Assignment 2

Learning latent space representations

and application to image generation

Benjamin Negrevergne, Alexandre Vérine

PSL University – Paris Dauphine – Équipes *MILES*

MILES
Machine Intelligence and Learning Systems

Outline

Reptile classification challenge

Task: classify pictures of crocodiles and alligators

Representation 1: features

$$\begin{bmatrix} \textit{beast\_color} & = & \textit{Light} \\ \textit{beast\_size} & = & \textit{Large} \end{bmatrix} \quad \rightarrow \quad f_1 \quad \rightarrow \quad \text{crocodile}$$

Reptile classification challenge

Task: classify pictures of crocodiles and alligators

Representation 1: features

$$\begin{bmatrix} beast\_color & = & Dark \\ beast\_size & = & Small \end{bmatrix} \quad \longrightarrow \quad f_1 \quad \longrightarrow \quad \text{alligator}$$

Reptile classification challenge

Task: classify pictures of crocodiles and alligators

Representation 1: features

$$\begin{bmatrix} \textit{beast\_color} & = & \textit{Dark} \\ \textit{beast\_size} & = & \textit{Small} \end{bmatrix} \quad \rightarrow \quad f_1 \quad \rightarrow \quad \text{alligator}$$

Representation 2: Use raw pixel data

$$\rightarrow \quad f_2 \quad \rightarrow \quad \text{crocodile}$$

Reptile classification challenge

Task: classify pictures of crocodiles and alligators

- **Representation 1: features**

$$\begin{bmatrix} \textit{beast\_color} & = & \textit{Dark} \\ \textit{beast\_size} & = & \textit{Small} \end{bmatrix} \rightarrow f_1 \rightarrow \text{alligator}$$

- **Representation 2: Use raw pixel data**

$$\rightarrow f_2 \rightarrow \text{alligator}$$

Reptile classification challenge

Task: classify pictures of crocodiles and alligators

Representation 1: features

$$\begin{bmatrix} \mathit{beast\_color} & = & \mathit{Dark} \\ \mathit{beast\_size} & = & \mathit{Small} \end{bmatrix} \quad \rightarrow \quad f_1 \quad \rightarrow \quad \text{alligator}$$

Representation 2: Use raw pixel data

$$\rightarrow \quad f_2 \quad \rightarrow \quad \text{alligator}$$

Which **representation** of the input is easier to work with ? Why ?

What's the difference?

$$f_1 \quad : \quad \mathbb{R}^2 \quad \rightarrow \quad \mathbb{R}$$

Input space 1: $\mathbb{R}^2$

$$f_2 \quad : \quad \mathbb{R}^{w \times h \times 3} \quad \rightarrow \quad \mathbb{R}$$

Input space 2: $\mathbb{R}^{w \times h \times 3}$

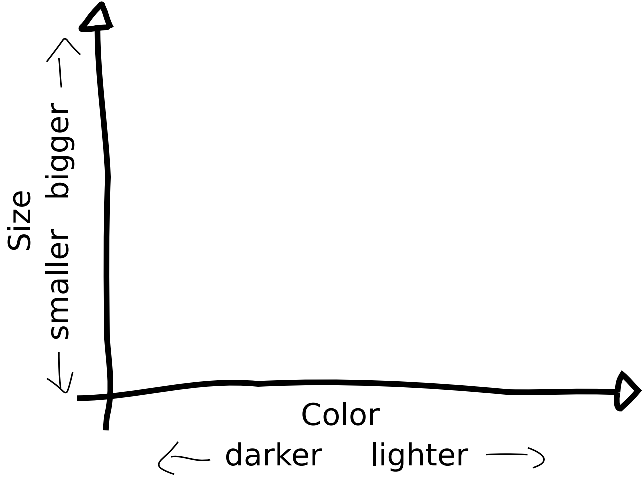
What's the difference?

