Advanced Course on Deep Generative Models

Lecture 6: Variational Inference, Variational Autoencoder (VAE)

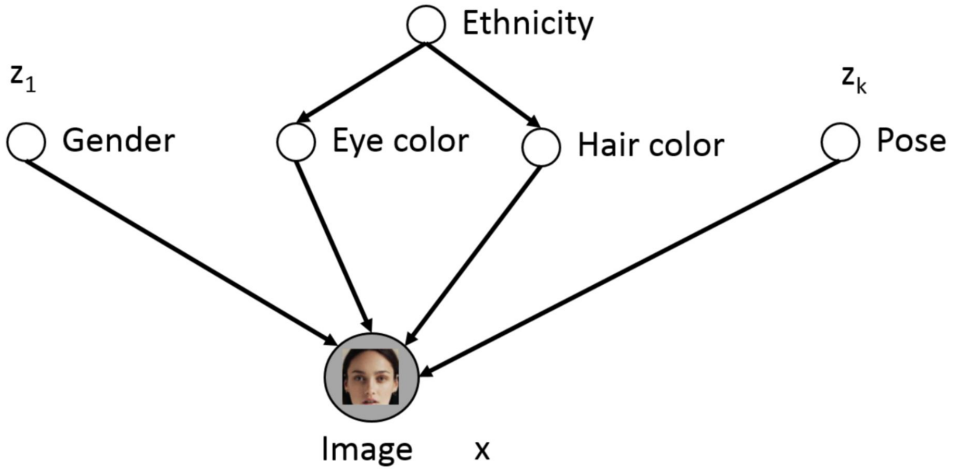
Dan Rosenbaum, CS Haifa

slides adapted from www.inf.ed.ac.uk/teaching/courses/pmr

Today

- KL divergence
- The variational lower bound (ELBO)
- Variational inference methods
- Amortized Variational Inference
- Variational Autoencoder (VAE)

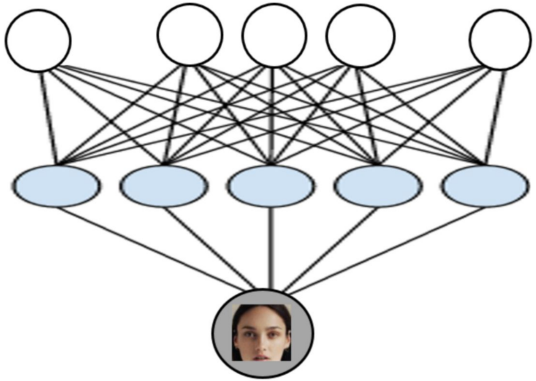
Latent variable models



1. Observed variables **x** are the pixels.
2. Ideally latent variables ***z*** capture interesting semantics.

Image: Volodymyr Kuleshov

Latent variable models - deep learning approach



1. Don’t specify structure.
2. $p_\theta(x \mid z)$ is modeled via a high capacity deep neural network.
3. Need to model $p(x) = \int p(x \mid z)\, p(z)\, dz$

Image: Volodymyr Kuleshov

Recap

- Learning and inference often involve hard integrals:
- For example:
  - Marginalization: $p(x) = \int p(x, y)\, dy$
  - Likelihood for latent variable models: $L(\theta) = \int p(D, z; \theta)\, dz$
- Can be approximated with sampling method

We will see today a different approximation method

Variational method

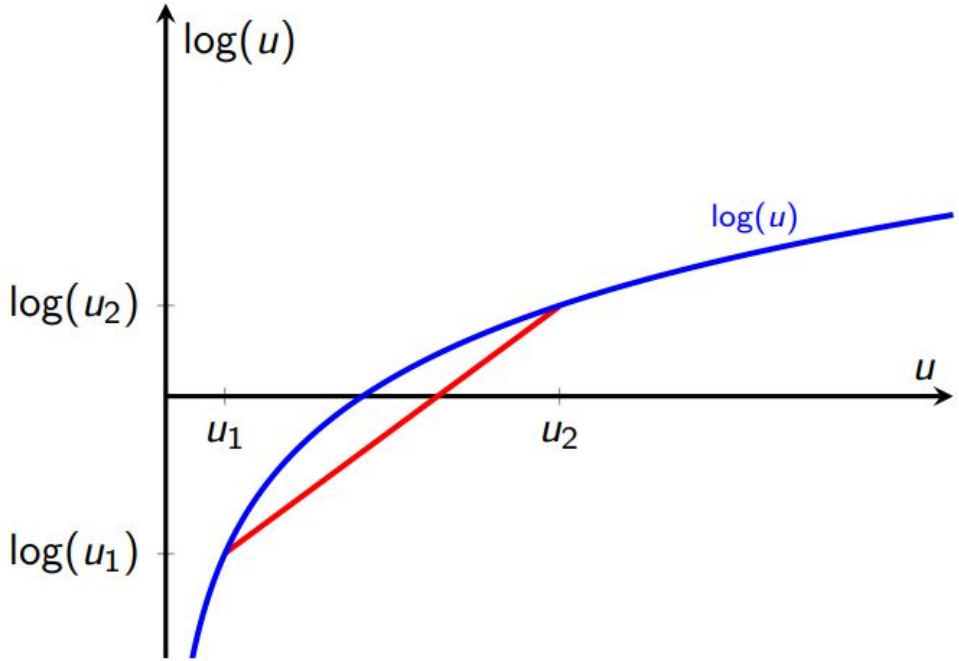
Originates in physics

- Fermat's principle (1650):
  "light travels between two given points
  along the path of shortest time"

- In general when looking for a (probability) function:
  frame the problem as an optimization problem.

log(u) is a concave function

- **log((1−a) u1 +a u2) ≥ (1−a) log u) +a log(u2) a ∈ [0, 1]**
- **log(average) ≥ average(log)**
- Generalization:
  log E g(x) ≥ E log g(x)
  with **g(x) > 0**
- Called Jensen's inequality



KL divergence

$$\mathrm{KL}(p||q) = \int p(\mathbf{x}) \log \frac{p(\mathbf{x})}{q(\mathbf{x})} \mathrm{d}\mathbf{x} = \mathbb{E}_{p(\mathbf{x})} \left[\log \frac{p(\mathbf{x})}{q(\mathbf{x})} \right]$$

- KL(p||q) ≠ KL(q||p)
- KL(p||q) ≥ 0
- KL(p||q) = 0 if and only if p=q

Non negativity

$$-\mathrm{KL}(p||q) = -\mathbb{E}_{p(\mathbf{x})}\left[\log \frac{p(\mathbf{x})}{q(\mathbf{x})}\right] \tag{2}$$

$$= \mathbb{E}_{p(\mathbf{x})}\left[\log \frac{q(\mathbf{x})}{p(\mathbf{x})}\right] \tag{3}$$

$$\leq \log \underbrace{\mathbb{E}_{p(\mathbf{x})}\left[\frac{q(\mathbf{x})}{p(\mathbf{x})}\right]}_{\int p(\mathbf{x})q(\mathbf{x})/p(\mathbf{x})d\mathbf{x}=1} \tag{4}$$

Hence $-\mathrm{KL}(p||q) \leq \log(1) = 0$ and thus

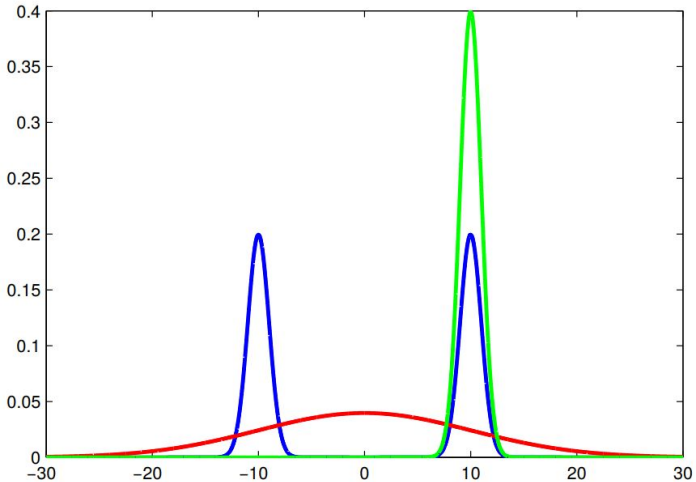
$$\mathrm{KL}(p||q) \geq 0 \tag{5}$$

Asymmetry

Blue: mixture of Gaussians $p(x)$ (fixed)

