Machine Learning and
Computer Vision Group

IST AUSTRIA
Institute of Science and Technology

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

Table of Contents

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

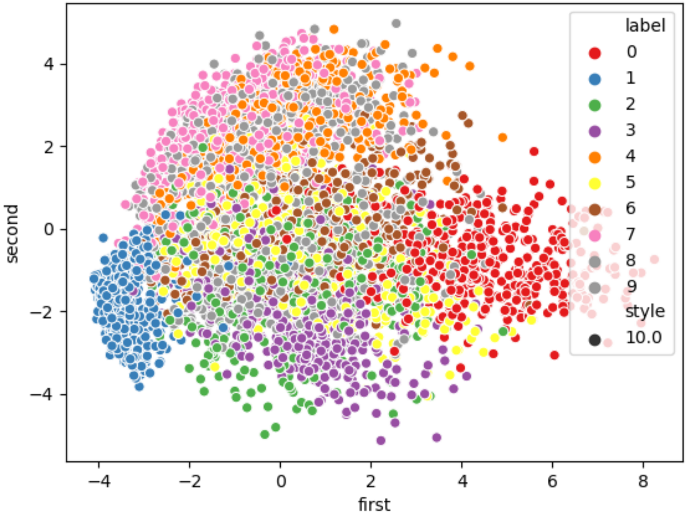
Unsupervised learning!

[1] Example from [Ger17]

Principal Component Analysis

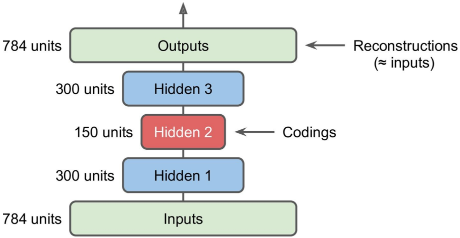
- $X \in \mathbb{R}^{N \times D}$ matrix data, with zero-mean columns
- Find the orthogonal directions $w \in \mathbb{R}^D$ along which the data has the greatest variance and project on them
- first principal component: $w_1 = \text{argmax}_{\|w\|=1} \|Xw\|^2$
- SVD: $X = U\Sigma W^T$, $W \in \mathbb{R}^{D \times D}$, $\quad W^T W = I_D$; PCA decomposition using the first $k < D$ components: find $W_k = [W^1, W^2, \ldots, W^k]$ and set $X_k = XW_k$

PCA Visualization of MNIST



Outline

General Architecture and Training of Autoencoders

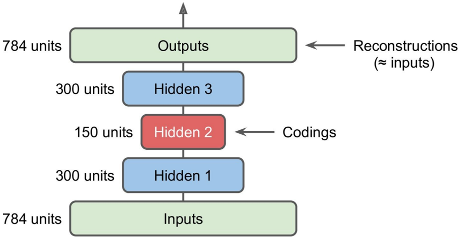


Stacked Autoencoder [2]

- Train by minimizing $\mathbb{E}_{x \sim D}[\|x - \hat{x}\|^2]$
- Avoid overfitting by tying the weights of the encoder and decoder: $W_{L-\ell+1} = W_\ell^T, \forall \ell \in \overline{1, L/2}$
- Can use the encoder to initialize a NN for classifiying 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]

General Architecture and Training of Autoencoders

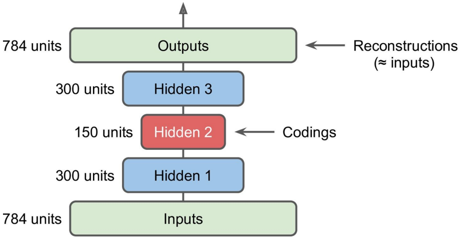


Stacked Autoencoder [2]

- Train by minimizing $\mathbb{E}_{x \sim D}[\|x - \hat{x}\|^2]$
- Avoid overfitting by tying the weights of the encoder and decoder: $W_{L-\ell+1} = W_\ell^T, \forall \ell \in \overline{1, L/2}$
- Can use the encoder to initialize a NN for classifiying 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]

General Architecture and Training of Autoencoders

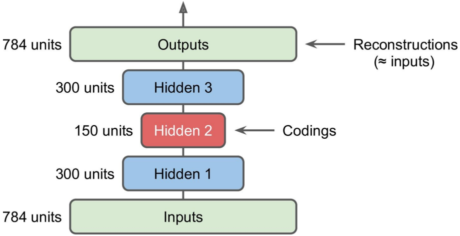


Stacked Autoencoder [2]