$f_1 \quad : \quad \mathbb{R}^2 \quad \rightarrow \quad \mathbb{R}$

Input space 1: $\mathbb{R}^2$

$f_2 \quad : \quad \mathbb{R}^{w \times h \times 3} \quad \rightarrow \quad \mathbb{R}$

Input space 2: $\mathbb{R}^{w \times h \times 3}$



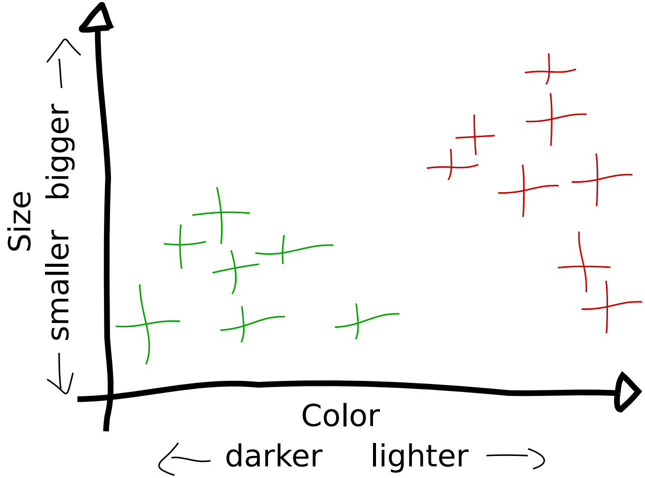
What's the difference?

$f_1 \quad : \quad \mathbb{R}^2 \quad \to \quad \mathbb{R}$

Input space 1: $\mathbb{R}^2$

$f_2 \quad : \quad \mathbb{R}^{w \times h \times 3} \quad \to \quad \mathbb{R}$

Input space 2: $\mathbb{R}^{w \times h \times 3}$



Data points **are linearly separable** in $\mathbb{R}^2$

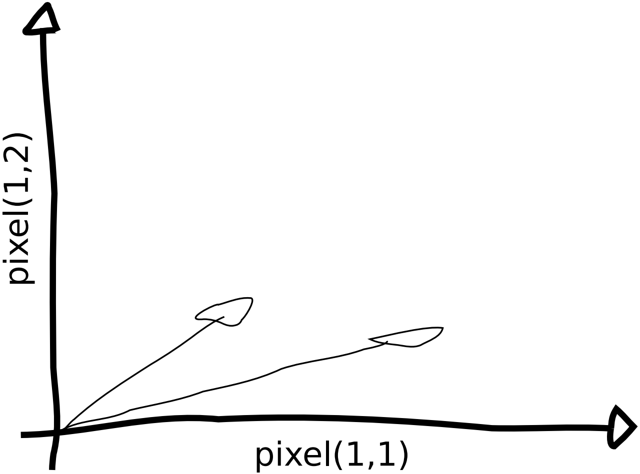
What's the difference?

$f_1 \quad : \quad \mathbb{R}^2 \quad \rightarrow \quad \mathbb{R}$

Input space 1: $\mathbb{R}^2$

$f_2 \quad : \quad \mathbb{R}^{w \times h \times 3} \quad \rightarrow \quad \mathbb{R}$

Input space 2: $\mathbb{R}^{w \times h \times 3}$



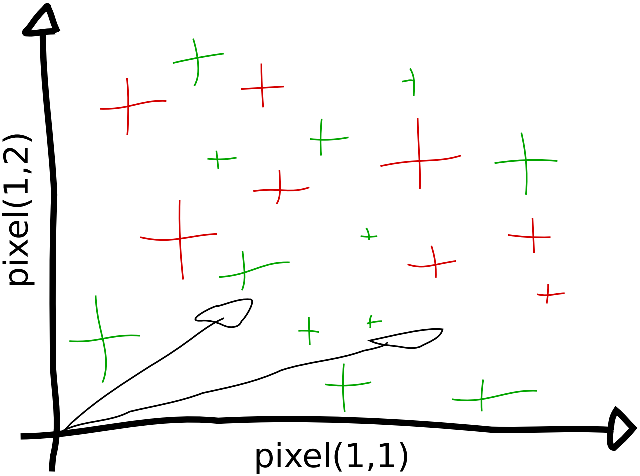
What's the difference?

$f_1 \quad : \quad \mathbb{R}^2 \quad \to \quad \mathbb{R}$

Input space 1: $\mathbb{R}^2$

$f_2 \quad : \quad \mathbb{R}^{w \times h \times 3} \quad \to \quad \mathbb{R}$

Input space 2: $\mathbb{R}^{w \times h \times 3}$



Data points **are not linearly separable** in $\mathbb{R}^{w \times h \times 3}$, Why?

