Deep generative models with latent variables

Brooks Paige

12 Nov 2025

Goals

Topics covered in these slides include

- why we might want to learn generative models
- what a deep generative model is, and how to perform inference in a variational autoencoder
- two different Monte Carlo estimators for gradients of expectations, and the reparameterization trick
- what some applications and extensions are for variational autoencoders.

Generative models and deep generative models

Most machine learning models you've run into are probably discriminative models for supervised learning.

- **Data**: pairs of instances and labels $(\mathbf{x}_i, y_i)$, for $i = 1, \ldots, N$.
- **Goal**: estimate $f_\theta(\mathbf{x}_i) \approx y_i$, or maximize $p_\theta(y_i|\mathbf{x}_i)$.

The learning task is to characterize the conditional distribution of **labels** y_i given **data** $\mathbf{x}_i$. We typically do this by estimating a parameter θ to minimize prediction error on the training set, or (equivalently) to maximize the training log-likelihood.

Generative models and deep generative models

In contrast, generative models also estimate the distribution of the data $\mathbf{x}_i$. This means we estimate a model for $p(\mathbf{x}_i)$, or the joint distribution $p(\mathbf{x}_i, y_i)$.

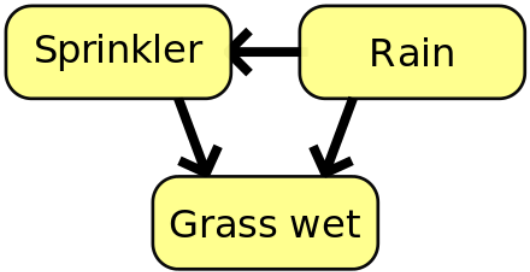
- **Data**: pairs of instances and labels $(\mathbf{x}_i, y_i)$, for $i = 1, \ldots, N$.
- **Generative model**: learn an approximation to $p_\theta(\mathbf{x}_i)$ or $p_\theta(\mathbf{x}_i, y_i)$.
- **Discriminative model**: learn an approximation to $p_\theta(y_i|\mathbf{x}_i)$.

Why use generative models?

Q: Why do this? If all we care about is prediction, this sounds like a lot of extra work!

Why use generative models?

One traditional answer: a joint model allows us to capture causal relationships and reason about how different unknown quantities relate to observed data.



Data: the grass is wet. **Prediction task**: Is it raining?

From generative models to discriminative models

$$p(x \mid y) = p(y \mid x)p(x)/p(y)$$

Posterior Likelihood Prior

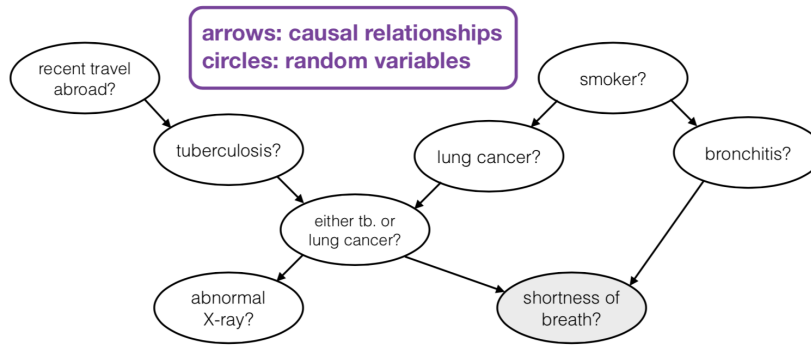
Bayes' rule relates joint distributions with conditional distributions.

- We can use Bayes' rule to run the generative model "backward" to do inference over unknown quantities (e.g. class labels, regression targets).

Latent variable modeling in medical diagnostics

Data: Patient exhibits shortness of breath.

Prediction task: Do they have tuberculosis?



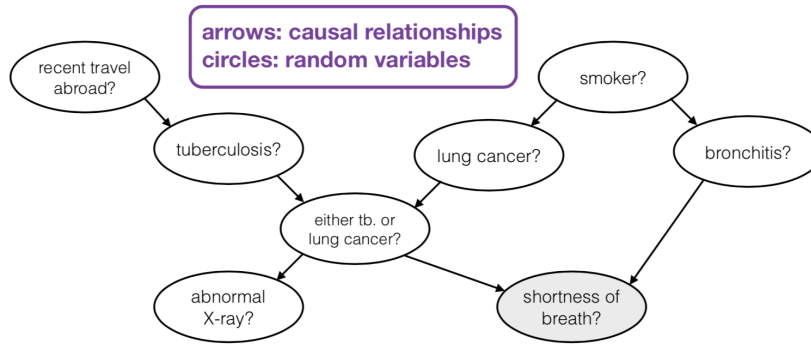
Lauritzen & Spiegelhalter (1988)

Latent variable modeling in medical diagnostics

Data: Patient exhibits shortness of breath.

Prediction task: Do they have tuberculosis?

Question: Should we run an X-ray?

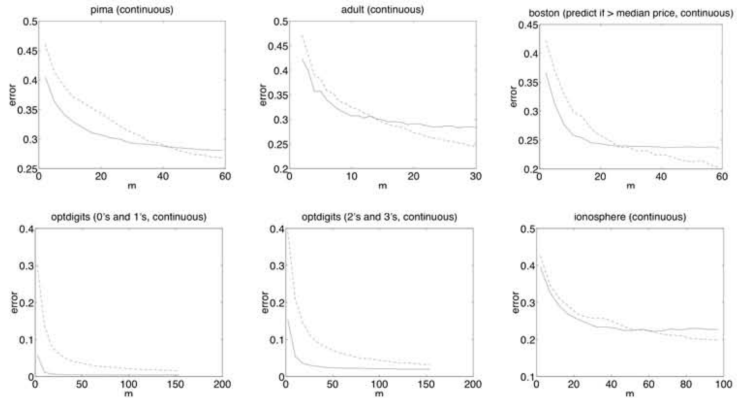


Lauritzen & Spiegelhalter (1988)

A tale of two binary linear classifiers

Logistic regression: learn $p(y|\mathbf{x})$

Gaussian naïve Bayes: learn $p(y)$, $p(\mathbf{x}|y=+1)$, and $p(\mathbf{x}|y=-1)$



Two distinct regimes! (**Small data** vs **Large data limit**...)

Ng & Jordan (2002)

Learning linear latent variable models

There's a long history of learning **linear** latent variable models directly from data, including

- Principal components analysis (PCA)
- Independent components analysis (ICA)
- Factor analysis
- Non-negative matrix factorization
- ...

Typically, these correspond to learning a low-rank factorization $\mathbf{X} \approx \mathbf{WV}$, where each instance $\mathbf{x}_i$ is a weighted sum of dictionary elements.

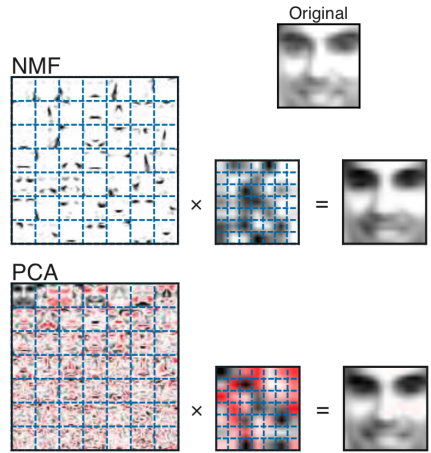


Figure: Hastie et al.

Probabilistic PCA (teaser)

While PCA is often introduced as a means for finding projections that explain the most possible variance, and estimated using SVD, PCA also can be formulated as a probabilistic generative model:

$$p(\mathbf{z}_i) = \mathcal{N}(\mathbf{z}_i|\mathbf{0}, \mathbf{I}) \qquad p(\mathbf{x}_i|\mathbf{z}_i) = \mathcal{N}(\mathbf{x}_i|\mathbf{W}\mathbf{z}_i + \boldsymbol{\mu}, \sigma^2\mathbf{I}),$$

for $i = 1, \ldots, N$, $\mathbf{z}_i \in \mathbb{R}^K$ and $\mathbf{x}_i \in \mathbb{R}^D$, with $K < D$. This model has a tractable marginal likelihood

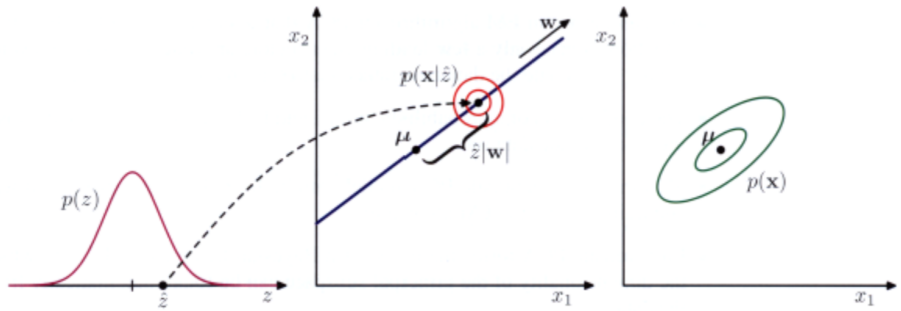
$$p(\mathbf{x}_i) = \int p(\mathbf{x}_i|\mathbf{z}_i)p(\mathbf{z}_i)d\mathbf{z}_i) = \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}, \mathbf{M}),$$

where $\mathbf{M} = \mathbf{W}\mathbf{W}^\top + \sigma^2\mathbf{I}$, and posterior distribution

$$p(\mathbf{z}_i|\mathbf{x}_i) = \mathcal{N}(\mathbf{z}_i|\mathbf{M}^{-1}\mathbf{W}^\top(\mathbf{x} - \boldsymbol{\mu}), \sigma^{-2}\mathbf{M}).$$

Roweis (1998), Tipping & Bishop (1999)

Probabilistic PCA: Generative viewpoint



This is a $K = 1$ dimensional latent space for $D = 2$ dimensional data. Data is generated by sampling a value of $\mathbf{z}$, projecting it up into 2d space with a $\mathbf{x} = \mathbf{W}\mathbf{z} + \boldsymbol{\mu}$, and adding noise σ^2. The rightmost plot shows the marginal density on $\mathbf{x}$.

Figure: Bishop (2006)

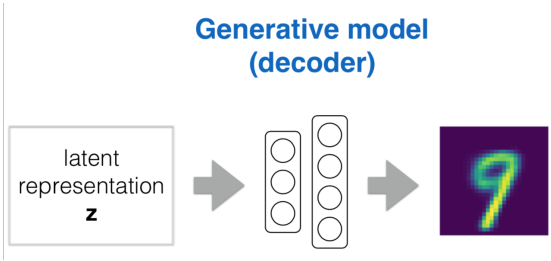
Auto-encoding variational Bayes

Variational Auto-Encoders

Now let's consider a deep generative model which has a Gaussian prior, and a likelihood defined by a deep net, e.g.

$$p(\mathbf{z}_i) = \mathcal{N}(\mathbf{z}_i|\mathbf{0}, \mathbf{I}) \qquad p(\mathbf{x}_i|\mathbf{z}_i) = \mathcal{N}(\mathbf{x}_i|f_\theta(\mathbf{z}_i), \sigma^2\mathbf{I}),$$

for $i = 1, \ldots, N$. As before, assume $\mathbf{z}_i \in \mathbb{R}^K$ and $\mathbf{x}_i \in \mathbb{R}^D$, with $K < D$.



