Deep Learning with TensorFlow
http://cvml.ist.ac.at/courses/DLWT_W18

Lecture 9:
Variational Autoencoders
Introduction to Variational Autoencoders

Alexandra Peste

IST Austria

January 14, 2019
Autoencoders
- The problem of dimensionality reduction
- Autoencoders
- Limitations

Variational Autoencoders
- Intuition behind VAEs
- General Architecture
- Probabilistic View of VAEs
- Learning in VAEs
- Applications
Outline

1. Autoencoders
   - The problem of dimensionality reduction
     - Autoencoders
     - Limitations

2. Variational Autoencoders
   - Intuition behind VAEs
   - General Architecture
   - Probabilistic View of VAEs
   - Learning in VAEs
   - Applications
Why care about dimensionality reduction?

Which sequence is easier to memorize? ¹

- 40, 27, 25, 36, 81, 57, 10, 73, 19, 68 ?
- 50, 25, 76, 38, 19, 58, 29, 88, 44, 22, 11, 34, 17, 52, 26, 13, 40, 20?

Reducing the dimensionality of the data helps to:

- store information more efficiently
- discover new patterns in the data, which were initially hidden from us

Unsupervised learning!

¹Example from [Ger17]
Principal Component Analysis

- \( \mathbf{X} \in \mathbb{R}^{N \times D} \) matrix data, with zero-mean columns

- Find the orthogonal directions \( \mathbf{w} \in \mathbb{R}^{D} \) along which the data has the greatest variance and project on them

- first principal component: \( w_1 = \arg\max_{\|\mathbf{w}\| = 1} \| \mathbf{Xw} \|^2 \)

- SVD: \( \mathbf{X} = \mathbf{U} \Sigma \mathbf{W}^T \), \( \mathbf{W} \in \mathbb{R}^{D \times D} \), \( \mathbf{W}^T \mathbf{W} = \mathbf{I}_D \); PCA decomposition using the first \( k < D \) components: find \( \mathbf{W}_k = [\mathbf{W}^1, \mathbf{W}^2, \ldots, \mathbf{W}^k] \) and set \( \mathbf{X}_k = \mathbf{X} \mathbf{W}_k \)
PCA Visualization of MNIST
Outline

1. Autoencoders
   - The problem of dimensionality reduction
   - Autoencoders
   - Limitations

2. Variational Autoencoders
   - Intuition behind VAEs
   - General Architecture
   - Probabilistic View of VAEs
   - Learning in VAEs
   - Applications
General Architecture and Training of Autoencoders

- **Train by minimizing**
  \[ \mathbb{E}_{x \sim D}[\|x - \hat{x}\|^2] \]

- Avoid overfitting by tying the weights of the encoder and decoder:
  \[ W_{L-\ell+1} = W^T_{\ell}, \forall \ell \in 1, L/2 \]

- Can use the encoder to initialize a NN for classifying the labels - AE learns more interesting features

- Caution: A too powerful AE might learn the identity map between input and reconstructions, making the coding layer represent just random noise

\[ ^2 \text{Image from [Ger17]} \]
General Architecture and Training of Autoencoders

- **Train by minimizing**
  \[ \mathbb{E}_{x \sim D}[\|x - \hat{x}\|^2] \]

- **Avoid overfitting by tying the weights of the encoder and decoder:**
  \[ W_{L-\ell+1} = W_{\ell}^T, \forall \ell \in 1, L/2 \]

- **Can use the encoder to initialize a NN for classifying the labels - AE learns more interesting features**

- **Caution:** A too powerful AE might learn the identity map between input and reconstructions, making the coding layer represent just random noise

---

\(^2\)Image from [Ger17]
Train by minimizing \( \mathbb{E}_{x \sim D}[\|x - \hat{x}\|^2] \)

Avoid overfitting by tying the weights of the encoder and decoder:
\[ W_{L-\ell+1} = W_{\ell}^T, \forall \ell \in 1, L/2 \]

Can use the encoder to initialize a NN for classifying the labels - AE learns more interesting features

Caution: A too powerful AE might learn the identity map between input and reconstructions, making the coding layer represent just random noise

\(^2\)Image from [Ger17]
Train by minimizing \( \mathbb{E}_{x \sim D} [\|x - \hat{x}\|^2] \)