Green: (unimodal) Gaussian q that minimises $\mathrm{KL}(q||p)$

Red: (unimodal) Gaussian q that minimises $\mathrm{KL}(p||q)$



Barber Figure 28.1, Section 28.3.4

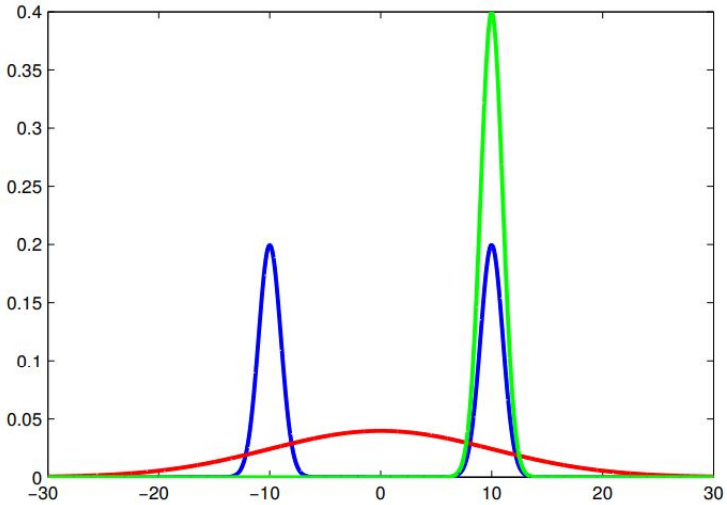
Asymmetry

$\operatorname{argmin}_q \mathrm{KL}(q||p) = \operatorname{argmin}_q \int q(\mathbf{x}) \log \frac{q(\mathbf{x})}{p(\mathbf{x})} \mathrm{d}\mathbf{x}$

- Optimal q avoids regions where p is small.
  (but can be small where p is large)
- Produces good local fit, "mode seeking"

$\operatorname{argmin}_q \mathrm{KL}(p||q) = \operatorname{argmin}_q \int p(\mathbf{x}) \log \frac{p(\mathbf{x})}{q(\mathbf{x})} \mathrm{d}\mathbf{x}$

- Optimal q is nonzero where p is nonzero
  (and does not care about regions where p is small)
- Corresponds to MLE; produces global fit/moment matching

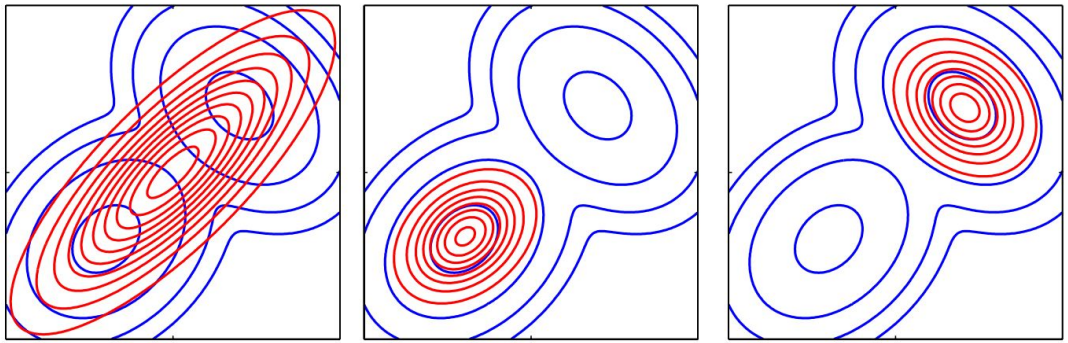


Asymmetry

Blue: mixture of Gaussians $p(\mathbf{x})$ (fixed)

Red: optimal (unimodal) Gaussians $q(\mathbf{x})$

Global moment matching (left) versus mode seeking (middle and right). (two local minima are shown)



$\min_q$ KL(p || q) $\min_q$ KL(q || p) $\min_q$ KL(q || p)

Bishop Figure 10.3

Variational lower bound

Consider joint pdf /pmf $p(\mathbf{x}, \mathbf{y})$ with marginal $p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{y}) \mathrm{d}\mathbf{y}$

- We can write $p(\mathbf{x})$ as

$$p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{y}) \frac{q(\mathbf{y}|\mathbf{x})}{q(\mathbf{y}|\mathbf{x})} \mathrm{d}\mathbf{y} = \mathbb{E}_{q(\mathbf{y}|\mathbf{x})} \left[\frac{p(\mathbf{x}, \mathbf{y})}{q(\mathbf{y}|\mathbf{x})} \right] \tag{9}$$

where $q(\mathbf{y}|\mathbf{x})$ is an auxiliary distribution (called the variational distribution in the context of variational inference/learning) for a given $\mathbf{x}$.

Variational lower bound

- Log marginal is

$$\log p(\mathbf{x}) = \log \mathbb{E}_{q(\mathbf{y}|\mathbf{x})} \left[\frac{p(\mathbf{x}, \mathbf{y})}{q(\mathbf{y}|\mathbf{x})} \right] \tag{10}$$

- Approximating the expectation with a sample average leads to importance sampling. Another approach is to work with the concavity of the logarithm instead.
- Concavity of the log gives

$$\log p(\mathbf{x}) = \log \mathbb{E}_{q(\mathbf{y}|\mathbf{x})} \left[\frac{p(\mathbf{x}, \mathbf{y})}{q(\mathbf{y}|\mathbf{x})} \right] \geq \mathbb{E}_{q(\mathbf{y}|\mathbf{x})} \left[\log \frac{p(\mathbf{x}, \mathbf{y})}{q(\mathbf{y}|\mathbf{x})} \right] \tag{11}$$

This is the variational lower bound for $\log p(\mathbf{x})$.

Variational lower bound

$$\log p(\mathbf{x}) = \log \mathbb{E}_{q(\mathbf{y}|\mathbf{x})}\left[\frac{p(\mathbf{x},\mathbf{y})}{q(\mathbf{y}|\mathbf{x})}\right] \geq \mathbb{E}_{q(\mathbf{y}|\mathbf{x})}\left[\log \frac{p(\mathbf{x},\mathbf{y})}{q(\mathbf{y}|\mathbf{x})}\right] \quad (11)$$

- Right-hand side is called the (variational) free energy $\mathcal{F}_{\mathbf{x}}(q)$ or the evidence lower bound (ELBO) $\mathcal{L}_{\mathbf{x}}(q)$

$$\mathcal{L}_{\mathbf{x}}(q) = \mathbb{E}_{q(\mathbf{y}|\mathbf{x})}\left[\log \frac{p(\mathbf{x},\mathbf{y})}{q(\mathbf{y}|\mathbf{x})}\right] \quad (12)$$

- Since q is a function, the ELBO is a functional, which is a mapping that depends on a function.

Properties of the ELBO

$$\mathcal{L}_{\mathbf{x}}(q) = \mathbb{E}_{q(\mathbf{y}|\mathbf{x})}\left[\log \frac{p(\mathbf{x},\mathbf{y})}{q(\mathbf{y}|\mathbf{x})}\right]$$

- By manipulating the definition of the ELBO, we obtain the following equivalent forms

$$\begin{aligned}
\mathcal{L}_{\mathbf{x}}(q) &= \log p(\mathbf{x}) - \mathrm{KL}(q(\mathbf{y}|\mathbf{x})||p(\mathbf{y}|\mathbf{x})) && (13)\\
&= \mathbb{E}_{q(\mathbf{y}|\mathbf{x})} \log p(\mathbf{x}|\mathbf{y}) - \mathrm{KL}(q(\mathbf{y}|\mathbf{x})||p(\mathbf{y})) && (14)\\
&= \mathbb{E}_{q(\mathbf{y}|\mathbf{x})} \log p(\mathbf{x},\mathbf{y}) + \mathcal{H}(q) && (15)
\end{aligned}$$

where $p(\mathbf{y})$ is the marginal of $p(\mathbf{x},\mathbf{y})$ and $\mathcal{H}(q)$ is the entropy of q.

Properties of the ELBO

$$\mathcal{L}_{\mathbf{x}}(q) = \mathbb{E}_{q(\mathbf{y}|\mathbf{x})}\left[\log \frac{p(\mathbf{x},\mathbf{y})}{q(\mathbf{y}|\mathbf{x})}\right] = \mathbb{E}_{q(\mathbf{y}|\mathbf{x})}\left[\log \frac{p(\mathbf{y}|\mathbf{x})p(\mathbf{x})}{q(\mathbf{y}|\mathbf{x})}\right]$$

$$= \mathbb{E}_{q(\mathbf{y}|\mathbf{x})}\left[\log \frac{p(\mathbf{y}|\mathbf{x})}{q(\mathbf{y}|\mathbf{x})} + \log p(\mathbf{x})\right]$$

$$= \mathbb{E}_{q(\mathbf{y}|\mathbf{x})}\left[\log \frac{p(\mathbf{y}|\mathbf{x})}{q(\mathbf{y}|\mathbf{x})}\right] + \log p(\mathbf{x})$$

$$= -\mathrm{KL}(q(\mathbf{y}|\mathbf{x})||p(\mathbf{y}|\mathbf{x})) + \log p(\mathbf{x})$$

Tightness of the ELBO

- From $\mathcal{L}_{\mathbf{x}}(q) = \log p(\mathbf{x}) - \mathrm{KL}(q(\mathbf{y}|\mathbf{x})||p(\mathbf{y}|\mathbf{x}))$ and non-negativity of the KL divergence, we have
  1. $\log p(\mathbf{x}) \geq \mathcal{L}_{\mathbf{x}}(q)$ (as before)
  2. $\log p(\mathbf{x}) = \mathcal{L}_{\mathbf{x}}(q) \Leftrightarrow q(\mathbf{y}|\mathbf{x}) = p(\mathbf{y}|\mathbf{x})$
- Maximising $\mathcal{L}_{\mathbf{x}}(q)$ with respect to q yields both $\log p(\mathbf{x})$ and the conditional $p(\mathbf{y}|\mathbf{x})$ at the same time.
- Makes sense because if we know $p(\mathbf{x}, \mathbf{y})$ and $p(\mathbf{x})$, we know $p(\mathbf{y}|\mathbf{x})$, and vice versa, since $p(\mathbf{y}|\mathbf{x}) = p(\mathbf{x}, \mathbf{y})/p(\mathbf{x})$.