Computing the marginal likelihood

Challenge: In this model, the marginal likelihood $p(\mathbf{X})$ is completely intractable.

$$\log p(\mathbf{X}) = \sum_{i=1}^{N} \log p(\mathbf{x}_i) = \sum_{i=1}^{N} \log \int p(\mathbf{x}_i|\mathbf{z}_i, \theta) p(\mathbf{z}_i) d\mathbf{z}_i$$

In the linear model, we had closed forms for both the marginal likelihood terms $p(\mathbf{x}_i)$ and the posterior distribution $p(\mathbf{z}_i|\mathbf{x}_i)$.

- If we have a nonlinear f_θ in $p(\mathbf{x}_i|\mathbf{z}_i) = \mathcal{N}(\mathbf{x}_i|f_\theta(\mathbf{z}_i), \sigma^2\mathbf{I})$, neither of these are the case.

Learning an approximate posterior

We will use an approximate inference approach known as **Variational Bayes**.

- Variational Bayes turns an inference problem into an optimization problem, by defining an approximating family of distributions $q_\phi(\mathbf{z}|\mathbf{x})$, parameterized by ϕ
- Inference entails finding a value of ϕ that makes $q_\phi(\mathbf{z}|\mathbf{x})$ "close" to the true posterior $p_\theta(\mathbf{z}|\mathbf{x})$

We now have **two questions** to answer:

1. What is a good choice of $q_\phi(\mathbf{z}|\mathbf{x})$?
2. What is an appropriate notion of "closeness" between distributions?

Amortized inference networks

What is a good choice of $q_\phi(\mathbf{z}|\mathbf{x})$? The typical choice is to use a Gaussian distribution whose mean and variance are output by **another** deep network, with

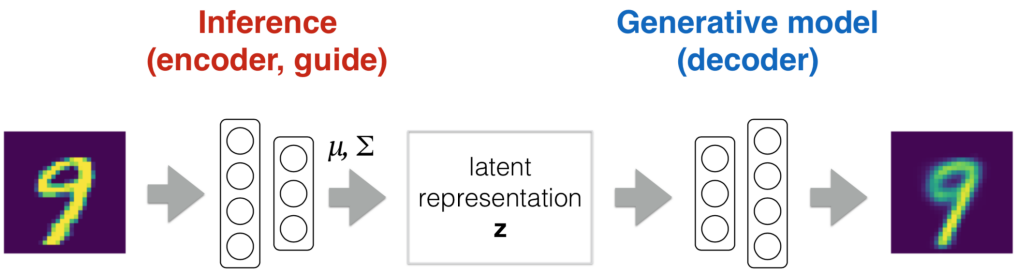
$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\mu(\mathbf{x}), \mathrm{diag}(\sigma(\mathbf{x})^2))$$

where

- $\mu : \mathbb{R}^D \to \mathbb{R}^K$ is a deep net which outputs a posterior mean, and
- $\sigma : \mathbb{R}^D \to \mathbb{R}^K$ is a deep net which outputs a posterior standard deviation.

Note that in practice the network for σ typically will output the log of the standard deviation, which is then exponentiated, in order to enforce a positivity constraint.

High-level diagram: autoencoder interpretation



The Kullback-Leibler divergence

The **Kullback-Leibler (KL) divergence** is a natural notion of "close":

$$D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x})) = \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})} d\mathbf{z}.$$

- $D_{KL}(q_\phi\|p_\theta) \geq 0$ for all q_ϕ, p_θ
- It takes a value 0 if and only if the two distributions are identical
- Note: it is **not symmetric** — it is a divergence measure, but not a distance.

The evidence lower bound (ELBO)

We can re-arrange the KL divergence to find a **tractable objective**,

$$D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x})) = \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})} d\mathbf{z}$$

The evidence lower bound (ELBO)

We can re-arrange the KL divergence to find a **tractable objective**,

$$
\begin{aligned}
D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x})) &= \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})} d\mathbf{z} \\
&= \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{x}) q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{x}, \mathbf{z})} d\mathbf{z}
\end{aligned}
$$

The evidence lower bound (ELBO)

We can re-arrange the KL divergence to find a **tractable objective**,

$$
\begin{aligned}
D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x})) &= \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})} d\mathbf{z} \\
&= \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{x}) q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{x}, \mathbf{z})} d\mathbf{z} \\
&= \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{x}, \mathbf{z})} d\mathbf{z} + \int q_\phi(\mathbf{z}|\mathbf{x}) \log p_\theta(\mathbf{x}) d\mathbf{z}
\end{aligned}
$$

The evidence lower bound (ELBO)

We can re-arrange the KL divergence to find a **tractable objective**,

$$
\begin{aligned}
D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x})) &= \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})} d\mathbf{z} \\
&= \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{x}) q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{x},\mathbf{z})} d\mathbf{z} \\
&= \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{x},\mathbf{z})} d\mathbf{z} + \int q_\phi(\mathbf{z}|\mathbf{x}) \log p_\theta(\mathbf{x}) d\mathbf{z} \\
&= \underbrace{-\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right]}_{\text{ELBO, } \mathcal{L}(\mathbf{x};\phi,\theta)} + \log p_\theta(\mathbf{x}).
\end{aligned}
$$

The evidence lower bound (ELBO)

We can re-arrange the KL divergence to find a **tractable objective**,

$$
\begin{aligned}
D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x})) &= \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})} d\mathbf{z} \\
&= \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{x}) q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{x}, \mathbf{z})} d\mathbf{z} \\
&= \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{x}, \mathbf{z})} d\mathbf{z} + \int q_\phi(\mathbf{z}|\mathbf{x}) \log p_\theta(\mathbf{x}) d\mathbf{z} \\
&= \underbrace{-\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right]}_{\text{ELBO, } \mathcal{L}(\mathbf{x};\phi,\theta)} + \log p_\theta(\mathbf{x}).
\end{aligned}
$$

The ELBO is tractable in the sense that it only includes the **joint distribution** $p_\theta(\mathbf{x}, \mathbf{z})$, not the (intractable) posterior $p(\mathbf{z}|\mathbf{x})$ or marginal likelihood $p_\theta(\mathbf{x})$.

The evidence lower bound (ELBO)

Re-arranging, we have

$$\log p_\theta(\mathbf{x}) = \mathcal{L}(\mathbf{x}; \phi, \theta) + D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})).$$

This is a very useful relationship between the KL-divergence, the marginal likelihood $p(\mathbf{x})$, and the ELBO

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

The evidence lower bound (ELBO)

Re-arranging, we have

$$\log p_\theta(\mathbf{x}) = \mathcal{L}(\mathbf{x}; \phi, \theta) + D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})).$$

Notably, since

- D_{KL} is **lower-bounded by zero**, and
- $\log p_\theta(\mathbf{x})$ is **constant** with respect to ϕ,

then, for the parameter ϕ, **maximizing the ELBO** is equivalent to **minimizing the KL divergence**.

The evidence lower bound (ELBO)

What about θ? Again,

$$\log p_\theta(\mathbf{x}) = \mathcal{L}(\mathbf{x}; \phi, \theta) + D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})).$$

Our desired objective function is the marginal likelihood $\log p_\theta(\mathbf{x})$, which we wish to maximize. However, again $D_{KL} \geq 0$ means we have

$$\mathcal{L}(\mathbf{x}; \phi, \theta) \leq \log p_\theta(\mathbf{x});$$

that is, the ELBO is a lower-bound on the log marginal likelihood, which we can use as a surrogate and maximize instead.

Note: This lower bound is tight when the KL divergence is zero, i.e. if we have learned the correct posterior!

Optimizing the ELBO

So far, we have seen that maximizing the ELBO

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

is a sensible objective function for both

- the generative parameters θ, and
- the inference network parameters ϕ.

But, we haven't discussed **how** to maximize the ELBO.

We'll use stochastic gradient descent, with one trick.

Estimators for gradients of expectations

Computing the gradient of the ELBO requires computing the gradient of an expectation. Is that a problem?

Let's step back from our problem for a moment and consider trying to compute the derivative of the expectation of a simple scalar function $f(x)$ under the distribution $p(x|\theta)$,

$$\frac{\partial}{\partial\theta}\mathbb{E}_{p(x|\theta)}[f(x)].$$

We will make use of the identity

$$p(x|\theta)\frac{\partial}{\partial\theta}\log p(x|\theta) = \frac{\partial}{\partial\theta}p(x|\theta).$$

Score-function estimator

The following derivation leads to the so-called **score-function estimator**:

$$\frac{\partial}{\partial \theta} \mathbb{E}_{p(x|\theta)}[f(x)]$$

Schulman et al., 2015

Score-function estimator

The following derivation leads to the so-called **score-function estimator**:

$$\frac{\partial}{\partial\theta}\mathbb{E}_{p(x|\theta)}[f(x)] = \frac{\partial}{\partial\theta}\int p(x|\theta)f(x)dx$$

Schulman et al., 2015

Score-function estimator

The following derivation leads to the so-called **score-function estimator**:

$$\begin{aligned}\frac{\partial}{\partial\theta}\mathbb{E}_{p(x|\theta)}[f(x)] &= \frac{\partial}{\partial\theta}\int p(x|\theta)f(x)dx \\ &= \int \frac{\partial}{\partial\theta}p(x|\theta)f(x)dx\end{aligned}$$

Schulman et al., 2015

Score-function estimator

The following derivation leads to the so-called **score-function estimator**:

$$\begin{aligned}\frac{\partial}{\partial\theta}\mathbb{E}_{p(x|\theta)}[f(x)] &= \frac{\partial}{\partial\theta}\int p(x|\theta)f(x)dx \\ &= \int \frac{\partial}{\partial\theta}p(x|\theta)f(x)dx \\ &= \int f(x)\frac{\partial}{\partial\theta}p(x|\theta)dx\end{aligned}$$

Schulman et al., 2015

Score-function estimator

The following derivation leads to the so-called **score-function estimator**:

$$\begin{aligned}
\frac{\partial}{\partial\theta}\mathbb{E}_{p(x|\theta)}[f(x)] &= \frac{\partial}{\partial\theta}\int p(x|\theta)f(x)dx \\
&= \int \frac{\partial}{\partial\theta}p(x|\theta)f(x)dx \\
&= \int f(x)\frac{\partial}{\partial\theta}p(x|\theta)dx \\
&= \int f(x)p(x|\theta)\frac{\partial}{\partial\theta}\log p(x|\theta)dx
\end{aligned}$$

Schulman et al., 2015

Score-function estimator

The following derivation leads to the so-called **score-function estimator**:

$$
\begin{aligned}
\frac{\partial}{\partial\theta}\mathbb{E}_{p(x|\theta)}[f(x)] &= \frac{\partial}{\partial\theta}\int p(x|\theta)f(x)dx \\
&= \int \frac{\partial}{\partial\theta}p(x|\theta)f(x)dx \\
&= \int f(x)\frac{\partial}{\partial\theta}p(x|\theta)dx \\
&= \int f(x)p(x|\theta)\frac{\partial}{\partial\theta}\log p(x|\theta)dx \\
&= \mathbb{E}_{p(x|\theta)}\left[f(x)\frac{\partial}{\partial\theta}\log p(x|\theta)\right]
\end{aligned}
$$

Schulman et al., 2015

Score-function estimator

That identity, and its Monte Carlo approximation,

$$\frac{\partial}{\partial\theta}\mathbb{E}_{p(x|\theta)}[f(x)] = \mathbb{E}_{p(x|\theta)}\left[f(x)\frac{\partial}{\partial\theta}\log p(x|\theta)\right]$$
$$\approx \frac{1}{S}\sum_{i=1}^{S} f(x^{(s)})\frac{\partial}{\partial\theta}\log p(x^{(s)}|\theta), \text{for } x^{(s)} \sim p(x|\theta)$$

holds even if $f(x)$ is non-differentiable, or x is discrete!

