques and the evidence lower bound.
  - Expectation maximisation.
  - Variational auto-encoder.

- Intriguing properties of the ELBO.
- The importance of the posterior distribution of latent factors given data.
- Reparametrisation.