Pro/cons

- **Representation 1: hand-crafted features**
  - \+ Can be processed with simple (linear) models
    - Individual features are highly discriminant
    - Input data points are (almost) linearly separable
  - – Requires expertise and manual labor to be built
  - – No extra information in case of ties

Pro/cons

- **Representation 1: hand-crafted features**
  - \+ Can be processed with simple (linear) models
    - Individual features are highly discriminant
    - Input data points are (almost) linearly separable
  - – Requires expertise and manual labor to be built
  - – No extra information in case of ties

- **Representation 2: raw pixel data**
  - \+ Contains all the information available
  - – Input data points are not (nearly) linearly separable

    Features are individually non-discriminant
  - – Difficult to process with simple models

Pro/cons

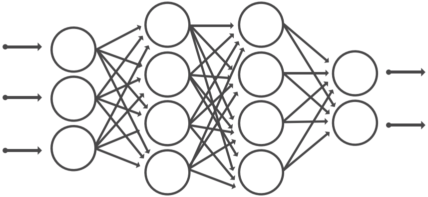
- **Representation 1: hand-crafted features**
  - \+ Can be processed with simple (linear) models
    - Individual features are highly discriminant
    - Input data points are (almost) linearly separable
  - – Requires expertise and manual labor to be built
  - – No extra information in case of ties

- **Representation 2: raw pixel data**
  - \+ Contains all the information available
  - – Input data points are not (nearly) linearly separable

    Features are individually non-discriminant
  - – Difficult to process with simple models

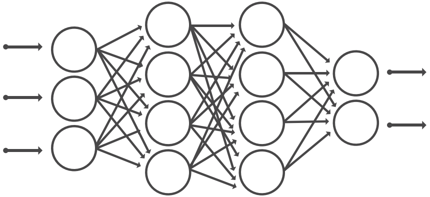
Can we build high level features automatically ?

Deep neural nets



$$f = f_1 \circ f_2 \circ \ldots \circ f_{n-1} \circ f_n$$

Deep neural nets



$$f = f_1 \circ f_2 \circ \ldots \circ f_{n-1} \circ f_n$$

- $f_1 : \mathcal{X} \to \mathcal{Z}_1$
- $f_i : \mathcal{Z}_{i-1} \to \mathcal{Z}_i$
- $f_n : \mathcal{Z}_{n-1} \to \mathcal{Y}$

Deep neural nets

$$f = \underbrace{f_1 \circ f_2 \circ \ldots \circ f_{n-1}}_{g} \circ \underbrace{f_n}_{h}$$

Remarks

- h is a linear classifier: $\mathcal{Z}_{n-1} \to \mathcal{Y}$
  ▶ Data points **must be** linearly separable in $\mathcal{Z}_{n-1}$
- Data points x **are not** linearly separable in the input space $\mathcal{X}$

Deep neural nets

$$f = \underbrace{f_1 \circ f_2 \circ \ldots \circ f_{n-1}}_{g} \circ \underbrace{f_n}_{h}$$

Remarks

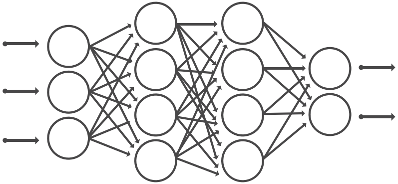
- h is a linear classifier: $\mathcal{Z}_{n-1} \to \mathcal{Y}$
  ▶ Data points **must be** linearly separable in $\mathcal{Z}_{n-1}$
- Data points x **are not** linearly separable in the input space $\mathcal{X}$

What is g?

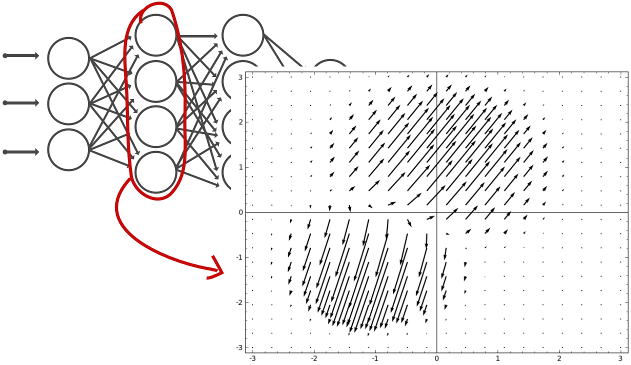
- a function $g : \mathcal{X} \to \mathcal{Z}_{n-1}$
- such that data points are linearly separable in $\mathcal{Z}_{n-1}$

▶ a representation of the point in $\mathcal{X}$ that is adequate for the task at hand