Unfortunately, it is often too high-variance to be useful, requiring many, many samples.

Schulman et al., 2015

Score-function estimator

That identity, and its Monte Carlo approximation,

$$\frac{\partial}{\partial\theta}\mathbb{E}_{p(x|\theta)}[f(x)] = \mathbb{E}_{p(x|\theta)}\left[f(x)\frac{\partial}{\partial\theta}\log p(x|\theta)\right]$$
$$\approx \frac{1}{S}\sum_{i=1}^{S} f(x^{(s)})\frac{\partial}{\partial\theta}\log p(x^{(s)}|\theta), \text{for } x^{(s)} \sim p(x|\theta)$$

holds even if $f(x)$ is non-differentiable, or x is discrete!

Unfortunately, it is often too high-variance to be useful, requiring many, many samples.

Problem: both the gradient, and the expectation, involved the same variable θ.

Schulman et al., 2015

Pathwise derivative estimator

Consider an alternative setting in which we have a separate random variable $\epsilon \sim p(\epsilon)$, and $x = x(\theta, \epsilon)$ is a deterministic function, e.g.

$$\frac{\partial}{\partial \theta} \mathbb{E}_{p(\epsilon)}[f(x(\theta, \epsilon))]$$

.

Schulman et al., 2015

Pathwise derivative estimator

Consider an alternative setting in which we have a separate random variable $\epsilon \sim p(\epsilon)$, and $x = x(\theta, \epsilon)$ is a deterministic function, e.g.

$$\frac{\partial}{\partial \theta} \mathbb{E}_{p(\epsilon)}[f(x(\theta, \epsilon))] = \frac{\partial}{\partial \theta} \int p(\epsilon) f(x(\theta, \epsilon)) dx$$

.

Schulman et al., 2015

Pathwise derivative estimator

Consider an alternative setting in which we have a separate random variable $\epsilon \sim p(\epsilon)$, and $x = x(\theta, \epsilon)$ is a deterministic function, e.g.

$$\begin{aligned}\frac{\partial}{\partial\theta}\mathbb{E}_{p(\epsilon)}[f(x(\theta,\epsilon))] &= \frac{\partial}{\partial\theta}\int p(\epsilon) f(x(\theta,\epsilon))dx \\ &= \int p(\epsilon)\frac{\partial}{\partial\theta} f(x(\theta,\epsilon))dx\end{aligned}$$

.

Schulman et al., 2015

Pathwise derivative estimator

Consider an alternative setting in which we have a separate random variable $\epsilon \sim p(\epsilon)$, and $x = x(\theta, \epsilon)$ is a deterministic function, e.g.

$$\begin{aligned}\frac{\partial}{\partial\theta}\mathbb{E}_{p(\epsilon)}[f(x(\theta,\epsilon))] &= \frac{\partial}{\partial\theta}\int p(\epsilon) f(x(\theta,\epsilon))dx \\ &= \int p(\epsilon)\frac{\partial}{\partial\theta} f(x(\theta,\epsilon))dx \\ &= \mathbb{E}_{p(\epsilon)}\left[\frac{\partial}{\partial\theta} f(x(\theta,\epsilon))\right].\end{aligned}$$

Schulman et al., 2015

Pathwise derivative estimator

Consider an alternative setting in which we have a separate random variable $\epsilon \sim p(\epsilon)$, and $x = x(\theta, \epsilon)$ is a deterministic function, e.g.

$$\begin{aligned}\frac{\partial}{\partial\theta}\mathbb{E}_{p(\epsilon)}[f(x(\theta,\epsilon))] &= \frac{\partial}{\partial\theta}\int p(\epsilon) f(x(\theta,\epsilon))dx \\ &= \int p(\epsilon)\frac{\partial}{\partial\theta} f(x(\theta,\epsilon))dx \\ &= \mathbb{E}_{p(\epsilon)}\left[\frac{\partial}{\partial\theta} f(x(\theta,\epsilon))\right].\end{aligned}$$

Here, the derivative of the expectation is the expectation of the derivative!

- While this does require $f(\cdot)$ and $x(\theta, \epsilon)$ to be differentiable, it has the advantage of leading to much lower-variance Monte Carlo estimators.

Schulman et al., 2015

Estimating gradients of the ELBO

Now, back to our ELBO. First, we will further decompose it as

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

Kingma & Welling, 2014

Estimating gradients of the ELBO

Now, back to our ELBO. First, we will further decompose it as

$$
\begin{aligned}
\mathcal{L}(\mathbf{x};\phi,\theta) &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z}) + \log p(\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right] \\
&= \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]}_{\textbf{reconstruction term}} - \underbrace{D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\textbf{regularization term}}
\end{aligned}
$$

Kingma & Welling, 2014

Estimating gradients of the ELBO

Now, back to our ELBO. First, we will further decompose it as

$$\begin{aligned}
\mathcal{L}(\mathbf{x};\phi,\theta) &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z}) + \log p(\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right] \\
&= \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]}_{\textbf{reconstruction term}} - \underbrace{D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\textbf{regularization term}}
\end{aligned}$$

You will often see the ELBO written this way. This is (partially) because $D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))$ can often be computed in closed form.

Kingma & Welling, 2014

Estimating gradients of the ELBO

Let's look at the **reconstruction term** in the ELBO, $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]$.

- Gradients ∇_θ: not a problem,

$$\nabla_\theta \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\nabla_\theta \log p_\theta(\mathbf{x}|\mathbf{z})]$$

- Gradients ∇_ϕ: **???**

We will use the **reparameterization trick**: define a function $g_\phi(\epsilon, \mathbf{x})$ and distribution $p(\epsilon)$ such that if

$$\epsilon \sim p(\epsilon), \qquad \tilde{\mathbf{z}} = g_\phi(\epsilon, \mathbf{x})$$

then $\tilde{\mathbf{z}} \sim q_\phi(\mathbf{z}|\mathbf{x})$.

Kingma & Welling, 2014

Reparameterization trick

Define a function $g_\phi(\epsilon, \mathbf{x})$ and distribution $p(\epsilon)$ such that if

$$\epsilon \sim p(\epsilon), \qquad \tilde{z} = g_\phi(\epsilon, \mathbf{x})$$

then $\tilde{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$.

Example: suppose

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\mu(\mathbf{x}), \mathrm{diag}(\sigma(\mathbf{x})^2)).$$

Then let

$$p(\epsilon) = \mathcal{N}(0, \mathbf{I})$$
$$\tilde{\mathbf{z}} = g_\phi(\epsilon, \mathbf{x}) = \mu(\mathbf{x}) + \sigma(\mathbf{x}) \odot \epsilon.$$

Kingma & Welling, 2014

Reparameterization trick

Define a function $g_\phi(\epsilon, \mathbf{x})$ and distribution $p(\epsilon)$ such that if

$$\epsilon \sim p(\epsilon), \qquad \tilde{z} = g_\phi(\epsilon, \mathbf{x})$$

then $\tilde{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$.

Example: suppose

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\mu(\mathbf{x}), \mathrm{diag}(\sigma(\mathbf{x})^2)).$$

Then let

$$p(\epsilon) = \mathcal{N}(0, \mathbf{I})$$
$$\tilde{\mathbf{z}} = g_\phi(\epsilon, \mathbf{x}) = \mu(\mathbf{x}) + \sigma(\mathbf{x}) \odot \epsilon.$$

Similar expressions hold for many common probability densities — we are **not limited to Gaussian** latent variables.

Kingma & Welling, 2014

Reparameterization trick

Using this "reparameterization trick", we can compute the gradient

$$\begin{aligned}\nabla_{\phi,\theta}\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] &= \nabla_{\phi,\theta}\mathbb{E}_{p(\epsilon)}[\log p_\theta(\mathbf{x}|\tilde{\mathbf{z}} = g_\phi(\epsilon, \mathbf{x}))] \\ &= \mathbb{E}_{p(\epsilon)}[\nabla_{\phi,\theta} \log p_\theta(\mathbf{x}|\tilde{\mathbf{z}} = g_\phi(\epsilon, \mathbf{x}))]\end{aligned}$$

This can be computed by **automatic differentiation**.

Note that if $p_\theta(\mathbf{x}|\mathbf{z})$ is Gaussian, e.g. $\mathcal{N}(\mathbf{x}|f_\theta(\mathbf{z}), \sigma^2\mathbf{I})$, then this term involves differentiating the squared error loss

$$\log p_\theta(\mathbf{x}|\tilde{\mathbf{z}} = g_\phi(\epsilon, \mathbf{x})) = -\frac{1}{2\sigma^2}\|\mathbf{x} - f_\theta(g_\phi(\epsilon, \mathbf{x}))\|_2^2 + \text{const.}$$

Kingma & Welling, 2014

Mini-batch gradient estimation

The last missing step is considering mini-batch gradient updates. Since

$$\log p_\theta(\mathbf{X}) = \sum_{i=1}^{N} \log p_\theta(\mathbf{x}_i) \geq \sum_{i=1}^{N} \mathcal{L}(\mathbf{x}_i; \phi, \theta)$$

then we can use a minibatch estimator

$$\sum_{i=1}^{N} \mathcal{L}(\mathbf{x}_i; \phi, \theta) \approx \frac{N}{M} \sum_{j=1}^{M} \mathcal{L}(\mathbf{x}_j; \phi, \theta),$$

where $\mathbf{x}_j$ are M random samples from the dataset.

Note: This is unbiased if $\mathbf{x}_i$ are i.i.d. samples from the underlying $\tilde{p}(\mathbf{x})$.

Summary

A **variational autoencoder** is

- a deep generative model for data $\mathbf{x}$, with a latent variable $\mathbf{z}$
- ...which takes the form

$$p_\theta(\mathbf{X}, \mathbf{Z}) = \prod_{i=1}^{N} p_\theta(\mathbf{x}_i|\mathbf{z}_i)p(\mathbf{z}_i)$$

- ...trained jointly with an approximation to the posterior

$$q_\phi(\mathbf{z}_i|\mathbf{x}_i) \approx p_\theta(\mathbf{z}_i|\mathbf{x}_i)$$

- ...by maximizing an ELBO.

