

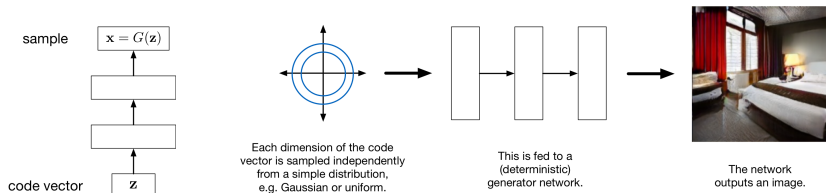
CS490/590 Lecture 17: Variational Autoencoders

Eren Gultepe
Department of Computer Science SIUE
Spring 2020

Adapted from Roger Grosse

Overview

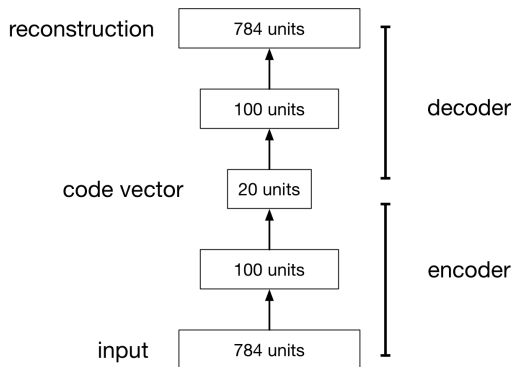
- Recall the generator network:



- One of the goals of unsupervised learning is to learn representations of images, sentences, etc.
- With reversible models, \mathbf{z} and \mathbf{x} must be the same size. Therefore, we can't reduce the dimensionality.
- Today, we'll cover the **variational autoencoder (VAE)**, a generative model that explicitly learns a low-dimensional representation.

Autoencoders

- An **autoencoder** is a feed-forward neural net whose job it is to take an input x and predict x .
- To make this non-trivial, we need to add a **bottleneck layer** whose dimension is much smaller than the input.



Autoencoders

Why autoencoders?

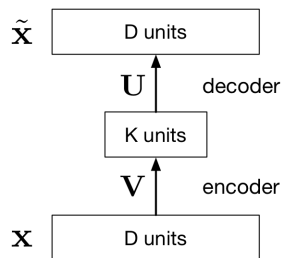
- Map high-dimensional data to two dimensions for visualization
- Compression (i.e. reducing the file size)
 - Note: this requires a VAE, not just an ordinary autoencoder.
- Learn abstract features in an unsupervised way so you can apply them to a supervised task
 - Unlabeled data can be much more plentiful than labeled data
- Learn a semantically meaningful representation where you can, e.g., interpolate between different images.

Principal Component Analysis

- The simplest kind of autoencoder has one hidden layer, linear activations, and squared error loss.

$$\mathcal{L}(\mathbf{x}, \tilde{\mathbf{x}}) = \|\mathbf{x} - \tilde{\mathbf{x}}\|^2$$

- This network computes $\tilde{\mathbf{x}} = \mathbf{U}\mathbf{V}\mathbf{x}$, which is a linear function.
- If $K \geq D$, we can choose \mathbf{U} and \mathbf{V} such that $\mathbf{U}\mathbf{V}$ is the identity. This isn't very interesting.
- But suppose $K < D$:
 - \mathbf{V} maps \mathbf{x} to a K -dimensional space, so it's doing dimensionality reduction.
 - The output must lie in a K -dimensional subspace, namely the column space of \mathbf{U} .