Variational principle: inference becomes optimization

- By maximising the ELBO

$$\mathcal{L}_{\mathbf{x}}(q) = \mathbb{E}_{q(\mathbf{y}|\mathbf{x})}\left[\log \frac{p(\mathbf{x},\mathbf{y})}{q(\mathbf{y}|\mathbf{x})}\right]$$

we can split the joint $p(\mathbf{x},\mathbf{y})$ into $p(\mathbf{x})$ and $p(\mathbf{y}|\mathbf{x})$

$$\log p(\mathbf{x}) = \max_q \mathcal{L}_{\mathbf{x}}(q)$$

$$p(\mathbf{y}|\mathbf{x}) = \operatorname*{argmax}_q \mathcal{L}_{\mathbf{x}}(q)$$

Solving the optimization problem

$$\mathcal{L}_{\mathbf{x}}(q) = \mathbb{E}_{q(\mathbf{y}|\mathbf{x})}\left[\log \frac{p(\mathbf{x},\mathbf{y})}{q(\mathbf{y}|\mathbf{x})}\right]$$

- Difficulties when maximising the ELBO:
  - Learning of a pdf/pmf $q(\mathbf{y}|\mathbf{x})$
  - Maximisation when objective involves $\mathbb{E}_{q(\mathbf{y}|\mathbf{x})}$ that depends on q
- Restrict search space to a family $\mathcal{Q}$ of variational distributions $q(\mathbf{y}|\mathbf{x})$ for which $\mathcal{L}_{\mathbf{x}}(q)$ is computable.
- Family $\mathcal{Q}$ specified by
  - independence assumptions, e.g. $q(\mathbf{y}|\mathbf{x}) = \prod_i q(y_i|\mathbf{x})$, which corresponds to "mean-field" variational inference
  - parametric assumptions, e.g. $q(y_i|\mathbf{x}) = \mathcal{N}(y_i; \mu_i(\mathbf{x}), \sigma_i^2(\mathbf{x}))$
- $\mathcal{L}_{\mathbf{x}}(q)$ can be computed analytically in closed form only in special cases.

Approximate posterior inference

- Inference task: given value $\mathbf{x} = \mathbf{x}_o$ and joint pdf/pmf $p(\mathbf{x}, \mathbf{y})$, compute $p(\mathbf{y}|\mathbf{x}_o)$.
- Variational approach: estimate the posterior by solving an optimisation problem

$$\hat{p}(\mathbf{y}|\mathbf{x}_o) = \operatorname*{argmax}_{q \in \mathcal{Q}} \mathcal{L}_{\mathbf{x}_o}(q) \tag{19}$$

  $\mathcal{Q}$ is the set of pdfs/pmfs in which we search for the solution
- From the basic property of the ELBO in Equation (13)

$$\log p(\mathbf{x}_o) = \mathrm{KL}(q(\mathbf{y}|\mathbf{x}_o)||p(\mathbf{y}|\mathbf{x}_o)) + \mathcal{L}_{\mathbf{x}_o}(q) = \text{const} \tag{20}$$

Approximate posterior inference

$$\log p(\mathbf{x}_o) = \mathrm{KL}(q(\mathbf{y}|\mathbf{x}_o)||p(\mathbf{y}|\mathbf{x}_o)) + \mathcal{L}_{\mathbf{x}_o}(q) = \text{const} \tag{20}$$

- Because the sum of the KL and ELBO is constant, we have

$$\underset{q \in \mathcal{Q}}{\operatorname{argmax}}\, \mathcal{L}_{\mathbf{x}_o}(q) = \underset{q \in \mathcal{Q}}{\operatorname{argmin}}\, \mathrm{KL}(q(\mathbf{y}|\mathbf{x}_o)||p(\mathbf{y}|\mathbf{x}_o)) \tag{21}$$

Compromise between prior and fit

- Equivalent forms of the ELBO:

$$\mathcal{L}_{\mathbf{x}_o}(q) = \mathbb{E}_{q(\mathbf{y}|\mathbf{x}_o)} \log p(\mathbf{x}_o|\mathbf{y}) - \mathrm{KL}(q(\mathbf{y}|\mathbf{x}_o)||p(\mathbf{y})) \qquad (22)$$

- By maximising $\mathcal{L}_{\mathbf{x}_o}(q)$ we find a q that
  - produces $\mathbf{y}$ which are likely explanations of $\mathbf{x}_o$
  - stays close to the prior $p(\mathbf{y})$
- If included in the search space $\mathcal{Q}$, $p(\mathbf{y}|\mathbf{x}_o)$ is the optimal q, which means that the posterior fulfils the two desiderata best.

Compromise between likely imputation and variability

- Equivalent forms of the ELBO:

$$\mathcal{L}_{\mathbf{x}_o}(q) = \mathbb{E}_{q(\mathbf{y}|\mathbf{x}_o)} \log p(\mathbf{x}_o, \mathbf{y}) + \mathcal{H}(q) \tag{23}$$

- By maximising $\mathcal{L}_{\mathbf{x}_o}(q)$ we find a q that
  - produces likely imputations (filled-in data) $\mathbf{y}$
  - is maximally variable
- If included in the search space $\mathcal{Q}$, $p(\mathbf{y}|\mathbf{x}_o)$ is the optimal q, which means that the posterior fulfils the two desiderata best.

Nature of approximation

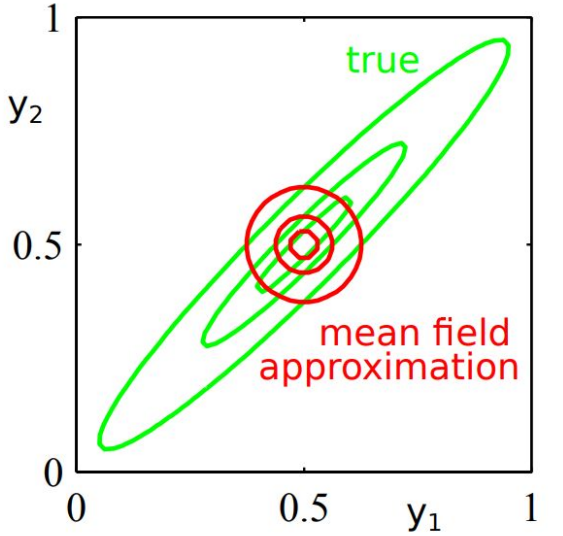
$$\text{argmax}_{q \in \mathcal{Q}} \mathcal{L}_{\mathbf{x}_o}(q) = \text{argmin}_{q \in \mathcal{Q}} \text{KL}(q(\mathbf{y}|\mathbf{x}_o)||p(\mathbf{y}|\mathbf{x}_o))$$

- When minimising $\text{KL}(q||p)$ with respect to q, q will try very hard to be zero where p is small.
- Assume true posterior is correlated bivariate Gaussian and we work with $\mathcal{Q} = \{q(\mathbf{y}|\mathbf{x}_o) : q(\mathbf{y}|\mathbf{x}_o) = q(y_1|\mathbf{x}_o)q(y_2|\mathbf{x}_o)\}$
  (independence but no parametric assumptions)

Nature of approximation

- Assume true posterior is correlated bivariate Gaussian and we work with $\mathcal{Q} = \{q(\mathbf{y}|\mathbf{x}_o) : q(\mathbf{y}|\mathbf{x}_o) = q(y_1|\mathbf{x}_o)q(y_2|\mathbf{x}_o)\}$
  (independence but no parametric assumptions)

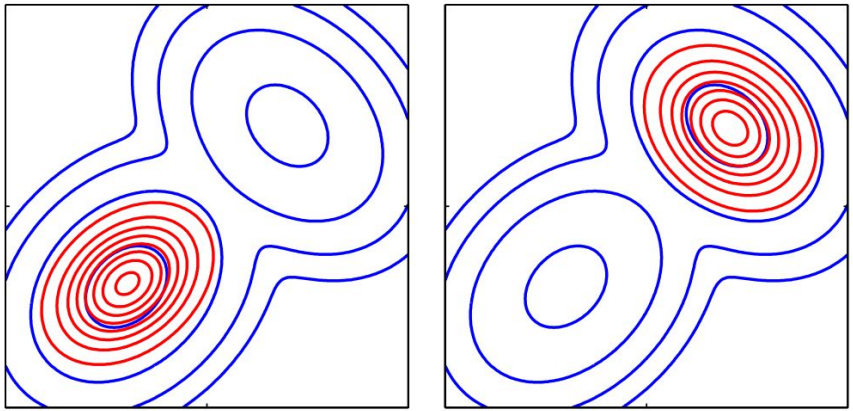
- Optimal q is Gaussian.
- Mean is correct but variances dictated by the variances of $p(\mathbf{y}|\mathbf{x}_o)$ along the y_1 and y_2 axes.
- Posterior variance is underestimated.



