

Sapienza University of Rome

Master in Artificial Intelligence and Robotics

Machine Learning

A.Y. 2024/2025

Prof. Luca Iocchi

Luca Iocchi

15. Dimensionality reduction

1 / 72

Sapienza University of Rome, Italy - Machine Learning (2024/2025)

WE USE MACHINE
LEARNING FOR
A TASK ON DATA THAT
WILL BE USE FOR
ANOTHER TASK

15. Dimensionality reduction

Luca Iocchi

“TRANSFORM DATA IN A LOWER DIMENSION”

with contributions from Valsamis Ntouskos

Luca Iocchi

15. Dimensionality reduction

2 / 72

Overview

- Continuous latent variables
- Principal Component Analysis (PCA)
- Probabilistic PCA
- Non-linear latent variable models
- Autoencoders
- Generative Models

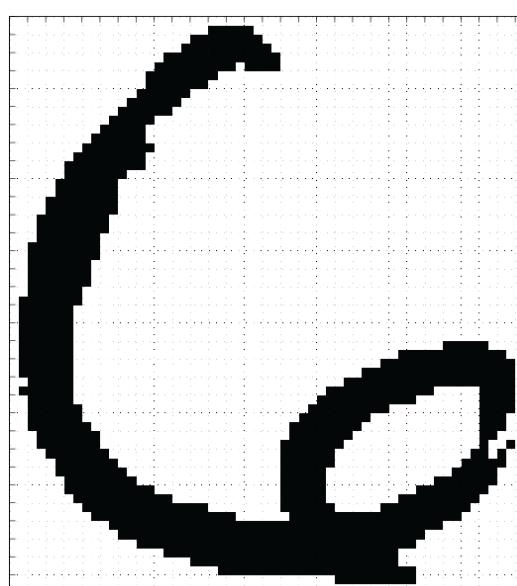
Reference

C. Bishop. Pattern Recognition and Machine Learning. Chapter 12.

Latent Variables

Example

USPS dataset: 64 rows by 57 columns



Latent Variables

Data space contains more than just digits



Latent Variables

Data space contains more than just digits



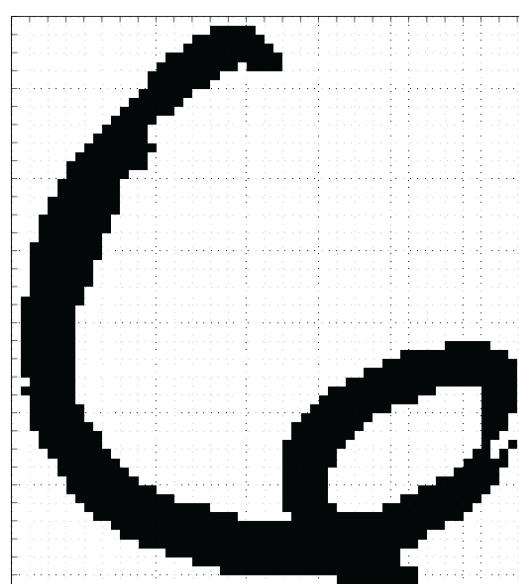
Latent Variables

Data space contains more than just digits



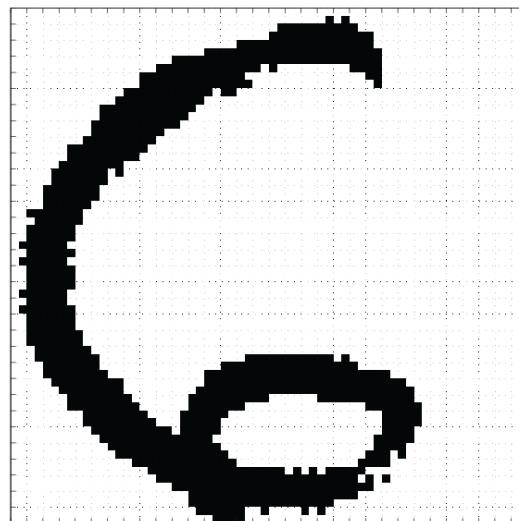
Latent Variables

Prototype rotation (1 dof transformation)



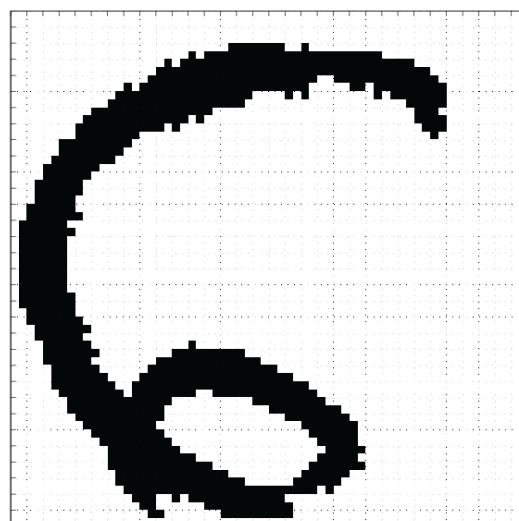
Latent Variables

Prototype rotation (1 dof transformation)



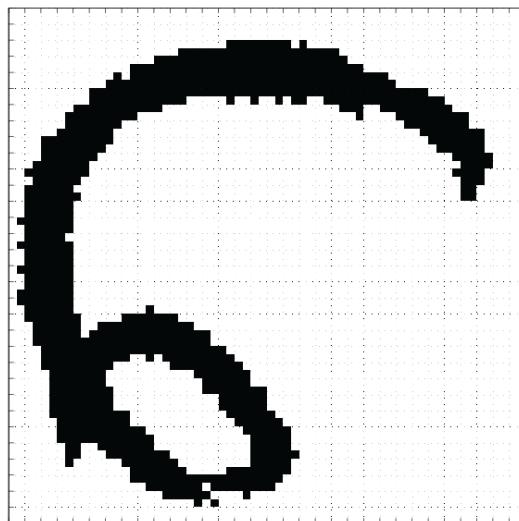
Latent Variables

Prototype rotation (1 dof transformation)



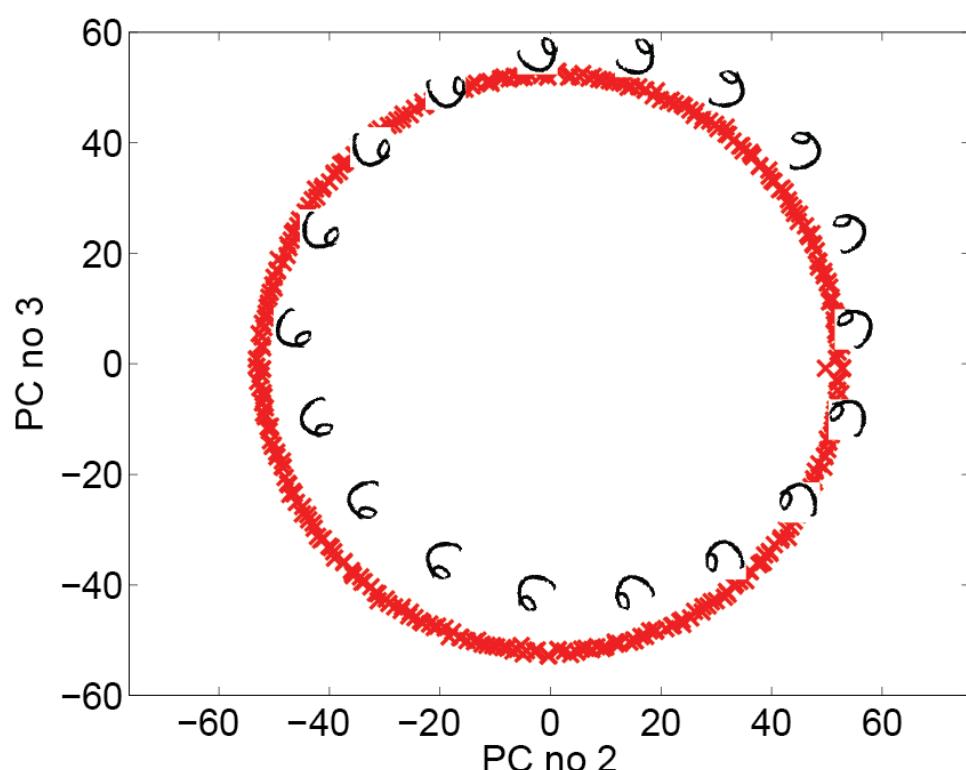
Latent Variables

Prototype rotation (1 dof transformation)