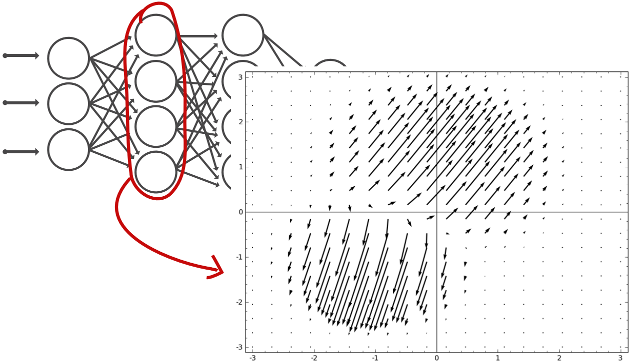
Learning deep representations

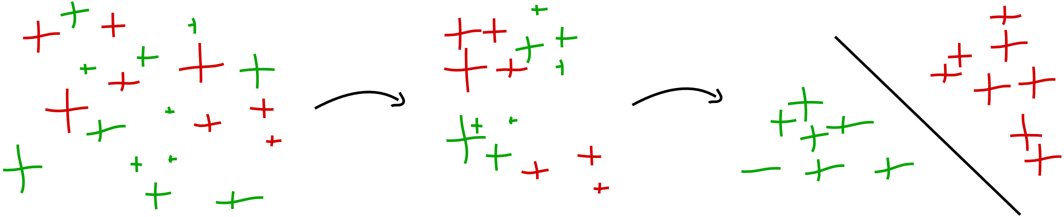


Learning deep representations

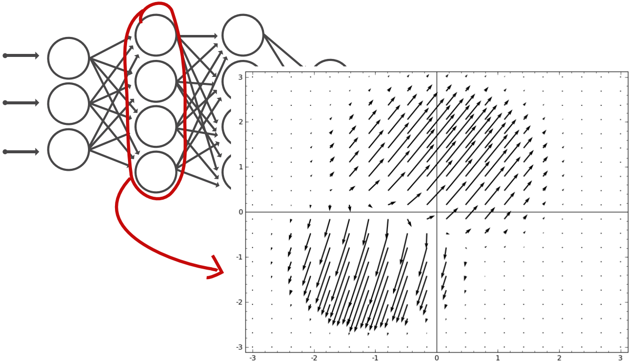


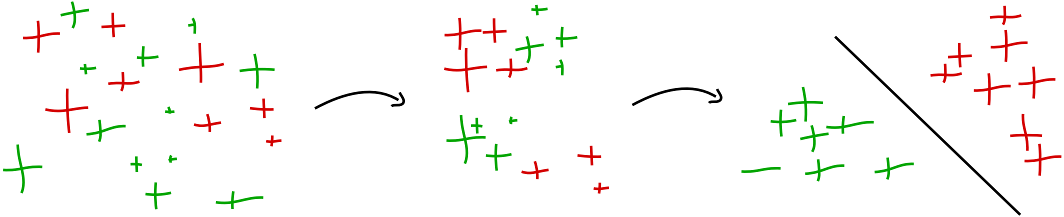
Learning deep representations





Learning deep representations





Note: the decision boundary is a complex high dimensional hypersurface in $\mathcal{X}$

Deep representations: first lessons

- DNN learn how to projet inputs into a latent space
- The structure of the latent space is useful for the task at hand.
- Given an input $x \in \mathcal{X}$ we say that $g(x)$ is an **embedding** of x, i.e. continuous vector representations of input data (image, text, graph ...)

Outline

Outline

Image embeddings

- D: a database with 80 000 pictures.
- $f = g \circ h$: a classifier trained on object recognition
  g non-linear function, h linear classifier
- x a random picture from the internet

$$x_1 = \arg\min_{x' \in D} ||g(x) - g(x')||$$

$$x_2 = \arg\min_{x' \in D \setminus \{x_1\}} ||g(x) - g(x')||$$

Image embeddings

- D: a database with 80 000 pictures.
- $f = g \circ h$: a classifier trained on object recognition
  g non-linear function, h linear classifier
- x a random picture from the internet

$$x_1 = \arg\min_{x' \in D} ||g(x) - g(x')||$$

$$x_2 = \arg\min_{x' \in D \setminus \{x_1\}} ||g(x) - g(x')||$$

x

Image embeddings

- D: a database with 80 000 pictures.
- $f = g \circ h$: a classifier trained on object recognition
  g non-linear function, h linear classifier
- x a random picture from the internet

$$x_1 = \arg\min_{x' \in D} ||g(x) - g(x')||$$

$$x_2 = \arg\min_{x' \in D \setminus \{x_1\}} ||g(x) - g(x')||$$

x

x_1

x_2

Query by image (2)