(Bishop, Figure 10.2)

Nature of approximation

- Assume that true posterior is multimodal, but that the family of variational distributions $\mathcal{Q}$ only includes unimodal distributions.
- The optimal $q(\mathbf{y}|\mathbf{x}_o)$ only covers one mode: "mode-seeking behaviour".



local optimum local optimum

Blue: true posterior
Red: approximation

Bishop Figure 10.3 (adapted)

Maximum Likelihood with Variational Inference

- Evidence lower bound (ELBO) holds for any $q(\mathbf{z}; \phi)$

$$\log p(\mathbf{x}; \theta) \geq \sum_{\mathbf{z}} q(\mathbf{z}; \phi) \log p(\mathbf{z}, \mathbf{x}; \theta) + H(q(\mathbf{z}; \phi)) = \underbrace{\mathcal{L}(\mathbf{x}; \theta, \phi)}_{\text{ELBO}}$$

- Maximum likelihood learning (over the entire dataset):

$$\ell(\theta; \mathcal{D}) = \sum_{\mathbf{x}^i \in \mathcal{D}} \log p(\mathbf{x}^i; \theta) \geq \sum_{\mathbf{x}^i \in \mathcal{D}} \mathcal{L}(\mathbf{x}^i; \theta, \phi^i)$$

- Therefore

$$\max_{\theta} \ell(\theta; \mathcal{D}) \geq \max_{\theta, \phi^1, \cdots, \phi^M} \sum_{\mathbf{x}^i \in \mathcal{D}} \mathcal{L}(\mathbf{x}^i; \theta, \phi^i)$$

Connection to EM

We can optimize the ELBO using coordinate ascent:

1. Optimize with respect to **q** (the proposal distribution)
2. Optimize with respect to $\boldsymbol{\theta}$ (the parameters of **p**)

If we can compute the posterior $\boldsymbol{p(h \mid D ; \theta_k)}$ (which is the best **q** in each iteration) we obtain the classical EM algorithm:

E-step: compute the expectation

$$\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}, q^*) = \underbrace{\mathbb{E}_{p(\mathbf{h}|\mathcal{D};\boldsymbol{\theta}_k)}[\log p(\mathcal{D}, \mathbf{h}; \boldsymbol{\theta})]}_{\substack{\text{interpretation: expected} \\ \text{completed log-likelihood of } \boldsymbol{\theta}}} - \underbrace{\mathbb{E}_{p(\mathbf{h}|\mathcal{D};\boldsymbol{\theta}_k)} \log p(\mathbf{h}|\mathcal{D}; \boldsymbol{\theta}_k)}_{\substack{\text{does not depend on } \boldsymbol{\theta} \text{ and} \\ \text{does not need to be computed}}}$$

Connection to EM

E-step: compute the expectation

$$\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}, q^*) = \underbrace{\mathbb{E}_{p(\mathbf{h}|\mathcal{D};\boldsymbol{\theta}_k)}[\log p(\mathcal{D}, \mathbf{h}; \boldsymbol{\theta})]}_{\substack{\text{interpretation: expected} \\ \text{completed log-likelihood of } \boldsymbol{\theta}}} - \underbrace{\mathbb{E}_{p(\mathbf{h}|\mathcal{D};\boldsymbol{\theta}_k)} \log p(\mathbf{h}|\mathcal{D}; \boldsymbol{\theta}_k)}_{\substack{\text{does not depend on } \boldsymbol{\theta} \text{ and} \\ \text{does not need to be computed}}}$$

M-step: maximise with respect to $\boldsymbol{\theta}$

$$\boldsymbol{\theta}_{k+1} = \operatorname*{argmax}_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}, q^*) = \operatorname*{argmax}_{\boldsymbol{\theta}} \mathbb{E}_{p(\mathbf{h}|\mathcal{D};\boldsymbol{\theta}_k)}[\log p(\mathcal{D}, \mathbf{h}; \boldsymbol{\theta})]$$

When the E-step or $p(h \mid D ; \theta_k)$ cannot be computed exactly, it is still useful to think of this coordinate ascent of **q** and θ ⇒ ***variational EM***

Maximum Likelihood with Variational Inference

- Can’t optimize the log likelihood directly.
- Optimize lower bound instead (ELBO) using a proposal distribution **q(z|x)**
- In classical variational inference **q** is optimized per data point x_i
- The role of $q(z \mid x_i)$ is to invert the decoder $p_\theta(x \mid z_i)$
- In EM we could compute the best **q** directly in each iteration,

  but now we need to optimize so this becomes an inner loop:

Training with Variational inference

While not converged:

While not converged:

$\phi_i \leftarrow$ update $\mathbf{q_\phi(z \mid x_i)}$ to optimize ELBO

$\theta \leftarrow \boldsymbol{p_\theta(x \mid z)}$ to optimize ELBO

Different parameter ϕ_i for each data point $\boldsymbol{x_i}$

- independence assumptions, e.g. $q(\mathbf{y}|\mathbf{x}) = \prod_i q(y_i|\mathbf{x})$, which corresponds to "mean-field" variational inference
- parametric assumptions, e.g. $q(y_i|\mathbf{x}) = \mathcal{N}(y_i; \mu_i(\mathbf{x}), \sigma_i^2(\mathbf{x}))$

Stochastic Variational Inference

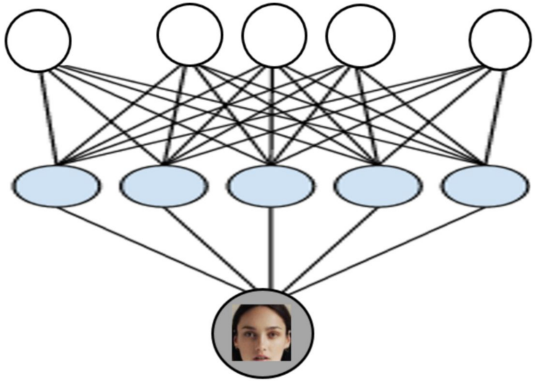
Optimize via stochastic variational inference

1. Initialize $\theta, \phi^1, \cdots, \phi^M$
2. Randomly sample a data point $\mathbf{x}^i$ from $\mathcal{D}$
3. Optimize $\mathcal{L}(\mathbf{x}^i; \theta, \phi^i)$ as a function of ϕ^i:
   1. Repeat $\phi^i = \phi^i + \eta \nabla_{\phi^i} \mathcal{L}(\mathbf{x}^i; \theta, \phi^i)$
   2. until convergence to $\phi^{i,*} \approx \arg\max_\phi \mathcal{L}(\mathbf{x}^i; \theta, \phi)$
4. Compute $\nabla_\theta \mathcal{L}(\mathbf{x}^i; \theta, \phi^{i,*})$
5. Update θ in the gradient direction. Go to step 2

Gradients through the expectations in the ELBO can be hard to compute.

We will see methods to do this.

Latent variable models - deep learning approach



1. Don’t specify structure.

2. $\mathbf{p_\theta(x \mid z)}$ is modeled via a high capacity deep neural network.

Image: Volodymyr Kuleshov

Simple prior - only learn a complex decoder

To keep it as simple as possible we will use the following modeling assumptions:

1. The prior $p(z)$ is fixed $N(\mu=0, \Sigma=I)$

2. The decoder $p_\theta(x \mid z)$ is computed via a deep neural network:

$$N(\mu_\theta(z), \text{diag}(\sigma_\theta(z)))$$

How can we train this decoder?

We want to train the decoder with maximum likelihood.

$$\log p(x) = \log \int p(x, z)\, dz$$

Usually this is intractable (in contrast to GMM which is based on a finite sum)

Option 1: Monte Carlo approximation by sampling from the prior

$$\log p(x) \approx \log 1/n \sum p_\theta(x \mid z_i), \; z_i \sim p(z)$$

Variance of this estimator is usually too high.