Avoid overfitting by tying the weights of the encoder and decoder: \( W_{L-\ell+1} = W_{\ell}^T, \forall \ell \in 1, L/2 \)

Can use the encoder to initialize a NN for classifying the labels - AE learns more interesting features

Caution: A too powerful AE might learn the identity map between input and reconstructions, making the coding layer represent just random noise

\(^2\)Image from [Ger17]
Outline

1. Autoencoders
   - The problem of dimensionality reduction
   - Autoencoders
   - Limitations

2. Variational Autoencoders
   - Intuition behind VAEs
   - General Architecture
   - Probabilistic View of VAEs
   - Learning in VAEs
   - Applications
Autoencoders have their limitations:

- they can memorize the train set $\implies$ representations learned are not meaningful
- the latent space has no structure, no guarantee that distances in the original space are preserved in the encoding space
- a small perturbation to an encoding should decode to something similar to the original image
- the encodings of the train set should cover the latent space nicely $\implies$ sampling any point from the latent space will decode into a reasonable image

MNIST latent space

Image from [https://github.com/greentfrapp/keras-aae](https://github.com/greentfrapp/keras-aae)
Autoencoders have their limitations:

- they can memorize the train set $\implies$ representations learned are not meaningful
- the latent space has no structure, no guarantee that distances in the original space are preserved in the encoding space
- a small perturbation to an encoding should decode to a something similar to the original image
- the encodings of the train set should cover the latent space nicely $\implies$ sampling any point from the latent space will decode into a reasonable image

Image from https://github.com/greentfrapp/keras-aae
Autoencoders have their limitations:

- they can memorize the train set $\implies$ representations learned are not meaningful
- the latent space has no structure, no guarantee that distances in the original space are preserved in the encoding space
- a small perturbation to an encoding should decode to a something similar to the original image
- the encodings of the train set should cover the latent space nicely $\implies$ sampling any point from the latent space will decode into a reasonable image
Autoencoders have their limitations:

- they can memorize the train set \(\implies\) representations learned are not meaningful
- the latent space has no structure, no guarantee that distances in the original space are preserved in the encoding space
- a small perturbation to an encoding should decode to a something similar to the original image
- the encodings of the train set should cover the latent space nicely \(\implies\) sampling any point from the latent space will decode into a reasonable image

---

\(^3\) Image from [https://github.com/greentfrapp/keras-aae](https://github.com/greentfrapp/keras-aae)
Outline

1. Autoencoders
   - The problem of dimensionality reduction
   - Autoencoders
   - Limitations

2. Variational Autoencoders
   - Intuition behind VAEs
   - General Architecture
   - Probabilistic View of VAEs
   - Learning in VAEs
   - Applications
“What I cannot create, I do not understand” - Richard Feynman

Real or generated? ⁴

⁴Images from [KALL17], [HHS⁺18]
What are VAEs?

- AEs with a distribution on the valid codes for each input (s.t. small perturbations don’t affect too much the reconstruction)
- the distributions of all latent codes cover the space nicely \( \implies \) initializing the decoder with a random code will result in a valid image
- Loss function: Reconstruction error + Regularization on the encoder
- they are probabilistic models, rooted in the field of variational inference \( \implies \) we actually have a theory why they work!

---

What are VAEs?

- AEs with a distribution on the valid codes for each input (s.t. small perturbations don’t affect too much the reconstruction)
- the distributions of all latent codes cover the space nicely $\implies$ initializing the decoder with a random code will result in a valid image
- Loss function: Reconstruction error + Regularization on the encoder
- they are probabilistic models, rooted in the field of variational inference $\implies$ we actually have a theory why they work! 😊

---

What are VAEs?

- AEs with a distribution on the valid codes for each input (s.t. small perturbations don’t affect too much the reconstruction)

- the distributions of all latent codes cover the space nicely \( \implies \) initializing the decoder with a random code will result in a valid image

- Loss function: Reconstruction error + Regularization on the encoder

- they are probabilistic models, rooted in the field of variational inference \( \implies \) we actually have a theory why they work! 🎉
What are VAEs?

- AEs with a distribution on the valid codes for each input (s.t. small perturbations don’t affect too much the reconstruction)
- the distributions of all latent codes cover the space nicely $\implies$ initializing the decoder with a random code will result in a valid image
- Loss function: Reconstruction error + Regularization on the encoder
- they are probabilistic models, rooted in the field of variational inference $\implies$ we actually have a theory why they work! 😊

---

Why VAEs?

