Sapienza University of Rome

Master in Artificial Intelligence and Robotics
Master in Engineering in Computer Science

Machine Learning

A.Y. 2019/2020

Prof. L. Iocchi, F. Patrizi, V. Ntouskos

15. Dimensionality reduction

L. Iocchi, F. Patrizi, V. Ntouskos

Overview

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

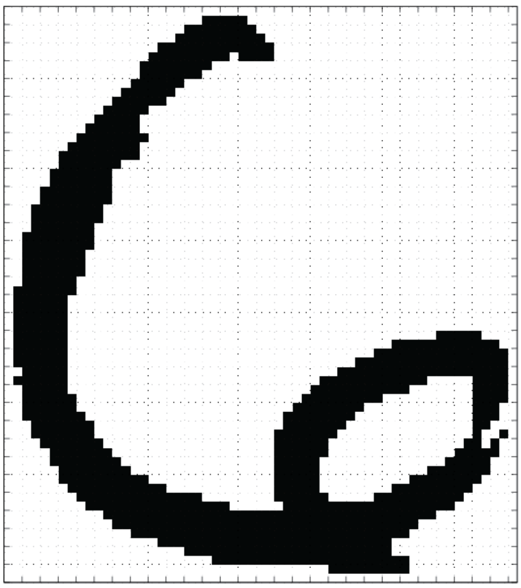
Reference

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

Latent Variables

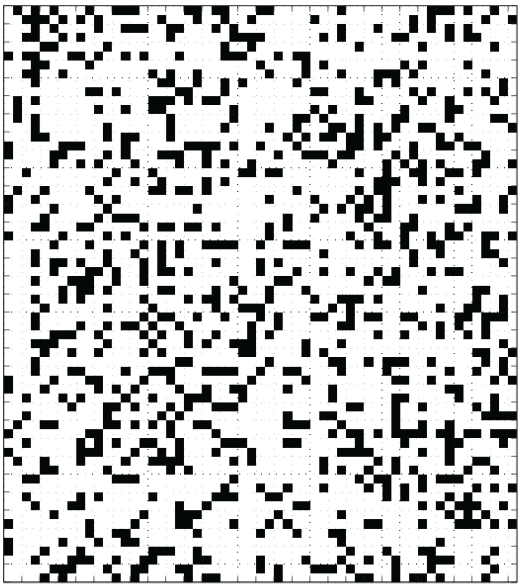
Example

USPS dataset: 64 rows by 57 columns



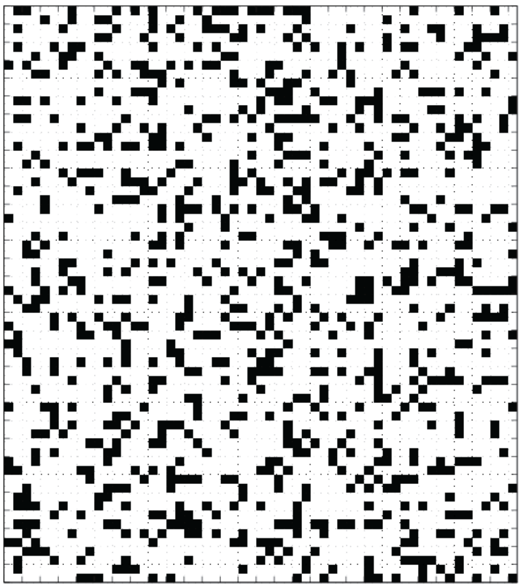
Latent Variables

