COMPSCI 689
Lecture 21: Variational Autoencoders

Benjamin M. Marlin

College of Information and Computer Sciences
University of Massachusetts Amherst

Slides by Benjamin M. Marlin (marlin@cs.umass.edu).

Autoencoders vs Factor Analysis

- Autoencoders and factor analysis can be applied to many of the same tasks including denoising and representation learning.
- Deep autoencoders can learn to represent data from complex manifolds using highly non-linear representations; however, they are not probabilistic models.
- Factor analysis is a probabilistic model, but the mean relationship between the latent variables and the visible variables is linear.
- What if we could merge the strengths of these two model classes and build deep, non-linear probabilistic models?

Non-Linear Factor Analysis

- This model generalizes the basic factor analysis model where the relationship between $\mathbf{z}$ and the mean of $\mathbf{X}$ is linear, to the case where the mean of $\mathbf{X}$ is generated from $\mathbf{z}$ using a deterministic non-linear transformation $f_{\mathbf{w}}(\mathbf{z})$ specified by a neural network.
- As with the basic factor analysis model $p(\mathbf{Z} = \mathbf{z}) = \mathcal{N}(\mathbf{z}; 0, I)$.
- The conditional probability of $\mathbf{X}$ is given by:

$$p(\mathbf{X} = \mathbf{x} | \mathbf{z}, \theta) = \mathcal{N}(\mathbf{x}; f_{\mathbf{w}}(\mathbf{z}), \Psi)$$

- $f_{\mathbf{w}}(\mathbf{z})$ represents a decoder/generator network. The model parameters are $\mathbf{w}$ and Ψ. Ψ is again a diagonal covariance matrix.

Non-Linear Factor Analysis: Joint Distribution

- The joint distribution of **X** and **Z** is thus given by:

$$p(\mathbf{X} = \mathbf{x}, \mathbf{Z} = \mathbf{z}|\theta) = \mathcal{N}(\mathbf{x}; f_{\mathbf{w}}(\mathbf{z}), \Psi)\mathcal{N}(\mathbf{z}; 0, I)$$

- Note that the joint distribution is computable and the computational complexity depends mostly on the complexity of the generator network $f_{\mathbf{w}}(\mathbf{z})$.
- Also note that given a trained model, we can efficiently generate new data from the model by sampling $\mathbf{z} \sim \mathcal{N}(0, I)$ and then $\mathbf{x} \sim \mathcal{N}(f_{\mathbf{w}}(\mathbf{z}), \Psi)$.

Non-Linear Factor Analysis: Marginal Distribution

- The marginal distribution of **X** is is given by:

$$\begin{aligned} p(\mathbf{X}=\mathbf{x}|\theta) &= \int p(\mathbf{X}=\mathbf{x}, \mathbf{Z}=\mathbf{z}|\theta) d\mathbf{z} \\ &= \int \mathcal{N}(\mathbf{x}; f_{\mathbf{w}}(\mathbf{z}), \Psi)\mathcal{N}(\mathbf{z}; 0, I) d\mathbf{z} \end{aligned}$$

Non-Linear Factor Analysis: Learning

- To learn the non-linear factor analysis model using maximum likelihood, we need to minimize the negative log marginal likelihood:

$$nlml(\mathcal{D}, \theta) = -\sum_{n=1}^{N} \log \left(\int p(\mathbf{X} = \mathbf{x}_n, \mathbf{Z} = \mathbf{z}|\theta) d\mathbf{z} \right)$$

Approximating the NLML

- We have seen one approach to dealing with intractable NLML functions that applies to real-valued latent variables.
- In this section, we'll look at a different approach that can be applied with any type of latent variables.
- This approach constructs an adjustable upper bound on the NLML function using a set of auxiliary distributions $q(\mathbf{Z} = \mathbf{z}|\phi_n) \approx p(\mathbf{Z} = \mathbf{z}|\mathbf{X} = \mathbf{x}, \theta)$, one per data case.
- We will follow the derivation for the case where **z** is a continuous random variable, but essentially the same results hold when it is discrete.