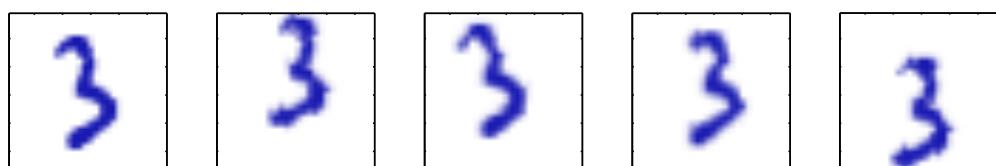
Latent Variables

Manifold



Latent Variables

Another example



3 degrees of freedom transformation (2D translation + rotation)

Latent Variables

For data with ‘structure’*

- We expect fewer distortions than dimensions
- data live on a lower dimensional manifold

Conclusion: deal with high dimensional data by looking for lower dimensional embedding

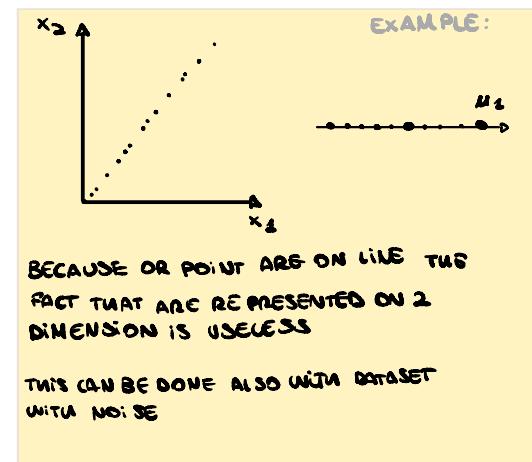
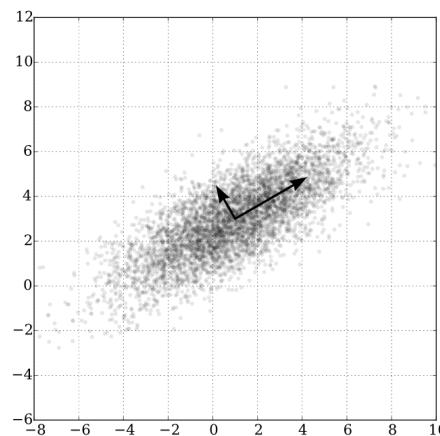
* from Raquel Urtasun’s slides

Principal Component Analysis

MAXIMIZE VARIANCE
OVER AN AXIS

Principal Component Analysis (PCA) is a widely used technique for various tasks as

- dimensionality reduction
- data compression (lossy)
- data visualization
- feature extraction



PCA - Variance Maximization

Given data $\{\mathbf{x}_n\} \in \mathbb{R}^d$

Goal: Maximize data variance after projection to some direction \mathbf{u}_1 (unit vector)

Projected points:

$$\mathbf{x}_n^T \mathbf{u}_1$$

Note: $\mathbf{u}_1^T \mathbf{u}_1 = 1$

PCA - Variance Maximization

Mean value of data points:

$$\bar{\mathbf{x}} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n$$

Data-centered matrix \mathbf{X} ($N \times d$):

COMPUTE MEAN OF DATA AND WE SUBTRACT
 EACH COMPONENT OF DATASET

$$\mathbf{X} = \begin{bmatrix} (\mathbf{x}_1 - \bar{\mathbf{x}})^T \\ \vdots \\ (\mathbf{x}_n - \bar{\mathbf{x}})^T \\ \vdots \\ (\mathbf{x}_N - \bar{\mathbf{x}})^T \end{bmatrix}$$

PCA - Variance Maximization

Mean of projected points:

$$\bar{\mathbf{x}}^T \mathbf{u}_1$$

Variance of projected points:

$$\frac{1}{N} \sum_{n=1}^N [\mathbf{u}_1^T \mathbf{x}_n - \mathbf{u}_1^T \bar{\mathbf{x}}]^2 = \mathbf{u}_1^T S \mathbf{u}_1$$

with S ($d \times d$) covariance matrix of the dataset

$$S = \frac{1}{N} \sum_{n=1}^N (\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^T = \frac{1}{N} \mathbf{X}^T \mathbf{X}$$

PCA - Variance Maximization

Problem definition

Maximize the projected variance

$$\max_{\mathbf{u}_1} \mathbf{u}_1^T S \mathbf{u}_1$$

subject to constraint $\mathbf{u}_1^T \mathbf{u}_1 = 1$

Equivalent to unconstrained maximization with a Lagrange multiplier

$$\max_{\mathbf{u}_1} \mathbf{u}_1^T S \mathbf{u}_1 + \lambda_1 (1 - \mathbf{u}_1^T \mathbf{u}_1)$$

PCA - Variance Maximization

Solution

Setting derivative w.r.t. \mathbf{u}_1 to zero we have

$$S \mathbf{u}_1 = \lambda_1 \mathbf{u}_1$$

\mathbf{u}_1 must be an eigenvector of S

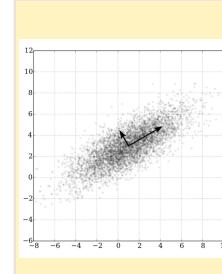
Left-multiplying by \mathbf{u}_1^T and using $\mathbf{u}_1^T \mathbf{u}_1 = 1$, we have

$$\mathbf{u}_1^T S \mathbf{u}_1 = \lambda_1$$

which is the variance after the projection.

PCA - Variance Maximization

Solution



1. COMPUTE COVARIANCE MATRIX
2. COMPUTE EIGENVECTOR SO Z HAVE THE DIRECTION

$$\mathbf{u}_1^T S \mathbf{u}_1 = \lambda_1$$

Variance is maximal when \mathbf{u}_1 is the eigenvector corresponding to the largest eigenvalue λ_1 .

This is called the first **principal component**.

PCA - Variance Maximization

Repeat to find other directions which

- maximize variance of projected data
- are orthogonal to the previous directions

Summary:

To perform PCA in a m -dimensional projection space, with $m < d$

- compute $\bar{\mathbf{x}}$: mean of the data
- compute S : covariance matrix of the dataset
- find m eigenvectors of S corresponding to the m largest eigenvalues

PCA - Error minimization

Consider a complete orthonormal D -dimensional basis such that

$$\mathbf{u}_i^T \mathbf{u}_j = \delta_{ij}$$

with $\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$

Each data point can be written as

$$\mathbf{x}_n = \sum_{i=1}^d \alpha_{ni} \mathbf{u}_i$$

Using the orthonormality property we have $\alpha_{ni} = \mathbf{x}_n^T \mathbf{u}_i$, hence

$$\mathbf{x}_n = \sum_{i=1}^d (\mathbf{x}_n^T \mathbf{u}_i) \mathbf{u}_i$$

PCA - Error minimization

Goal: Approximate \mathbf{x}_n using a lower-dimensional representation.