- **Encode** meaningfully the input in a lower dimensional space
- Initialize the decoder with $\mathcal{N}(0, I)$ to **generate** samples similar to the train set
Outline

1. Autoencoders
   - The problem of dimensionality reduction
   - Autoencoders
   - Limitations

2. Variational Autoencoders
   - Intuition behind VAEs
   - General Architecture
     - Probabilistic View of VAEs
     - Learning in VAEs
     - Applications
Encoder regularization

- The encoder with weights $\phi$ learns a distribution $q_\phi(z|x)$ over the valid codes $z \in \mathbb{R}^K$ of $x$; easiest choice for $q_\phi(z|x) : \mathcal{N}(\mu_\phi, \sigma_\phi^2 I)$

- We assume the space of our latent codes, before seeing $x$, is $p(z) \sim \mathcal{N}(0, I)$

- Enforce $q_\phi(z|x)$ for all $x$ to cover the latent space nicely $\implies$ we make $q_\phi(z|x)$ close to $\mathcal{N}(0, I)$ (hint: Use KL divergence)

- Define the regularization term:

$$\mathbb{E}_{x \sim \mathcal{D}} [KL(q_\phi(z|x) || p(z))]$$

- For independent Gaussian distributions: $KL = \frac{1}{2} \sum_{k=1}^{K} [\sigma_k^2 + \mu_k^2 - \ln \sigma_k^2 - 1]$
Encoder regularization

- The encoder with weights $\phi$ learns a distribution $q_\phi(z|x)$ over the valid codes $z \in \mathbb{R}^K$ of $x$; easiest choice for $q_\phi(z|x) : \mathcal{N}(\mu_\phi, \sigma_\phi^2 I)$

- We assume the space of our latent codes, before seeing $x$, is $p(z) \sim \mathcal{N}(0, I)$

- Enforce $q_\phi(z|x)$ for all $x$ to cover the latent space nicely $\implies$ we make $q_\phi(z|x)$ close to $\mathcal{N}(0, I)$ (hint: Use KL divergence)

- Define the regularization term:

$$\mathbb{E}_{x \sim D}[KL(q_\phi(z|x) || p(z))]$$

- For independent Gaussian distributions: $KL = \frac{1}{2} \sum_{k=1}^{K} [\sigma_k^2 + \mu_k^2 - \ln \sigma_k^2 - 1]$
The Reconstruction Term

- Given \( z \sim q_\phi(z|x) \), the decoder with weights \( \theta \) learns the most likely reconstruction of \( x \) \( \implies p_\theta(x|z) \)
- \( \|x - \hat{x}\|^2 \) is equivalent with \( p_\theta(\cdot|z) = \mathcal{N}(\hat{x}, I) \)
- other choices for \( p_\theta(x|z) \): Bernoulli distributions, if \( x \in \{0, 1\}^N \)

In general, the reconstruction term:

\[
E_{x \sim D} \left[ E_{q_\phi(z|x)} \left[ \ln p_\theta(x|z) \right] \right]
\]
The Reconstruction Term

- Given $z \sim q_\phi(z|x)$, the decoder with weights $\theta$ learns the most likely reconstruction of $x$ \iff $p_\theta(x|z)$

- $\|x - \hat{x}\|^2$ is equivalent with $p_\theta(\cdot|z) = \mathcal{N}(\hat{x}, I)$

- other choices for $p_\theta(x|z)$: Bernoulli distributions, if $x \in \{0, 1\}^N$

- In general, the reconstruction term:

$$E_{x \sim D}[E_{q_\phi(z|x)}[\ln p_\theta(x|z)]]$$
The Reconstruction Term

- Given $z \sim q_{\phi}(z|x)$, the decoder with weights $\theta$ learns the most likely reconstruction of $x$ \iff $p_{\theta}(x|z)$

- $\|x - \hat{x}\|^2$ is equivalent with $p_{\theta}(\cdot|z) = \mathcal{N}(\hat{x}, I)$

- other choices for $p_{\theta}(x|z)$: Bernoulli distributions, if $x \in \{0, 1\}^N$

- In general, the reconstruction term:

$$\mathbb{E}_{x \sim \mathcal{D}} \mathbb{E}_{q_{\phi}(z|x)}[\ln p_{\theta}(x|z)]$$
The Reconstruction Term