- Train by minimizing $\mathbb{E}_{x \sim D}[\|x - \hat{x}\|^2]$
- Avoid overfitting by tying the weights of the encoder and decoder: $W_{L-\ell+1} = W_\ell^T, \forall \ell \in \overline{1, L/2}$
- Can use the encoder to initialize a NN for classifiying 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]

General Architecture and Training of Autoencoders



Stacked Autoencoder [2]

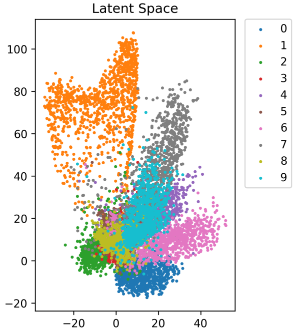
- Train by minimizing $\mathbb{E}_{x \sim D}[\|x - \hat{x}\|^2]$
- Avoid overfitting by tying the weights of the encoder and decoder: $W_{L-\ell+1} = W_\ell^T, \forall \ell \in \overline{1, L/2}$
- Can use the encoder to initialize a NN for classifiying the labels - AE learns more interesting features
- 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:

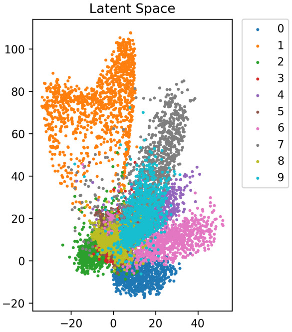


MNIST latent space [3]

- they can memorize the train set $\Longrightarrow$ 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 $\Longrightarrow$ sampling any point from the latent space will decode into a reasonable image

[3] Image from https://github.com/greentfrapp/keras-aae

Autoencoders have their limitations:

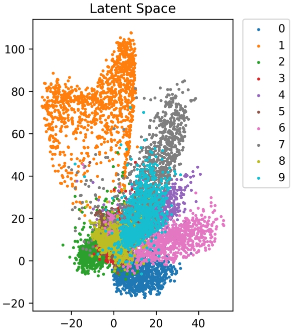


MNIST latent space [3]

- they can memorize the train set $\Longrightarrow$ 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 $\Longrightarrow$ sampling any point from the latent space will decode into a reasonable image

[3] Image from https://github.com/greentfrapp/keras-aae

Autoencoders have their limitations:

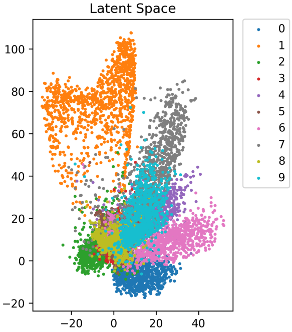


MNIST latent space [3]

- 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

Autoencoders have their limitations:



MNIST latent space [3]

- they can memorize the train set $\Longrightarrow$ 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 $\Longrightarrow$ sampling any point from the latent space will decode into a reasonable image

[3] Image from https://github.com/greentfrapp/keras-aae

Outline

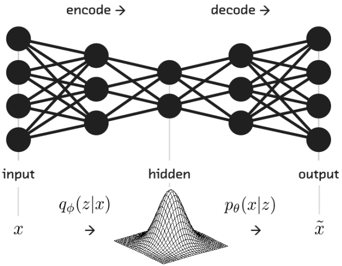
Generative Models in Deep Learning

"What I cannot create, I do not understand" - Richard Feynman

Real or generated? [4]

[4] Images from [KALL17], [HHS+18]

What are VAEs?

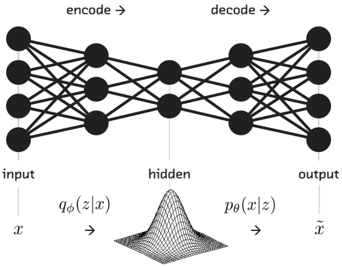


VAEs [5]

- 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 $\Longrightarrow$ 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 fied of variational inference $\Longrightarrow$ we actually have a theory why they work!

[5] Image from https://blog.fastforwardlabs.com/2016/08/22/under-the-hood-of-the-variational-autoencoder-in-html

What are VAEs?