We can write

$$\tilde{\mathbf{x}}_n = \sum_{i=1}^m z_{ni} \mathbf{u}_i + \sum_{i=m+1}^d b_i \mathbf{u}_i$$

HIGHEST EIGEN VALUE LOWEST EIGEN VALUE

WE ARE LOSING SOME INFORMATION
↓
SO WE WANT COMPUTE LOSS AND ERROR-FUNCTION

Note: b_i terms do not depend on sample \mathbf{x}_n .

Evaluate approximation error (MSE) as

$$J = \frac{1}{N} \sum_{n=1}^N \|\mathbf{x}_n - \tilde{\mathbf{x}}_n\|^2$$

Minimizing w.r.t. z_{nj} and b_i , we get

$$z_{ni} = \mathbf{x}_n^T \mathbf{u}_i, \quad i = 1, \dots, m$$

$$b_i = \bar{\mathbf{x}}^T \mathbf{u}_i, \quad i = M + 1, \dots, d$$

PCA - Error minimization

Using these expression we get

$$\mathbf{x}_n - \tilde{\mathbf{x}}_n = \sum_{i=m+1}^d [(\mathbf{x}_n - \bar{\mathbf{x}})^T \mathbf{u}_i] \mathbf{u}_i$$

The overall approximation error becomes

$$J = \frac{1}{N} \sum_{n=1}^N \sum_{i=m+1}^d (\mathbf{x}_n^T \mathbf{u}_i - \bar{\mathbf{x}}^T \mathbf{u}_i)^2 = \sum_{i=m+1}^d \mathbf{u}_i^T S \mathbf{u}_i$$

PCA - Error minimization

Minimize the approximation error subject to constraint $\mathbf{u}_i^T \mathbf{u}_i = 1$:

$$\tilde{J} = \sum_{i=m+1}^d \mathbf{u}_i^T S \mathbf{u}_i + \lambda_i (1 - \mathbf{u}_i^T \mathbf{u}_i)$$

Setting derivative of a \mathbf{u}_i to zero we have:

$$S \mathbf{u}_i = \lambda_i \mathbf{u}_i$$

Hence \mathbf{u}_i is an eigenvector of S with eigenvalue λ_i .

PCA - Error minimization

The approximation error is then given by

$$J = \sum_{i=m+1}^d \lambda_i$$

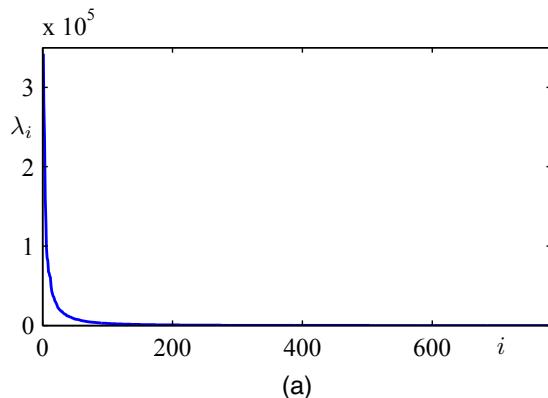
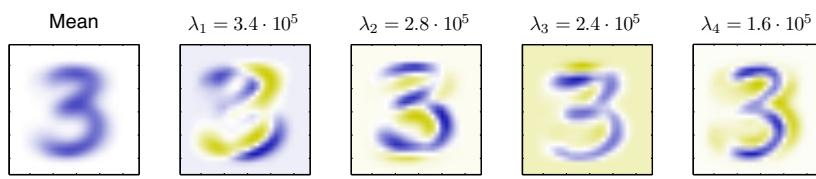
This is minimized by selecting \mathbf{u}_i as the eigenvectors corresponding to the $d - m$ smallest eigenvalues.

Note: Choosing $d - m$ smallest eigenvalues of S corresponds to finding the m highest eigenvalues of S as in the maximum variance formulation.

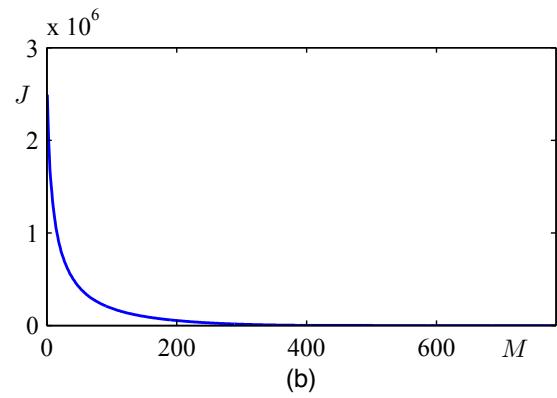
PCA - Algorithms

- ① Full eigenvalue decomposition of S (slow)
- ② Efficient eigenvalue decomposition - only m eigenvectors
- ③ Singular value decomposition of centered data matrix \mathbf{X}

PCA - Example



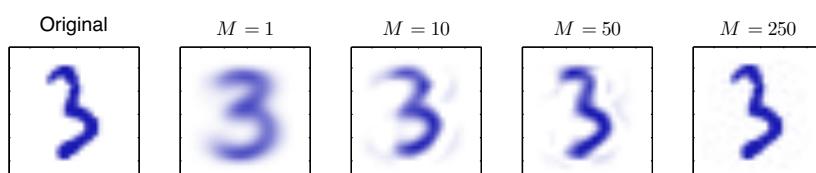
Eigenvalue spectrum



Sum of discarded eigenvalues (error)

PCA - Example

Reconstruction with a limited number of components



PCA for high-dimensional data

What if number of points is smaller than the dimensionality, i.e. $N < d$?
 At least $d-N+1$ eigenvalues of S are zero.

Example: small set of high-resolution images.

In this case finding eigenvalues of S ($d \times d$ matrix) is inefficient.

PCA for high-dimensional data

Solution for $N < d$:

Let \mathbf{X} be the $N \times d$ centered data matrix (i.e., n -th row is $(\mathbf{x}_n - \bar{\mathbf{x}})^T$)

and the corresponding covariance matrix:

$$S = \frac{1}{N} \mathbf{X}^T \mathbf{X}$$

The corresponding eigenvector equations is

$$\frac{1}{N} \mathbf{X}^T \mathbf{X} \mathbf{u}_i = \lambda_i \mathbf{u}_i$$

PCA for high-dimensional data

By left-multiplying by \mathbf{X} we obtain

$$\frac{1}{N} \mathbf{X} \mathbf{X}^T (\mathbf{X} \mathbf{u}_i) = \lambda_i (\mathbf{X} \mathbf{u}_i)$$

By defining $\mathbf{v}_i = \mathbf{X} \mathbf{u}_i$ we have

$$\frac{1}{N} \mathbf{X} \mathbf{X}^T \mathbf{v}_i = \lambda_i \mathbf{v}_i$$

$\mathbf{X} \mathbf{X}^T$ has the same $N - 1$ eigenvalues of $\mathbf{X}^T \mathbf{X}$ (the others are 0).

$\mathbf{X} \mathbf{X}^T$ is an $N \times N$ matrix whose eigenvalues can be computed efficiently.