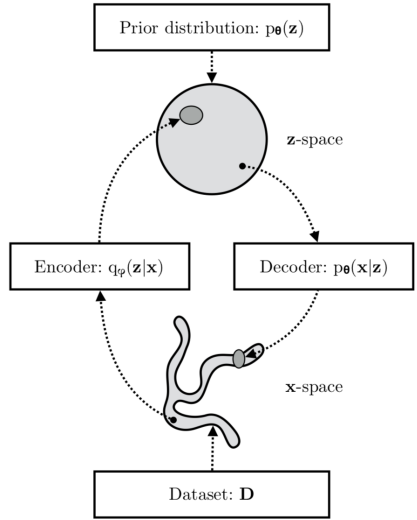


Figure: Kingma & Welling (2019)

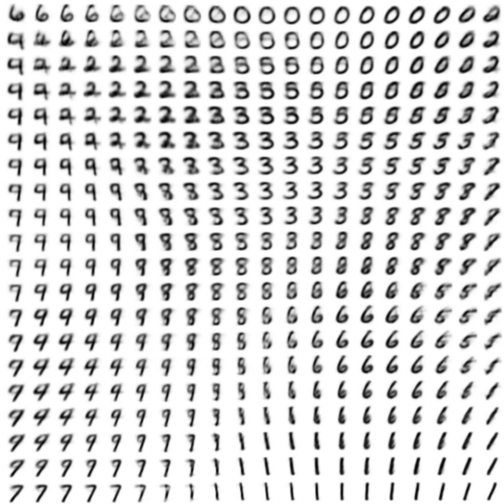
Examples

The "hello world" of machine learning

MNIST dataset.

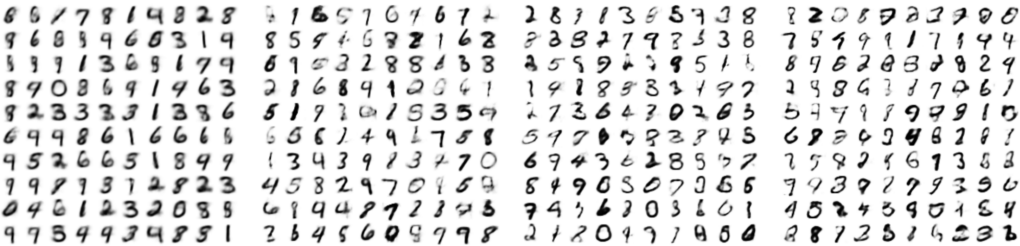
- **Encoder**: feed-forward network, Gaussian posterior
- **Decoder**: feed-forward network, Bernoulli or Gaussian likelihood
- 2-dimensional latent space $\mathbf{z}$

Plot shows the mean of the decoder for different values of the latent $\mathbf{z}$.



Kingma & Welling (2014)

MNIST: latent space comparison



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

- Non-obvious question: how big should the latent space be?

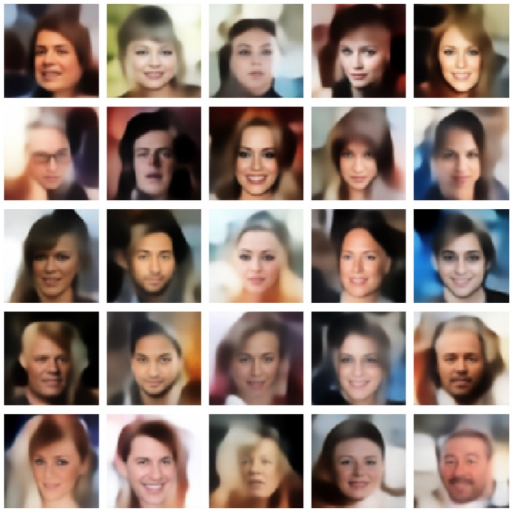
Kingma & Welling (2014)

Convolutional networks

CelebA dataset.

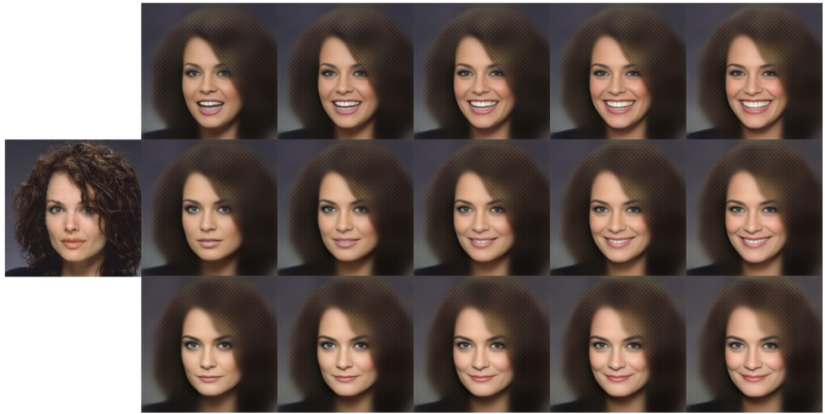
- **Encoder**: Fully convolutional network, Gaussian posterior
- **Decoder**: Fully convolutional network, Gaussian likelihood

Note: recent papers with different architectures and likelihoods can produce more photorealistic random samples.



Lamb et al. (2016)

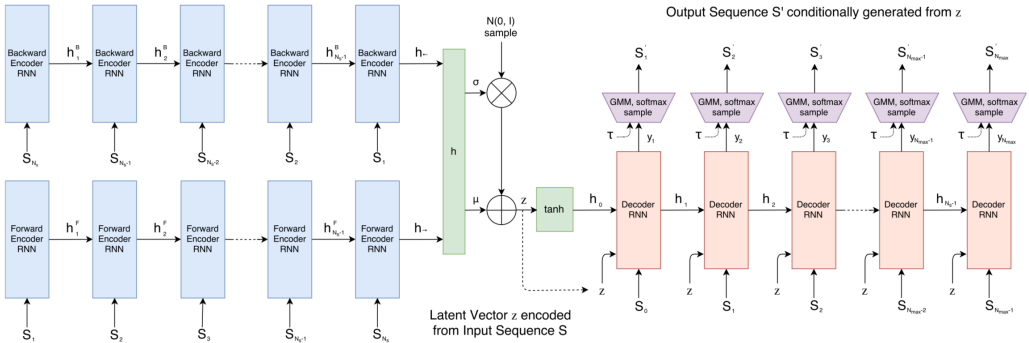
Perturbations in latent space



- **Uses**: We can identify directions in the latent space that correspond to properties, and manipulate them

White (2016)

A sequence model: SketchRNN



- **Encoder**: bi-directional LSTM, Gaussian posterior
- **Decoder**: LSTM that outputs parameters of a distribution over 5-tuples $(\Delta i, \Delta j,$ "pen up", "pen down", "stop"$)$

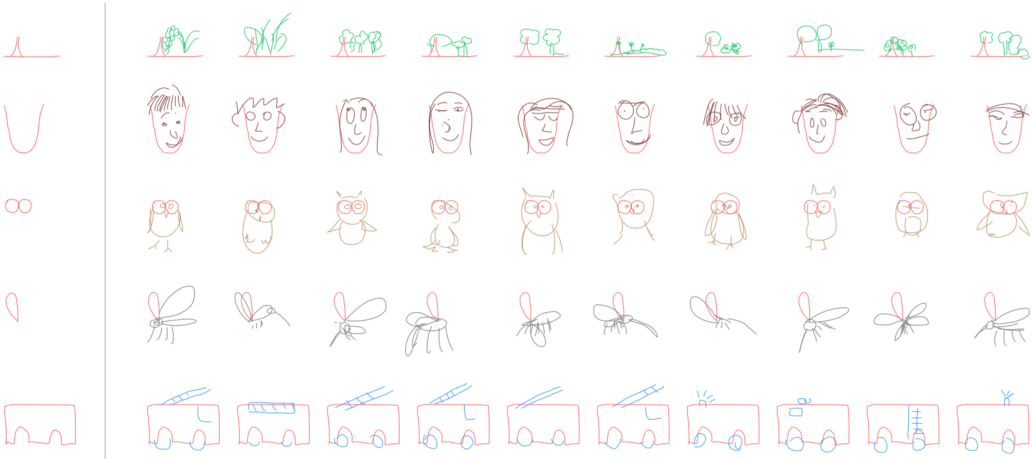
Ha & Eck (2017)

A sequence model: SketchRNN

- **Uses**: Reconstruction, generation, latent space manipulation, ...

Ha & Eck (2017)

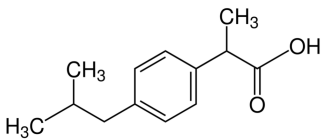
A sequence model: SketchRNN



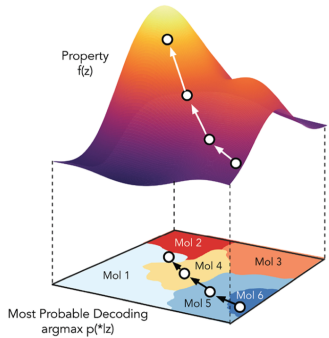
- **Uses**: Automatic completion of partial sketches

Ha & Eck (2017)

Sequence model for molecules



Ibuprofen: CC(C)CC1=CC=C(C=C1)C(C)C(=O)O



- Molecules are complex, discrete objects, but can be described succinctly as strings in a formal language
- **Uses**: Lift discrete optimization to a continuous latent space

Gomez-Bombarelli et al. (2018)

Application:

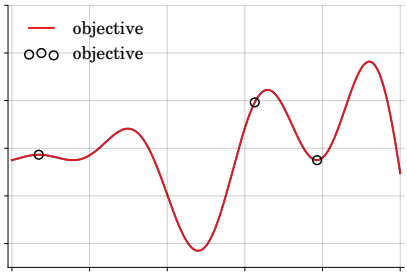
Latent space optimisation

Bayesian optimization cartoon

1. Get initial sample.

Figures: Miguel Hernández-Lobato

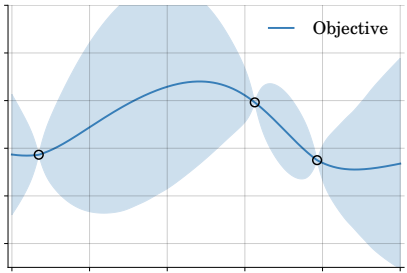
Bayesian optimization cartoon



❶ **Get initial sample.**

Figures: Miguel Hernández-Lobato

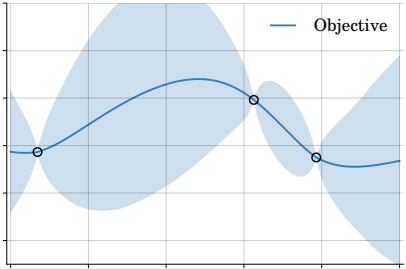
Bayesian optimization cartoon



1. Get initial sample.
2. **Fit a model to the data:**
$$p(y|\mathbf{x}, \mathcal{D}_n)\,.$$
3. Select data collection strategy:
$$\alpha(\mathbf{x}) = \mathbf{E}_{p(y|\mathbf{x},\mathcal{D}_n)}[U(y|\mathbf{x}, \mathcal{D}_n)]\,.$$

Figures: Miguel Hernández-Lobato

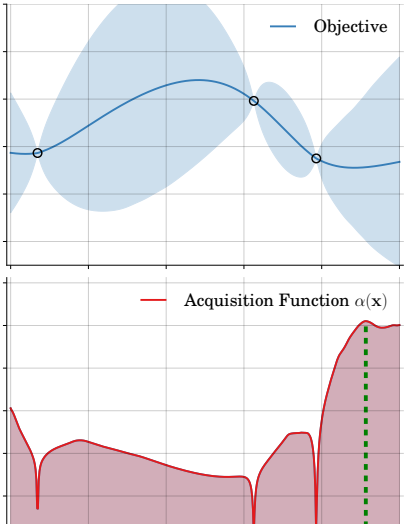
Bayesian optimization cartoon



1. Get initial sample.
2. Fit a model to the data:
$$p(y|\mathbf{x}, \mathcal{D}_n)\,.$$
3. **Select data collection strategy:**
$$\alpha(\mathbf{x}) = \mathbf{E}_{p(y|\mathbf{x},\mathcal{D}_n)}[U(y|\mathbf{x}, \mathcal{D}_n)]\,.$$
4. Optimize acquisition function $\alpha(\mathbf{x})$.