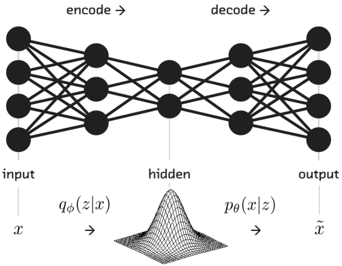


VAEs [5]

- 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 fied of variational inference $\implies$ we actually have a theory why they work!

[5] Image from https://blog.fastforwardlabs.com/2016/08/22/under-the-hood-of-the-variational-autoencoder-in.html

What are VAEs?

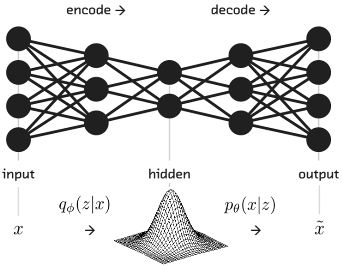


VAEs [5]

- 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 $\Longrightarrow$ 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 fied of variational inference $\Longrightarrow$ we actually have a theory why they work!

[5] Image from https://blog.fastforwardlabs.com/2016/08/22/under-the-hood-of-the-variational-autoencoder-in.html

What are VAEs?

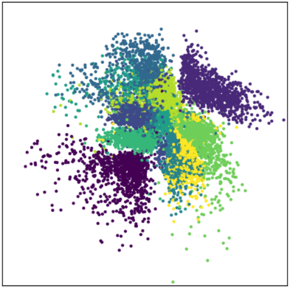


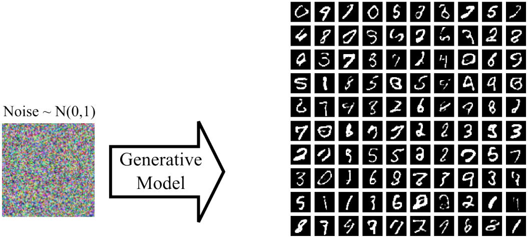
VAEs [5]

- 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 fied of variational inference $\implies$ we actually have a theory why they work! 👍

[5] Image from https://blog.fastforwardlabs.com/2016/08/22/under-the-hood-of-the-variational-autoencoder-in.html

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

Encoder regularization

- The encoder with weights ϕ learns a distribution $q_\phi(\mathbf{z}|\mathbf{x})$ over the valid codes $\mathbf{z} \in \mathbb{R}^K$ of $\mathbf{x}$; easiest choice for $q_\phi(\mathbf{z}|\mathbf{x})$: $\mathcal{N}(\mu_\phi, \sigma_\phi^2 I)$
- We assume the space of our latent codes, before seeing $\mathbf{x}$, is $p(\mathbf{z}) \sim \mathcal{N}(0, I)$
- Enforce $q_\phi(\mathbf{z}|\mathbf{x})$ for all $\mathbf{x}$ to cover the latent space nicely $\Longrightarrow$ we make $q_\phi(\mathbf{z}|\mathbf{x})$ close to $\mathcal{N}(0, I)$ (hint: Use KL divergence)
- Define the regularization term:

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

- 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 ϕ learns a distribution $q_\phi(\mathbf{z}|\mathbf{x})$ over the valid codes $\mathbf{z} \in \mathbb{R}^K$ of $\mathbf{x}$; easiest choice for $q_\phi(\mathbf{z}|\mathbf{x})$: $\mathcal{N}(\mu_\phi, \sigma_\phi^2 I)$
- We assume the space of our latent codes, before seeing $\mathbf{x}$, is $p(\mathbf{z}) \sim \mathcal{N}(0, I)$
- Enforce $q_\phi(\mathbf{z}|\mathbf{x})$ for all $\mathbf{x}$ to cover the latent space nicely $\Longrightarrow$ we make $q_\phi(\mathbf{z}|\mathbf{x})$ close to $\mathcal{N}(0, I)$ (hint: Use KL divergence)
- Define the regularization term:

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

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

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

The Reconstruction Term

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

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

The Reconstruction Term

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

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

The Reconstruction Term

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

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

The Evidence Lower Bound (ELBO)

- For a single data point $\mathbf{x}$, ELBO is defined as:

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

Reconstruction — Regularization

- Learning in VAEs $\iff$ finding:

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

- ELBO is a lower bound on $\ln p_\theta(\mathbf{x}) \implies$ VAE does MLE implicitly!
- ϕ and θ are learned simultaneously by backpropagation
(⚠ Sampling layer! How do we backprop through stochastic layers?)

