Variational Autoencoders

References

These notes are based on the following:

- Kingma, D. and Welling, M. (2014). Auto-Encoding Variational Bayes. In ICLR 2014. https://arxiv.org/pdf/1312.6114.pdf
- Carl Doersch's tutorial on Variational Auto-encoders. https://arxiv.org/pdf/1606.05908.pdf
- Stefano Ermon's CS 228 notes: https://ermongroup.github.io/cs228-notes/extras/vae/

Introduction

- Suppose you had a dataset with N instances $\boldsymbol{X} = \{\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^N\}$ where N is very big
- Each instance has d features i.e. $x^i = \{x_1^i, \ldots, x_d^i\}$
- E.g. each data instance $\boldsymbol{x}_i$ is a 20x28=560 dimenisional image of a face as shown below.

Source: Frey Face dataset.
https://cs.nyu.edu/~roweis/data/frey_rawface.mat

Latent Variable Models

- We would like to learn the joint distribution $P(\boldsymbol{X})$ that generates this dataset (i.e. learn a generative model)
- We also assume that there are K unobserved variables (called latent factors) $\boldsymbol{Z} = \{z_{1}\text{ˎ}, z_2, \ldots, z_K\}$
- The latent variables control variation in the features e.g. smiling/frowning, facing left/center/right, etc.

Latent Variable Models

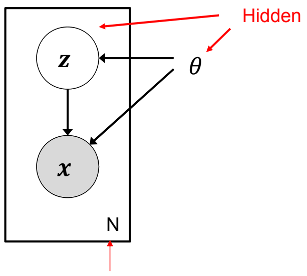
We assume the following generative process:

1. Sample $\boldsymbol{z}^i \sim P_{\boldsymbol{\theta}^*}(\boldsymbol{z})$ where $\boldsymbol{z}$ is an unobserved continuous random variable and $\boldsymbol{\theta}^*$ are the true parameters. Note: $\boldsymbol{z}^i$ is k dimensional i.e. $\boldsymbol{z}^i = \{z_1^i, \dots, z_K^i\}$
2. Sample $\boldsymbol{x}_i \sim P_{\boldsymbol{\theta}^*}(\boldsymbol{x}|\boldsymbol{z})$. Note: $\boldsymbol{x}^i$ is d dimensional i.e. $\boldsymbol{x}^i = \{x_1^i, \dots, x_d^i\}$.

Note: $\boldsymbol{\theta}^*$ and $\boldsymbol{z}_i$ are unknown but we observe $\boldsymbol{x}_i$

Latent Variable Models

This leads to the following probabilistic graphical model:

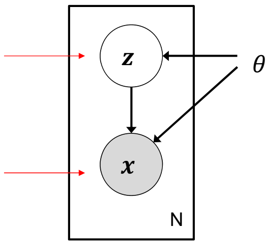


Hidden

This box is a plate, meaning you replicate the contents by the number in the lower right corner. Each replicate corresponds to a data instance $\boldsymbol{x}_i$ and its corresponding latent factor $\boldsymbol{z}_i$.

Latent Variable Models

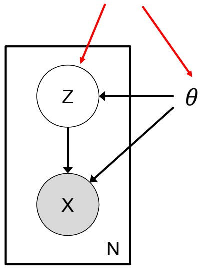
Note: This is multivariate data. The diagram is deliberately vague to abstract away the complex relationships between the dimensions of z and x



This is a k dimensional vector.

This is a d dimensional vector.

Latent Variable Models

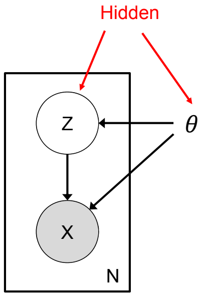


Hidden

There are 3 tasks of interest:

1. Learn the parameters $\boldsymbol{\theta}$ from data
2. Learn the latent factors $\boldsymbol{Z}$ given $\boldsymbol{X}$
3. Fill in missing values in a new data point $\boldsymbol{x}_{new}$ e.g. for image inpainting

Latent Variable Models



These tasks involve computation of the following:

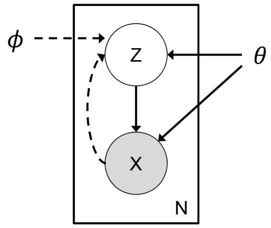
$$p_\theta(\mathbf{z}|\boldsymbol{x}) = \frac{p_\theta(\mathbf{z})p_\theta(\boldsymbol{x}|\mathbf{z})}{p_\theta(\boldsymbol{x})}$$

But this is intractable:

$$p_\theta(\boldsymbol{x}) = \int p_\theta(\mathbf{z})p_\theta(\boldsymbol{x}|\mathbf{z})d\mathbf{z}$$

We also assume there is so much data we fit it into memory and have to work with minibatches

The Variational Bound



Let $q_\phi(\mathbf{z}|\boldsymbol{x})$ be a variational approximation to the intractable posterior $p_\theta(\mathbf{z}|\boldsymbol{x})$

