Variational Autoencoders

Tutorial

Harry Ross & Luke McEachern

03/10/2016

Generative modelling

- Assume that the original dataset is drawn from a distribution $P(X)$.
- Attempt to model $P(X)$ in order to synthesize novel data (e.g. images) that are likely in the original $P(X)$ (i.e. similar, to our original data, but not the same).

Note the difference between these types of models and the autoencoder that we trained as part of assignment 2.

Motivation behind VAEs

Classic approaches

- Require strong assumptions about the structure in the data.
- Make severe approximations, leading to suboptimal models.
- Are computationally expensive (think intractable integrals, and MCMC).

Latent variable models

Latent variables: unobserved.

In general latent variables represent the true nature of the data, such that the process in which the data was originally generated is fully quantified. Understanding the process for which data of interest can be produced in a non trivial way is vital to being able to produce novel data. With this objective in mind it is clear that having a good way to estimate the true latents is essential to data generation.

Consider the simple example of generation of handwritten numbers:

Latent variables models cont...

- Before generation, an LVM samples a latent variable $z \in \{0, \ldots, 9\}$ representing the decision of which number to generate. Here, each number can be broadly considered a class.
- During the generation process, the LVM 'checks' that the stokes made match those consistent with the chosen character
- Note that we never observe which latent variable has been sampled. The task of computer vision is essentially to determine which latent variables were used in the process of generating a certain image

Note that the above slide was a bit of a simplification: In practice, even when performing the simple task of drawing a number, we sample *many* latent variables ranging from the size and width of the number, to the thickness, roundness etc.

Latent variable models cont...

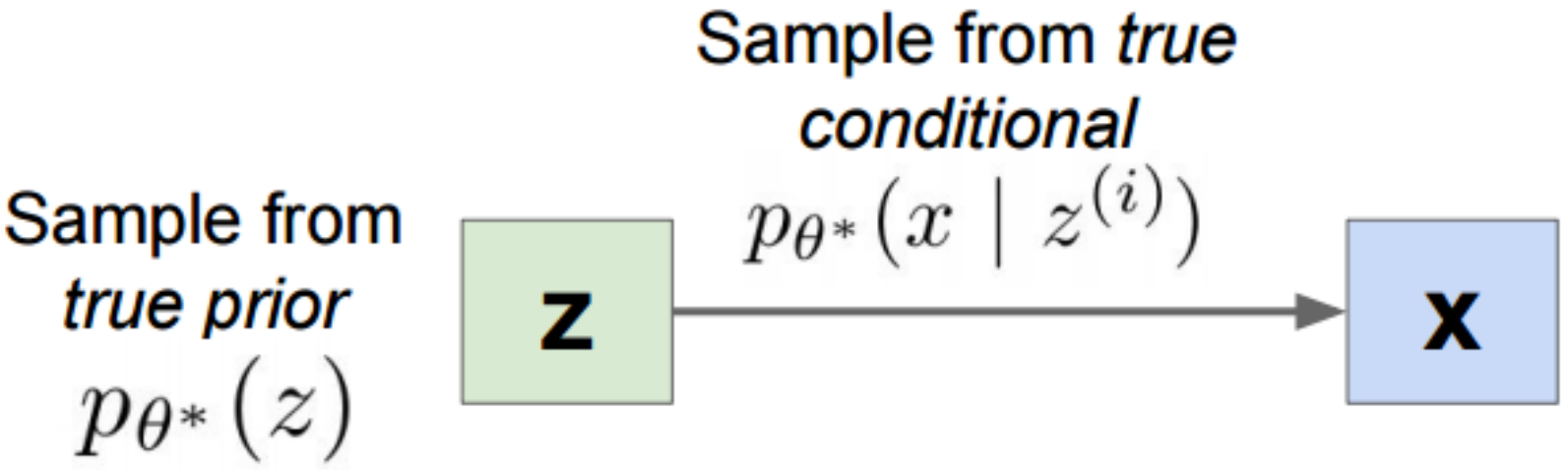
A more realistic model:

- We have a vector of latent variables $\vec{z}$, sampled from a high dimensional space $\mathcal{Z}$ according to a pdf $P(\vec{z})$ over $\mathcal{Z}$.