The Evidence Lower Bound (ELBO)

- For a single data point $\mathbf{x}$, ELBO is defined as:

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

Reconstruction　　　　Regularization

- Learning in VAEs $\Longleftrightarrow$ finding:

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

- ELBO is a lower bound on $\ln p_\theta(\mathbf{x}) \Longrightarrow$ VAE does MLE implicitly!
- ϕ and θ are learned simultaneously by backpropagation (⚠ Sampling layer! How do we backprop through stochastic layers?)

The Evidence Lower Bound (ELBO)

- For a single data point $\mathbf{x}$, ELBO is defined as:

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

Reconstruction   Regularization

- Learning in VAEs $\iff$ finding:

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

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

- ϕ and θ are learned simultaneously by backpropagation
(⚠ Sampling layer! How do we backprop through stochastic layers?)

The Evidence Lower Bound (ELBO)

- For a single data point $\mathbf{x}$, ELBO is defined as:

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

Reconstruction   Regularization

- Learning in VAEs $\Longleftrightarrow$ finding:

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

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

- ϕ and θ are learned simultaneously by backpropagation
(⚠ Sampling layer! How do we backprop through stochastic layers?)

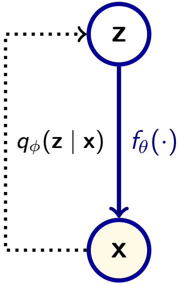
Outline

VAEs as PGMs

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

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

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

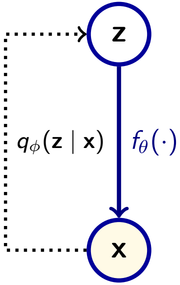


VAEs as PGMs

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

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

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

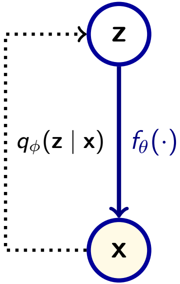


VAEs as PGMs

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

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

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



Defining the Evidence Lower Bound (ELBO)

How to derive the bound?

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

Therefore,

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

ELBO can be further rewritten into the familiar form

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

Defining the Evidence Lower Bound (ELBO)

How to derive the bound?

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

Therefore,

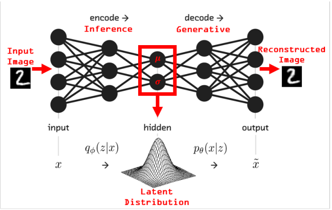
$$\ln p_\theta(\mathbf{x}) \geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\ln \frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right] = \mathcal{L}(\theta,\phi;\mathbf{x})$$

ELBO can be further rewritten into the familiar form

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

Outline

How to deal with stochastic layers?

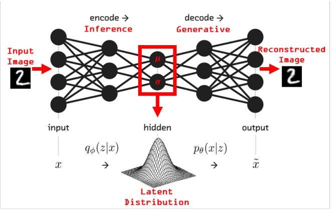


sampling from $\mathcal{N}(\mu, \sigma^2 I)$ in the middle [6]

- If we sample directly from $\mathcal{N}(\mu_\phi, \sigma_\phi^2 I)$, the graph losses the dependence on the encoder's parameters $\Longrightarrow$ we can't backpropagate
- Use **reparameterization trick**: first sample $\epsilon \sim \mathcal{N}(0, I)$, and feed $\mathbf{z} = \mu_\phi + \sigma_\phi \odot \epsilon$ into the decoder (the same as $z \sim \mathcal{N}(\mu_\phi, \sigma_\phi^2 I)$, but backprop-friendly!)

[7] Image from https://www.kaggle.com/rvislaywade/visualizing-mnist-using-a-variational-autoencoder

How to deal with stochastic layers?

