Lecture 22: Variational Autoencoder
Reference: Kingma & Welling (2013)

Mark Hasegawa-Johnson

University of Illinois

ECE 417: Multimedia Signal Processing, Fall 2020

Outline

Autoencoder

An **autoencoder** is a neural net that learns a hidden code, $\vec{h}$, that is sufficient to reconstruct $\vec{x}$:

$$\vec{h} = f(\vec{x})$$
$$\hat{x} = g(\vec{h})$$

An autoencoder is usually trained to minimize mean-squared error, equivalent to maximizing the likelihood of a spherical Gaussian model:

$$\mathcal{L} = \sum_{i=1}^{n} \|\hat{x}_i - \vec{x}_i\|^2$$

Autoencoder

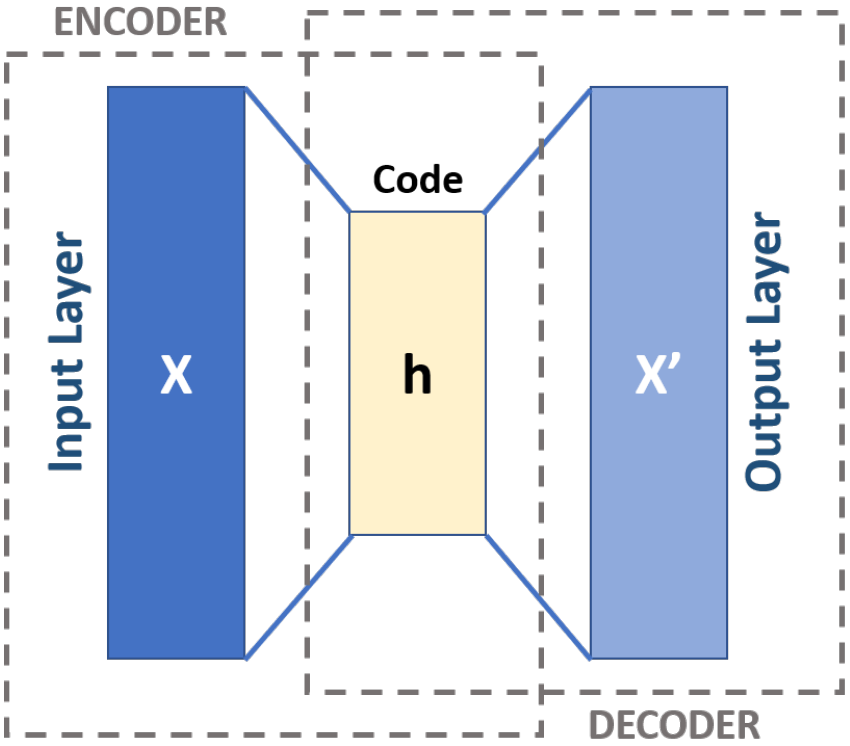

Image Michela Massi, 2019, CC-SA 4.0, https://commons.wikimedia.org/wiki/File:Autoencoder_schema.png

Linear Two-layer Autoencoder = PCA

Consider a two-layer linear autoencoder:

$$\vec{h} = W^T(\vec{x} - \vec{b})$$
$$\hat{x} = W\vec{h} + \vec{b}$$

The loss becomes

$$\mathcal{L} = \sum_{i=1}^{n} \|\vec{x}_i - WW^T(\vec{x} - \vec{b}) - \vec{b}\|^2$$

Linear Two-layer Autoencoder = PCA

$$\mathcal{L} = \sum_{i=1}^{n} \|\vec{x}_i - WW^T(\vec{x} - \vec{b}) - \vec{b}\|^2$$

- If $\text{len}(\vec{h}) \geq \text{len}(\vec{x})$, then the optimum solution is $W = I$, $\vec{b} = \vec{0}$, and $\hat{x} = \vec{x}$.
- If $\text{len}(\vec{h}) < \text{len}(\vec{x})$, then PCA minimizes $\mathcal{L}$.
  - $\vec{b}$ = the data mean,
  - W = the matrix of principal component directions.