- Given $z \sim q_\phi(z|x)$, the decoder with weights $\theta$ learns the most likely reconstruction of $x \implies p_\theta(x|z)$

- $\|x - \hat{x}\|^2$ is equivalent with $p_\theta(\cdot|z) = \mathcal{N}(\hat{x}, I)$

- Other choices for $p_\theta(x|z)$: Bernoulli distributions, if $x \in \{0, 1\}^N$

- In general, the reconstruction term:

$$\mathbb{E}_{x \sim D}[\mathbb{E}_{q_\phi(z|x)}[\ln p_\theta(x|z)]]$$
For a single data point $x$, ELBO is defined as:

$$\mathcal{L}(\theta, \phi; x) = \mathbb{E}_{q_{\phi}(z|x)}[\ln p_\theta(x|z)] - KL(q_{\phi}(z|x) \parallel p_\theta(z))$$

**Reconstruction**

**Regularization**

Learning in VAEs $\iff$ finding:

$$\phi^*, \theta^* = \arg \max_{\phi, \theta} \mathbb{E}_{x \sim D}[\mathcal{L}(\theta, \phi; x)] \approx \arg \max_{\phi, \theta} \frac{1}{N} \sum_{n=1}^{N} \mathcal{L}(\theta, \phi; x_n)$$

ELBO is a lower bound on $\ln p_\theta(x) \implies$ VAE does MLE implicitly!

$\phi$ and $\theta$ are learned simultaneously by backpropagation

(⚠️ Sampling layer! How do we backprop through stochastic layers?)
For a single data point $x$, ELBO is defined as:

$$\mathcal{L}(\theta, \phi; x) = \mathbb{E}_{q_\phi(z|x)}[\ln p_\theta(x|z)] - KL(q_\phi(z|x) \parallel p_\theta(z))$$

Learning in VAEs $\iff$ finding:

$$\phi^*, \theta^* = \arg \max_{\phi, \theta} \mathbb{E}_{x \sim D}[\mathcal{L}(\theta, \phi; x)] \approx \arg \max_{\phi, \theta} \frac{1}{N} \sum_{n=1}^{N} \mathcal{L}(\theta, \phi; x_n)$$

ELBO is a lower bound on $\ln p_\theta(x)$ $\implies$ VAE does MLE implicitly!

$\phi$ and $\theta$ are learned simultaneously by backpropagation

⚠️ Sampling layer! How do we backprop through stochastic layers?
The Evidence Lower Bound (ELBO)

- For a single data point \( x \), ELBO is defined as:

\[
\mathcal{L}(\theta, \phi; x) = \mathbb{E}_{q_{\phi}(z|x)}[\ln p_{\theta}(x|z)] - KL(q_{\phi}(z|x) \| p_{\theta}(z))
\]

- Learning in VAEs \( \iff \) finding:

\[
\phi^*, \theta^* = \arg \max_{\phi, \theta} \mathbb{E}_{x \sim D}[\mathcal{L}(\theta, \phi; x)] \approx \arg \max_{\phi, \theta} \frac{1}{N} \sum_{n=1}^{N} \mathcal{L}(\theta, \phi; x_n)
\]

- ELBO is a lower bound on \( \ln p_{\theta}(x) \) \( \iff \) VAE does MLE implicitly!

- \( \phi \) and \( \theta \) are learned simultaneously by backpropagation

(⚠️ Sampling layer! How do we backprop through stochastic layers?)
For a single data point $x$, ELBO is defined as:

$$L(\theta, \phi; x) = \mathbb{E}_{q(\phi|z|x)}[\ln p(\theta|x|z)] - KL(q(\phi|z|x) \parallel p(\theta|z))$$

Reconstruction

Regularization

Learning in VAEs $\iff$ finding:

$$\phi^*, \theta^* = \arg \max_{\phi, \theta} \mathbb{E}_{x \sim D}[L(\theta, \phi; x)] \approx \arg \max_{\phi, \theta} \frac{1}{N} \sum_{n=1}^{N} L(\theta, \phi; x_n)$$

ELBO is a lower bound on $\ln p(\theta|x)$ $\implies$ VAE does MLE implicitly!