Figures: Miguel Hernández-Lobato

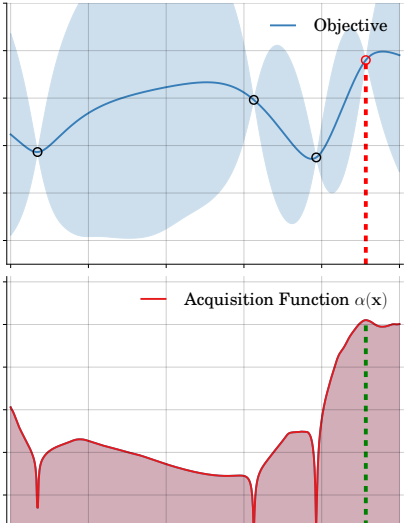
Bayesian optimization cartoon



1. Get initial sample.
2. Fit a model to the data:
$$p(y|\mathbf{x}, \mathcal{D}_n)\,.$$
3. Select data collection strategy:
$$\alpha(\mathbf{x}) = \mathbf{E}_{p(y|\mathbf{x},\mathcal{D}_n)}[U(y|\mathbf{x}, \mathcal{D}_n)]\,.$$
4. **Optimize acquisition function $\alpha(\mathbf{x})$.**
5. Collect data and update model.

Figures: Miguel Hernández-Lobato

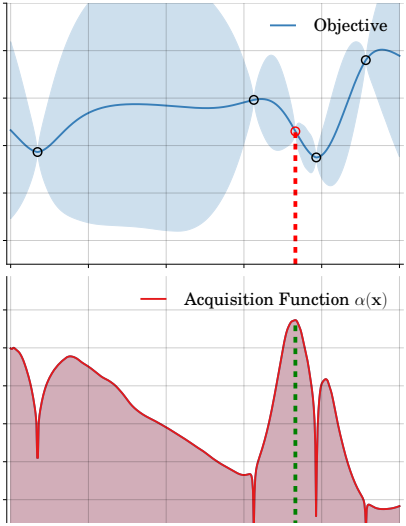
Bayesian optimization cartoon



1. Get initial sample.
2. Fit a model to the data:
$$p(y|\mathbf{x}, \mathcal{D}_n)\,.$$
3. Select data collection strategy:
$$\alpha(\mathbf{x}) = \mathbf{E}_{p(y|\mathbf{x},\mathcal{D}_n)}[U(y|\mathbf{x}, \mathcal{D}_n)]\,.$$
4. Optimize acquisition function $\alpha(\mathbf{x})$.
5. **Collect data and update model.**
6. Repeat!

Figures: Miguel Hernández-Lobato

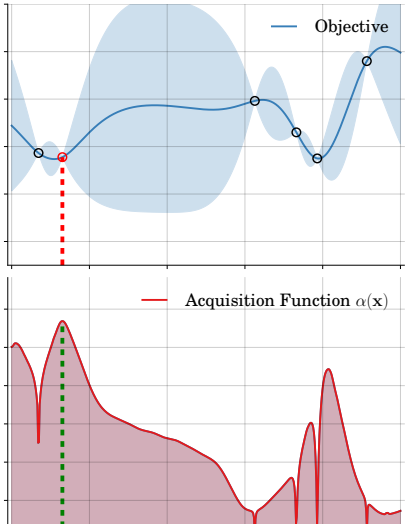
Bayesian optimization cartoon



1. Get initial sample.
2. Fit a model to the data:
$$p(y|\mathbf{x}, \mathcal{D}_n)\,.$$
3. **Select data collection strategy:**
$$\alpha(\mathbf{x}) = \mathbf{E}_{p(y|\mathbf{x},\mathcal{D}_n)}[U(y|\mathbf{x}, \mathcal{D}_n)]\,.$$
4. **Optimize acquisition function $\alpha(\mathbf{x})$.**
5. **Collect data and update model.**
6. **Repeat!**

Figures: Miguel Hernández-Lobato

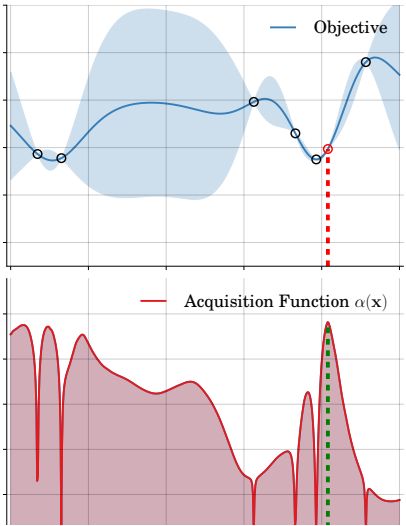
Bayesian optimization cartoon



1. Get initial sample.
2. Fit a model to the data:
$$p(y|\mathbf{x}, \mathcal{D}_n)\,.$$
3. **Select data collection strategy:**
$$\alpha(\mathbf{x}) = \mathbf{E}_{p(y|\mathbf{x},\mathcal{D}_n)}[U(y|\mathbf{x}, \mathcal{D}_n)]\,.$$
4. **Optimize acquisition function $\alpha(\mathbf{x})$.**
5. **Collect data and update model.**
6. **Repeat!**

Figures: Miguel Hernández-Lobato

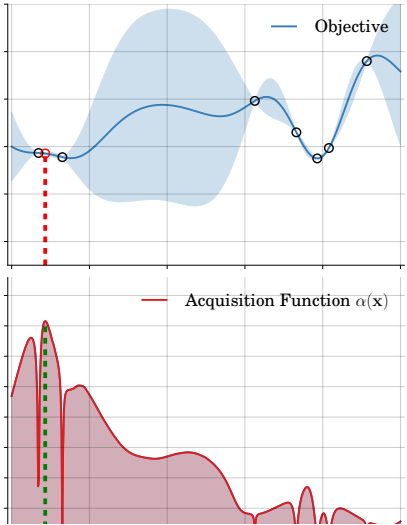
Bayesian optimization cartoon



1. Get initial sample.
2. Fit a model to the data:
$$p(y|\mathbf{x}, \mathcal{D}_n)\,.$$
3. **Select data collection strategy:**
$$\alpha(\mathbf{x}) = \mathbf{E}_{p(y|\mathbf{x},\mathcal{D}_n)}[U(y|\mathbf{x}, \mathcal{D}_n)]\,.$$
4. **Optimize acquisition function $\alpha(\mathbf{x})$.**
5. **Collect data and update model.**
6. **Repeat!**

Figures: Miguel Hernández-Lobato

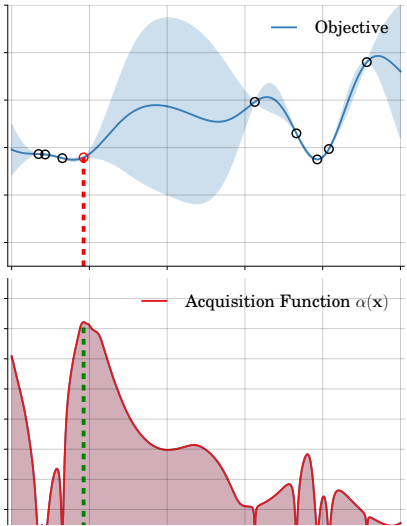
Bayesian optimization cartoon



1. Get initial sample.
2. Fit a model to the data:
$$p(y|\mathbf{x}, \mathcal{D}_n)\,.$$
3. **Select data collection strategy:**
$$\alpha(\mathbf{x}) = \mathbf{E}_{p(y|\mathbf{x},\mathcal{D}_n)}[U(y|\mathbf{x}, \mathcal{D}_n)]\,.$$
4. **Optimize acquisition function $\alpha(\mathbf{x})$.**
5. **Collect data and update model.**
6. **Repeat!**

Figures: Miguel Hernández-Lobato

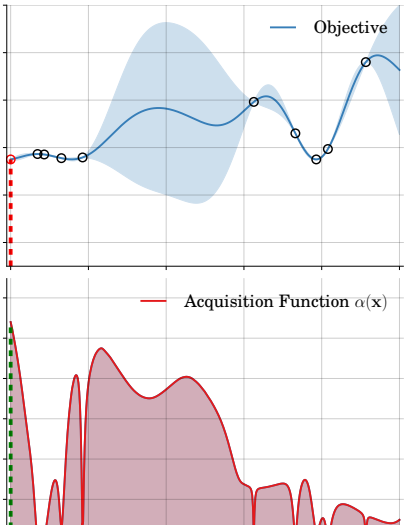
Bayesian optimization cartoon



1. Get initial sample.
2. Fit a model to the data:
$$p(y|\mathbf{x}, \mathcal{D}_n)\,.$$
3. **Select data collection strategy:**
$$\alpha(\mathbf{x}) = \mathbf{E}_{p(y|\mathbf{x},\mathcal{D}_n)}[U(y|\mathbf{x}, \mathcal{D}_n)]\,.$$
4. **Optimize acquisition function $\alpha(\mathbf{x})$.**
5. **Collect data and update model.**
6. **Repeat!**

Figures: Miguel Hernández-Lobato

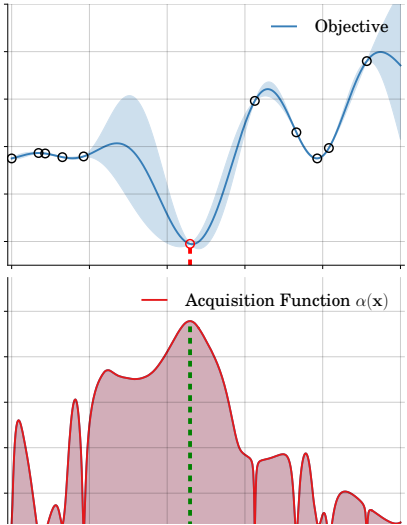
Bayesian optimization cartoon



1. Get initial sample.
2. Fit a model to the data:
$$p(y|\mathbf{x}, \mathcal{D}_n)\,.$$
3. **Select data collection strategy:**
$$\alpha(\mathbf{x}) = \mathbf{E}_{p(y|\mathbf{x},\mathcal{D}_n)}[U(y|\mathbf{x}, \mathcal{D}_n)]\,.$$
4. **Optimize acquisition function $\alpha(\mathbf{x})$.**
5. **Collect data and update model.**
6. **Repeat!**