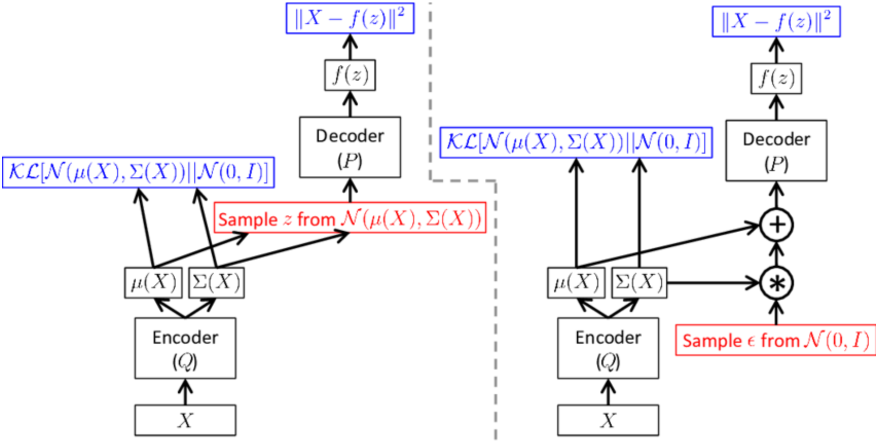


sampling from $\mathcal{N}(\mu, \sigma^2 I)$ in the middle [6]

- If we sample directly from $\mathcal{N}(\mu_\phi, \sigma_\phi^2 I)$, 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 $\mathbf{z} = \mu_\phi + \sigma_\phi \odot \epsilon$ into the decoder (the same as $z \sim \mathcal{N}(\mu_\phi, \sigma_\phi^2 I)$, but backprop-friendly!)

[7] Image from https://www.kaggle.com/rvislaywade/visualizing-mnist-using-a-variational-autoencoder

Reparameterization Trick Visualized



8

[8] Image from [Doe16]

Backpropagation Formulas

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

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

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

Backpropagation Formulas

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

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

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

Backpropagation Formulas

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

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

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

Reparameterization Trick - General Case

- Find r.v. $\epsilon \sim r(\cdot)$ and $g_\phi(\cdot)$ diff. function, s.t. $\mathbf{z} = g_\phi(\epsilon)$
- $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\ln p_\theta(\mathbf{x}|\mathbf{z})] = \mathbb{E}_{r(\epsilon)}[\ln p_\theta(\mathbf{x}|g_\phi(\epsilon))]$
- $\nabla_\phi \mathcal{L}(\phi, \theta; \mathbf{x}) = \nabla_\phi \mathbb{E}_{r(\epsilon)}[\nabla_\phi \ln p_\theta(\mathbf{x}|g_\phi(\epsilon))]$
- Can use MC approx. $\nabla_\phi \mathcal{L}(\phi, \theta; \mathbf{x}) \approx \frac{1}{S} \sum_{s=1}^{S} \nabla_\phi \ln p_\theta(\mathbf{x}|g_\phi(\epsilon^{(s)}))$ ✓
- 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. $\mathbf{z} = g_\phi(\epsilon)$
- $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\ln p_\theta(\mathbf{x}|\mathbf{z})] = \mathbb{E}_{r(\epsilon)}[\ln p_\theta(\mathbf{x}|g_\phi(\epsilon))]$
- $\nabla_\phi \mathcal{L}(\phi, \theta; \mathbf{x}) = \nabla_\phi \mathbb{E}_{r(\epsilon)}[\nabla_\phi \ln p_\theta(\mathbf{x}|g_\phi(\epsilon))]$
- Can use MC approx. $\nabla_\phi \mathcal{L}(\phi, \theta; \mathbf{x}) \approx \frac{1}{S}\sum_{s=1}^{S} \nabla_\phi \ln p_\theta(\mathbf{x}|g_\phi(\epsilon^{(s)}))$ ✓
- 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. $\mathbf{z} = g_\phi(\epsilon)$
- $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\ln p_\theta(\mathbf{x}|\mathbf{z})] = \mathbb{E}_{r(\epsilon)}[\ln p_\theta(\mathbf{x}|g_\phi(\epsilon))]$
- $\nabla_\phi \mathcal{L}(\phi, \theta; \mathbf{x}) = \nabla_\phi \mathbb{E}_{r(\epsilon)}[\nabla_\phi \ln p_\theta(\mathbf{x}|g_\phi(\epsilon))]$
- Can use MC approx. $\nabla_\phi \mathcal{L}(\phi, \theta; \mathbf{x}) \approx \frac{1}{S} \sum_{s=1}^{S} \nabla_\phi \ln p_\theta(\mathbf{x}|g_\phi(\epsilon^{(s)}))$ ✓
- 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. $\mathbf{z} = g_\phi(\epsilon)$
- $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\ln p_\theta(\mathbf{x}|\mathbf{z})] = \mathbb{E}_{r(\epsilon)}[\ln p_\theta(\mathbf{x}|g_\phi(\epsilon))]$
- $\nabla_\phi \mathcal{L}(\phi, \theta; \mathbf{x}) = \nabla_\phi \mathbb{E}_{r(\epsilon)}[\nabla_\phi \ln p_\theta(\mathbf{x}|g_\phi(\epsilon))]$
- Can use MC approx. $\nabla_\phi \mathcal{L}(\phi, \theta; \mathbf{x}) \approx \frac{1}{S}\sum_{s=1}^{S} \nabla_\phi \ln p_\theta(\mathbf{x}|g_\phi(\epsilon^{(s)}))$ ✓
- For Gaussian distributions, $g_\phi(\epsilon) = \mu_\phi + \sigma_\phi \odot \epsilon$, with $\epsilon \sim \mathcal{N}(0, I)$