Types of Autoencoders

If the hidden layer has too few constraints, we can get perfect reconstruction without learning anything useful. In order to learn useful hidden representations, a few common constraints are:

- Low-dimensional hidden layer. In this case, $\vec{h}$ is a nonlinear generalization of PCA, sometimes called a **bottleneck.**
- **Sparse autoencoder**: use a large hidden layer, but regularize the loss using a penalty that encourages $\vec{h}$ to be mostly zeros, e.g.,

$$\mathcal{L} = \sum_{i=1}^{n} \|\hat{x}_i - \vec{x}_i\|^2 + \lambda \sum_{i=1}^{n} \|\vec{h}_i\|_1$$

- **Variational autoencoder:** like a sparse autoencoder, but the penalty encourages $\vec{h}$ to match a predefined prior distribution, $p_\theta(\vec{h})$.

Outline

Bayesian Learning

The goal of Bayesian learning is to learn a complete model of the probability density function, $p(\vec{x})$. We usually assume that there is some latent variable $\vec{z}$, and a set of parameters θ, such that

- $\vec{z}$ is a latent variable generated randomly according to $p_\theta(\vec{z})$
- $\vec{x}$ is then generated randomly according to $p_\theta(\vec{x}|\vec{z})$

Bayesian Learning

A hidden Markov model is an example of Bayesian learning. We can generalize the three problems of an HMM, using the words of (Kingma and Welling, 2013):

1. **Recognition:** Efficient approximate marginal inference of the variable $\vec{x}$. Besides comparing different models, this can also allow us to generate synthetic data using $p_\theta(\vec{x})$.
2. **Segmentation:** Efficient approximate posterior inference of the latent variable $\vec{z}$ given an observed value $\vec{x}$ for a choice of parameters $\vec{\theta}$.
3. **Learning:** Efficient approximate ML or MAP estimation for the parameters $\vec{\theta}$.

Variational Bayes: Intractable Posteriors

- Variational Bayes is used in cases like the HMM, when
  - the **likelihood**, $p_\theta(\vec{x}|\vec{z})$, is easy to compute, but
  - the **posterior**, $p_\theta(\vec{z}|\vec{x})$, is intractable.
- If $p_\theta(\vec{z}|\vec{x})$ is intractable, then we can't exactly solve the segmentation or learning problems. Instead, we introduce a **variational approximation**, $q_\phi(\vec{z}|\vec{x}) \approx p_\theta(\vec{z}|\vec{x})$, and try to learn parameters ϕ and θ in order to match the data as well as possible.

The Evidence Lower Bound

Variational Bayes learns parameters θ and ϕ in order to maximize the **evidence** distribution:

$$\ln p_\theta(\vec{x})$$

- Averaging over the training data is understood.

The Evidence Lower Bound

Since the evidence doesn't depend on $\vec{z}$ at all, it doesn't hurt to compute its expected value w.r.t. $\vec{z}$:

$$\ln p_\theta(\vec{x}) = E_{q_\phi(\vec{z}|\vec{x})}\left[\ln p_\theta(\vec{x})\right]$$

The Evidence Lower Bound

We can introduce $\vec{z}$ inside the expectation by using the definition of conditional probability, $p(\vec{x}, \vec{z}) = p(\vec{x})p(\vec{z}|\vec{x})$, to get:

$$\ln p_\theta(\vec{x}) = E_{q_\phi(\vec{z}|\vec{x})}\left[\ln\left(\frac{p_\theta(\vec{x}, \vec{z})}{p_\theta(\vec{z}|\vec{x})}\right)\right]$$

The Evidence Lower Bound

Now, we can introduce q_ϕ inside the expectation as:

$$\ln p_\theta(\vec{x}) = E_{q_\phi(\vec{z}|\vec{x})}\left[\ln\left(\frac{p_\theta(\vec{x},\vec{z})}{p_\theta(\vec{z}|\vec{x})}\frac{q_\phi(\vec{z}|\vec{x})}{q_\phi(\vec{z}|\vec{x})}\right)\right]$$

Kullback-Leibler Divergence

Claude Shannon introduced a measure of the difference between two probability densities, called the Kullback-Leibler divergence:

$$D_{KL}\left(q_\phi(\vec{z}|\vec{x})\|p_\theta(\vec{z}\|\vec{x})\right) = E_{q(\vec{z}|\vec{x})}\left[\ln\left(\frac{q_\phi(\vec{z}|\vec{x})}{p_\theta(\vec{z}|\vec{x})}\right)\right]$$

A useful thing to know about KLD is that it's always non-negative: $D_{KL}(q\|p) \geq 0$, with equality if and only if $q = p$.