Principal Component Analysis

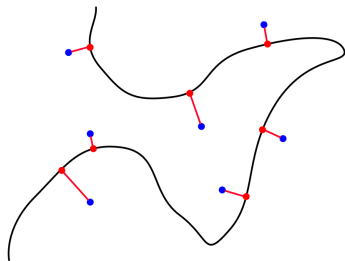
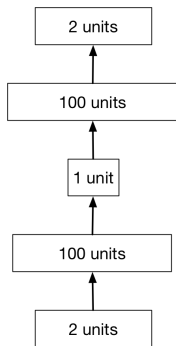
- Linear autoencoders with squared error loss are equivalent to Principal Component Analysis (PCA). Two equivalent formulations:
 - Find the subspace that minimizes the reconstruction error.
 - Find the subspace that maximizes the projected variance.
- The optimal subspace is spanned by the dominant eigenvectors of the empirical covariance matrix.



“Eigenfaces”

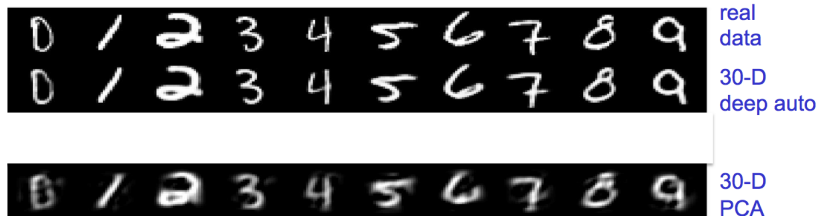
Deep Autoencoders

- Deep nonlinear autoencoders learn to project the data, not onto a subspace, but onto a nonlinear **manifold**
- This manifold is the image of the decoder.
- This is a kind of **nonlinear dimensionality reduction**.



Deep Autoencoders

- Nonlinear autoencoders can learn more powerful codes for a given dimensionality, compared with linear autoencoders (PCA)



Deep Autoencoders

- Some limitations of autoencoders
 - They're not generative models, so they don't define a distribution
 - How to choose the latent dimension?

Observation Model

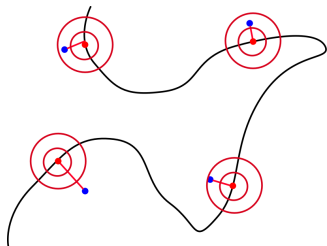
- Consider training a generator network with maximum likelihood.

$$p(\mathbf{x}) = \int p(\mathbf{z})p(\mathbf{x} | \mathbf{z}) d\mathbf{z}$$

- One problem: if \mathbf{z} is low-dimensional and the decoder is deterministic, then $p(\mathbf{x}) = 0$ almost everywhere!
 - The model only generates samples over a low-dimensional sub-manifold of \mathcal{X} .
- Solution: define a noisy observation model, e.g.

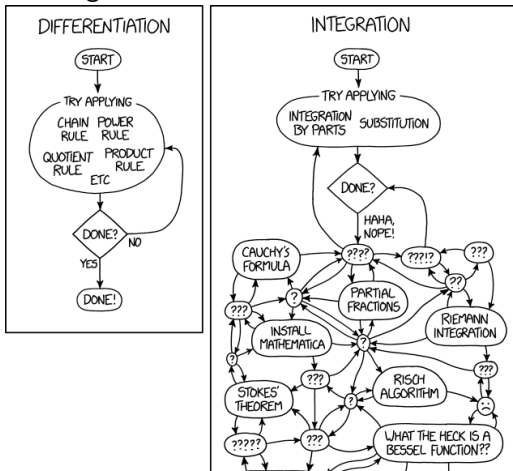
$$p(\mathbf{x} | \mathbf{z}) = \mathcal{N}(\mathbf{x}; G_{\theta}(\mathbf{z}), \eta \mathbf{I}),$$

where G_{θ} is the function computed by the decoder with parameters θ .



Observation Model

- At least $p(\mathbf{x}) = \int p(\mathbf{z})p(\mathbf{x} | \mathbf{z}) d\mathbf{z}$ is well-defined, but how can we compute it?
- Integration, according to XKCD:



Observation Model

- At least $p(\mathbf{x}) = \int p(\mathbf{z})p(\mathbf{x} | \mathbf{z}) d\mathbf{z}$ is well-defined, but how can we compute it?
 - The decoder function $G_\theta(\mathbf{z})$ is very complicated, so there's no hope of finding a closed form.
- Instead, we will try to maximize a lower bound on $\log p(\mathbf{x})$.
 - The math is essentially the same as in the EM algorithm from CSC411.