$\phi$ and $\theta$ are learned simultaneously by backpropagation

(⚠️ Sampling layer! How do we backprop through stochastic layers?)
Outline

1 Autoencoders
   - The problem of dimensionality reduction
   - Autoencoders
   - Limitations

2 Variational Autoencoders
   - Intuition behind VAEs
   - General Architecture
   - Probabilistic View of VAEs
   - Learning in VAEs
   - Applications
We assume our observations $x$ are the result of a hidden random variable $z \sim p(z)$, through $f_\theta$. Our goal: infer $p_\theta(z|x) \rightarrow$ intractable problem

Use variational inference to find $q_\phi(z|x) \approx p_\theta(z|x)$, by minimizing $\text{KL}(q_\phi(z|x) \parallel p_\theta(z|x))$.

Equivalent easier problem: maximize a lower bound of the log likelihood.
We assume our observations $\mathbf{x}$ are the result of a hidden random variable $\mathbf{z} \sim p(\mathbf{z})$, through $f_\theta$.

Our goal: infer $p_\theta(\mathbf{z}|\mathbf{x}) \rightarrow$ intractable problem

Use **variational inference** to find $q_\phi(\mathbf{z}|\mathbf{x}) \approx p_\theta(\mathbf{z}|\mathbf{x})$, by minimizing $\text{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p_\theta(\mathbf{z}|\mathbf{x}))$.

Equivalent easier problem:
maximize a lower bound of the log likelihood.
VAEs as PGMs

We assume our observations $x$ are the result of a hidden random variable $z \sim p(z)$, through $f_\theta$.
Our goal: infer $p_\theta(z|x) \rightarrow$ intractable problem

Use **variational inference** to find $q_\phi(z|x) \approx p_\theta(z|x)$, by minimizing $\text{KL}(q_\phi(z|x) \parallel p_\theta(z|x))$.

Equivalent easier problem:
maximize a lower bound of the log likelihood.
Defining the Evidence Lower Bound (ELBO)

How to derive the bound?

\[
\ln p_\theta(x) = \int q_\phi(z|x) \ln \frac{p_\theta(x, z)}{p_\theta(z|x)} \, dz = \mathbb{E}_{q_\phi(z|x)} \left[ \ln \frac{p_\theta(x, z)}{q_\phi(z|x)} \right] + KL(q_\phi(z|x) \| p_\theta(z|x))
\]

Therefore,

\[
\ln p_\theta(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[ \ln \frac{p_\theta(x, z)}{q_\phi(z|x)} \right] = \mathcal{L}(\theta, \phi; x)
\]

ELBO can be further rewritten into the familiar form

\[
\mathcal{L}(\theta, \phi; x) = \mathbb{E}_{q_\phi(z|x)} [\ln p_\theta(x|z)] - KL(q_\phi(z|x) \| p(z))
\]
Defining the Evidence Lower Bound (ELBO)

How to derive the bound?

\[ \ln p_\theta(x) = \int q_\phi(z|x) \ln \frac{p_\theta(x, z)}{p_\theta(z|x)} \, dz = \mathbb{E}_{q_\phi(z|x)} \left[ \ln \frac{p_\theta(x, z)}{q_\phi(z|x)} \right] + \text{KL}(q_\phi(z|x) \| p_\theta(z|x)) \]

Therefore,

\[ \ln p_\theta(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[ \ln \frac{p_\theta(x, z)}{q_\phi(z|x)} \right] = \mathcal{L}(\theta, \phi; x) \]

ELBO can be further rewritten into the familiar form

\[ \mathcal{L}(\theta, \phi; x) = \mathbb{E}_{q_\phi(z|x)} [\ln p_\theta(x|z)] - \text{KL}(q_\phi(z|x) \| p(z)) \]
Outline

1. Autoencoders
   - The problem of dimensionality reduction
   - Autoencoders
   - Limitations

2. Variational Autoencoders
   - Intuition behind VAEs
   - General Architecture
   - Probabilistic View of VAEs
   - Learning in VAEs
   - Applications
How to deal with stochastic layers?

If we sample directly from $\mathcal{N}(\mu_\phi, \sigma^2_\phi I)$, the graph loses the dependence on the encoder’s parameters $\implies$ we can’t backpropagate.