The Evidence Lower Bound

Re-arranging terms inside the expectation, we get

$$\ln p_\theta(\vec{x}) = D_{KL}\left(q_\phi(\vec{z}|\vec{x}) \| p_\theta(\vec{z}|\vec{x})\right) + \mathcal{L}\left(\theta, \phi; \vec{x}\right)$$

and therefore

$$\ln p_\theta(\vec{x}) \geq \mathcal{L}\left(\theta, \phi; \vec{x}\right)$$

with equality if and only if $q_\phi(\vec{z}|\vec{x}) = p_\theta(\vec{z}|\vec{x})$. The term $\mathcal{L}\left(\theta, \phi; \vec{x}\right)$ is therefore called the evidence lower bound or ELBO, and is given by

$$\mathcal{L}\left(\theta, \phi; \vec{x}\right) = E_{q(\vec{z}|\vec{x})}\left[\ln p_\theta(\vec{x}, \vec{z}) - \ln q_\phi(\vec{z}|\vec{x})\right]$$

Summary: Variational Bayes

- Variational Bayes is a method for learning a latent-variable model that can be used to generate synthetic data, encode the data, or recognize the data.
- It solves the intractability of $p_\theta(\vec{z}|\vec{x})$ by introducing a variational approximation, $q_\phi(\vec{z}|\vec{x}) \approx p_\theta(\vec{z}|\vec{x})$.
- The intractable term $p_\theta(\vec{z}|\vec{x})$ is then eliminated from the evidence by wrapping it up inside $D_{KL}\left(q_\phi(\vec{z}|\vec{x})||p_\theta(\vec{z}||\vec{x})\right)$. Since KLD is always non-negative, we can eliminate it from training criterion, leaving us with the evidence lower bound:

$$\mathcal{L}\left(\theta, \phi; \vec{x}\right) = E_{q(\vec{z}|\vec{x})}\left[\ln p_\theta(\vec{x}, \vec{z}) - \ln q_\phi(\vec{z}|\vec{x})\right]$$

Outline

The Problem with Variational Bayes

All previous VB algorithms struggled with the problem of efficiently computing expectations of the form $E_{q(\vec{z}|\vec{x})}[f(\vec{z})]$. Two tricks were commonly used, each with its own problems:

- **Factoring:** assume that $q(\vec{z}|\vec{x})$ has some simple form, e.g., assume that the dimensions of $\vec{z}$ are independent given $\vec{x}$. Problem: sometimes, no reasonable simplification of this type is possible.
- **Sampling (Monte Carlo methods):** draw L samples $\vec{z}^{(l)}$ from the distribution $q(\vec{z}|\vec{x})$, and then approximate

$$E_{q(\vec{z}|\vec{x})}[f(\vec{z})] \approx \frac{1}{L}\sum_{l=1}^{L} f(\vec{z}^{(l)})$$

Problem: if $q(\vec{z}|\vec{x})$ is a complicated distribution with many local maxima, then the approximation may be very bad, even for relatively large values of L.