Variational Inference

- We obtain the lower bound using **Jensen's Inequality**: for a convex function h of a random variable X ,

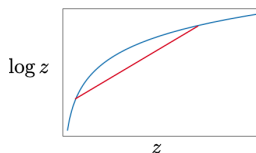
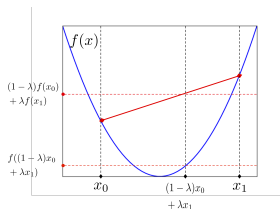
$$\mathbb{E}[h(X)] \geq h(\mathbb{E}[X])$$

Therefore, if h is **concave** (i.e. $-h$ is convex),

$$\mathbb{E}[h(X)] \leq h(\mathbb{E}[X])$$

- The function $\log z$ is concave. Therefore,

$$\mathbb{E}[\log X] \leq \log \mathbb{E}[X]$$



Variational Inference

- Suppose we have some distribution $q(\mathbf{z})$. (We'll see later where this comes from.)
- We use Jensen's Inequality to obtain the lower bound.

$$\begin{aligned}\log p(\mathbf{x}) &= \log \int p(\mathbf{z}) p(\mathbf{x}|\mathbf{z}) d\mathbf{z} \\ &= \log \int q(\mathbf{z}) \frac{p(\mathbf{z})}{q(\mathbf{z})} p(\mathbf{x}|\mathbf{z}) d\mathbf{z} \\ &\geq \int q(\mathbf{z}) \log \left[\frac{p(\mathbf{z})}{q(\mathbf{z})} p(\mathbf{x}|\mathbf{z}) \right] d\mathbf{z} && \text{(Jensen's Inequality)} \\ &= \mathbb{E}_q \left[\log \frac{p(\mathbf{z})}{q(\mathbf{z})} \right] + \mathbb{E}_q [\log p(\mathbf{x}|\mathbf{z})]\end{aligned}$$

- We'll look at these two terms in turn.

Variational Inference

- The first term we'll look at is $\mathbb{E}_q [\log p(\mathbf{x}|\mathbf{z})]$
- Since we assumed a Gaussian observation model,

$$\begin{aligned}\log p(\mathbf{x}|\mathbf{z}) &= \log \mathcal{N}(\mathbf{x}; G_{\theta}(\mathbf{z}), \eta \mathbf{I}) \\ &= \log \left[\frac{1}{(2\pi\eta)^{D/2}} \exp \left(-\frac{1}{2\eta} \|\mathbf{x} - G_{\theta}(\mathbf{z})\|^2 \right) \right] \\ &= -\frac{1}{2\eta} \|\mathbf{x} - G_{\theta}(\mathbf{z})\|^2 + \text{const}\end{aligned}$$

- So this term is the expected squared error in reconstructing \mathbf{x} from \mathbf{z} . We call it the **reconstruction term**.

Variational Inference

- The second term is $\mathbb{E}_q \left[\log \frac{p(\mathbf{z})}{q(\mathbf{z})} \right]$.
- This is just $-\text{D}_{\text{KL}}(q(\mathbf{z})\|p(\mathbf{z}))$, where D_{KL} is the **Kullback-Leibler (KL) divergence**

$$\text{D}_{\text{KL}}(q(\mathbf{z})\|p(\mathbf{z})) \triangleq \mathbb{E}_q \left[\log \frac{q(\mathbf{z})}{p(\mathbf{z})} \right]$$

- KL divergence is a widely used measure of distance between probability distributions, though it doesn't satisfy the axioms to be a distance metric.
- More details in tutorial.
- Typically, $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. Hence, the KL term encourages q to be close to $\mathcal{N}(\mathbf{0}, \mathbf{I})$.
- We'll give the KL term a much more interesting interpretation when we discuss Bayesian neural nets.