Variational bound

$$p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{y}) \frac{q(\mathbf{y}|\mathbf{x})}{q(\mathbf{y}|\mathbf{x})} \mathrm{d}\mathbf{y} = \mathbb{E}_{q(\mathbf{y}|\mathbf{x})} \left[\frac{p(\mathbf{x}, \mathbf{y})}{q(\mathbf{y}|\mathbf{x})} \right]$$

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

Variational bound, Evidence lower bound (ELBO)

Variational bound

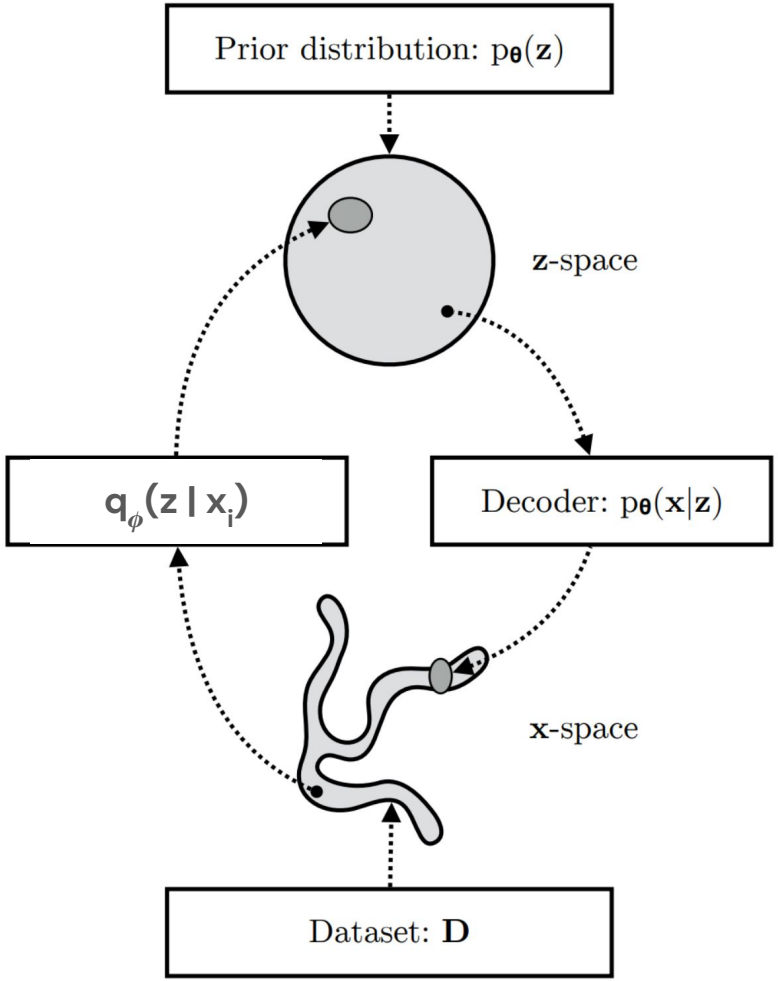
$$\mathcal{L}_{\mathbf{x}}(q) = \mathbb{E}_{q(\mathbf{y}|\mathbf{x})}\left[\log \frac{p(\mathbf{x},\mathbf{y})}{q(\mathbf{y}|\mathbf{x})}\right]$$

By manipulating the definition of the ELBO, we obtain the following equivalent forms

$$\begin{aligned}
\mathcal{L}_{\mathbf{x}}(q) &= \log p(\mathbf{x}) - \mathrm{KL}(q(\mathbf{y}|\mathbf{x})||p(\mathbf{y}|\mathbf{x})) && (13)\\
&= \mathbb{E}_{q(\mathbf{y}|\mathbf{x})} \log p(\mathbf{x}|\mathbf{y}) - \mathrm{KL}(q(\mathbf{y}|\mathbf{x})||p(\mathbf{y})) && (14)\\
&= \mathbb{E}_{q(\mathbf{y}|\mathbf{x})} \log p(\mathbf{x},\mathbf{y}) + \mathcal{H}(q) && (15)
\end{aligned}$$

The role of q

- The role of $q_\phi(z \mid x_i)$ is to invert the decoder $p_\theta(x \mid z_i)$
- Different parameter ϕ_i for each data point x_i
- Can $q_\phi(z \mid x_i)$ be a simple Gaussian?



Amortized inference

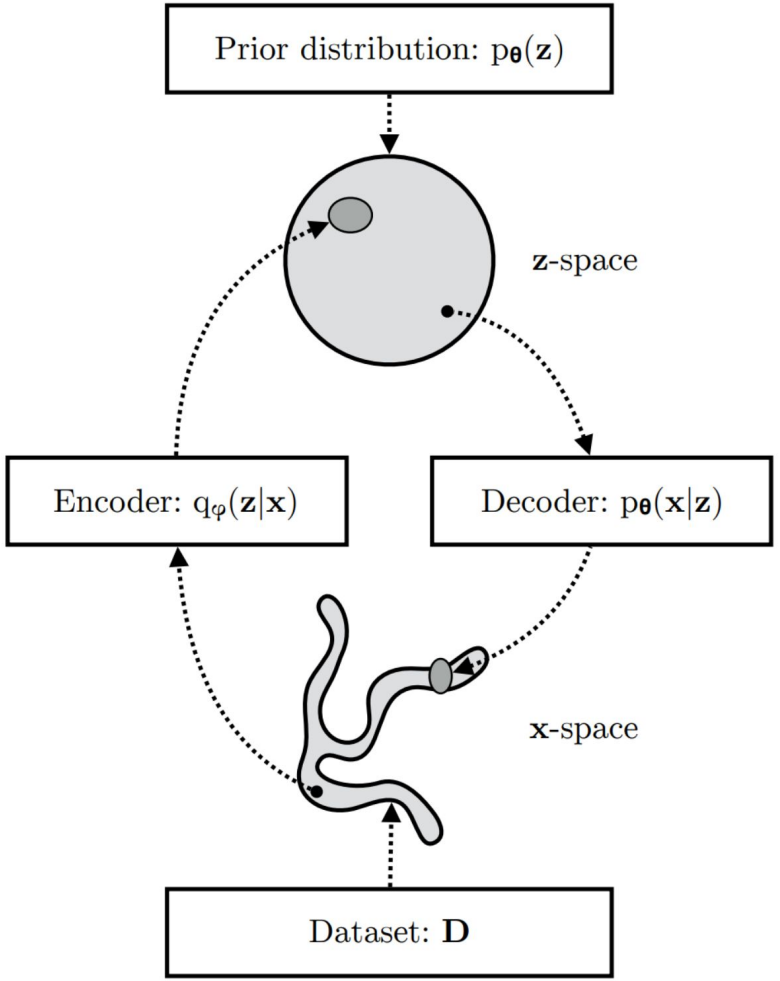
The double-loop optimization of variational inference is inefficient in the setup of deep learning (we need many iterations and many data points)

Idea: Learn one set of parameters that map each data point **x** to a posterior distribution over **z**

Use neural networks to parameterize a Gaussian distribution:

$$q_\phi(z \mid x) = \mathcal{N}(\mu_\phi(x), \mathrm{diag}(\sigma_\phi(x)))$$

Amortized inference



Training with the ELBO objective

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

$$\begin{aligned}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) &= \log p_{\boldsymbol{\theta}}(\mathbf{x}) - D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})) \\ &\leq \log p_{\boldsymbol{\theta}}(\mathbf{x})\end{aligned}$$

So the KL determines two ‘distances’:

1. Between the approximate posterior **q** and the true posterior.
2. Between the ELBO and the log-likelihood.

Optimizing the ELBO w.r.t. $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ we get both a better bound and a better model.

Optimizing w.r.t. θ

$$
\begin{aligned}
\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) &= \nabla_{\boldsymbol{\theta}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \left[\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \right] \\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \left[\nabla_{\boldsymbol{\theta}} (\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})) \right] \\
&\simeq \nabla_{\boldsymbol{\theta}} (\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})) \\
&= \nabla_{\boldsymbol{\theta}} (\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}))
\end{aligned}
$$

The expectation is approximated using Monte Carlo estimation:

z in the last two lines is a random sample from $\mathbf{q_{\phi}(z \mid x)}$

Optimizing w.r.t. ϕ

$$\begin{aligned}\nabla_{\phi}\mathcal{L}_{\theta,\phi}(\mathbf{x}) &= \nabla_{\phi}\mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})}\left[\log p_{\theta}(\mathbf{x},\mathbf{z}) - \log q_{\phi}(\mathbf{z}|\mathbf{x})\right] \\ &\neq \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})}\left[\nabla_{\phi}(\log p_{\theta}(\mathbf{x},\mathbf{z}) - \log q_{\phi}(\mathbf{z}|\mathbf{x}))\right]\end{aligned}$$

Harder to optimize because ϕ appears both inside the expectation, and in the probability that is used for the expectation.

Score function estimator

$$\nabla_{\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[f(\mathbf{z})\right] = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[f(\mathbf{z}) \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\right]$$
$$\simeq f(\mathbf{z}) \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$$

where $\mathbf{z} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$.

Also known as the likelihood-ratio estimator or REINFORCE.

We will not use this method here, and rather find a method to compute the gradients through the expectation probability.

Reparameterization trick

Idea: separate the parameters and the stochasticity in $\mathbf{q_\phi(z \mid x)}$

Such that a sample of **z** can be computed as:

$$\mathbf{z = g(\epsilon, \phi, x)}$$

Where $\boldsymbol{\epsilon}$ is random variable without parameters.

The expectation over **z** can be formulated as an expectation over $\boldsymbol{\epsilon}$.

Reparameterization trick

The gradient of the expectation becomes computable via Monte Carlo:

$$\nabla_{\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} [f(\mathbf{z})] = \nabla_{\boldsymbol{\phi}} \mathbb{E}_{p(\boldsymbol{\epsilon})} [f(\mathbf{z})] \tag{2.22}$$

$$= \mathbb{E}_{p(\boldsymbol{\epsilon})} [\nabla_{\boldsymbol{\phi}} f(\mathbf{z})] \tag{2.23}$$

$$\simeq \nabla_{\boldsymbol{\phi}} f(\mathbf{z}) \tag{2.24}$$

where in the last line, $\mathbf{z} = \mathbf{g}(\boldsymbol{\phi}, \mathbf{x}, \boldsymbol{\epsilon})$ with random noise sample $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$.