Use reparameterization trick: first sample $\epsilon \sim \mathcal{N}(0, I)$, and feed $z = \mu_\phi + \sigma_\phi \odot \epsilon$ into the decoder (the same as $z \sim \mathcal{N}(\mu_\phi, \sigma^2_\phi I)$, but backprop-friendly!

Image from https://www.kaggle.com/rvislaywade/visualizing-mnist-using-a-variational-autoencoder
How to deal with stochastic layers?

- If we sample directly from $\mathcal{N}(\mu_\phi, \sigma^2_\phi I)$, the graph losses the dependence on the encoder’s parameters $\implies$ we can’t backpropagate.

- Use **reparameterization trick**: first sample $\epsilon \sim \mathcal{N}(0, I)$, and feed $z = \mu_\phi + \sigma_\phi \odot \epsilon$ into the decoder (the same as $z \sim \mathcal{N}(\mu_\phi, \sigma^2_\phi I)$, but backprop-friendly!)

---

*Image from https://www.kaggle.com/rvislaywade/visualizing-mnist-using-a-variational-autoencoder*
Reparameterization Trick Visualized

\[ \mathcal{KL}[\mathcal{N}(\mu(X), \Sigma(X)) || \mathcal{N}(0, I)] \]

Sample \( z \) from \( \mathcal{N}(\mu(X), \Sigma(X)) \)

\[ \|X - f(z)\|^2 \]

Image from [Doe16]
Backpropagation Formulas

- **Gradient w.r.t. $\theta$:**
  \[ \nabla_{\theta} \mathcal{L}(\phi, \theta; x) = \nabla_{\theta} \mathbb{E}_{q_{\phi}(z|x)}[\ln p_{\theta}(x|z)] = \mathbb{E}_{q_{\phi}(z|x)}[\nabla_{\theta} \ln p_{\theta}(z|x)] \]
  \[ \nabla_{\theta} \mathcal{L}(\phi, \theta; x) \approx \frac{1}{S} \sum_{s=1}^{S} \nabla_{\theta} \ln p_{\theta}(x|z^{(s)}) \]

- **Gradient w.r.t $\phi$:**
  \[ \nabla_{\phi} \mathcal{L}(\phi, \theta; x) = \nabla_{\phi} \mathbb{E}_{q_{\phi}(z|x)}[\ln p_{\theta}(x|z)] - \nabla_{\phi} \text{KL}(q_{\phi}(z|x)||p(z)) \]
  \[ \mathbb{E}_{q_{\phi}(z|x)}[\ln p_{\theta}(x|z)] \approx \frac{1}{S} \sum_{s=1}^{S} \ln p_{\theta}(x|z^{(s)}) \rightarrow \text{no explicit dependence on } \phi \]

- **How to compute $\nabla_{\phi} \mathbb{E}_{q_{\phi}(z|x)}[\ln p_{\theta}(x|z)]$ ?**
Backpropagation Formulas

- **Gradient w.r.t. $\theta$:**
  \[
  \nabla_\theta \mathcal{L}(\phi, \theta; x) = \nabla_\theta \mathbb{E}_{q_\phi(z|x)}[\ln p_\theta(x|z)] = \mathbb{E}_{q_\phi(z|x)}[\nabla_\theta \ln p_\theta(z|x)]
  \]
  \[
  \nabla_\theta \mathcal{L}(\phi, \theta; x) \approx \frac{1}{S} \sum_{s=1}^S \nabla_\theta \ln p_\theta(x|z^{(s)})
  \]

- **Gradient w.r.t $\phi$:**
  \[
  \nabla_\phi \mathcal{L}(\phi, \theta; x) = \nabla_\phi \mathbb{E}_{q_\phi(z|x)}[\ln p_\theta(x|z)] - \nabla_\phi \text{KL}(q_\phi(z|x)\|p(z))
  \]
  \[
  \mathbb{E}_{q_\phi(z|x)}[\ln p_\theta(x|z)] \approx \frac{1}{S} \sum_{s=1}^S \ln p_\theta(x|z^{(s)}) \rightarrow \text{no explicit dependence on } \phi
  \]

- How to compute $\nabla_\phi \mathbb{E}_{q_\phi(z|x)}[\ln p_\theta(x|z)]$?
Backpropagation Formulas