Variational Inference

- Hence, we're trying to maximize the **variational lower bound**, or **variational free energy**:

$$\log p(\mathbf{x}) \geq \mathcal{F}(\boldsymbol{\theta}, q) = \mathbb{E}_q [\log p(\mathbf{x}|\mathbf{z})] - D_{\text{KL}}(q\|p).$$

- The term “variational” is a historical accident: “variational inference” used to be done using variational calculus, but this isn't how we train VAEs.
- We'd like to choose q to make the bound as tight as possible.
- It's possible to show that the gap is given by:

$$\log p(\mathbf{x}) - \mathcal{F}(\boldsymbol{\theta}, q) = D_{\text{KL}}(q(\mathbf{z})\|p(\mathbf{z}|\mathbf{x})).$$

Therefore, we'd like q to be as close as possible to the posterior distribution $p(\mathbf{z}|\mathbf{x})$.

- Let's think about the role of each of the two terms.
- The reconstruction term

$$\mathbb{E}_q[\log p(\mathbf{x}|\mathbf{z})] = -\frac{1}{2\sigma^2}\mathbb{E}_q[\|\mathbf{x} - G_\theta(\mathbf{z})\|^2] + \text{const}$$

is minimized when q is a **point mass** on

$$\mathbf{z}_* = \arg \min_{\mathbf{z}} \|\mathbf{x} - G_\theta(\mathbf{z})\|^2.$$

- But a point mass would have infinite KL divergence. (Exercise: check this.) So the KL term forces q to be more spread out.

Reparameterization Trick

- To fit q , let's assign it a parametric form, in particular a Gaussian distribution: $q(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu} = (\mu_1, \dots, \mu_K)$ and $\boldsymbol{\Sigma} = \text{diag}(\sigma_1^2, \dots, \sigma_K^2)$.
- In general, it's hard to differentiate through an expectation. But for Gaussian q , we can apply the **reparameterization trick**:

$$z_i = \mu_i + \sigma_i \epsilon_i,$$

where $\epsilon_i \sim \mathcal{N}(0, 1)$.

- Hence,

$$\overline{\mu}_i = \overline{z}_i \quad \overline{\sigma}_i = \overline{z}_i \epsilon_i.$$

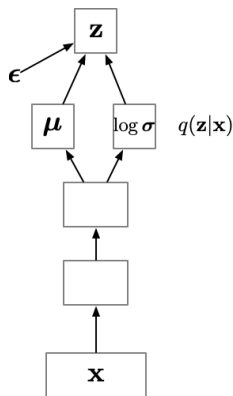
- This is exactly analogous to how we derived the backprop rules for dropout.

Amortization

- This suggests one strategy for learning the decoder. For each training example,
 - ① Fit q to approximate the posterior for the current \mathbf{x} by doing many steps of gradient ascent on \mathcal{F} .
 - ② Update the decoder parameters θ with gradient ascent on \mathcal{F} .
- **Problem:** this requires an expensive iterative procedure for every training example, so it will take a long time to process the whole training set.

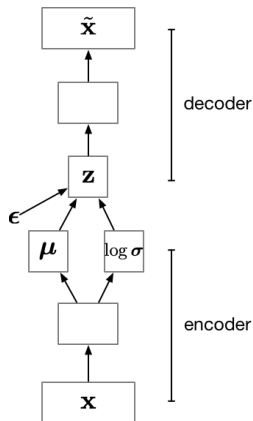
Amortization

- **Idea:** amortize the cost of inference by learning an **inference network** which predicts (μ, Σ) as a function of \mathbf{x} .
- The outputs of the inference net are μ and $\log \sigma$. (The log representation ensures $\sigma > 0$.)
- If $\sigma \approx \mathbf{0}$, then this network essentially computes \mathbf{z} deterministically, by way of μ .
 - But the KL term encourages $\sigma > 0$, so in general \mathbf{z} will be noisy.
- The notation $q(\mathbf{z}|\mathbf{x})$ emphasizes that q depends on \mathbf{x} , even though it's not actually a conditional distribution.