Figures: Miguel Hernández-Lobato

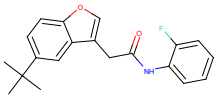
Bayesian optimization cartoon



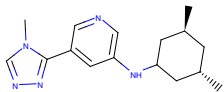
1. Get initial sample.
2. Fit a model to the data:
$$p(y|\mathbf{x}, \mathcal{D}_n)\,.$$
3. **Select data collection strategy:**
$$\alpha(\mathbf{x}) = \mathbf{E}_{p(y|\mathbf{x},\mathcal{D}_n)}[U(y|\mathbf{x}, \mathcal{D}_n)]\,.$$
4. **Optimize acquisition function $\alpha(\mathbf{x})$.**
5. **Collect data and update model.**
6. **Repeat!**

Figures: Miguel Hernández-Lobato

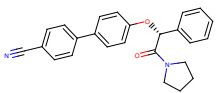
Representation: Molecules as text



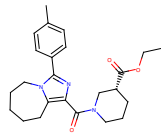
CC(C)(C)c1ccc2occ(CC(=O)Nc3ccccc3F)c2c1



C[C@@H]1CC(Nc2cncc(-c3nncn3C)c2)C[C@@H](C)C1

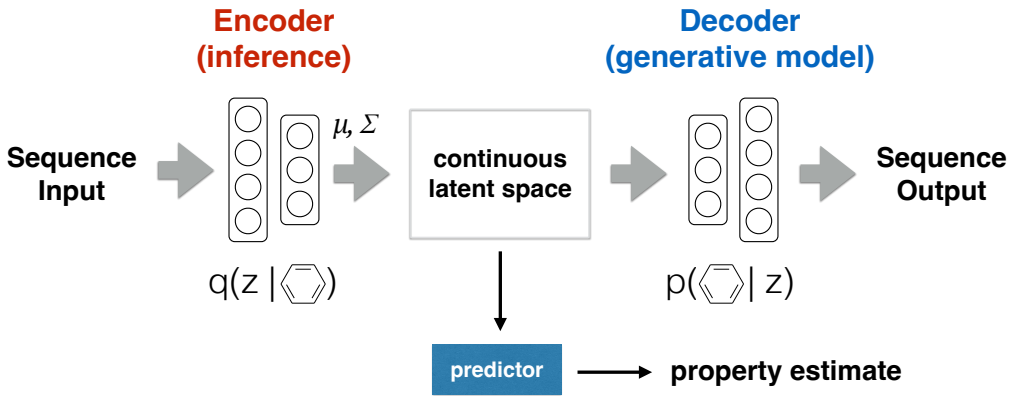


N#Cc1ccc(-c2ccc(O[C@@H](C(=O)N3CCCC3)c3ccccc3)cc2)cc1



CCOC(=O)[C@@H]1CCCN(C(=O)c2nc(-c3ccc(C)cc3)n3c2CCCCC3)C1

Variational autoencoder for "text"



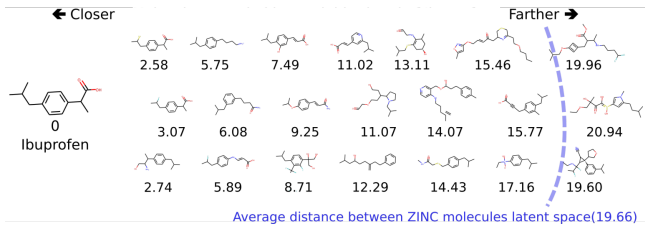
Gómez-Bombarelli et al, 2018

A deep generative model for molecules

- Sampled molecules have statistics similar to real molecules

source	data set	logP	SAS	QED
Data	ZINC	2.46 (1.43)	3.05 (0.83)	0.73 (0.14)
GA	ZINC	2.84 (1.86)	3.80 (1.01)	0.57 (0.20)
VAE	ZINC	2.67 (1.46)	3.18 (0.86)	0.70 (0.14)
Data	QM9	0.30 (1.00)	4.25 (0.94)	0.48 (0.07)
GA	QM9	0.96 (1.53)	4.47 (1.01)	0.53 (0.13)
VAE	QM9	0.30 (0.97)	4.34 (0.98)	0.47 (0.08)

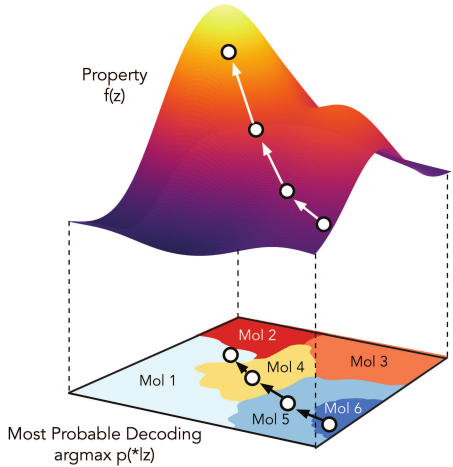
- Nearby latent representations decode into similar molecules



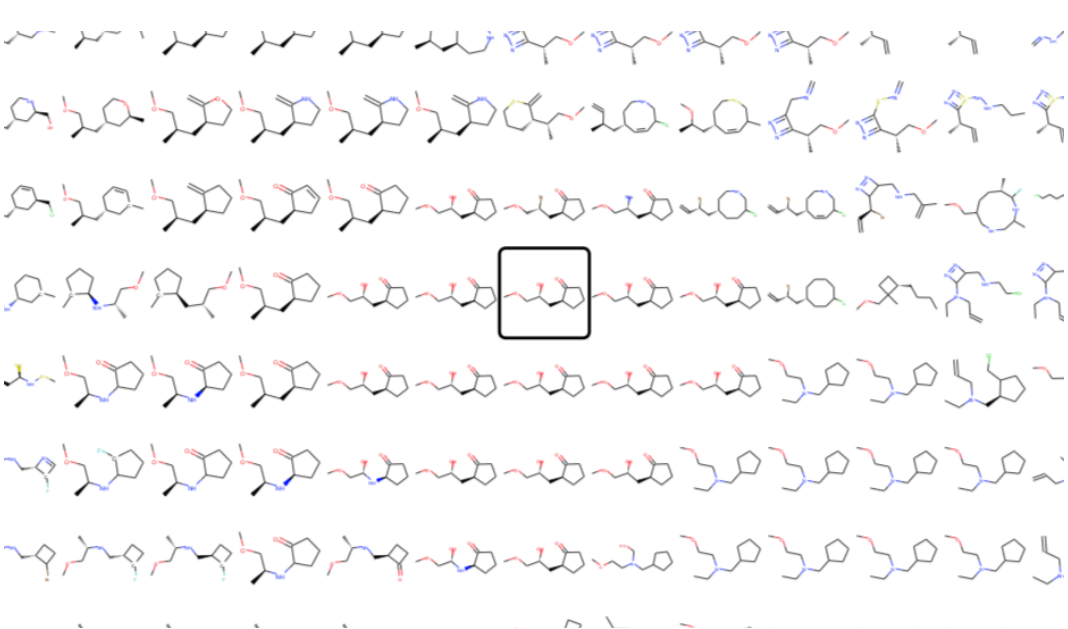
Gómez-Bombarelli et al, 2018

A deep generative model for molecules

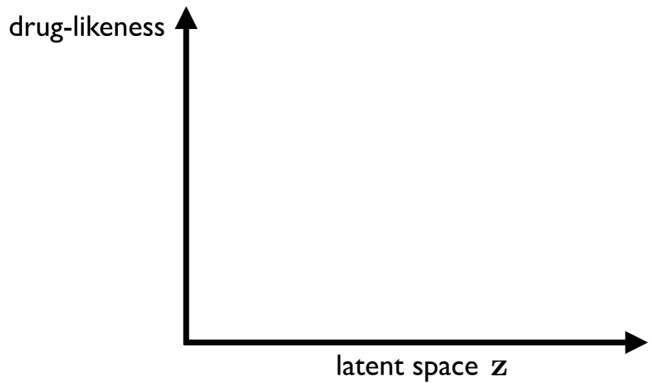
- Learn a “map” for molecules
- Now it is possible to apply continuous optimization methods (including Bayesian optimization)
- This can also be used for local, gradient-based optimization