X

Query by image (2)

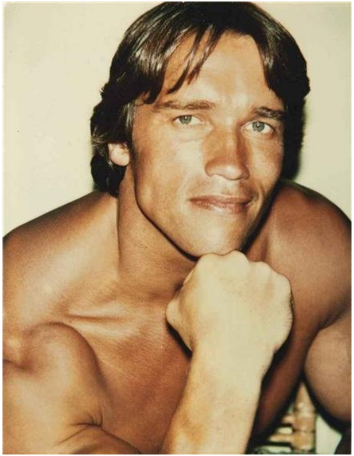
x

x_1

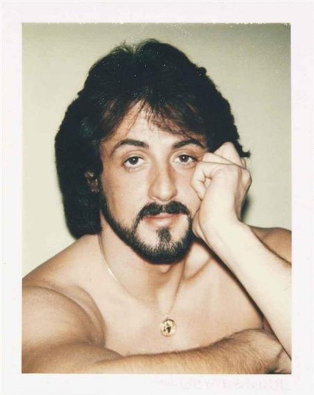
x_2

Query by image (2)

x



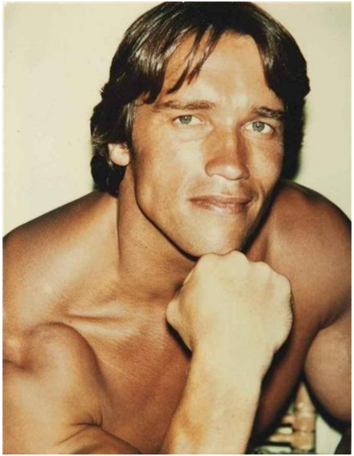
x_1



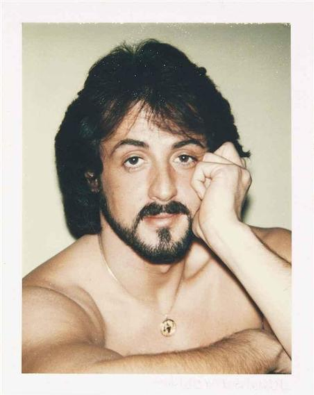
x_2

Query by image (2)

x



x_1



x_2

Result: The distance in the latent space seems to be meaningful!

Word embeddings

We can train (monolingual) *word embeddings*,
i.e. representations trained to predict well words that appear in its context (ref here)

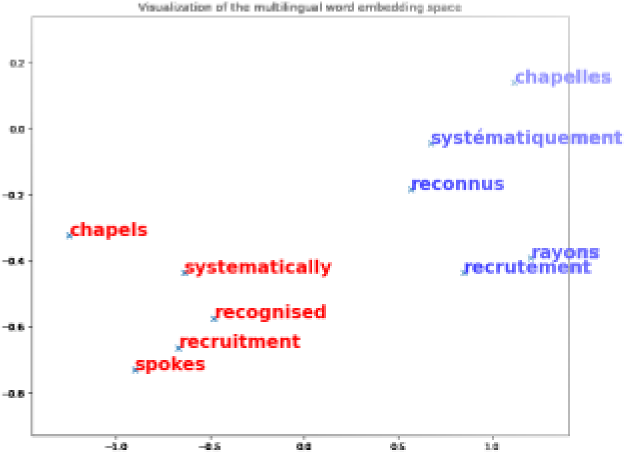


Image generated using pretrained embeddings available here. *En avant guiGuan team*, 2021.