Data space contains more than just digits



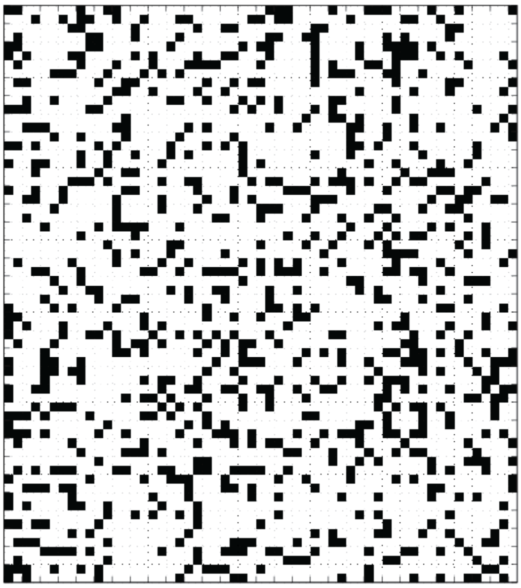
Latent Variables

Data space contains more than just digits



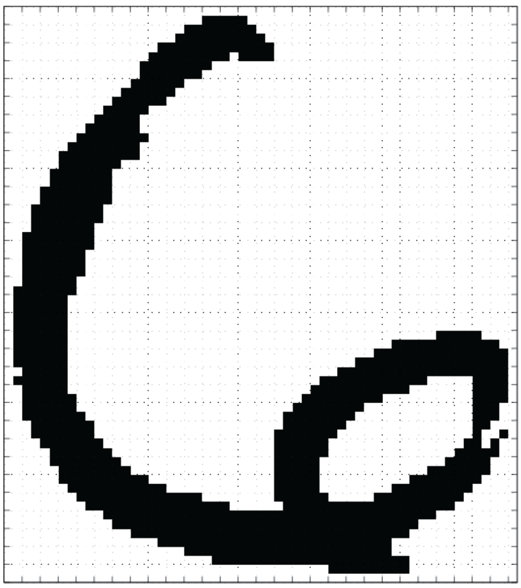
Latent Variables

Data space contains more than just digits



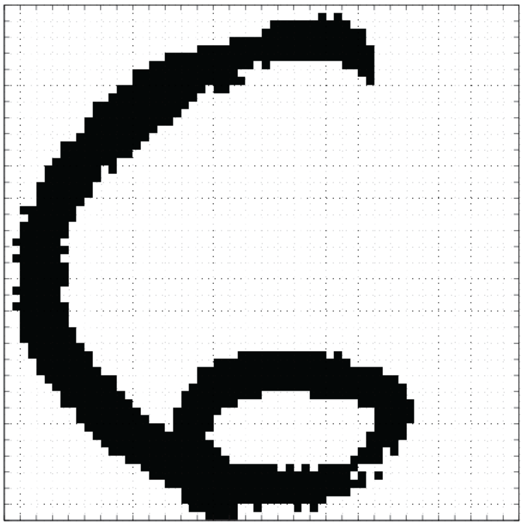
Latent Variables

Prototype rotation (1 dof transformation)



Latent Variables

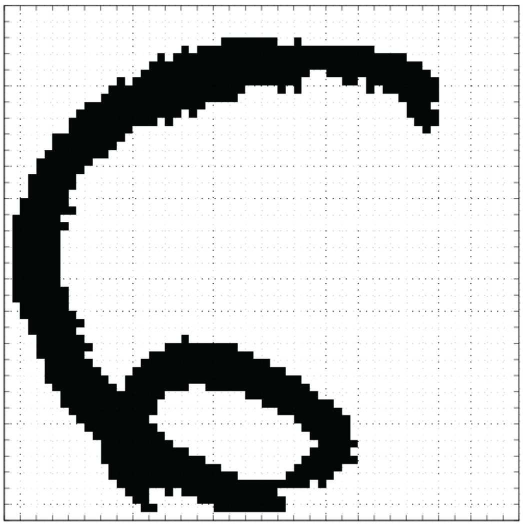
Prototype rotation (1 dof transformation)



Sapienza University of Rome, Italy - Machine Learning (2019/2020)

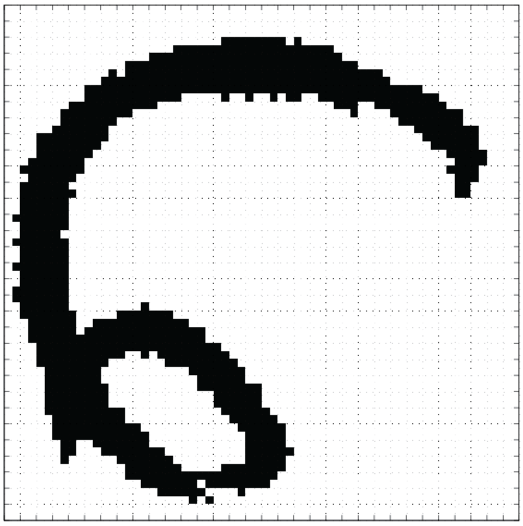
Latent Variables

Prototype rotation (1 dof transformation)