Reparameterization Trick - General Case

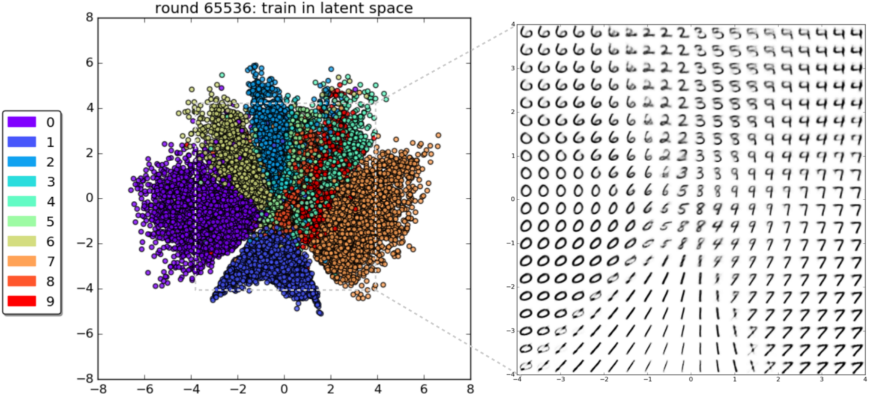
- Find r.v. $\epsilon \sim r(\cdot)$ and $g_\phi(\cdot)$ diff. function, s.t. $\mathbf{z} = g_\phi(\epsilon)$
- $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\ln p_\theta(\mathbf{x}|\mathbf{z})] = \mathbb{E}_{r(\epsilon)}[\ln p_\theta(\mathbf{x}|g_\phi(\epsilon))]$
- $\nabla_\phi \mathcal{L}(\phi, \theta; \mathbf{x}) = \nabla_\phi \mathbb{E}_{r(\epsilon)}[\nabla_\phi \ln p_\theta(\mathbf{x}|g_\phi(\epsilon))]$
- Can use MC approx. $\nabla_\phi \mathcal{L}(\phi, \theta; \mathbf{x}) \approx \frac{1}{S}\sum_{s=1}^{S} \nabla_\phi \ln p_\theta(\mathbf{x}|g_\phi(\epsilon^{(s)}))$ ✓
- For Gaussian distributions, $g_\phi(\epsilon) = \mu_\phi + \sigma_\phi \odot \epsilon$, with $\epsilon \sim \mathcal{N}(0, I)$

Outline

Implementation using tf.contrib.distributions

```
# Full example at:
# https://danijar.com/building-variational-auto-encoders-in-tensorflow/

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 [9]

[9] Image from https://blog.fastforwardlabs.com/2016/08/12/introducing-variational-autoencoders-in-prose-and.html

Thank you!

References I

[Doe16] Carl Doersch.
Tutorial on variational autoencoders.
CoRR, abs/1606.05908, 2016.

[Ger17] Aurelien Geron.
Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems.
O'Reilly Media, 2017.

[HHS+18] Huaibo Huang, Ran He, Zhenan Sun, Tieniu Tan, et al.
Introvae: Introspective variational autoencoders for photographic image synthesis.
In *Advances in Neural Information Processing Systems*, pages 52–63, 2018.

[KALL17] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen.
Progressive growing of gans for improved quality, stability, and variation.
arXiv preprint arXiv:1710.10196, 2017.

References II

[KW13] Diederik P Kingma and Max Welling.
Auto-encoding variational bayes.
arXiv preprint arXiv:1312.6114, 2013.