Word embeddings

We can train (monolingual) *word embeddings*,
i.e. representations trained to predict well words that appear in its context (ref here)

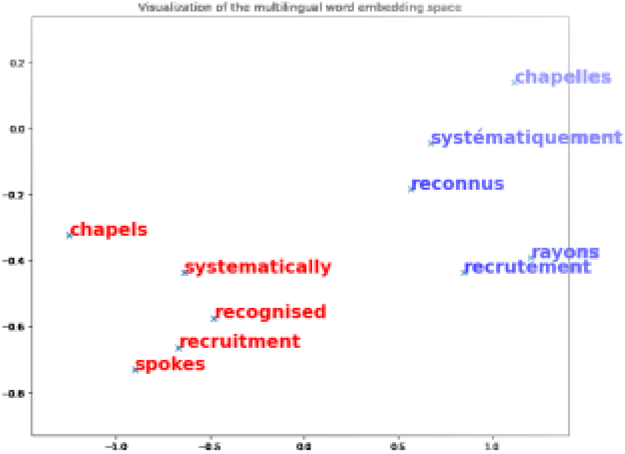
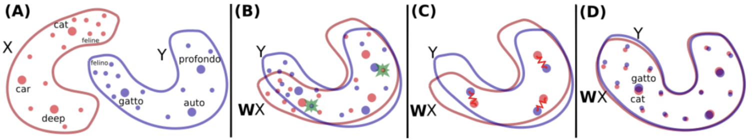


Image generated using pretrained embeddings available here. *En avant guiGuan team*, 2021.

- The structure between word embeddings is preserved across languages

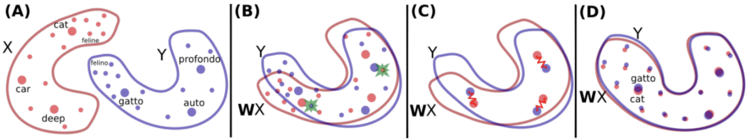
Word to word translation using word embeddings



Exploit linear transformations and rotations to translate a word.

$$Y = \mathbf{W}X$$

Word to word translation using word embeddings



Exploit linear transformations and rotations to translate a word.

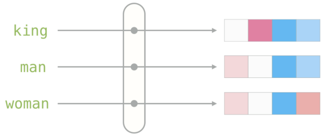
$$Y = \mathbf{W}X$$

Learn **W**

- with a parallel corpus (e.g. supervised dataset FR-EN)
- without a parallel corpus using a GAN

Text manipulation with word embeddings

The embedding space is geometric!

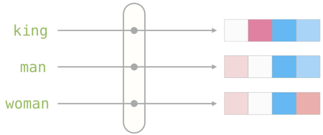


The embeddings space is geometric:

$$v(king) - v(man) + v(woman) = \quad ?$$

Text manipulation with word embeddings

The embedding space is geometric!



The embeddings space is geometric:

$$v(king) - v(man) + v(woman) = \quad \text{queen}$$

Visualising the data manifold

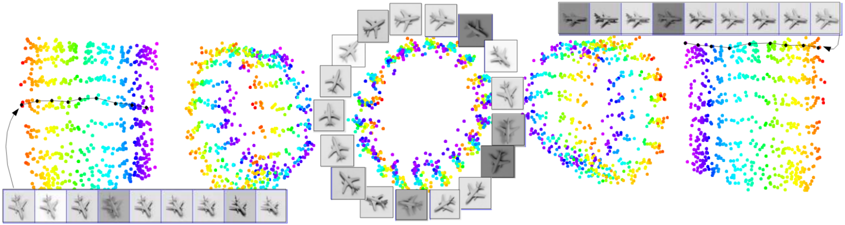
Dimensionality reduction by learning an invariant mapping.
Hadsell, R., Chopra, S., LeCun, Y. CVPR (2006)
Learns a mapping g that maps inputs with **few controlled variations** to a **low dimensional space**

- all pictures are pictures of planes with different poses
- 9 different elevations and 18 different azimuth (orientation).
- input pictures are projected into a low 3-dimensional space

Visualising the data manifold

Dimensionality reduction by learning an invariant mapping.
Hadsell, R., Chopra, S., LeCun, Y. CVPR (2006)
Learns a mapping g that maps inputs with **few controlled variations** to a **low dimensional space**

- all pictures are pictures of planes with different poses
- 9 different elevations and 18 different azimuth (orientation).
- input pictures are projected into a low 3-dimensional space

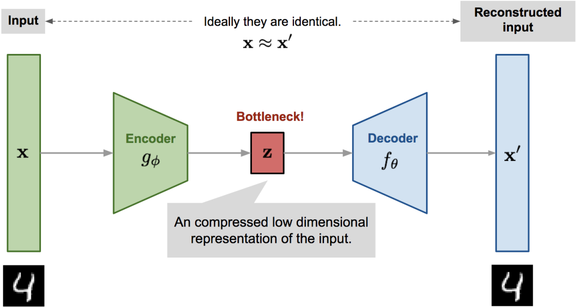


Result:

- Most input data lies on a well defined subspace of the output space
- There is a clear relation between spacial coordinates and features (elevation, azimuth)

Outline

Data generation with autoencoders



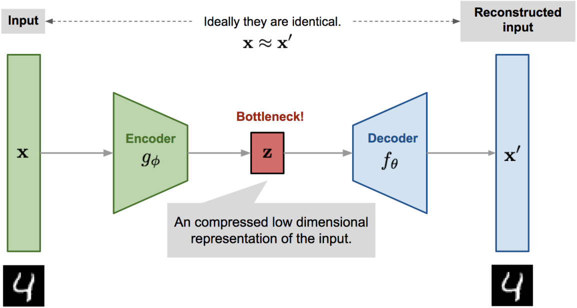
AE, image from https://lilianweng.github.io

First Idea:

- train an autoencoder with the reconstruction error loss
- sample a vector z in the latent space $\mathcal{Z}$
- decode z into an image $x = d(z)$

Problem?

Data generation with autoencoders



AE, image from https://lilianweng.github.io

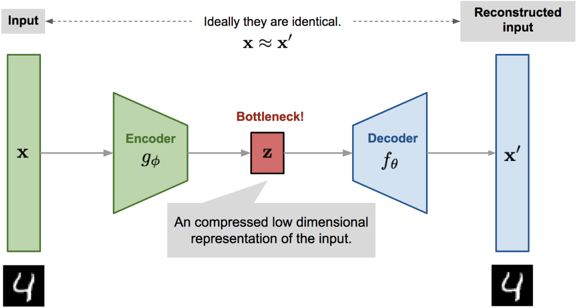
First Idea:

- train an autoencoder with the reconstruction error loss
- sample a vector z in the latent space $\mathcal{Z}$
- decode z into an image $x = d(z)$

Problem

How can we sample z so that $d(z)$ is mapped to a point in the *data manifold*?

Data generation with autoencoders



AE, image from https://lilianweng.github.io

First Idea:

- train an autoencoder with the reconstruction error loss
- sample a vector z in the latent space $\mathcal{Z}$
- decode z into an image $x = d(z)$

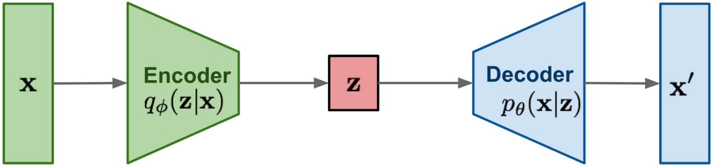
Problem

How can we sample z so that $d(z)$ is mapped to a point in the *data manifold*?

Solution

- regularize the latent space so that the encoded training data in normally distributed in $\mathcal{Z}$
- sample z from a normal distribution

Data generation with VAE

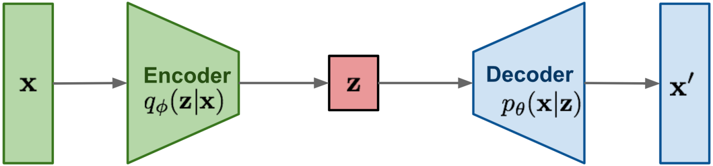


VAE, image from https://lilianweng.github.io

Main differences with AE

- Use a probabilistic encoder
- Regularize the latent space during training

Data generation with VAE



VAE, image from https://lilianweng.github.io

Main differences with AE

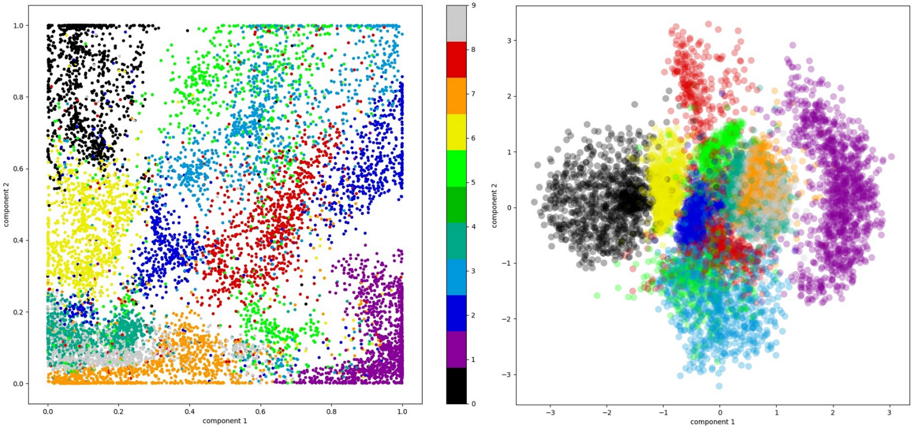
- Use a probabilistic encoder
- Regularize the latent space during training