Since many different pdfs are possible, each with subtle differences, let's call an individual sample from the family of pdfs $f(\vec{z}; \theta)$, where θ is a parameter vector in space Θ that fully describes the pdf.

Finally, VAEs...

Assumption:



[1]

Goal:

- To model the distribution from which the input data, X, was generated, by learning its defining features
- To use this model to generate novel data that reflects $\vec{z}$, without direct replication of X

[1] CS231n, Stanford: 2016

Put formally, the objective of a VAE is to maximize the probability of our training data:

$$P(X) = \int P(X|\vec{z};\theta)P(\vec{z})d\vec{z} \tag{1}$$

Where, $P(X|z;\theta)$ is the conditional probability of X, given the latents and parameters, θ. $P(\vec{z})$ is the prior for $\vec{z}$ from which we sample from.

i.e. estimate θ, without access to $\vec{z}$.

VAEs - Assumptions

In the VAE framework, we assume:

- $P(\vec{z})$ is a unit Guassian (for mathematical convenience).
- $P(X|\vec{z})$ is a diagonal Gaussian. But why?

VAEs - Setting up the objective: computation of $P(X)$

As we can see, our task is to maximise equation (1). To do this, our VAE must define the latent variables and deal with the integral over $\vec{z}$.

But

- If $f(\vec{z}; \theta)$ is a multilayered neural network, we can just let that define the latent variable right?
- And we can just approximate the integral as follows, right?

$$P(X) \approx \frac{1}{n} \sum_i P(X|\vec{z}_i);$$

Issue: $\mathcal{Z}$ is a high-dimensional space and thus n has to be very large to get an accurate estimate of $P(X)$

VAEs - Setting up the objective: computation of $P(X)$

Solution: In latent space $\mathcal{Z}$, sampling and computing $P(X|\vec{z})$ will be ≈ 0 for most $\vec{z}$. A key feature of VAE is to sample only latents $\vec{z}$ that have high probabilities of having produced X. $Q(\vec{z}|X)$ defines a function that samples these high probability z values only to estimate $P(X)$. The space of z values under Q is (hopefully) *much* smaller than $\mathcal{Z}$. Means that $E_{\vec{z}\sim Q}P(X|\vec{z})$ is now relatively easy to calculate.

(for later) We assume $Q(z|X)$ takes the form:

$$\mathcal{N}(\vec{z}|\vec{\mu}(X;\theta),\Sigma(X;\theta))$$

VAEs - Setting up the objective: Using $Q(\vec{z})$

But hold on...

- $Q(\vec{z})$ could be any pdf
- How does it help us to optimize $P(X)$?

VAEs - setting up the objective: Using $Q(\vec{z})$

We use the Kullback-Leibler divergence to relate $Q(\vec{z})$ to $P(X)$:

$$
\begin{aligned}
&D_{KL}[Q(\vec{z})||P(\vec{z}|X)] \\
&\qquad = E_{\vec{z}\sim Q}[\log Q(\vec{z}) - \log P(\vec{z}|X)] \\
&\qquad = E_{\vec{z}\sim Q}[\log Q(\vec{z}) - \log P(X|\vec{z}) - \log P(\vec{z})] + \log P(X) \qquad (2)
\end{aligned}
$$

Where (2) follows from Bayes' theorem on $P(\vec{z}|X)$.

VAEs - Setting up the objective: Using $Q(\vec{z})$

And we want our $Q(\vec{z})$ function to be conditioned on the input X, so:

$$-D_{KL}[Q(\vec{z})||P(\vec{z}|X)] = E_{z\sim Q}[\log P(x|\vec{z})] - D_{KL}[Q(\vec{z})||P(\vec{z})] - \log P(X) \tag{3}$$

$$\log P(X) - D_{KL}[Q(\vec{z})||P(\vec{z}|X)] = E_{\vec{z}\sim Q}[\log P(X|\vec{z})] - D_{KL}[Q(\vec{z})||P(\vec{z})]$$

$$\log P(X) - D_{KL}[Q(\vec{z}|X)||P(\vec{z}|X)] = E_{\vec{z}\sim Q}[\log P(X|\vec{z})] - D_{KL}[Q(\vec{z}|X)||P(\vec{z})] \tag{4}$$

Where (3) follows from (2) through negation and the definition of D_{KL}.