Note that $q_\phi(\mathbf{z}|\boldsymbol{x})$ has parameters ϕ that we need to optimize

The Variational Bound

Recall from the variational inference section that we can write:

$$\log p_\theta(\boldsymbol{x}) = KL(q_\phi(\mathbf{z}|\boldsymbol{x})||p_\theta(\mathbf{z}|\boldsymbol{x})) + ELBO(\phi,\theta,\boldsymbol{x})$$

where

$$ELBO(\phi,\theta,\boldsymbol{x}) = E_{q_\phi(\mathbf{z}|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x},\mathbf{z}) - \log q_\phi(\mathbf{z}|\boldsymbol{x})\right]$$

We can also rewrite:

$$ELBO(\phi,\theta,\boldsymbol{x}) = E_{q_\phi(\mathbf{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\mathbf{z})] - KL\big(q_\phi(\mathbf{z}|\boldsymbol{x})||p_\theta(\mathbf{z})\big)$$

The Variational Bound

$$ELBO(\phi,\theta,\boldsymbol{x}) = E_{q_\phi(\mathbf{z}|\boldsymbol{x})}\underbrace{[\log p_\theta(\boldsymbol{x}|\mathbf{z})]}_{} - \underbrace{KL\big(q_\phi(\mathbf{z}|\boldsymbol{x})||p_\theta(\mathbf{z})\big)}_{}$$

Reconstruction error: Needs to reconstruct $\boldsymbol{x}$ from $\boldsymbol{z}$ such that $\boldsymbol{x}$ has high probability under generative distribution P_θ

Regularization term: Makes z look like the prior (Gaussian) to prevent learning the identity function.

The Variational Bound

$$ELBO(\phi,\theta,\boldsymbol{x}) = \underbrace{E_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})]}_{} - \underbrace{KL\big(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z})\big)}_{}$$

Think of $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ as a probabilistic decoder that takes a code $\boldsymbol{z}$ and decodes it to an instance $\boldsymbol{x}$

Think of $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ as an probabilistic encoder that takes an instance $\boldsymbol{x}$ and encodes it to a code $\boldsymbol{z}$. It approximates the true posterior $p_\theta(\boldsymbol{z}|\boldsymbol{x})$.

This is the connection to autoencoders in deep learning!

The Variational Bound

$$ELBO(\phi,\theta,\boldsymbol{x}) = E_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - KL\big(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z})\big)$$

We want to differentiate the ELBO above w.r.t:

1. The generative parameters θ (straightforward)
2. The variational parameters ϕ (not straightforward)

The Reparameterization Trick

Need to compute gradient of an expectation:

$$\nabla_{\boldsymbol{\phi}} E_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}\,(\boldsymbol{z})}[f(\boldsymbol{z})] = \nabla_{\boldsymbol{\phi}} \int q_{\boldsymbol{\phi}}(\boldsymbol{z})\, f(\boldsymbol{z})d\boldsymbol{z}$$

$$= \int \nabla_{\boldsymbol{\phi}} q_{\boldsymbol{\phi}}\,(\boldsymbol{z}) f(\boldsymbol{z})d\boldsymbol{z}$$

$$= \int \nabla_{\boldsymbol{\phi}} log\big(q_{\boldsymbol{\phi}}(\boldsymbol{z})\big) q_{\boldsymbol{\phi}}\,(z) f(\boldsymbol{z})d\boldsymbol{z}$$

$$= E_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z})}\big[f(\boldsymbol{z})\nabla_{\boldsymbol{\phi}}\log(q_{\boldsymbol{\phi}}(\boldsymbol{z})\big]$$

$\simeq \frac{1}{L}\sum_{l=1}^{L} f(\boldsymbol{z})\nabla_{q_{\boldsymbol{\phi}}(\boldsymbol{z}^l)} \log q_{\boldsymbol{\phi}}(\boldsymbol{z}^l)$ with $\boldsymbol{z}^{\boldsymbol{l}} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}^i)$

Note:

$$\nabla_{\boldsymbol{\phi}} log\big(q_{\boldsymbol{\phi}}(\boldsymbol{z})\big) q_{\boldsymbol{\phi}}\,(\boldsymbol{z})$$
$$= \frac{1}{q_{\boldsymbol{\phi}}(\boldsymbol{z})} \nabla_{\boldsymbol{\phi}}\, q_{\boldsymbol{\phi}}(\boldsymbol{z})\, q_{\boldsymbol{\phi}}(\boldsymbol{z})$$
$$= \nabla_{\boldsymbol{\phi}}\, q_{\boldsymbol{\phi}}(\boldsymbol{z})$$

Estimating integral with L samples

The Reparameterization Trick

This is called the score function estimator:

$$\nabla_{\boldsymbol{\phi}} E_{z\sim q_{\boldsymbol{\phi}}\,(\boldsymbol{z})}[f(\boldsymbol{z})] = E_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z})}\big[f(\boldsymbol{z})\nabla_{\boldsymbol{\phi}}\log(q_{\boldsymbol{\phi}}(\boldsymbol{z}))\big]$$