Training procedure

1. take a training data point x, obtain μ_x and σ_x from the encoder
2. sample $z \sim \mathcal{N}(\mu_x, \sigma_x)$
3. decode z into $\tilde{x}$
4. compute the loss and update parameters

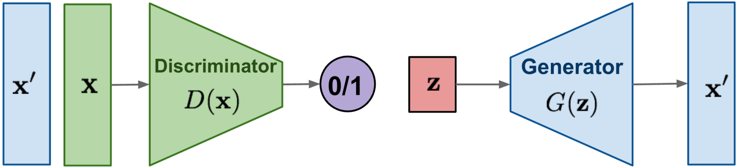
$$loss(x, \tilde{x}) = \|x - \tilde{x}\| + KL(\mathcal{N}(\mu_x, \sigma_x), \mathcal{N}(0, I))$$

AE vs. VAE



1.0
0.8
0.6
0.4
0.2
0.0
component 2
0.0
0.2
0.4
0.6
0.8
1.0
component 1
9
8
7
6
5
4
3
2
1
0
3
2
1
0
−1
−2
−3
component 2
−3
−2
−1
0
1
2
3
component 1

Generative Adversarial Networks



GAN, image from https://lilianweng.github.io

Main difference with VAE

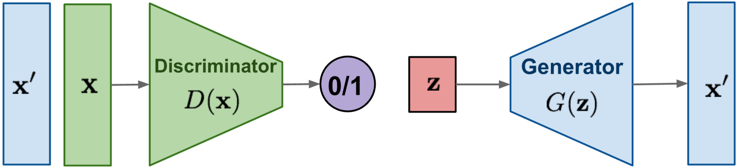
- Minimize an estimate of the divergence $\mathcal{D}(P\|\hat{P}_G)$, instead of the reconstruction error
  - P is the true data distribution
  - $\hat{P}_G$ is the model distribution induced by a generator function $G : \mathcal{Z} \to \mathcal{X}$ (i.e. a decoder)
  - Since P is unavailable, we use a discriminator $D : \mathcal{X} \to [0, 1]$ to estimate $\mathcal{D}(P\|\hat{P}_G)$
  - D is trained to distinguish samples from P and samples from $\hat{P}_G$

Training procedure

G and D are trained simultaneously to solve the following min-max problem:

$$\min_G \max_D \mathbb{E}_{x_r\sim P}[\log D(x)] + \mathbb{E}_{x_g\sim \hat{P}_G}[\log 1 - D(x)]$$

Generative Adversarial Networks



GAN, image from https://lilianweng.github.io

References:

- Generative Adversarial Nets
  https://arxiv.org/pdf/1406.2661
- Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks https://arxiv.org/abs/1511.06434

Goal of Assignment 2

1. Train a GAN on MNIST.
2. The structure of the Generator is fixed.
3. Use different one or two possible improvements to improve the data generation.

representation_learning/img/mnist.jpeg

Possible improvements

- f-GANs
  1. f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization
- WGAN
  1. Wasserstein GAN
- Rejection Sampling
  1. Discriminator Rejection Sampling
  2. Metropolis-Hastings Generative Adversarial Networks
- Latent Rejection sampling
  1. Latent reweighting, an almost free improvement for GANs
- Gradient ascent
  1. Discriminator optimal transport
  2. Refining Deep Generative Models via Discriminator Gradient Flow
  3. Your GAN is Secretly an Energy-based Model and You Should use Discriminator Driven Latent Sampling
- Classifier guidance generation
  1. MMGAN: Generative Adversarial Networks for Multi-Modal Distributions
  2. Gaussian Mixture Generative Adversarial Networks for Diverse Datasets, and the Unsupervised Clustering of Images

Requirements Assignment 2

1. Train a vanilla GAN
2. Write a script generate.py that generate 10000 samples in the folder samples (use mine).
3. Based on these 10k samples, you will be evaluated on FID, Precision and Recall.
   Precision/Recall