Jensen’s Inequality: Continuous Case

- Suppose $f()$ is a convex function of a scalar real argument.
- Let $g(x)$ be an arbitrary real-valued function of $x \in \mathcal{X}$ and $q(X = x)$ be a probability density function over the same space.
- The continuous form of Jensen’s Inequality states that the following upper bound holds:

$$f\left(\int q(X = x)g(x)dx\right) \leq \int q(X = x)f(g(x))dx = \mathbb{E}_{q(X)}[f(g(x))]$$

$$-\log\left(\int q(X = x)g(x)dx\right) \leq -\int q(X = x)\log(g(x))dx$$

Jensen's Inequality: Discrete Case

- Suppose $f()$ is a convex function of a scalar real argument.
- Let $g(k)$ be an arbitrary real-valued function of $k \in \{1, ..., C\}$ and let $Q(K = k)$ be a PMF such that $Q(K = k) \geq 0$ for all k and $\sum_{k=1}^{K} Q(K = k) = 1$.
- The discrete form of Jensen's Inequality states that for any valid choice of PMF Q, the following upper bound holds:

$$f\Big(\sum_{k=1}^{K} Q(K=k)g(k)\Big) \leq \sum_{k=1}^{K} Q(K=k)f(g(k)) = \mathbb{E}_{Q(K)}[f(g(k))]$$

An Upper Bound on the NLML

$$
\begin{aligned}
nlml(\mathcal{D}, \theta) &= -\sum_{n=1}^{N} \log\left(\int p(\mathbf{x}_n, \mathbf{z}|\theta) d\mathbf{z}\right) \\
&= -\sum_{n=1}^{N} \log\left(\int \frac{q(\mathbf{z}|\phi_n)}{q(\mathbf{z}|\phi_n)} p(\mathbf{x}_n, \mathbf{z}|\theta) d\mathbf{z}\right) \\
&\leq -\sum_{n=1}^{N} \int q(\mathbf{z}|\phi_n) \log\left(\frac{p(\mathbf{x}_n, \mathbf{z}|\theta)}{q(\mathbf{z}|\phi_n)}\right) d\mathbf{z} \\
&= -\sum_{n=1}^{N} \int q(\mathbf{z}|\phi_n) \Big(\log p(\mathbf{x}_n, \mathbf{z}|\theta) - \log q(\mathbf{z}|\phi_n) \Big) d\mathbf{z} \\
&= -\sum_{n=1}^{N} E_{q(\mathbf{Z}|\phi_n)}[\log p(\mathbf{x}_n, \mathbf{z}|\theta) - \log q(\mathbf{z}|\phi_n)] = Q(\mathcal{D}, \theta, \phi)
\end{aligned}
$$

Minimizing the Q Function

- Our task is now to minimize the upper bound $Q(\mathcal{D}, \theta, \phi)$. We can do this by minimizing $Q(\mathcal{D}, \theta, \phi)$ with respect to the θ parameters and the ϕ parameters.
- If the distributions are such that the expectation in $Q(\mathcal{D}, \theta, \phi)$ can be computed analytically, we obtain an algorithm referred to as the *variational expectation maximization* or variational EM algorithm:

$$\text{E-Step: } \phi_n^{t+1} \leftarrow \arg\min_{\phi_n} Q(\mathcal{D}, \theta^t, \phi)$$

$$\text{M-Step: } \theta^{t+1} \leftarrow \arg\min_{\theta} Q(\mathcal{D}, \theta, \phi^{t+1})$$

Tightness of the Bound

Minimizing $Q(\mathcal{D}, \theta, \phi)$ is only going to be meaningful if the upper bound is reasonably tight. So how tight is it?