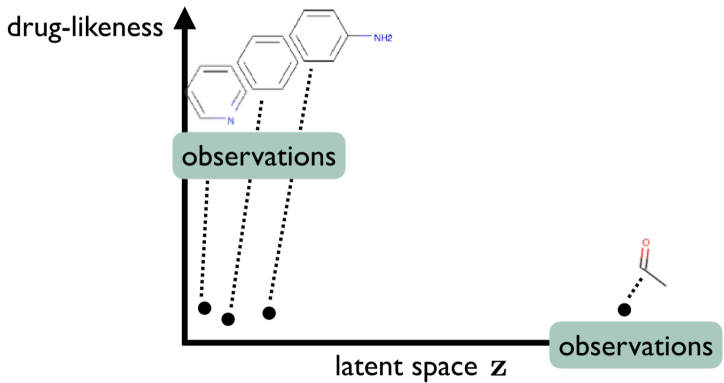
Gómez-Bombarelli et al, 2018



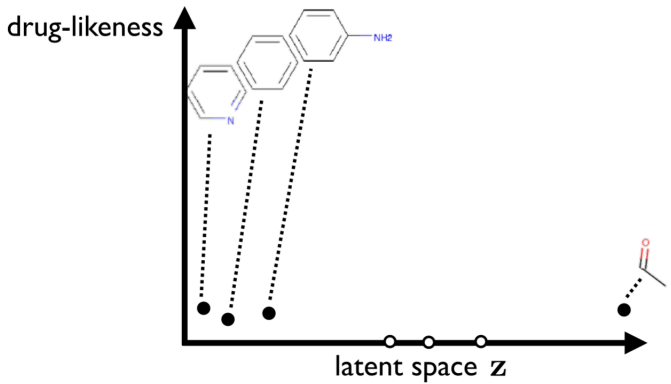
Searching for new drugs



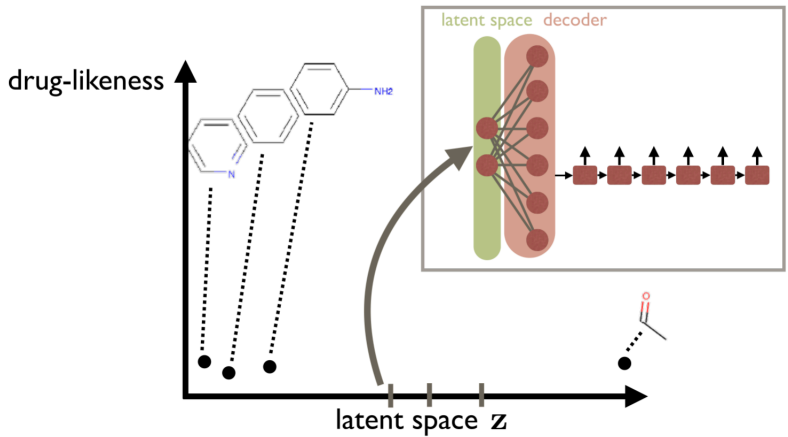
Searching for new drugs



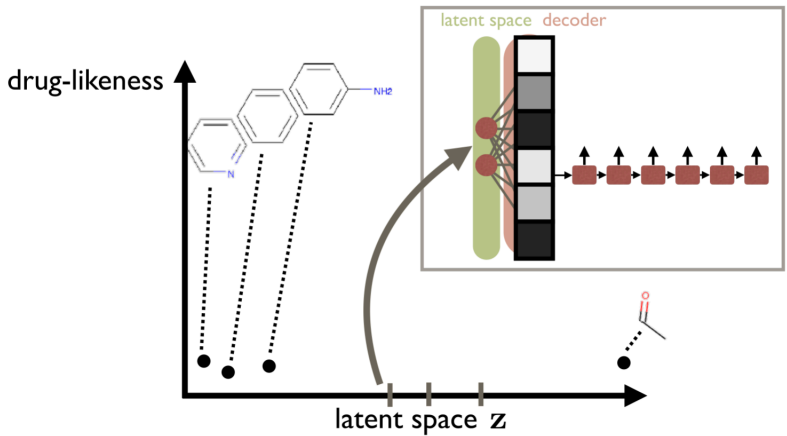
Searching in latent space



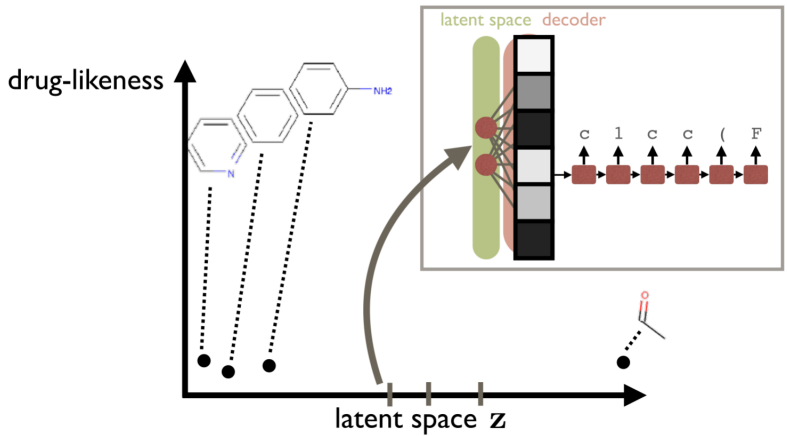
Searching in latent space



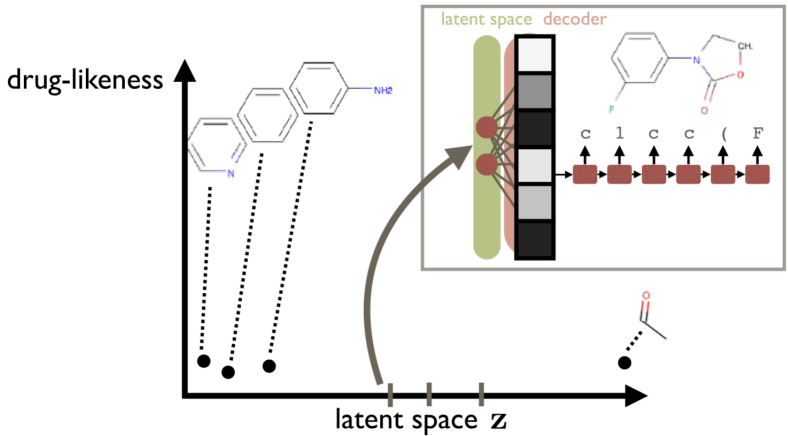
Searching in latent space



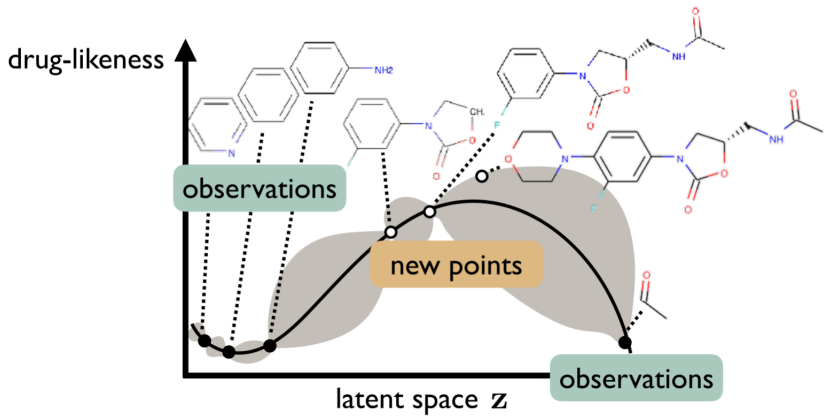
Searching in latent space



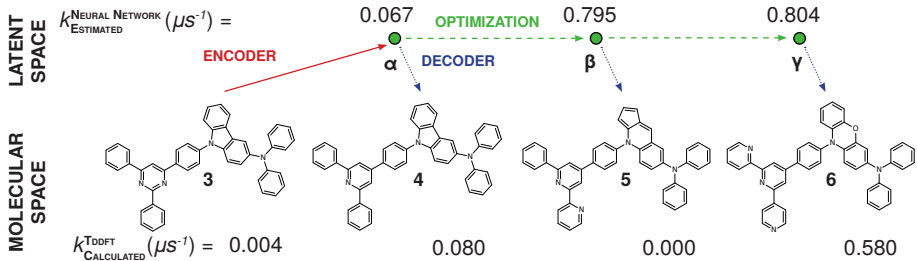
Searching in latent space



Searching in latent space



Example: OLED



- Optimized fluorescence decay rate, estimated from computations on 150k molecules
- Challenge: many “decoded” SMILES are not valid…

Gómez-Bombarelli et al, 2018

Application:

Semi-supervised learning

Learning with labels

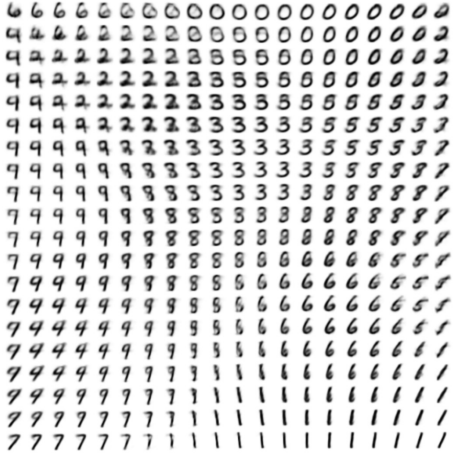
So far, we have only considered unlabelled data $\mathbf{x}$.
Question: What should we do with labeled pairs $(\mathbf{x}_i, y_i)$?

Learning with labels

Let's take MNIST as a running example.

- Images of handwritten digits $\mathbf{x}_i$
- Digit labels $y_i \in 0, \ldots, 9$
- The 2d learned latent space is shown again on the right.

Reminder: this was trained on an unsupervised model, that only had access to images $\mathbf{x}_i$!



Defining a joint generative model

We're going to define a generative model over the joint distribution $\mathbf{x}, y$, with a latent variable $\mathbf{z}$.

- **Question**: How should we factorize $p_\theta(\mathbf{x}, y, \mathbf{z})$?

Defining a joint generative model

We're going to define a generative model over the joint distribution $\mathbf{x}, y$, with a latent variable $\mathbf{z}$.

- **Question**: How should we factorize $p_\theta(\mathbf{x}, y, \mathbf{z})$?
- How do we actually draw images? Generally, it's helpful to mirror the real-world causal direction as much as possible.

Defining a joint generative model

We're going to define a generative model over the joint distribution $\mathbf{x}, y$, with a latent variable $\mathbf{z}$.

- **Question**: How should we factorize $p_\theta(\mathbf{x}, y, \mathbf{z})$?
- How do we actually draw images? Generally, it's helpful to mirror the real-world causal direction as much as possible.

A reasonable choice:

$$p_\theta(\mathbf{x}, y, \mathbf{z}) = p_\theta(\mathbf{x}|y, \mathbf{z})p(\mathbf{z})p(y)$$

"First, independently pick a digit y, and a latent style vector $\mathbf{z}$. Then, go draw the corresponding character $\mathbf{x}$."

Generative model and inference networks

We'll mostly follow the same structure used before, but now with an extra random variable y.

Generative model:

$$
\begin{aligned}
p(\mathbf{z}) &= \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I}) \\
p(y) &= \text{Discrete}(y|\boldsymbol{\pi}) \\
p_\theta(\mathbf{x}|y, \mathbf{z}) &= f(\mathbf{x}; y, \mathbf{z}, \theta)
\end{aligned}
$$

where $\boldsymbol{\pi}$ is a prior probability vector, and f is an appropriate probability density (e.g. Gaussian) with parameters given by a nonlinear transformation of $\mathbf{z}, y$.

Generative model and inference networks

For the inference network, we also need to choose a factorization.

Approximate posterior:

$$q(y|\mathbf{x}) = \text{Discrete}(y|\pi_\phi(\mathbf{x}))$$
$$q(\mathbf{z}|\mathbf{x}, y) = \mathcal{N}(\mathbf{z}|\mu_\phi(\mathbf{x}, y), \text{diag}(\sigma_\phi(\mathbf{x}, y)^2))$$

where π_ϕ, μ_ϕ, and σ_ϕ are all deep networks.

Questions: Why infer y first? Why does $\mathbf{z}$ depend on both $\mathbf{x}$ and y?

Kingma et al. (2014)

Semi-supervised setting

In the **semi-supervised setting**, we assume that the labels y are known for some instances, but not for others.

This effectively partitions the dataset into

- N^s image, label pairs $(\mathbf{x}_i, y_i)$
- N^u unlabelled images $\mathbf{x}_j$.

We'll treat these two sets of data separately, looking at their per-datapoint contributions to an ELBO.

Kingma et al. (2014)

Supervised ELBO

Let's start with the **supervised case**, focusing only on instances $\mathbf{x}$ where y is known, with

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

This ELBO is essentially identical to the one we derived before, aside from the fact that our observed "data" now also includes y.

$$\mathcal{L}(\mathbf{x}, y; \phi, \theta) \leq \log p_\theta(\mathbf{x}, y)$$

Kingma et al. (2014)

Unsupervised ELBO

In the **unsupervised case**, we do not know y, and need to

$$\mathcal{U}(\mathbf{x}; \phi, \theta) = \mathbb{E}_{q_\phi(y|\mathbf{x})q_\phi(\mathbf{z}|\mathbf{x},y)}[\log p_\theta(\mathbf{x}, y, \mathbf{z}) - \log q_\phi(y|\mathbf{x}) - \log q_\phi(\mathbf{z}|\mathbf{x}, y)].$$

This ELBO is **also essentially identical** to the one we derived before, except now our latent space also includes y.

$$\mathcal{U}(\mathbf{x}; \phi, \theta) \leq \log p_\theta(\mathbf{x})$$

Kingma et al. (2014)

Semi-supervised objective function

Putting these together, we can define an overall objective function

$$\mathcal{J} = \sum_{i=1}^{N^s} \mathcal{L}(\mathbf{x}_i, y_i; \phi, \theta) + \sum_{j=1}^{N^u} \mathcal{U}(\mathbf{x}_j; \phi, \theta).$$

There's just one thing odd about this objective: the supervised terms do not include the quantity we would traditionally call a classifier,

$$q_\phi(y|\mathbf{x}) = \text{Discrete}(y|\pi_\phi(\mathbf{x})),$$

which means the classifier network $\pi_\phi(\mathbf{x})$ is *only* estimated using the unsupervised data $\mathbf{x}_i$.