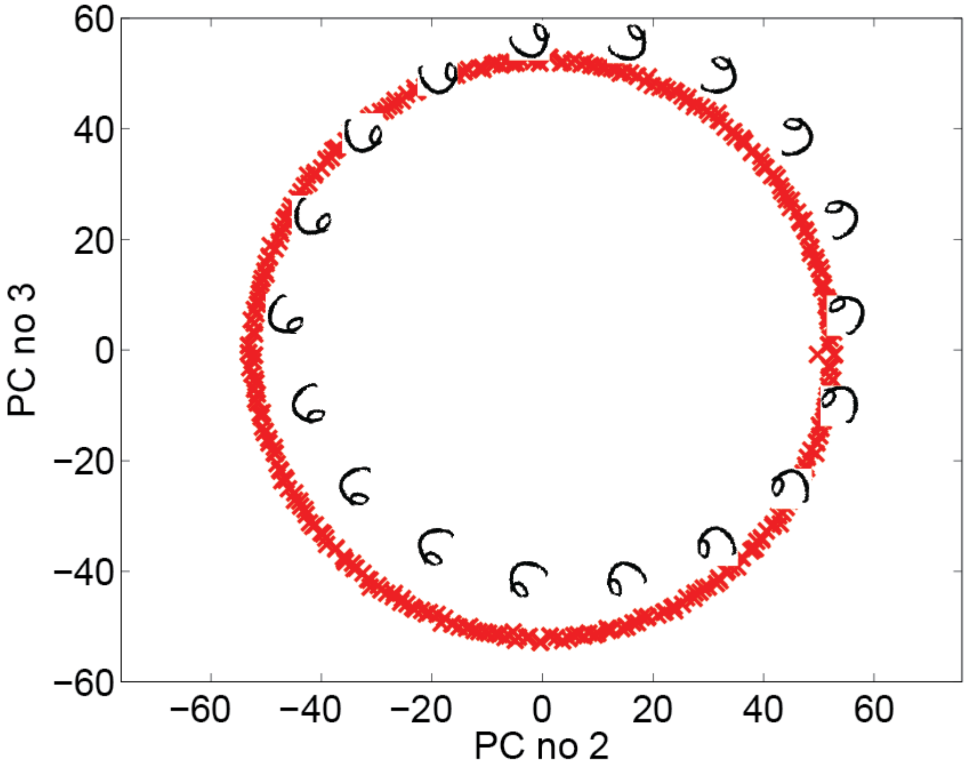
Latent Variables

Prototype rotation (1 dof transformation)



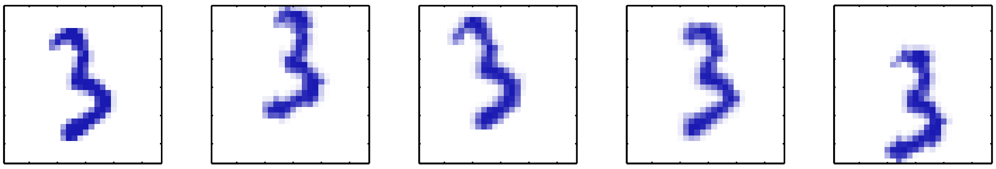
Latent Variables

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

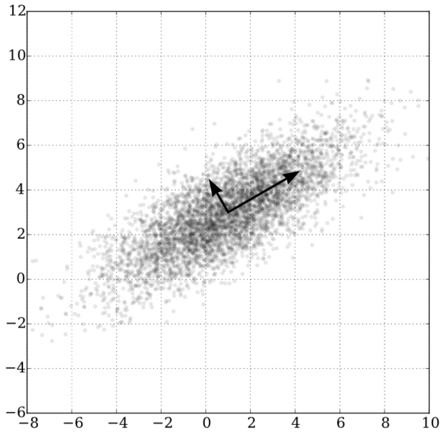
Conclusion: deal with high dimensional data by looking for lower dimensional embedding

*from Raquel Urtasun's slides

Principal Component Analysis

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 $\mathbf{u}_1$

Projected points:

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

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

PCA - Variance Maximization

Mean value of data points:

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

Mean of projected points:

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

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

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

Sapienza University of Rome, Italy - Machine Learning (2019/2020)

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.