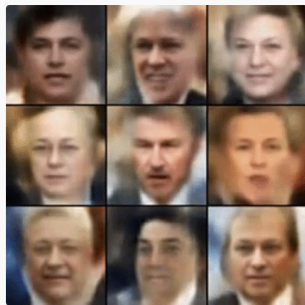
Amortization

- Combining this with the decoder network, we see the structure closely resembles an ordinary autoencoder. The inference net is like an encoder.
- Hence, this architecture is known as a **variational autoencoder (VAE)**.
- The parameters of both the encoder and decoder networks are updated using a single pass of ordinary backprop.
 - The reconstruction term corresponds to squared error $\|\mathbf{x} - \tilde{\mathbf{x}}\|^2$, like in an ordinary VAE.
 - The KL term regularizes the representation by encouraging \mathbf{z} to be more stochastic.



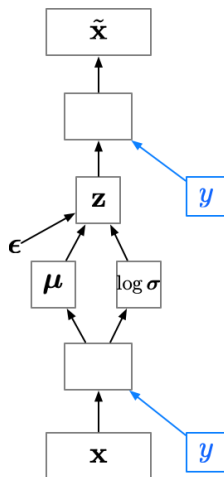
VAEs vs. Other Generative Models

- In short, a VAE is like an autoencoder, except that it's also a generative model (defines a distribution $p(\mathbf{x})$).
- Unlike autoregressive models, generation only requires one forward pass.
- Unlike reversible models, we can fit a low-dimensional latent representation. We'll see we can do interesting things with this...



Class-Conditional VAE

- So far, we haven't used the labels y . A **class-conditional VAE** provides the labels to both the encoder and the decoder.
- Since the latent code z no longer has to model the image category, it can focus on modeling the stylistic features.
- If we're lucky, this lets us **disentangle** style and content. (Note: disentanglement is still a dark art.)
- See Kingma et al., "Semi-supervised learning with deep generative models."



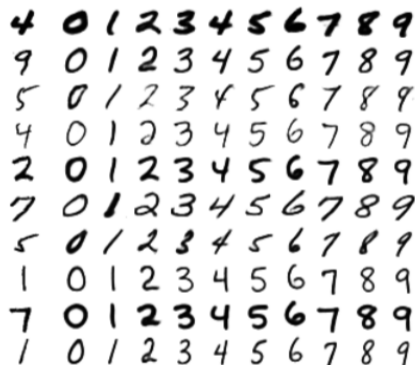
Class-Conditional VAE

- By varying two **latent dimensions** (i.e. dimensions of \mathbf{z}) while holding y fixed, we can visualize the **latent space**.



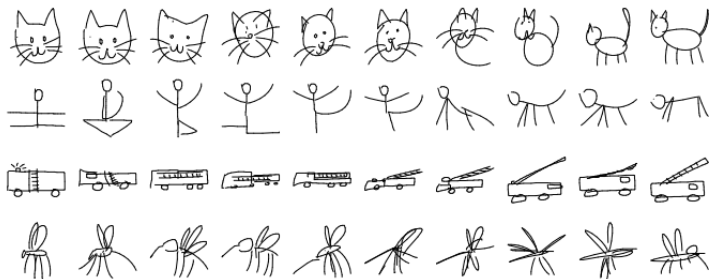
Class-Conditional VAE

- By varying the label y while holding z fixed, we can solve image analogies.



Latent Space Interpolations

- You can often get interesting results by interpolating between two vectors in the latent space:



Ha and Eck, "A neural representation of sketch drawings"

Latent Space Interpolations

- Latent space interpolation of music:
<https://magenta.tensorflow.org/music-vae>