- **Gradient w.r.t. \( \theta \):**
  \[
  \nabla_\theta L(\phi, \theta; x) = \nabla_\theta \mathbb{E}_{q_\phi(z|x)}[\ln p_\theta(x|z)] = \mathbb{E}_{q_\phi(z|x)}[\nabla_\theta \ln p_\theta(z|x)]
  \]
  \[
  \nabla_\theta L(\phi, \theta; x) \approx \frac{1}{S} \sum_{s=1}^{S} \nabla_\theta \ln p_\theta(x|z^{(s)}) \quad \checkmark
  \]

- **Gradient w.r.t \( \phi \):**
  \[
  \nabla_\phi L(\phi, \theta; x) = \nabla_\phi \mathbb{E}_{q_\phi(z|x)}[\ln p_\theta(x|z)] - \nabla_\phi \text{KL}(q_\phi(z|x)||p(z))
  \]
  \[
  \mathbb{E}_{q_\phi(z|x)}[\ln p_\theta(x|z)] \approx \frac{1}{S} \sum_{s=1}^{S} \ln p_\theta(x|z^{(s)}) \quad \rightarrow \text{no explicit dependence on } \phi \quad \xmark
  \]

- **How to compute \( \nabla_\phi \mathbb{E}_{q_\phi(z|x)}[\ln p_\theta(x|z)] \)?**
Reparameterization Trick - General Case

- Find r.v. $\epsilon \sim r(\cdot)$ and $g_\phi(\cdot)$ diff. function, s.t. $z = g_\phi(\epsilon)$

- $\mathbb{E}_{q_\phi(z|x)}[\ln p_\theta(x|z)] = \mathbb{E}_{r(\epsilon)}[\ln p_\theta(x|g_\phi(\epsilon))]$

- $\nabla_\phi \mathcal{L}(\phi, \theta; x) = \nabla_\phi \mathbb{E}_{r(\epsilon)}[\nabla_\phi \ln p_\theta(x|g_\phi(\epsilon))]$

- Can use MC approx. $\nabla_\phi \mathcal{L}(\phi, \theta; x) \approx \frac{1}{S} \sum_{s=1}^{S} \nabla_\phi \ln p_\theta(x|g_\phi(\epsilon^{(s)}))$

- For Gaussian distributions, $g_\phi(\epsilon) = \mu_\phi + \sigma_\phi \odot \epsilon$, with $\epsilon \sim \mathcal{N}(0, I)$
Reparameterization Trick - General Case

- Find r.v. $\epsilon \sim r(\cdot)$ and $g_\phi(\cdot)$ diff. function, s.t. $z = g_\phi(\epsilon)$

- $\mathbb{E}_{q_\phi(z|x)}[\ln p_\theta(x|z)] = \mathbb{E}_{r(\epsilon)}[\ln p_\theta(x|g_\phi(\epsilon))]$

- $\nabla_\phi \mathcal{L}(\phi, \theta; x) = \nabla_\phi \mathbb{E}_{r(\epsilon)}[\nabla_\phi \ln p_\theta(x|g_\phi(\epsilon))]$

- Can use MC approx. $\nabla_\phi \mathcal{L}(\phi, \theta; x) \approx \frac{1}{S} \sum_{s=1}^{S} \nabla_\phi \ln p_\theta(x|g_\phi(\epsilon^{(s)}))$

- For Gaussian distributions, $g_\phi(\epsilon) = \mu_\phi + \sigma_\phi \odot \epsilon$, with $\epsilon \sim \mathcal{N}(0, I)$
Reparameterization Trick - General Case

- Find r.v. $\epsilon \sim r(\cdot)$ and $g_\phi(\cdot)$ diff. function, s.t. $z = g_\phi(\epsilon)$

- $\mathbb{E}_{q_\phi(z|x)}[\ln p_\theta(x|z)] = \mathbb{E}_{r(\epsilon)}[\ln p_\theta(x|g_\phi(\epsilon))]$

- $\nabla_\phi \mathcal{L}(\phi, \theta; x) = \nabla_\phi \mathbb{E}_{r(\epsilon)}[\nabla_\phi \ln p_\theta(x|g_\phi(\epsilon))]$

- Can use MC approx. $\nabla_\phi \mathcal{L}(\phi, \theta; x) \approx \frac{1}{S} \sum_{s=1}^{S} \nabla_\phi \ln p_\theta(x|g_\phi(\epsilon^{(s)}))$