Sapienza University of Rome, Italy - Machine Learning (2019/2020)

PCA - Variance Maximization

Solution

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

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 $\alpha_{nj} = \mathbf{x}_n^T \mathbf{u}_j$, 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

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

Evaluate approximation error as

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

Minimizing w.r.t. z_{nj} we get

$$z_{nj} = \mathbf{x}_n^T \mathbf{u}_j, \;\; j = 1, \ldots, M$$

Minimizing w.r.t. b_j we get

$$b_j = \bar{\mathbf{x}}^T \mathbf{u}_j, \;\; j = M+1, \ldots, D$$

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.

Sapienza University of Rome, Italy - Machine Learning (2019/2020)

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.

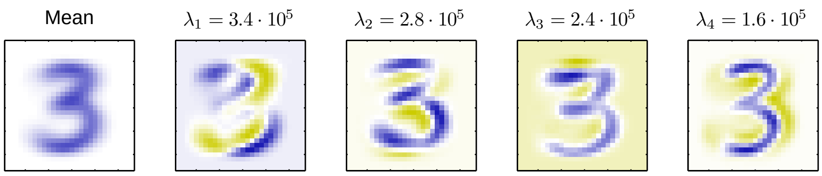
Note: Choosing $D - M$ smallest eigenvalues of S corresponds to finding M highest eigenvalues of S as in the maximum variance formulation.

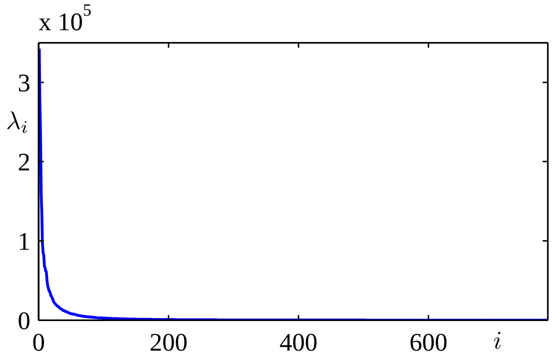
PCA - Algorithms

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

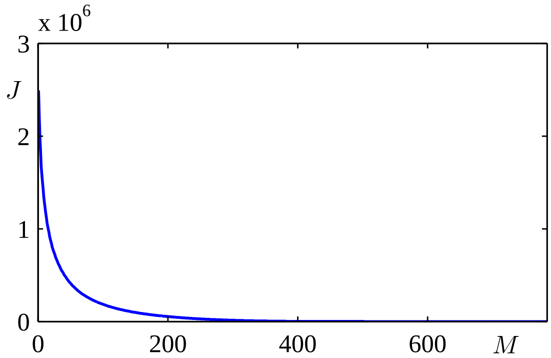
Sapienza University of Rome, Italy - Machine Learning (2019/2020)

PCA - Example





(a)

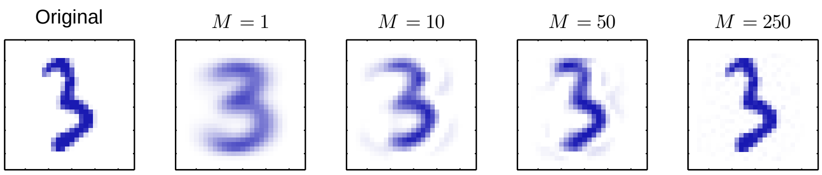


(b)

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 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 $N < D$:

Define $\mathbf{X}$ as the $N \times D$ centered data matrix whose n-th row is $(\mathbf{x}_n - \bar{\mathbf{x}})^T$

The covariance matrix can be written as

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

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

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

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 $\mathbf{u}_i^T\mathbf{u}_i = 1$

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

Sapienza University of Rome, Italy - Machine Learning (2019/2020)

Probabilistic PCA

Linear Latent Variable Model

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

$$\mathbf{x} = \mathbf{W}\mathbf{z} + \boldsymbol{\mu}$$

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

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

- Assume Linear-Gaussian relationship between latent variables and data

$$P(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; \mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \sigma^2\mathbf{I})$$