VAEs - Setting up the objective

And so our objective function has reached its final form:

$$\begin{aligned} &\log P(X) - D_{KL}[Q(\vec{z}|X)||P(\vec{z}|X)] \\ &\qquad = E_{\vec{z}\sim Q}[\log P(X|\vec{z})] - D_{KL}[Q(\vec{z}|X)||P(\vec{z})] \end{aligned} \tag{4}$$

Note that by maximising (4), we maximise the $P(X)$, while simultaneously "pulling" $Q(\vec{z}|X)$ closer to the true distribution, $P(\vec{z}|X)$.

VAEs - Optimising the objective

But hang on...

- How can we back propagate through the stochastic process of sampling $\vec{z}$ from $Q(\vec{z}|X)$?

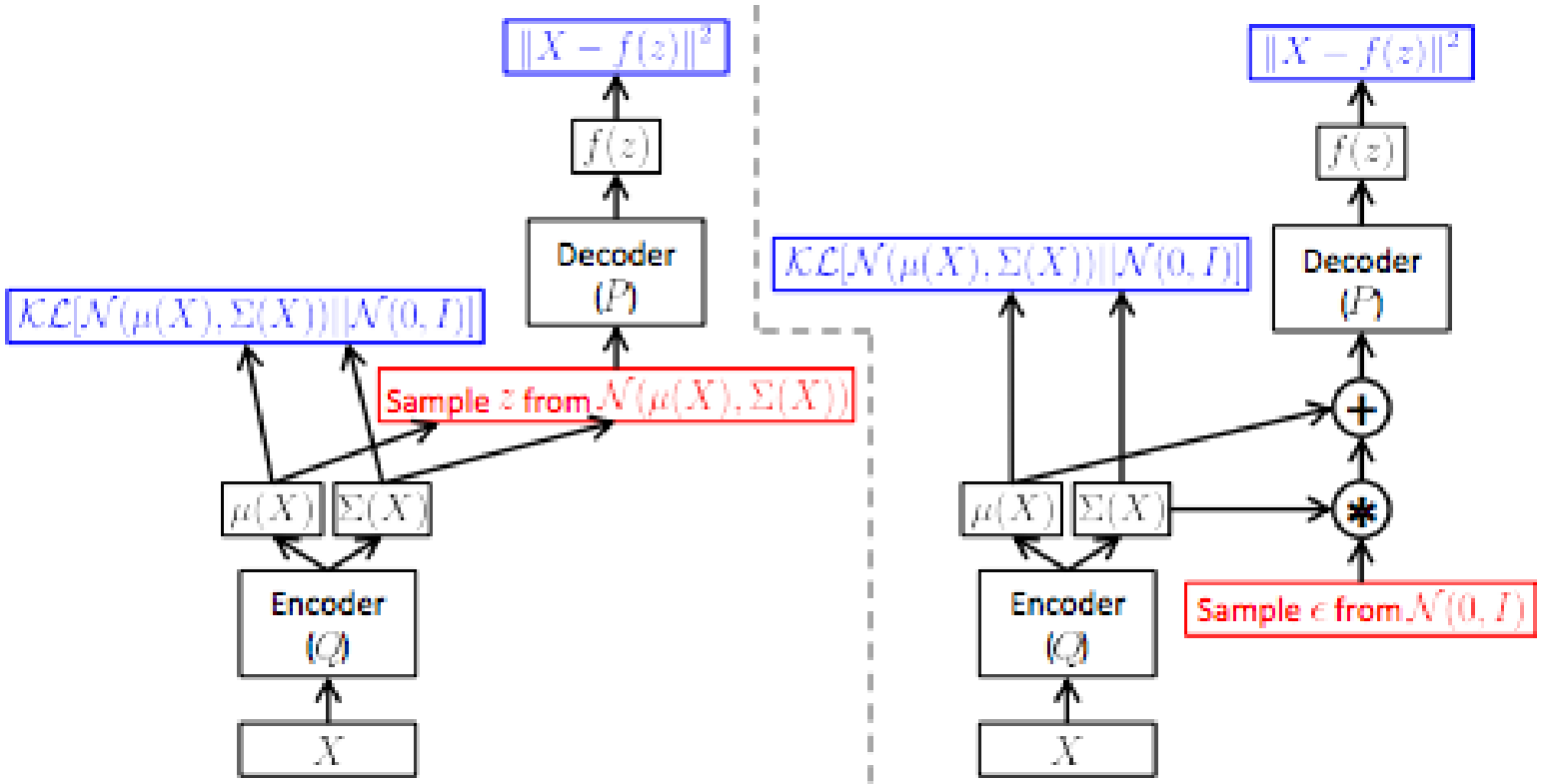
We can't. But we can use the (pretty cool) "reparameterization trick": $\vec{z} = \vec{\mu}(X) + \Sigma^{1/2}(X) * \epsilon$ where $\vec{\mu}(X)$ and $\Sigma(X)$ are the mean and covariance of $Q(X)$.