- For Gaussian distributions, $g_\phi(\epsilon) = \mu_\phi + \sigma_\phi \odot \epsilon$, with $\epsilon \sim \mathcal{N}(0, I)$
Reparameterization Trick - General Case

- Find r.v. $\epsilon \sim r(\cdot)$ and $g_\phi(\cdot)$ diff. function, s.t. $z = g_\phi(\epsilon)$

- $E_{q_\phi(z|x)}[\ln p_\theta(x|z)] = E_{r(\epsilon)}[\ln p_\theta(x|g_\phi(\epsilon))]$

- $\nabla_\phi L(\phi, \theta; x) = \nabla_\phi E_{r(\epsilon)}[\nabla_\phi \ln p_\theta(x|g_\phi(\epsilon))]$

- Can use MC approx. $\nabla_\phi L(\phi, \theta; x) \approx \frac{1}{S} \sum_{s=1}^{S} \nabla_\phi \ln p_\theta(x|g_\phi(\epsilon^{(s)}))$

- For Gaussian distributions, $g_\phi(\epsilon) = \mu_\phi + \sigma_\phi \odot \epsilon$, with $\epsilon \sim \mathcal{N}(0, I)$
Reparameterization Trick - General Case

- Find r.v. $\epsilon \sim r(\cdot)$ and $g_\phi(\cdot)$ diff. function, s.t. $z = g_\phi(\epsilon)$

- $\mathbb{E}_{q_\phi(z|x)}[\ln p_\theta(x|z)] = \mathbb{E}_{r(\epsilon)}[\ln p_\theta(x|g_\phi(\epsilon))]$

- $\nabla_\phi \mathcal{L}(\phi, \theta; x) = \nabla_\phi \mathbb{E}_{r(\epsilon)}[\nabla_\phi \ln p_\theta(x|g_\phi(\epsilon))]$

- Can use MC approx. $\nabla_\phi \mathcal{L}(\phi, \theta; x) \approx \frac{1}{S} \sum_{s=1}^{S} \nabla_\phi \ln p_\theta(x|g_\phi(\epsilon^{(s)}))$

- For Gaussian distributions, $g_\phi(\epsilon) = \mu_\phi + \sigma_\phi \odot \epsilon$, with $\epsilon \sim \mathcal{N}(0, I)$
Outline

1. Autoencoders
   - The problem of dimensionality reduction
   - Autoencoders
   - Limitations

2. Variational Autoencoders
   - Intuition behind VAEs
   - General Architecture
   - Probabilistic View of VAEs
   - Learning in VAEs
   - Applications
Implementation using `tf.contrib.distributions`

```python
# Full example at:

import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data
tfd = tf.contrib.distributions

def make_encoder(data, code_size):
    x = tf.layers.flatten(data)
    x = tf.layers.dense(x, 200, tf.nn.relu)
    x = tf.layers.dense(x, 200, tf.nn.relu)
    loc = tf.layers.dense(x, code_size)
    scale = tf.layers.dense(x, code_size, tf.nn.softplus)
    return tfd.MultivariateNormalDiag(loc, scale)

def make_prior(code_size):
    loc = tf.zeros(code_size)
    scale = tf.ones(code_size)
    return tfd.MultivariateNormalDiag(loc, scale)

def make_decoder(code, data_shape):
    x = code
    x = tf.layers.dense(x, 200, tf.nn.relu)
    x = tf.layers.dense(x, 200, tf.nn.relu)
    logit = tf.layers.dense(x, np.prod(data_shape))
    logit = tf.reshape(logit, [-1, ] + data_shape)
    return tfd.Independent(tfd.Bernoulli(logit), 2)

data = tf.placeholder(tf.float32, [None, 28, 28])
make_encoder = tf.make_template('encoder', make_encoder)
make_decoder = tf.make_template('decoder', make_decoder)

# Define the model.
prior = make_prior(code_size=2)
posterior = make_encoder(data, code_size=2)
code = posterior.sample()

# Define the loss.
likelihood = make_decoder(code, [28, 28]).log_prob(data)
divergence = tfd.kl_divergence(posterior, prior)
elbo = tf.reduce_mean(likelihood - divergence)
optimize = tf.train.AdamOptimizer(0.001).minimize(-elbo)
samples = make_decoder(prior.sample(10), [28, 28]).mean()
```
Learned Latent Manifold

MNIST latent manifold

Thank you!
References I