Kingma & Welling: The Reparameterization Trick

Kingma & Welling (2013) proposed a reparameterization trick: assume

$$\vec{z}^{(l)} = g_\phi(\vec{\epsilon}^{(l)}, \vec{x}),$$

where g_ϕ is a flexible universal approximator (a neural net), and $\vec{\epsilon}$ is drawn from a predefined unimodal compact probability density function, e.g., a unit-normal Gaussian or uniform distribution. The sample average of a Gaussian or uniform distribution approaches its true average very quickly, e.g., even for $L \approx 5J$, where $J = \text{len}(\vec{z})$. Kingma & Welling recommend using a minibatch of about 100 training tokens, with $L = 1$ for each training token.

$$E_{q(\vec{z}|\vec{x})}[f(\vec{z})] \approx \frac{1}{L}\sum_{l=1}^{L} f(g_\phi(\vec{\epsilon}^{(l)}, \vec{x}))$$

Kingma & Welling: The Reparameterization Trick

The reparameterization trick is most useful if you assume that the latent variable has a parameter-independent prior, e.g., $p_\theta(\vec{z}) = p(\vec{z}) = \mathcal{N}(\vec{0}, I)$. Then you can re-write the ELBO as

$$\begin{aligned}\mathcal{L}(\theta, \phi; \vec{x}) &= -D_{KL}\left(q_\phi(\vec{z}|\vec{x}) \| p_\theta(\vec{z})\right) + E_{q(\vec{z}|\vec{x})}\left[\ln p_\theta(\vec{x}|\vec{z})\right] \\ &\approx -D_{KL}\left(q_\phi(\vec{z}|\vec{x}) \| p(\vec{z})\right) + \frac{1}{L}\sum_{l=1}^{L} \ln p_\theta(\vec{x}|\vec{z}^{(l)})\end{aligned}$$

- The second term, $\ln p_\theta(\vec{x}|\vec{z}^{(l)})$, is a neural network parameterized by θ.
- The first term, $-D_{KL}\left(q_\phi(\vec{z}|\vec{x}) \| p(\vec{z})\right)$, needs to be solved by pencil and paper.

Variational Autoencoder

The variational autoencoder has the following steps. For each training token,

1. Use a neural network, parameterized by ϕ, to compute the posterior mean and variance of the latent variable:

$$q_\phi(\vec{z}|\vec{x}) = \mathcal{N}\left(\vec{\mu}_{\vec{z}}(\vec{x};\phi), \Sigma_{\vec{z}}(\vec{x};\phi)\right)$$

   where $\mathcal{N}(\vec{\mu}, \Sigma)$ is the normal distribution, $\vec{\mu}_{\vec{z}}(\vec{x};\phi)$ and $\Sigma_{\vec{z}}(\vec{x};\phi)$ are the mean and variance of $\vec{z}$ conditioned on $\vec{x}$ and ϕ.
2. Draw L random $\vec{z}$ vectors from the distribution (e.g., $L = 1$).
3. Estimate $p_\theta(\vec{x}|\vec{z})$ using another neural network, parameterized by θ.
4. Update θ and ϕ using gradient ascent.

Benefits and Drawbacks of VAE

- A key benefit of the VAE is interpretability. You have two networks: one that generates $\vec{z}$ from $\vec{x}$, and one that generates $\vec{x}$ from $\vec{z}$. Therefore, you can map out the latent space, observing what types of data vectors $\vec{x}$ are generated from each point in the latent space.
- Another key benefit is generation. The latent variable is forced to have the distribution $\mathcal{N}(\vec{0}, I)$, so it's easy to generate synthetic data.
- A drawback is over-smoothing. VAE is trained to maximize the evidence of the training data, i.e., the likelihood marginalized over all latent variables. Maximum-likelihood models tend to generate synthetic data that is over-smoothed, more like an average of many images, instead of any particular image.

Outline

Summary

- Autoencoders try to find a latent vector that encodes as much information as possible about $\vec{x}$, subject to some constraints.
- Bayesian learning methods try to find a latent vector that encodes the data, subject to a known model of the prior and likelihood distributions.
- Variational Bayes avoids the intractability of the Bayesian posterior by using, instead, a variational approximation, $q_\phi(\vec{z}|\vec{x}) \approx p_\theta(\vec{z}|\vec{x})$. The difference between q and p is captured by their KLD, which is known to be non-negative, therefore VB can just maximize the ELBO.
- VAE models q_ϕ and p_θ using neural networks. The ELBO then has two terms: reconstruction error (parameterized by θ), and KLD between q_ϕ and a desired latent prior.