PCA for high-dimensional data

Given the eigenvalues λ_i of $\mathbf{X} \mathbf{X}^T$, to find the eigenvectors we left-multiply by \mathbf{X}^T

$$\left(\frac{1}{N} \mathbf{X}^T \mathbf{X} \right) (\mathbf{X}^T \mathbf{v}_i) = \lambda_i (\mathbf{X}^T \mathbf{v}_i)$$

This makes clear that $(\mathbf{X}^T \mathbf{v}_i)$ is an eigenvector of S with eigenvalue λ_i .

To find \mathbf{u}_i we have to normalize these eigenvectors such that $\mathbf{u}_i^T \mathbf{u}_i = 1$

$$\mathbf{u}_i = \frac{1}{\sqrt{N \lambda_i}} \mathbf{X}^T \mathbf{v}_i$$

PCA for high-dimensional data

Summing up, when $N \ll d$:

- Consider the centered data matrix \mathbf{X}
- Compute (efficiently) the $N - 1$ eigenvalues λ_i and eigenvectors \mathbf{v}_i of $\mathbf{X}\mathbf{X}^T$
- Let $\mathbf{u}_i = \frac{1}{\sqrt{N\lambda_i}}\mathbf{X}^T\mathbf{v}_i$

The so-obtained $N - 1$ vectors \mathbf{u}_i are the eigenvectors of $S = \mathbf{X}^T\mathbf{X}$, with non-null eigenvalue λ_i

Probabilistic PCA

\rightarrow WE HAVE A PROBABILISTIC
REPRESENTATION OF DATA

Linear Latent Variable Model

- Represent data \mathbf{x} with lower dimensional latent variables \mathbf{z}
- Assume linear relationship

$$\xrightarrow{\substack{\text{WE CAN} \\ \text{ONLY} \\ \text{OBSERVE}}} \mathbf{x} = \mathbf{W}\mathbf{z} + \boldsymbol{\mu}$$

- Assume Gaussian distribution of latent variables \mathbf{z}

$$P(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$$

\hookrightarrow GAUSSIAN SAMPLE O MEAN AND
IDENTITY MATRIX AS VARIANCE

- Assume Linear-Gaussian relationship between latent variables and data

$$P(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; \underbrace{\mathbf{W}\mathbf{z} + \boldsymbol{\mu}}_{\substack{\text{WHERE} \\ \text{IS CENTERED}}}, \sigma^2 \mathbf{I})$$

- Model parameters: \mathbf{W} , $\boldsymbol{\mu}$, σ

Probabilistic PCA

Marginal distribution

$$P(\mathbf{x}) = \int P(\mathbf{x}|\mathbf{z})P(\mathbf{z})d\mathbf{z} = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \mathbf{C})$$

with

$$\mathbf{C} = \mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I}$$

Posterior distribution

$$P(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \mathbf{M}^{-1}\mathbf{W}^T(\mathbf{x} - \boldsymbol{\mu}), \sigma^2\mathbf{M})$$

with

$$\mathbf{M} = \mathbf{W}^T\mathbf{W} + \sigma^2\mathbf{I}$$

Maximum likelihood PCA

Maximum likelihood: given data \mathbf{X}

$$\underset{\mathbf{W}, \boldsymbol{\mu}, \sigma}{\operatorname{argmax}} \ln P(\mathbf{X}|\mathbf{W}, \boldsymbol{\mu}, \sigma^2) = \sum_{n=1}^N \ln P(\mathbf{x}_n|\mathbf{W}, \boldsymbol{\mu}, \sigma^2)$$

Setting derivatives to 0, we have a closed form solution

$$\boldsymbol{\mu}_{ML} = \bar{\mathbf{x}} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n$$

$$\mathbf{W}_{ML} = \dots$$

$$\sigma_{ML}^2 = \dots$$

\mathbf{W} depends on the eigenvalues and eigenvectors of S (not trivial proof)

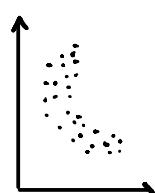
Maximum likelihood PCA

Maximum likelihood solution for the probabilistic PCA model can be obtained also with EM algorithm.

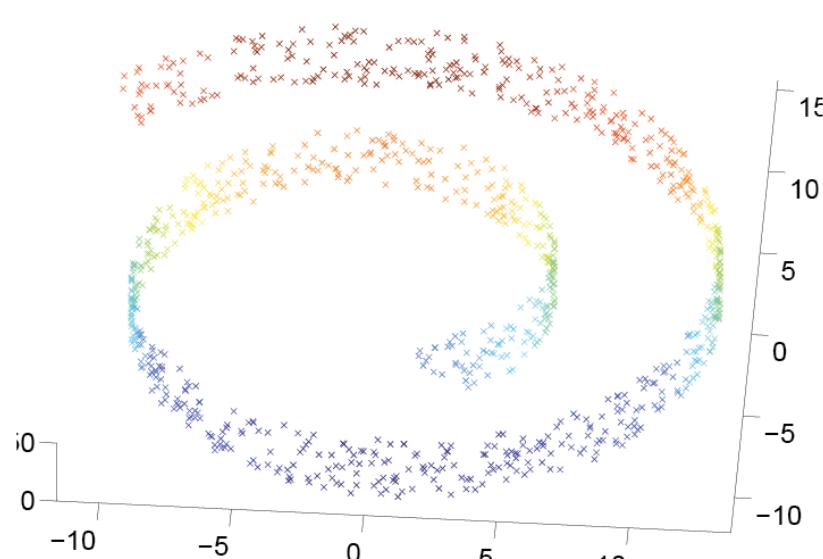
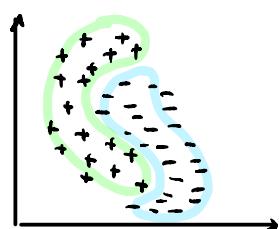
Non-Linear Latent Variable Models

Motivation: Linear representations are not sufficient for complex data

BANANA DISTRIBUTION :



IS A PROBLEM if we HAVE GAUSSIAN DISTRIBUTION



The 'Swiss Roll' dataset. 2D manifold embedded in 3D space.

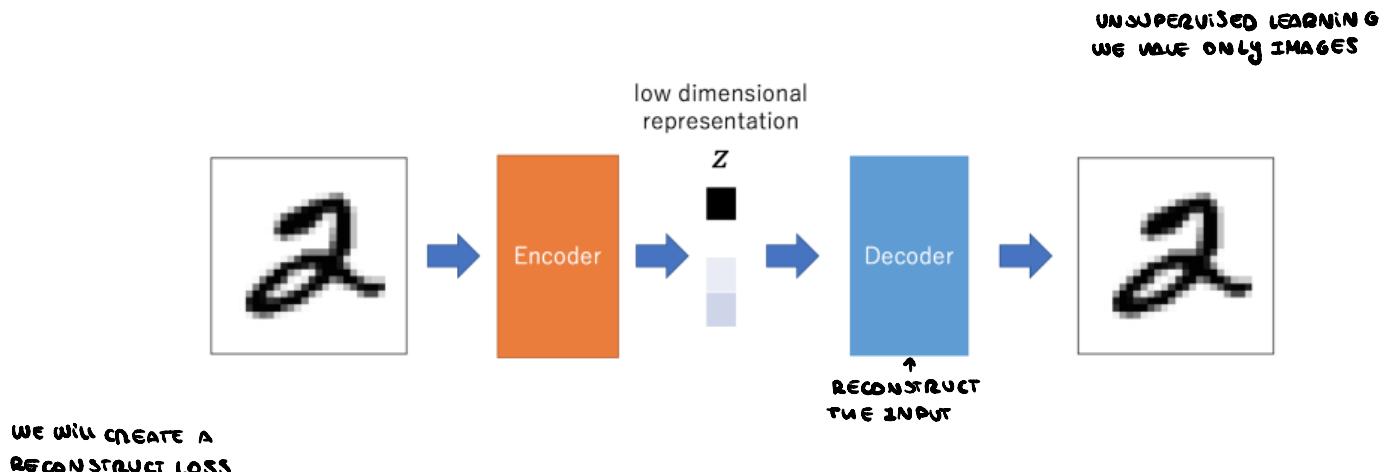
Autoassociative Neural Networks (Autoencoders)