The intuition here is that we want to reparameterize $\vec{z}$ such that it's node can be considered deterministic.
Specifically, the equation that we take the gradient of (under this reparameterization) is:

$$E_{X\sim D}\left[E_{\epsilon\sim\mathcal{N}(0,1)}[logP(X|z^*)] - \mathcal{D}[Q(z|X)||P(z)]\right]$$

Where z^* is defined as the above reparameterization

VAEs - Training

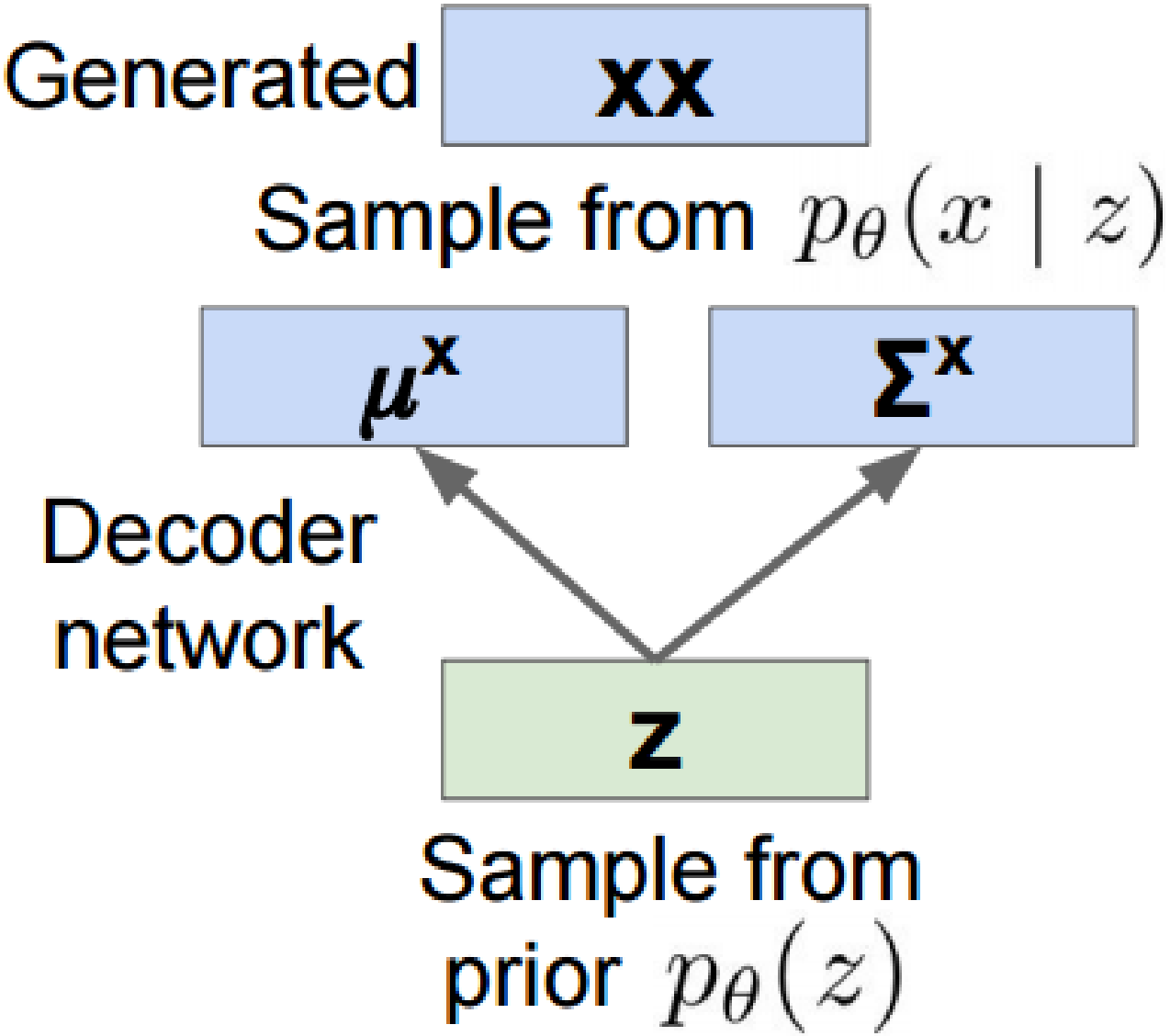


VAEs - Training

Process

- An image, X_i, is input
- The "encoder", generates a likely mean and covariances for each latent variable, given the input X_i (from $Q(\vec{z}|X_i)$)
- The mean and covariances are transformed into a sample $\vec{z}$ from the latent space $\mathcal{Z}$ using the reparameterization trick
- The "decoder" generates an image by maximising the likelihood $P(X|\vec{z})$
- The error term is calculated (see above), and back propagated through the network
- Repeat