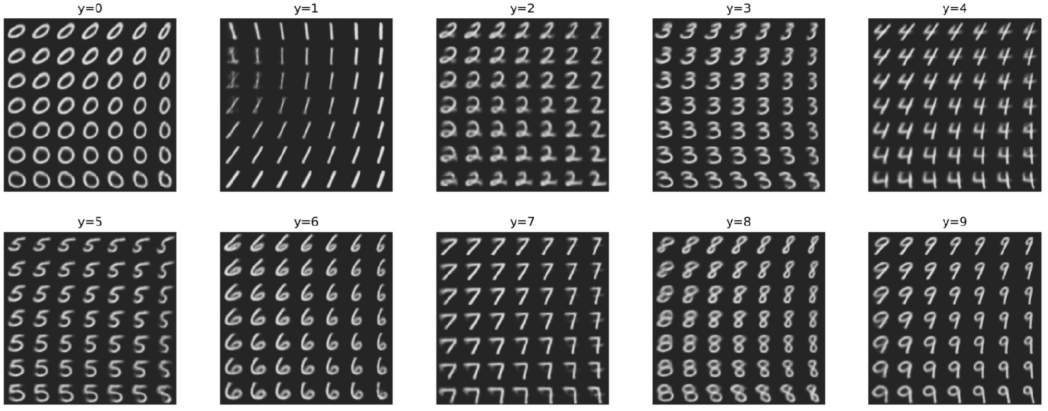
This counterintuitive behaviour can be avoided by adding in an additional term

$$\tilde{\mathcal{J}} = \mathcal{J} + \alpha \sum_{i=1}^{N^s} \log q_\phi(y_i|\mathbf{x}_i).$$

Check-in

- **Question**: Compared to the previous VAE, what do we think the latent space $\mathbf{z}$ will look like?
- **Question**: What is the difference between training this model when fully supervised, and learning ten independent VAEs (one per class)?

Class-conditional simulation

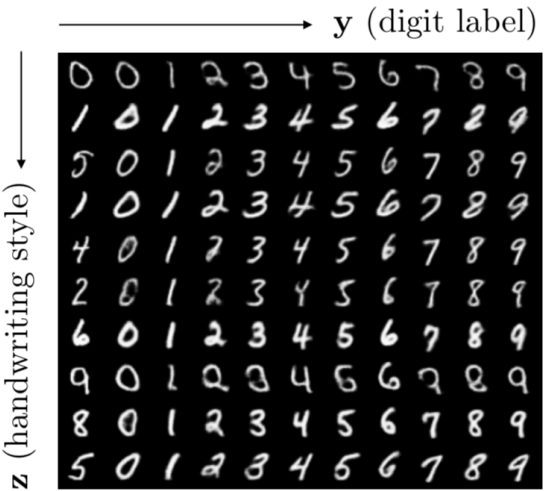


- First: the generated images do represent the target class.
- Second: the latent factor $\mathbf{z}$ seems to capture other independent variation in a consistent manner.

Kingma et al. (2014)

Style transfer, or visual analogies

- Holding the random variable $\mathbf{z}$ constant while modifying y transfers *style* across classes
- Each row corresponds to a fixed value of $\mathbf{z}_i$
- Each column past the first shows the mean $p_\theta(\mathbf{x}|y_k, \mathbf{z}_i)$, for $k = 0, \ldots, 9$



Kingma et al. (2014)

Street view house numbers data

- Same thing, on a potentially more impressive dataset
- First column shows real data, the rest show label-conditional reconstructions

Kingma et al. (2014)

How good is the classifier?

- Kingma et al. (2014) report 3.33% test error on MNIST when using only 100 labels, i.e. 10 labels per class...
- ...and down to 2.18% error for 3000 labels.
- Later work, Siddharth et al. (2017), uses different network architectures and reaches 1.57% test error on 3000 total labels, which is fairly comparable to looking at the full dataset.

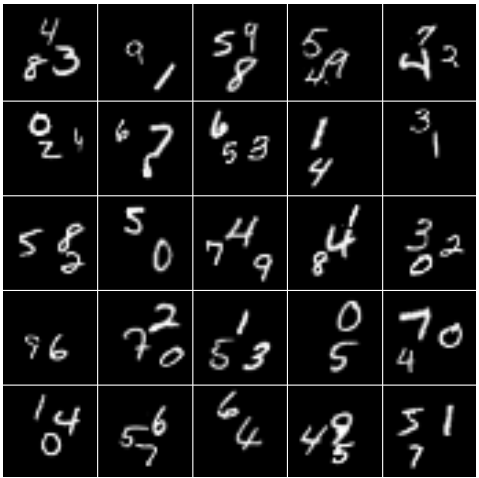
The full MNIST dataset has 60k examples.

SVHN classification performance tells a similar story.

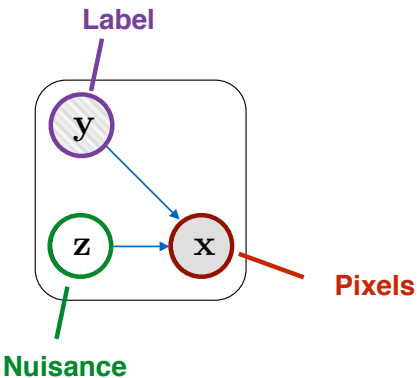
Extension:

Compositional models

From one digit to many digits

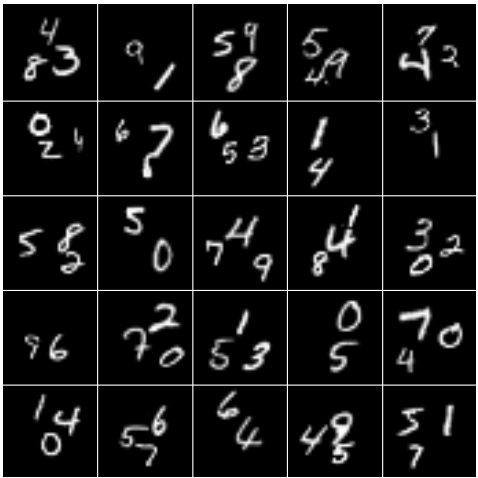


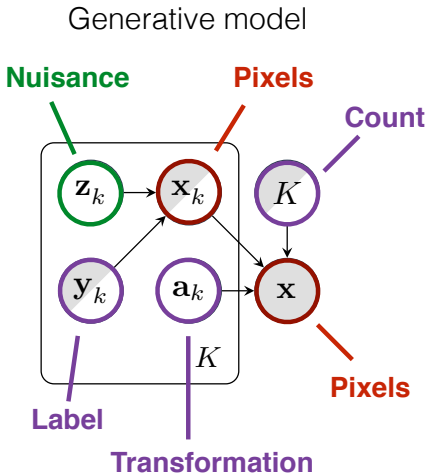
Generative model



Siddharth et al., 2017

From one digit to many digits

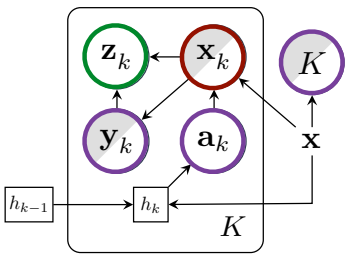




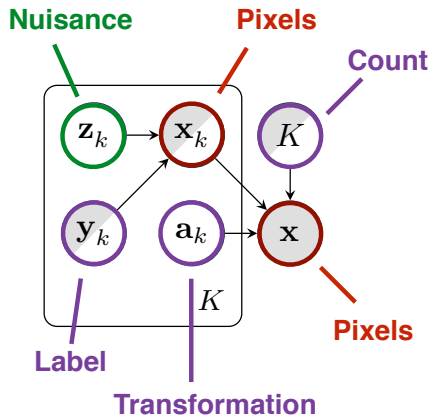
Siddharth et al., 2017

How do we build these models?

Inference model
(recurrent neural network)



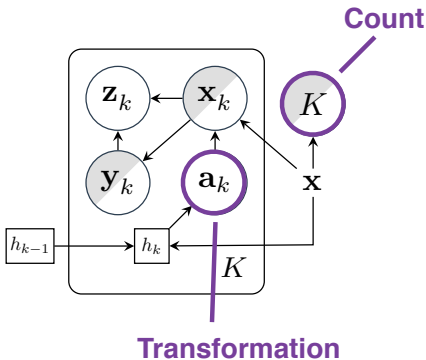
Generative model

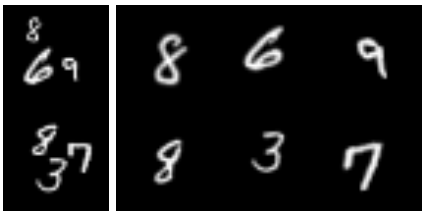


Siddharth et al., 2017

Inference: counting and locating

Inference model
(recurrent neural network)



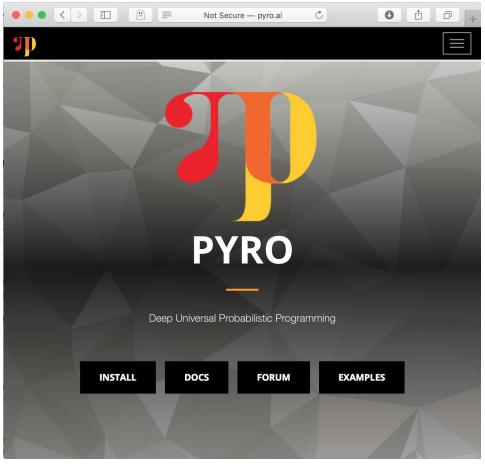


Siddharth et al., 2017

How to do all this (easily)

Pyro: deep "probabilistic programming"

- Adds probabilistic modeling on top of PyTorch
- Write two programs: a `model` and a `guide`, defined over the same random variables
- Implements automatic inference by computing and optimizing the ELBO



Other VAE extensions

How can you improve a VAE?

Improve the inference over $\mathbf{z}$, given $\mathbf{x}$?

- Use $q(\mathbf{z}|\mathbf{x})$ as an importance sampling proposal (IWAE; Burda et al., 2015)
- Use $q(\mathbf{z}|\mathbf{x})$ as an initialization for MCMC (Hoffman, 2015)
- Use a much more powerful $q(\mathbf{z}|\mathbf{x})$, instead of a factorized Gaussian

Improve the prior over $\mathbf{z}$?

- Mixture models as priors
- Autoregressive models as priors

Automatically "disentangle" the dimensions of $\mathbf{z}$ to have axis-aligned features?

- Beta-VAE, Higgins et al.
- Total correlation VAE, Chen et al.
- ...

Thanks!