TO DEAL WITH DATA FARAWAY FROM GAUSSIAN DISTRIBUTION

=
ENCODER + DECODERS

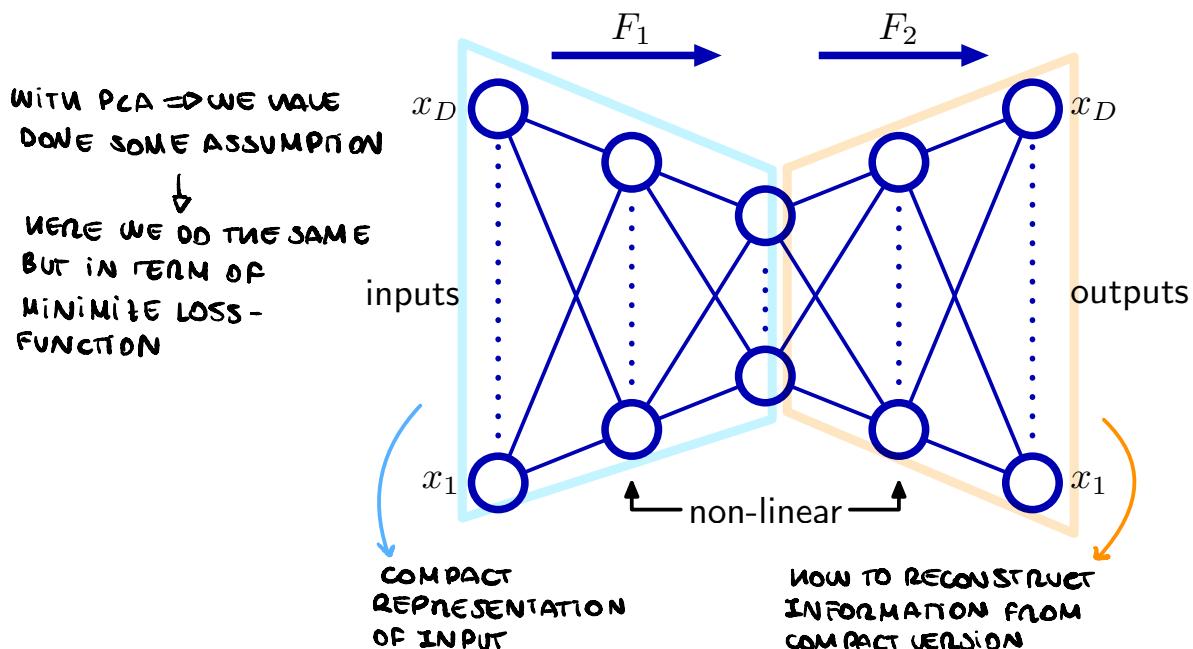
What is an autoencoder?

- a combination of two NN: an encoder and a decoder
- trained with input=output based on reconstruction loss
- provides low-dimensional representation



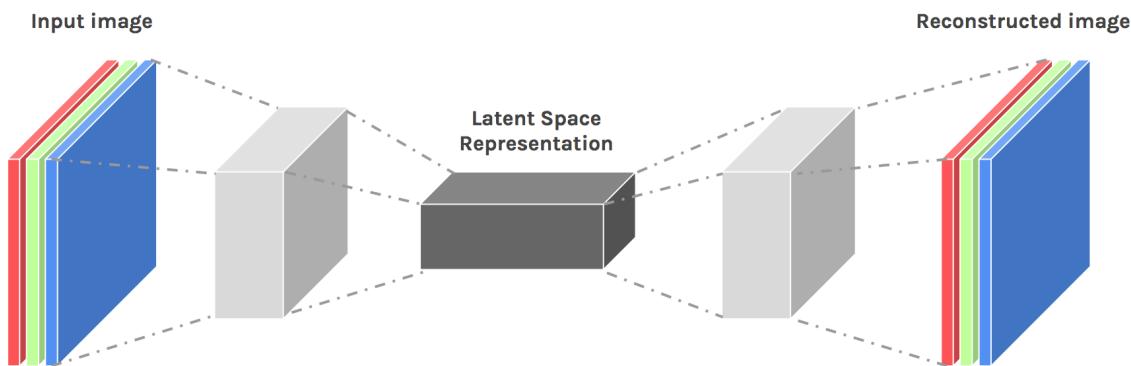
Autoencoders

Neural networks with reduced sized hidden layers (bottleneck) which learn to reconstruct their input by minimizing a loss function.



Convolutional Autoencoders

AE for images based on Convolutional and Convolutional Transposed layers



Autoencoders

Given a dataset $\{x_n\}$, Autoencoders are trained with the same sample x_n both in input and in output.

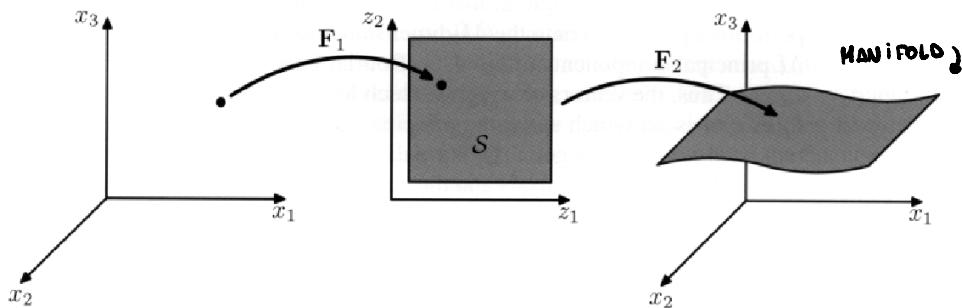
Autoencoders learn how to encode/decode the samples in a dataset in a low-dimensional space.

Autoencoders can be seen as a method for non-linear principal component analysis.

Autoencoders

Autoencoder example:

Input: 3-D, Hidden layer: 2-D, Output: 3-D



The 3D reconstruction from 2D hidden layer generates a 2D manifold (surface) in 3D.

Autoencoder Example

Autoencoder for data set representing values $\{0, \dots, 7\}$ with 1-out-of-8 encoding:

$$D = \{\langle 1, 0, \dots, 0 \rangle, \langle 0, 1, \dots, 0 \rangle, \dots, \langle 0, 0, \dots, 1 \rangle\}$$

Autoencoder layout

- 8 input nodes
- 3 hidden nodes
- 8 output nodes

Observe representation of the 8 values in the latent space.