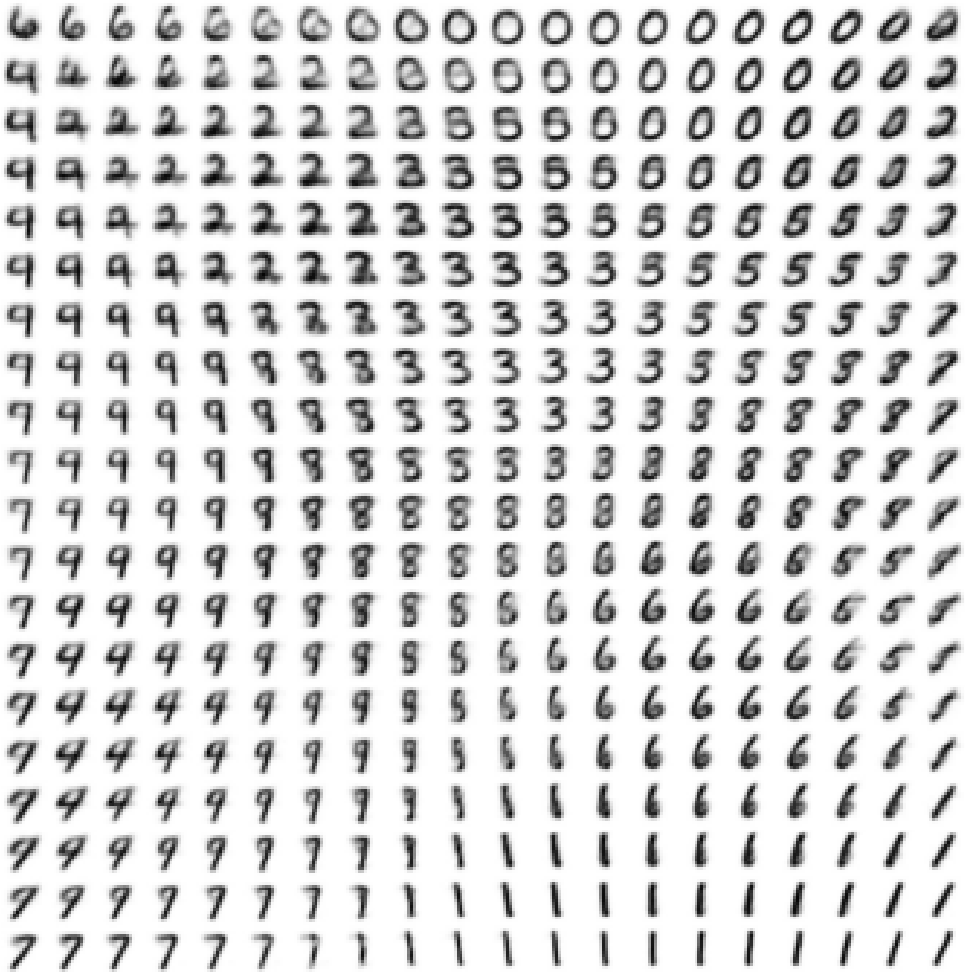
VAEs - Generation



[2]

[2] CS231n, Stanford: 2016

Example - Pretty pictures



[3]

[3] CS231n, Stanford: 2016

Example - Pretty pictures cont.

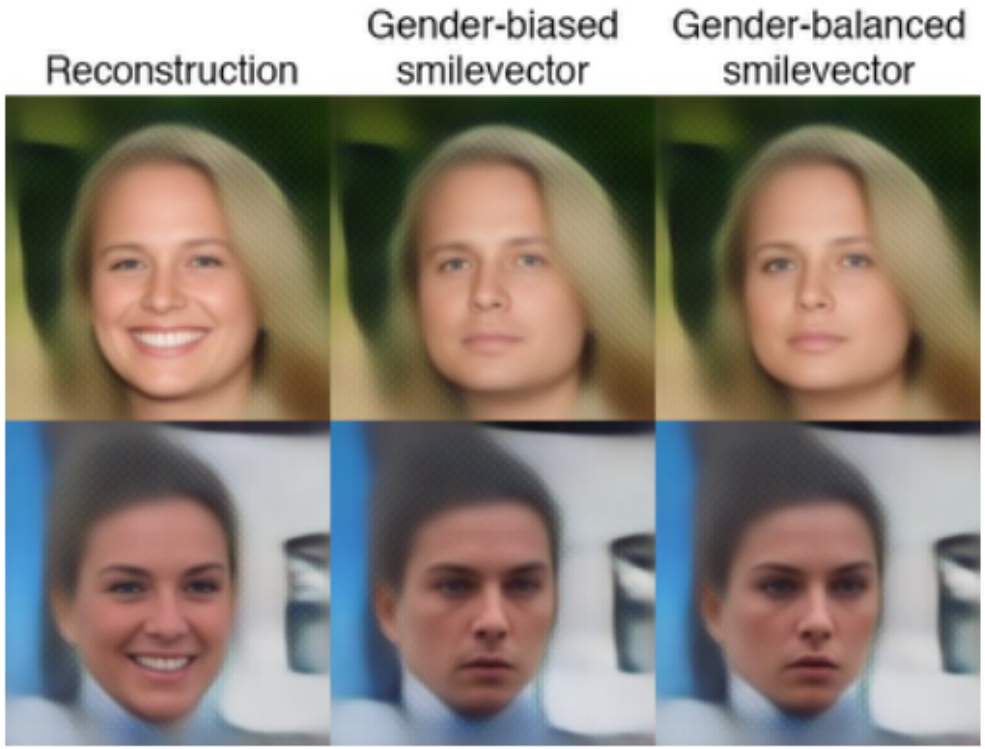


Figure 6: Initial attempts to build a smile vector suffered from sampling bias. The effect was that removing smiles from reconstructions (left) also added male attributes (center). By using replication to balance the data across both attributes before computing the attribute vectors, the gender bias was removed (right). (model: VAE from Lamb 16 on CelebA) 4

[4] Tom White, VUW: 2016