COMS 4995 Lecture 13: Variational Autoencoders

Richard Zemel
Overview

Four modern approaches to generative modeling:

- Autoregressive models
- Generative adversarial networks
- Reversible architectures
- Variational autoencoders (this lecture)

All four approaches have different pros and cons.
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.
Why autoencoders?

- Map high-dimensional data to two dimensions for visualization
- Compression (i.e. reducing the file size)
- 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 (optional)

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

\[ \mathcal{L}(x, \tilde{x}) = \|x - \tilde{x}\|^2 \]

• This network computes \( \tilde{x} = UVx \), which is a linear function.

• If \( K \geq D \), we can choose \( U \) and \( V \) such that \( UV \) is the identity. This isn’t very interesting.

• But suppose \( K < D \):
  • \( V \) maps \( x \) 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 \( U \).
Principal Component Analysis (optional)

- 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.
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.

\[ p(x) = \int p(z)p(x \mid z) \, dz \]

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

- Solution: define a noisy observation model, e.g.

\[ p(x \mid z) = \mathcal{N}(x; G_\theta(z), \eta I), \]

where \( G_\theta \) is the function computed by the decoder with parameters \( \theta \).
Observation Model

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

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

$$\mathbb{E}[\log X] \leq \log \mathbb{E}[X]$$
Suppose we have some distribution $q(z)$. (We’ll see later where this comes from.)

We use Jensen’s Inequality to obtain the lower bound.

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

$$= \log \int q(z) \frac{p(z)}{q(z)} p(x|z) \, dz$$

$$\geq \int q(z) \log \left[ \frac{p(z)}{q(z)} p(x|z) \right] \, dz \quad \text{(Jensen’s Inequality)}$$

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

We’ll look at these two terms in turn.
The first term we’ll look at is \( \mathbb{E}_q [\log p(x|z)] \).

Since we assumed a Gaussian observation model,

\[
\log p(x|z) = \log \mathcal{N}(x; G_\theta(z), \eta I)
\]

\[
= \log \left[ \frac{1}{(2\pi\eta)^{D/2}} \exp \left( -\frac{1}{2\eta} \| x - G_\theta(z) \|^2 \right) \right]
\]

\[
= -\frac{1}{2\eta} \| x - G_\theta(z) \|^2 + \text{const}
\]

So this term is the expected squared error in reconstructing \( x \) from \( z \). We call it the reconstruction term.
Variational Inference

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

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

- 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, $p(z) = \mathcal{N}(0, I)$. Hence, the KL term encourages $q$ to be close to $\mathcal{N}(0, I)$.
- 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:

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

- 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:

\[
\log p(x) - \mathcal{F}(\theta, q) = D_{KL}(q(z)\|p(z|x)).
\]

Therefore, we’d like \( q \) to be as close as possible to the posterior distribution \( p(z|x) \).
Let’s think about the role of each of the two terms.

The reconstruction term

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

is minimized when $q$ is a point mass on

$$z_* = \arg\min_z \|x - G_\theta(z)\|^2.$$

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: $q(z) = \mathcal{N}(z; \mu, \Sigma)$, where $\mu = (\mu_1, \ldots, \mu_K)$ and $\Sigma = \text{diag}(\sigma^2_1, \ldots, \sigma^2_K)$.

- 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.
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 $\theta$ 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 $\left(\mu, \Sigma\right)$ as a function of $x$.
- The outputs of the inference net are $\mu$ and $\log \sigma$. (The log representation ensures $\sigma > 0$.)
- If $\sigma \approx 0$, then this network essentially computes $z$ deterministically, by way of $\mu$.
  - But the KL term encourages $\sigma > 0$, so in general $z$ will be noisy.
- The notation $q(z|x)$ emphasizes that $q$ depends on $x$, 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 \( \| x - \tilde{x} \|^2 \), like in an ordinary VAE.
  - The KL term regularizes the representation by encouraging \( z \) to be more stochastic.
In short, a VAE is like an autoencoder, except that it’s also a generative model (defines a distribution $p(x)$).

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...
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 $z$) while holding $y$ fixed, we can visualize the latent space.
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