Autoencoders for anomaly detection

Autoencoders are very useful for anomaly detection (one-class classification)

Problem

Learn $f : X \rightarrow \{n, a\}$ with data set $D = \{(x_n, n)\}$ (samples from only one class)

WE CONSIDER AS
UNSUPERVISED DATASET
BECAUSE LABEL IS NOT
RELEVANT ↴

Train with only samples from normal class (x_n, n)

Given test $x' \notin D$, predict $\hat{f}(x') \in \{n, a\}$ (normal vs. abnormal)

Autoencoders for anomaly detection

Solution

1. Train an autoencoder AE with $\{x_n\}$ (i.e, learn a latent representation for normal samples), compute final train *loss*
2. Determine a threshold δ , e.g. $\delta = \text{mean}(\text{loss}) + \text{std}(\text{loss})$
2. Given $x' \notin D$, reconstruct x' with AE and compute loss'
3. If $\text{loss}' < \delta$ return *normal*, otherwise return *abnormal*

Note: works even better with ensembles of autoencoders.

Autoencoders for anomaly detection

Example

- Take logs of nominal behaviors during development
- Train an anomaly detection system
- Use anomaly detection during deployment



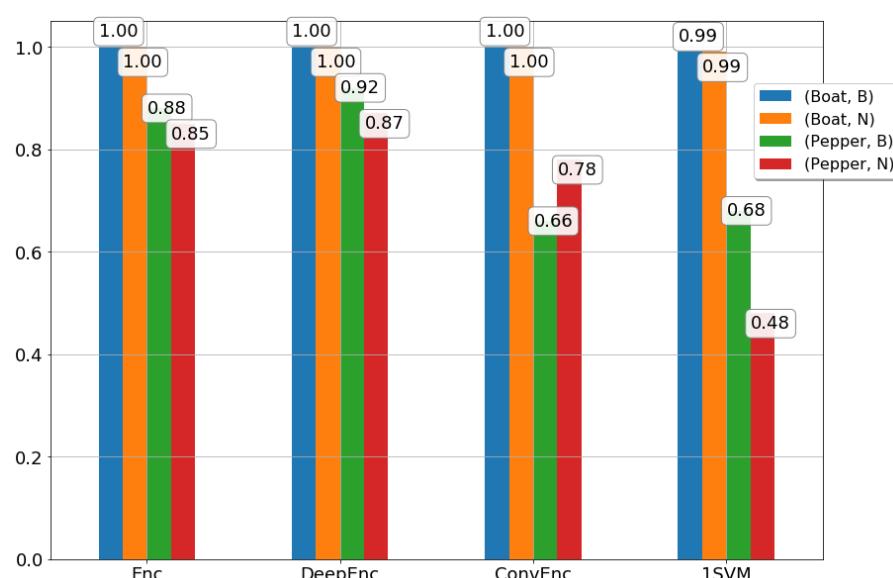
(a)



(b)

Autoencoders for anomaly detection

Results



Generative models

Variational Auto-Encoders (VAEs)

- focus on learning latent space structure

Generative Adversarial Networks (GANs)

- focus on learning a distribution
- no latent space (in general)

VAEs

VARIATIONAL AUTOENCODERS

Goal

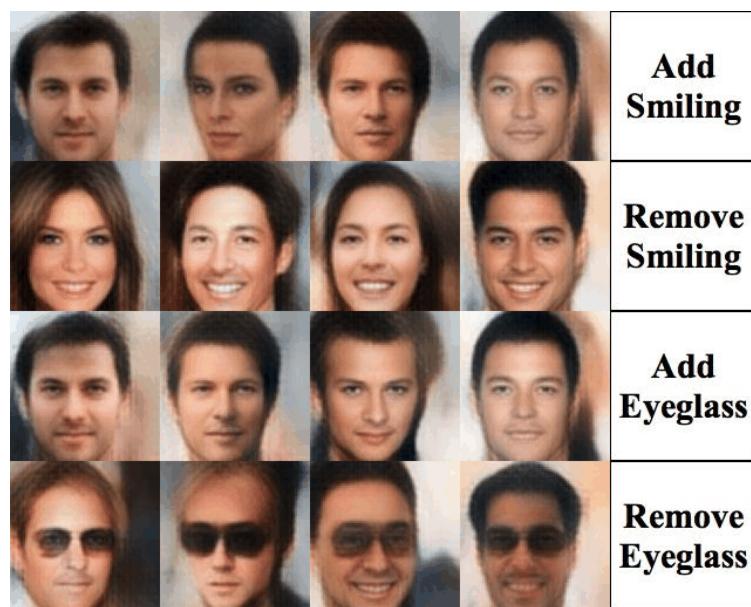
- modify data in specific directions
- identify meaningful directions in latent space

Examples

- ‘distort’ faces (change expression, add glasses)
- produce digits from different hand-writing styles
- distort 3D meshes