Optimizing the ELBO

$$\begin{aligned}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) &= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\right] \\ &= \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\right]\end{aligned}$$

where $\mathbf{z} = g(\boldsymbol{\epsilon}, \boldsymbol{\phi}, \mathbf{x})$.

Use Monte-Carlo estimator by sampling $\boldsymbol{\epsilon}$:

$$\begin{aligned}\boldsymbol{\epsilon} &\sim p(\boldsymbol{\epsilon}) \\ \mathbf{z} &= \mathbf{g}(\boldsymbol{\phi}, \mathbf{x}, \boldsymbol{\epsilon}) \\ \tilde{\mathcal{L}}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) &= \log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\end{aligned}$$

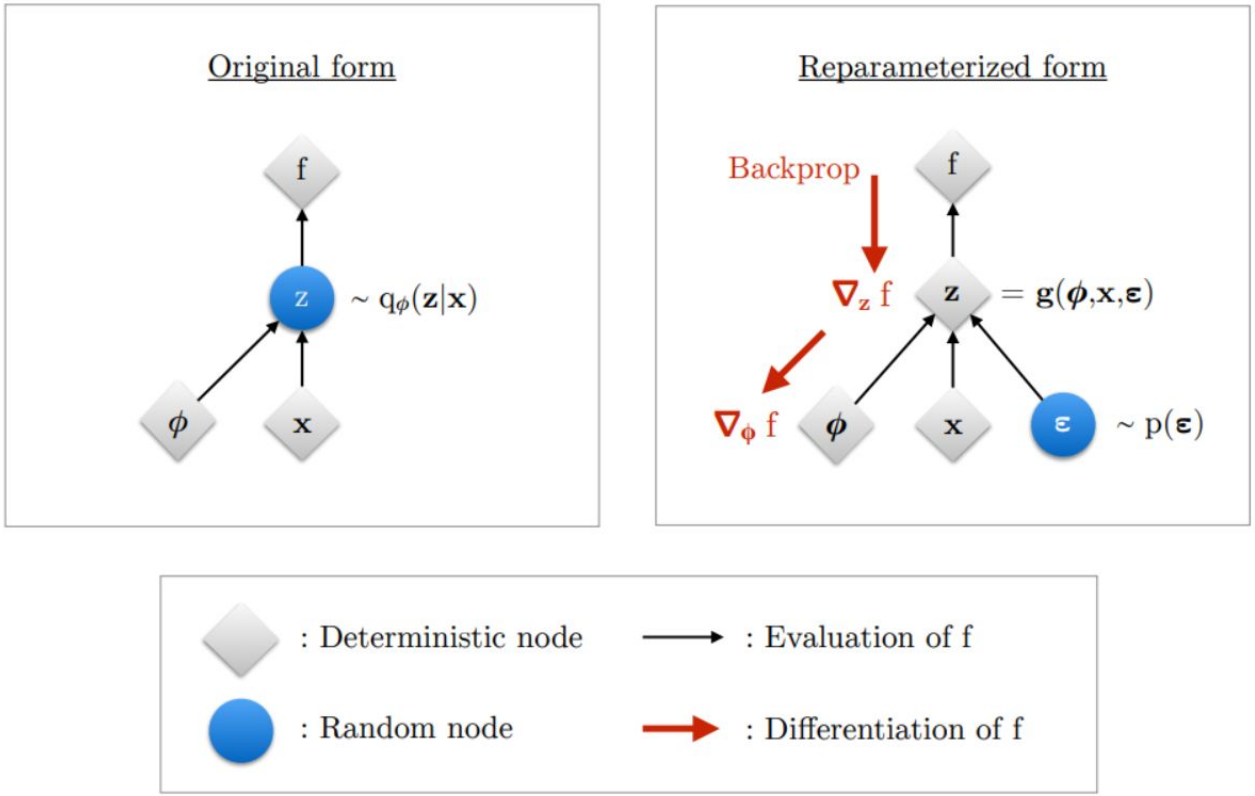
Reparameterization trick for Gaussians

$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}^2))$:

$$(\boldsymbol{\mu}, \log \boldsymbol{\sigma}) = \text{EncoderNeuralNet}_{\boldsymbol{\phi}}(\mathbf{x})$$
$$q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = \prod_i q_{\boldsymbol{\phi}}(z_i|\mathbf{x}) = \prod_i \mathcal{N}(z_i; \mu_i, \sigma_i^2)$$

After reparameterization:

$$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$$
$$(\boldsymbol{\mu}, \log \boldsymbol{\sigma}) = \text{EncoderNeuralNet}_{\boldsymbol{\phi}}(\mathbf{x})$$
$$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$$


Original form
f
z
$\sim q_\phi(\mathbf{z}|\mathbf{x})$
ϕ
x
Reparameterized form
Backprop
f
$\nabla_z f$
z
$= g(\phi,\mathbf{x},\varepsilon)$
$\nabla_\phi f$
ϕ
x
ε
$\sim p(\varepsilon)$
: Deterministic node
: Evaluation of f
: Random node
: Differentiation of f

Putting it all together - The Variational Autoencoder (VAE)

Data:

$\mathcal{D}$: Dataset

$q_{\phi}(\mathbf{z}|\mathbf{x})$: Inference model

$p_{\theta}(\mathbf{x}, \mathbf{z})$: Generative model

Result:

$\boldsymbol{\theta}, \boldsymbol{\phi}$: Learned parameters

$(\boldsymbol{\theta}, \boldsymbol{\phi}) \leftarrow$ Initialize parameters

while *SGD not converged* **do**

  $\mathcal{M} \sim \mathcal{D}$ (Random minibatch of data)

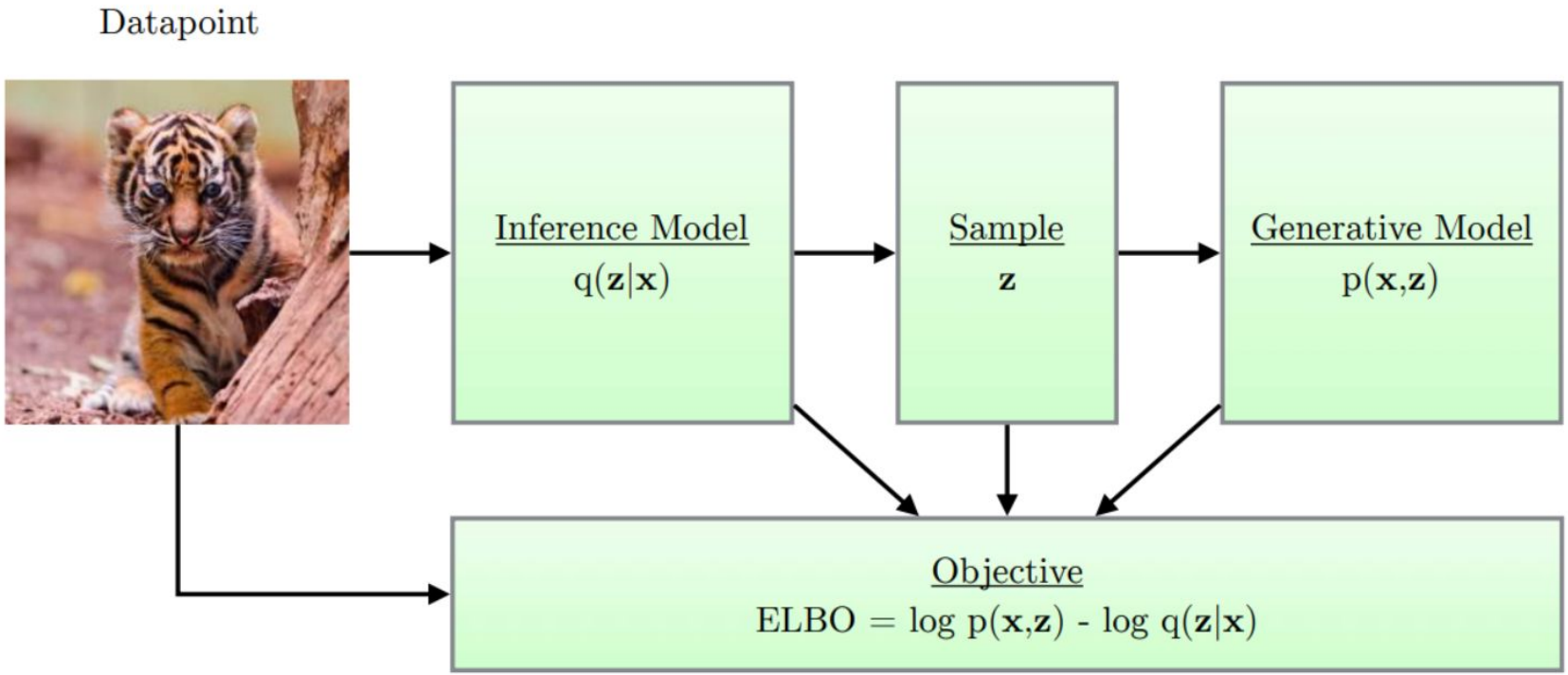
  $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$ (Random noise for every datapoint in $\mathcal{M}$)