$$Q(\mathcal{D}, \theta, \phi) = -\sum_{n=1}^{N} \left(\mathbb{E}_{q_n}[\log p(\mathbf{x}_n, \mathbf{z}|\theta) - \log q(\mathbf{z}|\phi_n)] \right)$$

$$= -\sum_{n=1}^{N} \left(\mathbb{E}_{q_n}[\log p(\mathbf{x}_n|\theta) + \log p(\mathbf{z}|\mathbf{x}_n, \theta) - \log q(\mathbf{z}|\phi_n)] \right)$$

$$= -\sum_{n=1}^{N} \left(\log p(\mathbf{x}_n|\theta) + \mathbb{E}_{q_n}\left[\log \left(\frac{p(\mathbf{z}|\mathbf{x}_n, \theta)}{q(\mathbf{z}|\phi_n)} \right) \right] \right)$$

Tightness of the Bound

We can recognize the first term as the NLML and the second term as $-KL(q(\mathbf{z}|\phi_n)\|p(\mathbf{z}|\mathbf{x}_n,\theta))$ giving the following result:

$$Q(\mathcal{D},\theta,\phi) = nlml(\mathcal{D},\theta) + \sum_{n=1}^{N} KL(q(\mathbf{z}|\phi_n)\|p(\mathbf{z}|\mathbf{x}_n,\theta))$$

So, if we can select $q(\mathbf{z}|\phi_n)$ so that it can represent $p(\mathbf{z}|\mathbf{x}_n,\theta)$ for all n and θ, then the bound is perfectly tight. Often we need to use a simpler approximating family, and this approach will incur bias.

The classical simplified approximating family is to construct $q(\mathbf{z}|\phi_n)$ as a product of marginals. This is referred to as the *variational mean field* approximation.

Amortized Variational Learning

- In the algorithm presented so far, we need to solve an optimization problem in the E-Step to determine the parameter values of the approximating distributions $q(\mathbf{z}|\phi_n)$ for each data case $\mathbf{x}_n$.
- This can be quite costly. We can instead learn an auxiliary neural network model (called an inference network or a recognition network) $g_v(\mathbf{x}_n)$ that predicts the optimal parameters ϕ_n for the model $q(\mathbf{z}|\phi_n)$ using $\mathbf{x}_n$ as input.
- The parameters of the auxiliary network $g_v(\mathbf{x}_n)$ are learned along with the primary model parameters.
- This is referred to as *amortized variational inference*.

Stochastic Variational EM

- Even when using a simplified approximating distribution $q_n(\mathbf{z}|\phi_n)$, its still possible for the term $\mathbb{E}_{q(\mathbf{z}|\phi_n)}[\log p(\mathbf{x}_n, \mathbf{z}, |\theta)]$ to contain a computationally intractable integral.
- In such cases, we can consider a stochastic approximation to the upper bound based on drawing samples $z_{sn} \sim q_n(\mathbf{z}|\phi_n)$.
- We obtain an approximate objective:

$$\frac{1}{S}\sum_{s=1}^{S}\sum_{n=1}^{N}(\log p(\mathbf{x}_n, \mathbf{z}_{sn}, |\theta) - \log q(\mathbf{z}_{sn}|\phi_n))$$

Reparameterization Trick

- Note that while this approximation is a standard Monte Carlo approximation to an integral, it can not be directly used for learning the variational parameters. Why?
- The distribution we sampled from depends on the variational parameters, and we can't take gradients through sampling.
- Instead, we can often draw samples from a parameter-free base distribution, and then transform them into samples from $q(\mathbf{z}|\phi_n)$ using a deterministic, differentiable function.