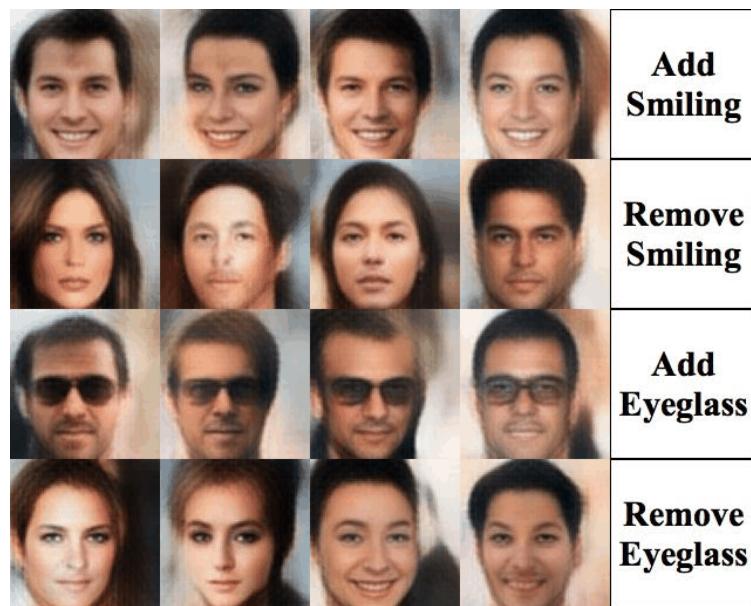
VAEs

Example: faces



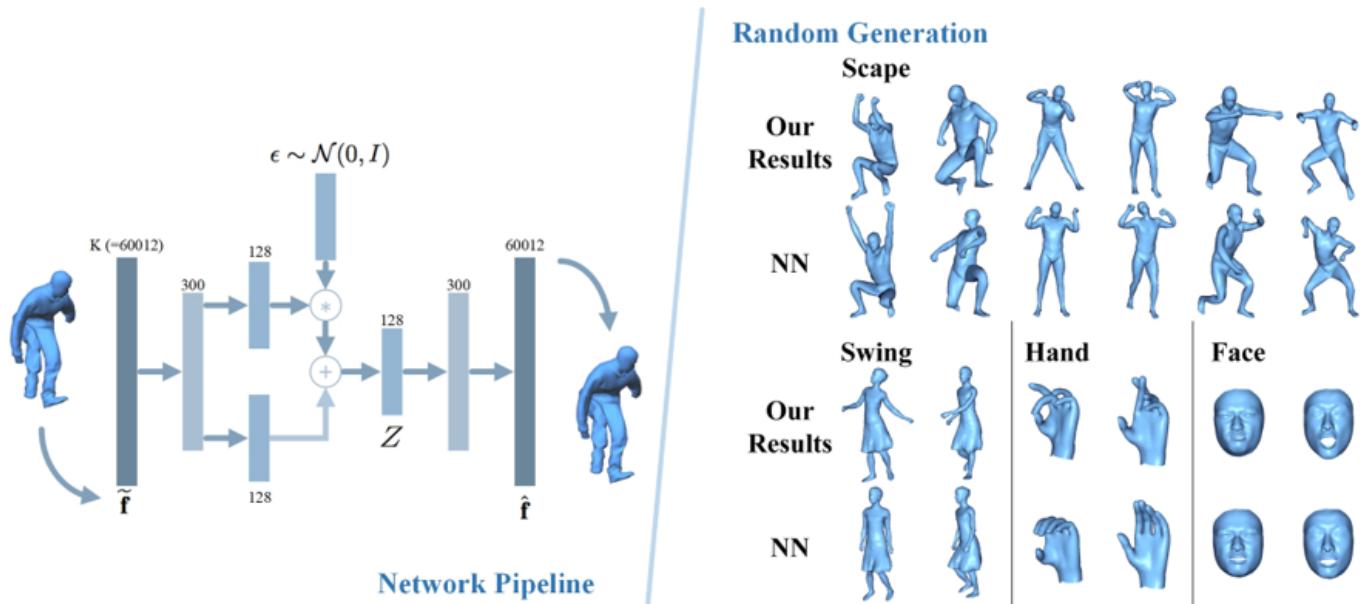
VAEs

Example: faces



VAEs

Example: 3D Mesh deformation

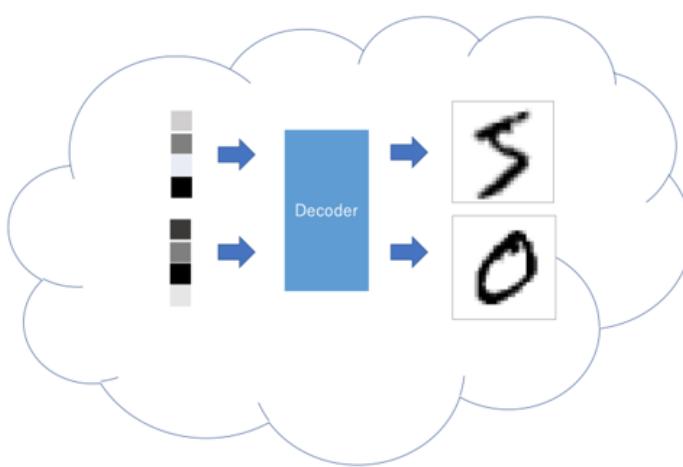


VAEs

Goal

Feed latent vectors and get realistic samples of $P(X)$

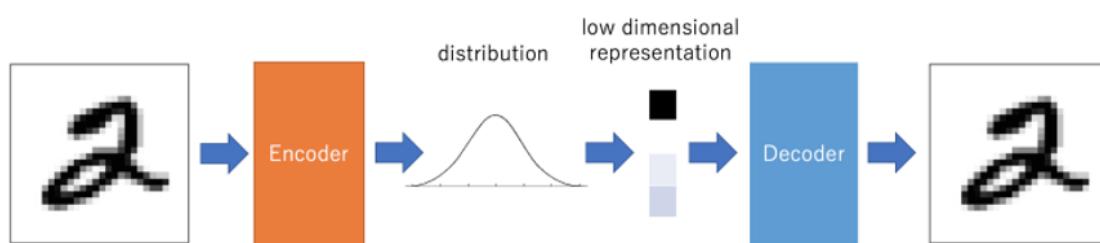
Similar vector values should produce similar generated instances.



VAEs

Main idea

Encoder produces a *distribution* instead of a vector
 Decoder operates on samples from this distribution



VAEs

Problems

- ① How to produce a distribution?
- ② How to prevent degeneration?

Solutions

- ① consider parametric distributions – typically Gaussian
 - produce mean μ and variance Σ
- ② add loss term based on Kullback-Leibler divergence (Evidence Lower Bound)

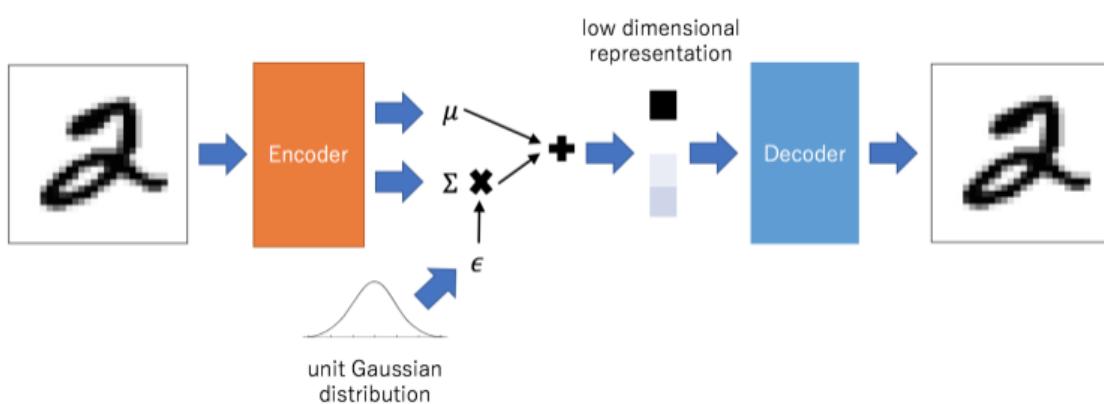
VAEs

One more problem:

- Sampling operation is not differentiable

Solution:

- re-parametrization



GANs

Goal

- Sample from the input data distribution \mathcal{X}

Idea

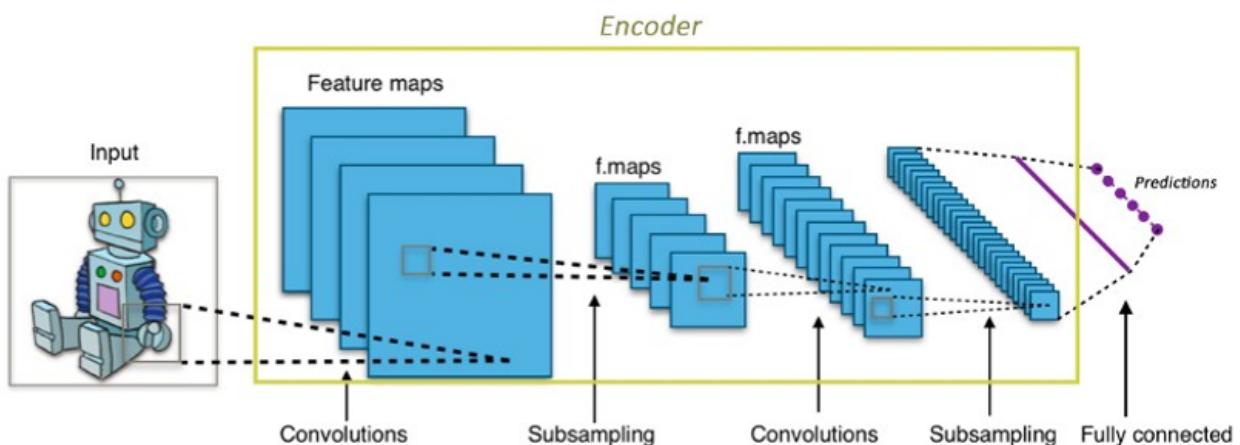
- Invert a (Convolutional) Neural Network
- Use adversarial training

GANs

→ ENCODE IMAGES TO A VECTOR

Encoder (CNN)