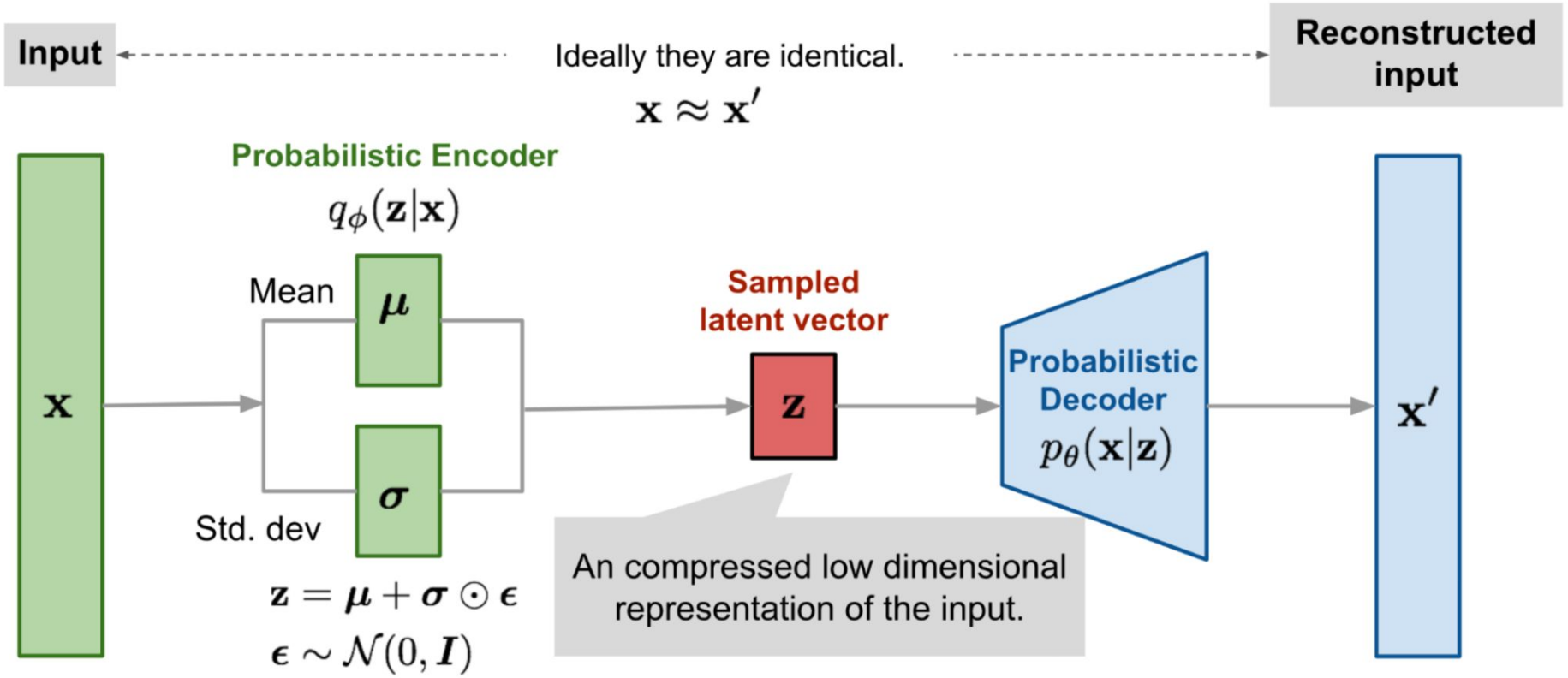
  Compute $\tilde{\mathcal{L}}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathcal{M}, \boldsymbol{\epsilon})$ and its gradients $\nabla_{\boldsymbol{\theta},\boldsymbol{\phi}} \tilde{\mathcal{L}}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathcal{M}, \boldsymbol{\epsilon})$

  Update $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ using SGD optimizer

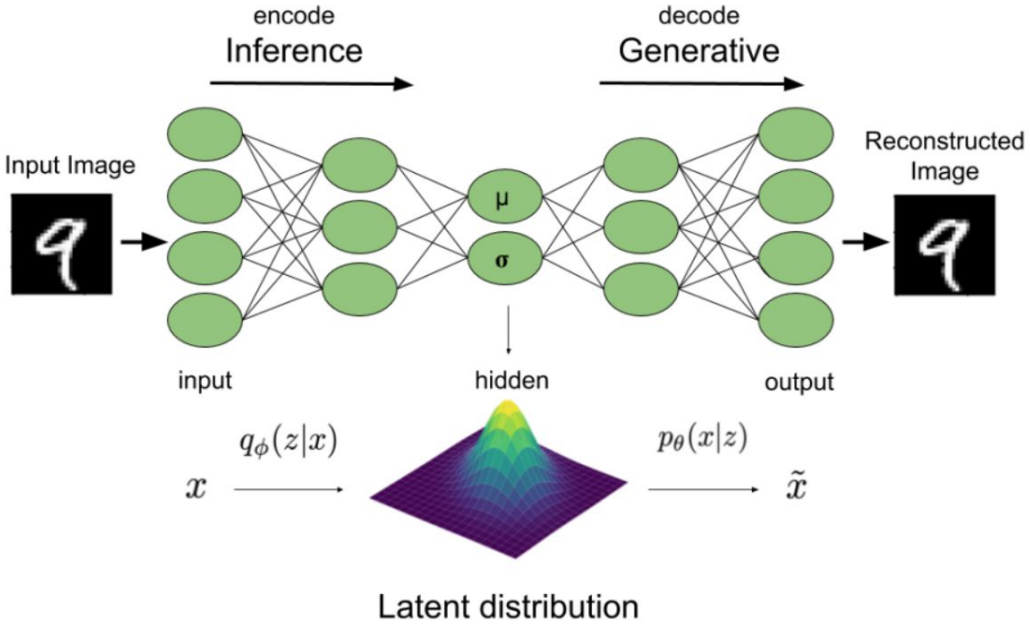
end

Putting it all together - The Variational Autoencoder (VAE)




Input
Ideally they are identical.
$\mathbf{x} \approx \mathbf{x}'$
Reconstructed input
Probabilistic Encoder
$q_\phi(\mathbf{z}|\mathbf{x})$
Mean
$\boldsymbol{\mu}$
$\boldsymbol{\sigma}$
Std. dev
$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$
$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$
$\mathbf{x}$
Sampled latent vector
$\mathbf{z}$
An compressed low dimensional representation of the input.
Probabilistic Decoder
$p_\theta(\mathbf{x}|\mathbf{z})$
$\mathbf{x}'$

Connection to autoencoders



$$E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z};\theta)] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))$$

Image: Volodymyr Kuleshov

Log-Likelihood for continuous data

- In contrast to discrete modeling, the log-likelihood of continuous data can be tricky to interpret.
- What happens to the log-likelihood when one pixel is always black?
- Need to consider this in training and evaluation:
    - Training: limit the variance from below
    - Evaluation: Add uniform noise [0,1/256] to the values.

KL between Gaussians

$$\int q_{\boldsymbol{\theta}}(\mathbf{z}) \log p(\mathbf{z})\, d\mathbf{z} = \int \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2) \log \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})\, d\mathbf{z}$$

$$= -\frac{J}{2} \log(2\pi) - \frac{1}{2} \sum_{j=1}^{J} (\mu_j^2 + \sigma_j^2)$$

And:

$$\int q_{\boldsymbol{\theta}}(\mathbf{z}) \log q_{\boldsymbol{\theta}}(\mathbf{z})\, d\mathbf{z} = \int \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2) \log \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2)\, d\mathbf{z}$$

$$= -\frac{J}{2} \log(2\pi) - \frac{1}{2} \sum_{j=1}^{J} (1 + \log \sigma_j^2)$$

Therefore:

$$-D_{KL}((q_{\boldsymbol{\phi}}(\mathbf{z}) || p_{\boldsymbol{\theta}}(\mathbf{z})) = \int q_{\boldsymbol{\theta}}(\mathbf{z}) \left(\log p_{\boldsymbol{\theta}}(\mathbf{z}) - \log q_{\boldsymbol{\theta}}(\mathbf{z})\right)\, d\mathbf{z}$$