Example: Diagonal Gaussian Re-parameterization

- Suppose that $q(\mathbf{z}|\phi)$ is a diagonal Gaussian with mean m_k and standard deviation s_k on dimension k.
- All the dimensions of a diagonal Gaussian are independent, so we can sample $\mathbf{z}$ by sampling each of its dimensions z_k from the corresponding marginal $\mathcal{N}(m_k, s_k^2)$.
- To begin, sample $\eta_k \sim \mathcal{N}(0, 1)$ for each k.
- Now define $z_k = m_k + \eta_k s_k$ for each k.
- Then z_k is a sample from $\mathcal{N}(m_k, s_k^2)$ and $\mathbf{z}$ is a sample from $q(\mathbf{z}|\phi)$

Re-parameterization Trick

- This re-parameterization gives the final approximate objective:

$$\frac{1}{S}\sum_{s=1}^{S}\sum_{n=1}^{N}(\log P(\mathbf{x}_n, \mathbf{z}_{ns}, |\theta) - \log q(\mathbf{z}_{ns}|\phi_n))$$

$$\mathbf{z}_{kns} = m_{kn} + \eta_{kns}s_{kn}$$

$$\eta_{kns} \sim \mathcal{N}(0, 1)$$

Learning with Stochastic Approximation

- On each learning iteration, given a set of samples η_{kns}, we can compute the gradient of the approximate objective and do gradient-based learning.
- Note that since all of the integrals are gone, we can implement the computation for $p(\mathbf{x}_n, \mathbf{z}_{ns}, |\theta)$ and $q(\mathbf{z}_{ns}|\phi_n)$ and use automatic differentiation to obtain the required gradients.
- This allows us to more easily build complex distributions for both p and q. So long as we can sample from q using re-parameterization, and both p and q are differentiable, we can apply this framework.

Variational Autoencoder

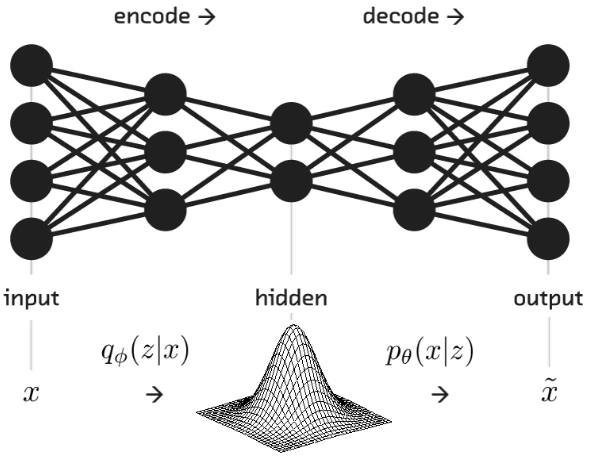
- A variational autoencoder combines all of the ideas introduced in this lecture.
- The model is non-linear factor analysis. The learning approach minimizes the stochastic variational upper bound on the NLML with amortized variational mean field inference.
- For this specific model, we assume a factorized auxiliary distribution $q(\mathbf{z}|\mathbf{x}_n,\phi) = \prod_{k=1}^{K} q(z_k|\mathbf{x}_n,\phi_k) = \mathcal{N}(\mathbf{z}; m_k(\mathbf{x}_n), s_k^2(\mathbf{x}_n))$.
- $m_k(\mathbf{x}_n)$ is a neural network that predicts the mean for latent dimension k, while $s_k(\mathbf{x}_n)$ is a neural network that predicts the standard deviation for latent dimension k, both taking $\mathbf{x}_n$ as input.

Variational Autoencoders

- This model family has the advantage that it is a full probabilistic model for $\mathbf{X}$.
- The neural network model used in the generator $\mathcal{N}(\mathbf{x}; f_{\mathbf{w}}(\mathbf{z}), \Psi)$ can model complex non-linear manifolds.
- The neural network models used in the recognition network enable optimization-free approximate inference during and after model learning.
- Like with factor analysis, the model can also be generalized to produce distributions over different types of variables by using the generator network to predict the parameters of appropriately chosen base distributions.

Stochastic Variational Autoencoder

trained using variational learning. Amortized inference using



Demo!