- processes an image and produces a vector/*code*

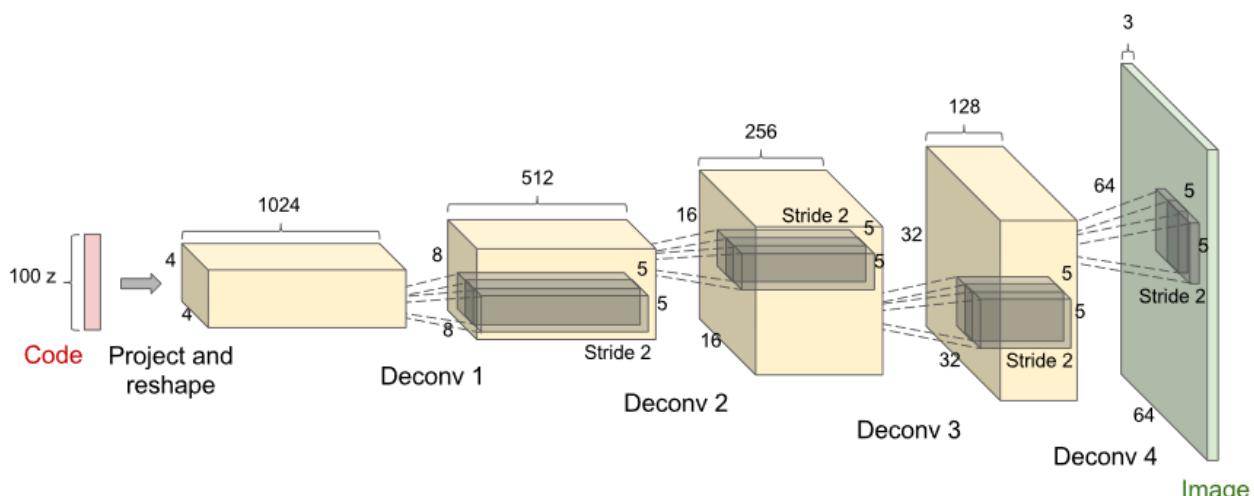


GANs

→ FROM A VECTOR TO IMAGE

Decoder

- receives a code (random vector) and produces an image
- uses “deconvolutional” (transposed convolution) layers



GANs

Problem

How to train the decoder to produce meaningful data?

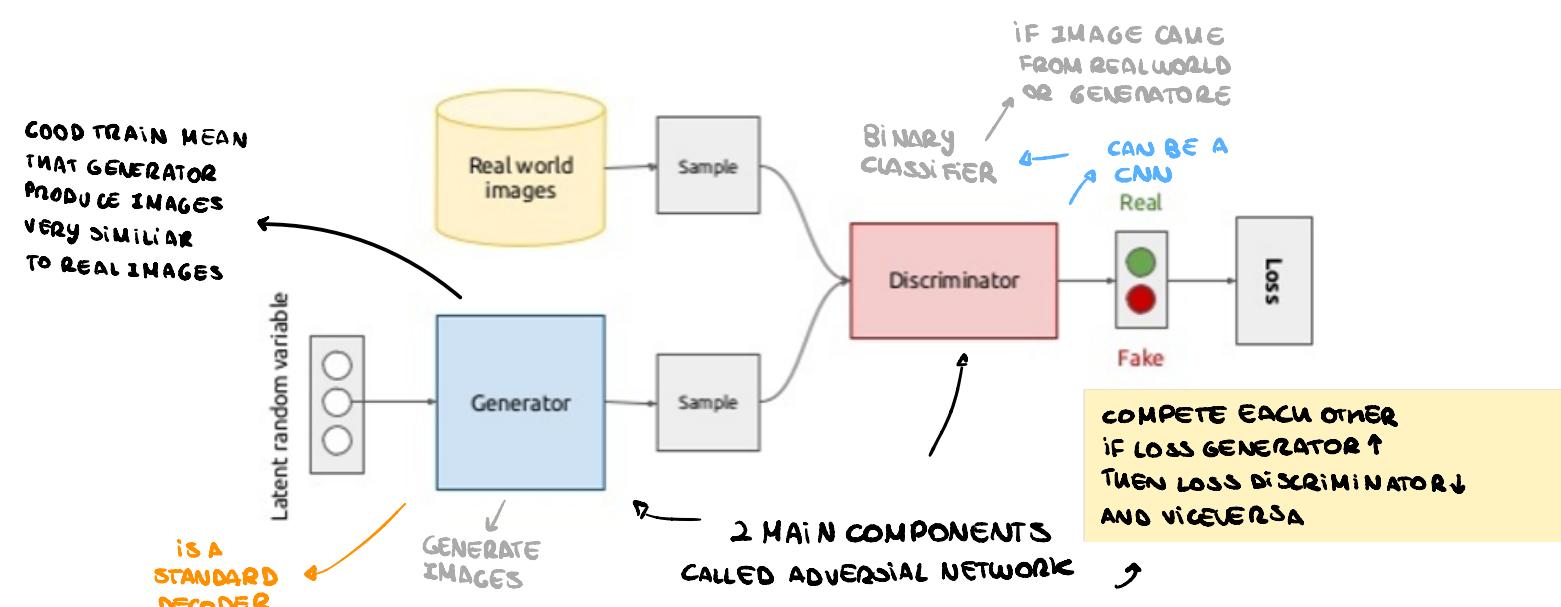
Idea

Use Adversarial Training

GANs

A GAN is a combination of two networks

- ① a generator network (decoder)
- ② a discriminator network (critic)



GANs

Roles

Generator produces samples of the distribution $P(X)$

Discriminator identifies if a sample actually comes from the (unknown) $P(X)$ or not

Training

make the networks compete with each other

- generator tries to fool the discriminator in believing that the sample is 'real'
- discriminator tries to discriminate as good as possible 'real' from 'fake' samples

LOSS OF GENERATOR AND DISCRIMINATOR
DOESN'T DECREASE MONOTONIC

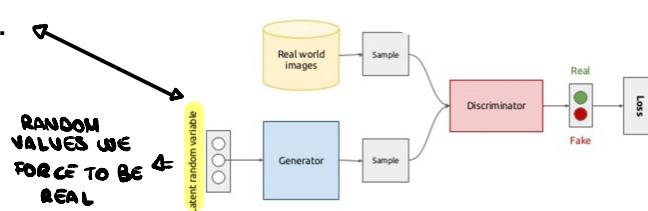
GANs

Training

AT BEGIN THIS IMAGES ARE COMPLETELY RANDOM

Repeat:

- Train the discriminator with a batch of data $\{(x_n, \text{Real})\} \cup \{(x'_m, \text{Fake})\}$, where x_n comes from the data set, while x'_m are images generated from the generator with random values of the latent variable.
- Train the generator by using the entire model (generator + discriminator) with discriminator layers fixed (i.e., not trainable) with a batch of data $\{(r_k, \text{Real})\}$, where r_k are random values of the latent variable.



GANs

Example on CIFAR dataset (32x32 images)

Are these real images or generated?



GANs

Example on CIFAR dataset (32x32 images)

Are these real images or generated?



GANs

Example on CIFAR dataset (32x32 images)
 Results at 300, 900 and 5700 iterations



GANs

Example: Celebrity faces (1024x1024 images)



Adversarial attacks to ML models

Unfortunately, GANs can be used to attack ML models.

Examples:



<https://spectrum.ieee.org/cars-that-think/transportation/sensors/>

slight-street-sign-modifications-can-fool-machine-learning-algorithms

Summary

- Dimensionality reduction aims at identifying the real/intrinsic degrees of freedom of a data set
- Analysis of latent variables helps in understanding the variability of the input data
- Deep associative neural networks provide a general tool for non-linear PCA
- Special architectures can be used to sample the latent space for generating realistic samples of the data distribution