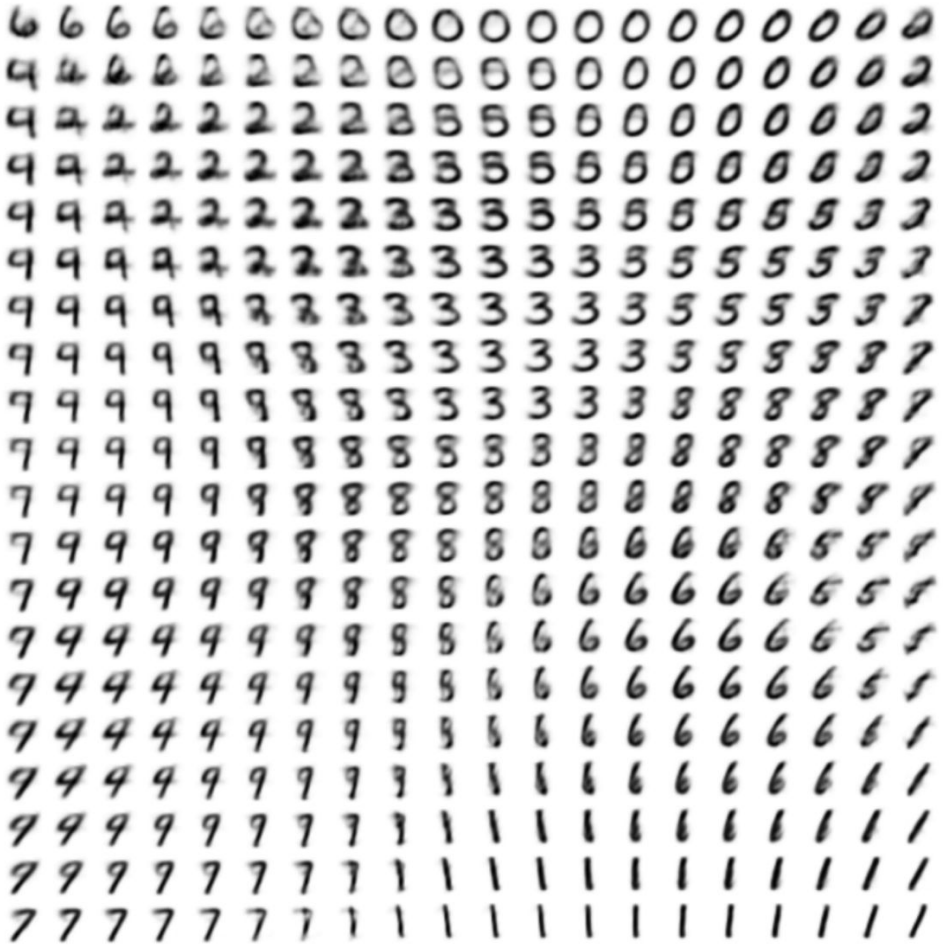
$$= \frac{1}{2} \sum_{j=1}^{J} \left(1 + \log((\sigma_j)^2) - (\mu_j)^2 - (\sigma_j)^2\right)$$

Results

(a) 2-D latent space (b) 5-D latent space (c) 10-D latent space (d) 20-D latent space

Figure 5: Random samples from learned generative models of MNIST for different dimensionalities of latent space.

2D latents



Results

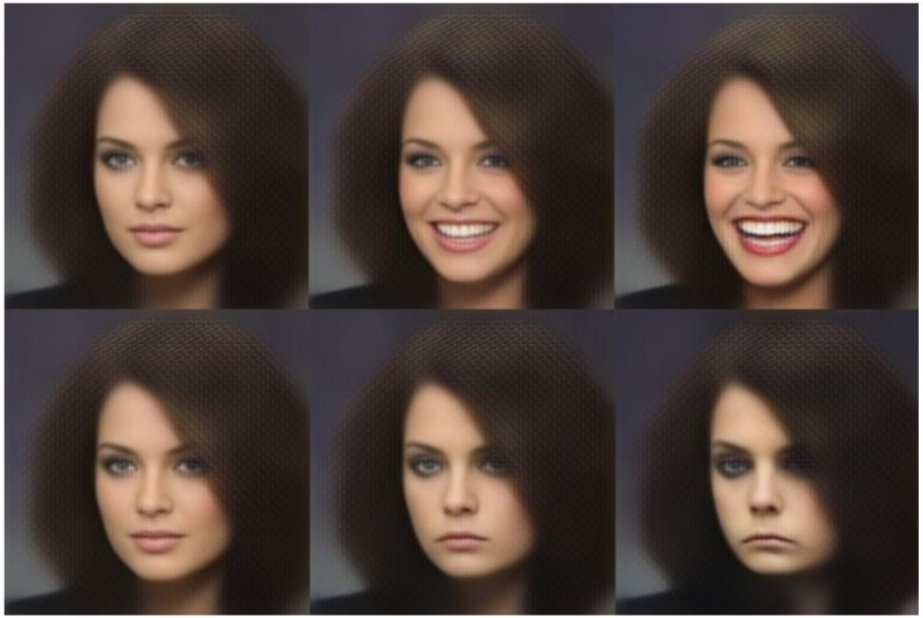
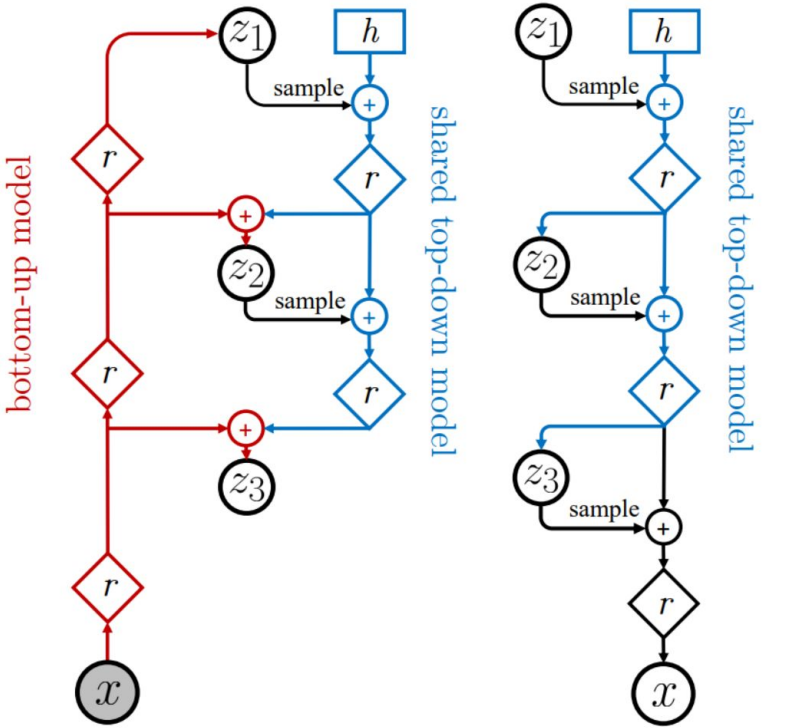


Figure 4.4: VAEs can be used for image resynthesis. In this example by White, 2016, an original image (left) is modified in a latent space in the direction of a *smile vector*, producing a range of versions of the original, from smiling to sadness.

Extension - Hierarchical VAE

NVAE



(a) Bidirectional Encoder (b) Generative Model

Figure 2: The neural networks implementing an encoder $q(\boldsymbol{z}|\boldsymbol{x})$ and generative model $p(\boldsymbol{x}, \boldsymbol{z})$ for a 3-group hierarchical VAE. ◇ denotes residual neural networks, ⊕ denotes feature combination (e.g., concatenation), and [h] is a trainable parameter.

(a) MNIST ($t = 1.0$)

(b) CIFAR-10 ($t = 0.7$)

(c) CelebA 64 ($t = 0.6$)

(d) CelebA HQ ($t = 0.6$)

(e) FFHQ ($t = 0.5$)

Negative log-likelihood

bit/dimension for all datasets except for MNIST which is in nats.

Method	MNIST 28×28	CIFAR-10 32×32	ImageNet 32×32	CelebA 64×64	CelebA HQ 256×256	FFHQ 256×256
NVAE w/o flow	**78.01**	2.93	-	2.04	-	0.71
NVAE w/ flow	78.19	**2.91**	3.92	**2.03**	**0.70**	**0.69**
VAE Models with an Unconditional Decoder						
BIVA [36]	78.41	3.08	3.96	2.48	-	-
IAF-VAE [4]	79.10	3.11	-	-	-	-
DVAE++ [20]	78.49	3.38	-	-	-	-
Conv Draw [42]	-	3.58	4.40	-	-	-
Flow Models without any Autoregressive Components in the Generative Model						
VFlow [59]	-	2.98	-	-	-	-
ANF [60]	-	3.05	3.92	-	0.72	-
Flow++ [61]	-	3.08	**3.86**	-	-	-
Residual flow [50]	-	3.28	4.01	-	0.99	-
GLOW [62]	-	3.35	4.09	-	1.03	-
Real NVP [63]	-	3.49	4.28	3.02	-	-
VAE and Flow Models with Autoregressive Components in the Generative Model						
δ-VAE [25]	-	2.83	3.77	-	-	-
PixelVAE++ [35]	78.00	2.90	-	-	-	-
VampPrior [64]	78.45	-	-	-	-	-
MAE [65]	77.98	2.95	-	-	-	-
Lossy VAE [66]	78.53	2.95	-	-	-	-
MaCow [67]	-	3.16	-	-	0.67	-
Autoregressive Models						
SPN [68]	-	-	3.85	-	0.61	-
PixelSNAIL [34]	-	2.85	3.80	-	-	-
Image Transformer [69]	-	2.90	3.77	-	-	-
PixelCNN++ [70]	-	2.92	-	-	-	-
PixelRNN [41]	-	3.00	3.86	-	-	-
Gated PixelCNN [71]	-	3.03	3.83	-	-	-

Summary

- Pros:
  - Simple implementation
  - Easy to sample from (ancestral sampling from prior and decoder)
  - Latent representation can be inferred for new data points using q
  - Can be used to approximate a bound over p(x)
- Cons:
  - Simple versions do not lead to very good results (a hard optimization problem)
  - Cannot be used to compute p(x) exactly
  - When performance is bad not clear if because of q or p