Refence: M. C. Fu. Gradient estimation.Handbooks in operations research and management science, 13:575–616,2006

Suffers from high variance. To see why:

- Doesn't use information about $f(\boldsymbol{z})$ to guide sampling of $z^l \sim q^{\boldsymbol{\phi}}(\boldsymbol{z})$
- Could sample z that is in low probability regions of $f(\boldsymbol{z})$
- Needs lots of samples to get a good estimate. With small # of samples, you get high variance

The Reparameterization Trick

To get a low variance estimator, we rewrite $z \sim q_\phi(z|x)$ as a two step process:

1. Sample noise term $\varepsilon \sim p(\varepsilon)$ where $p(\varepsilon)$ is a simple distribution
2. Create a deterministic function that combines ε with x i.e. $z = g_\phi(\varepsilon, x)$

The Reparameterization Trick

Example: suppose $q_\phi(z|x)$ is a normal distribution. Previously we wrote $z \sim N(z; \mu, \sigma)$. Now we write:

1) Sample $\varepsilon \sim N(0,1)$

2) $z = \mu + \sigma\varepsilon$

Note: This produces the same distribution

The Reparameterization Trick

More generally, it allows us to do the following:

$$
\begin{aligned}
&\nabla_\phi E_{z \sim q_\phi\,(z|x)}[f(x,z)] \\
&= \nabla_\phi E_{\varepsilon \sim p(\varepsilon)}\left[f(x, g_\phi(\varepsilon, x))\right] \\
&= E_{\varepsilon \sim p(\varepsilon)}\left[\nabla_\phi f(x, g_\phi(\varepsilon, x))\right]
\end{aligned}
$$

The gradient moves inside the expectation. This estimator has much lower variance than the score function estimator.

See Appendix D of Rezende, D. J., Mohamed, S. and Wiestra, D. (2014). Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In Proceedings of the 31st International Conference on Machine Learning.

The Reparameterization Trick

$$\widetilde{ELBO}(\phi, \theta, \mathbf{x}) = -KL\big(q_\phi(\mathbf{z}|\mathbf{x}) || p_\theta(\mathbf{z})\big) + \underbrace{E_{q_\phi(z|x)}[\log p_\theta(\mathbf{x}|\mathbf{z})]}$$

$$E_{q_\phi(z|x)}\left[\log q_\phi(\mathbf{z}|\mathbf{x})\right] \simeq \frac{1}{L}\sum_{l=1}^{L}(\log p_\theta(\mathbf{x}|\mathbf{z}^l))$$

Where $\mathbf{z}^l = g_\phi(\boldsymbol{\varepsilon}^l, \mathbf{x})$
and $\boldsymbol{\varepsilon}^l \sim p(\boldsymbol{\varepsilon})$
and L = # of samples

The Reparameterization Trick

With minibatches:

- Sample M datapoints $\boldsymbol{X}^M = \{\boldsymbol{x}^1, \dots, \boldsymbol{x}^M\}$ from the full dataset of N datapoints
- Compute

$$\widetilde{ELBO}^M(\phi, \theta, \boldsymbol{X}^M) = \frac{N}{M}\sum_{i=1}^{M}\widetilde{ELBO}(\phi, \theta, \boldsymbol{x}^i)$$

The Variational Autoencoder

How do we choose p_θ and q_ϕ?

- Could use standard probabilistic graphical models
- Or you could use a neural network to parameterize the distributions p_θ and q_ϕ

The Variational Autoencoder

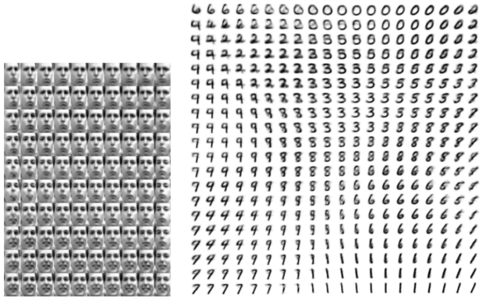
Example:

- $p(\boldsymbol{x}|\boldsymbol{z}) = N\big(\boldsymbol{x}; \mu(\boldsymbol{z}), diag(\sigma(\boldsymbol{z})^2)\big)$

Outputs of neural networks

- $q(\boldsymbol{z}|\boldsymbol{x}) = N(\boldsymbol{z}; \mu(\boldsymbol{x}), diag(\sigma(\boldsymbol{x})^2))$
- $p(\boldsymbol{z}) = N(\boldsymbol{z}; 0, \boldsymbol{I})$

The Variational Autoencoder



(a) Learned Frey Face manifold (b) Learned MNIST manifold

Results on 2D latent space. From: Kingma, D. and Welling, M. (2014). Auto-Encoding Variational Bayes. In ICLR 2014. https://arxiv.org/pdf/1312.6114.pdf