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

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

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

$$\sigma^2_{ML} = ...$$

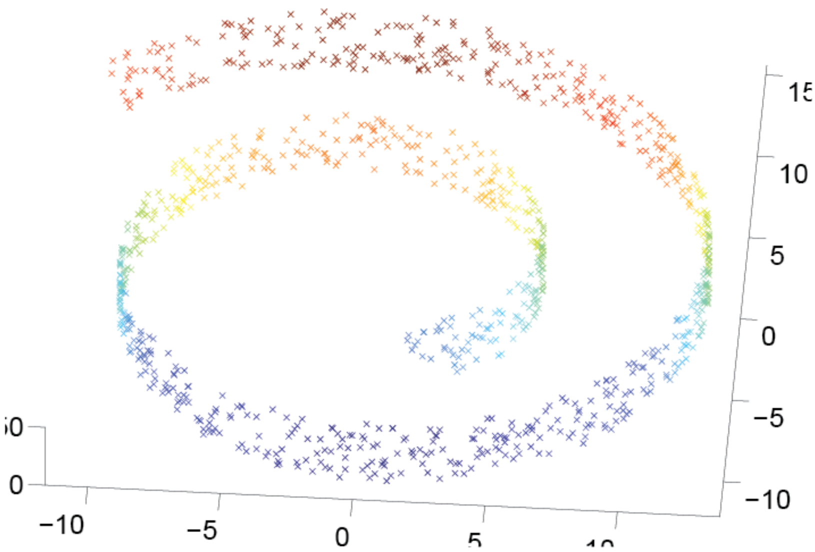
$\mathbf{W}$ depends on the eigenvalues and eigenvectors of S (not trivial proof)

Maximum likelihood PCA

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

Non-Linear Latent Variable Models

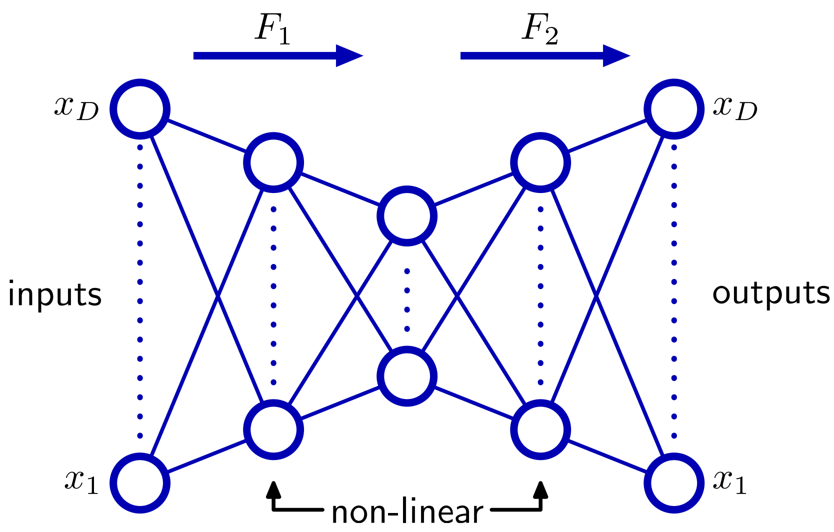
Motivation: Linear representations are not sufficient for complex data



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

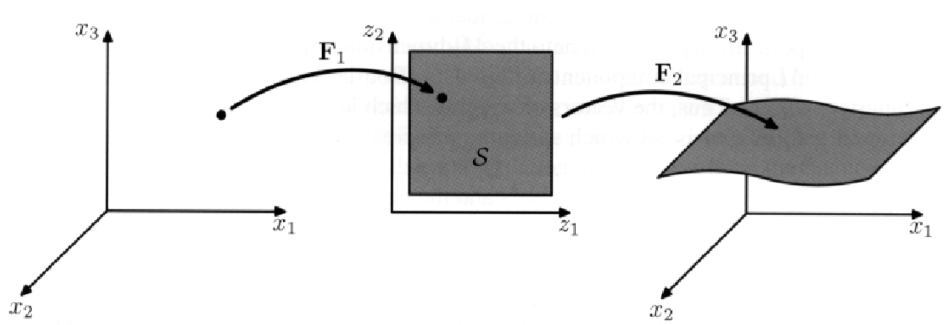
Autoassociative Neural Networks (Autoencoders)

Neural networks with reduced sized hidden layers (bottleneck) which learn to reconstruct their input by minimizing a sum-of-squares error .



Autoencoders

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



Non-linear PCA

Beyond Regression/Classification

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)

Sapienza University of Rome, Italy - Machine Learning (2019/2020)

GANs

Goal

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

Idea

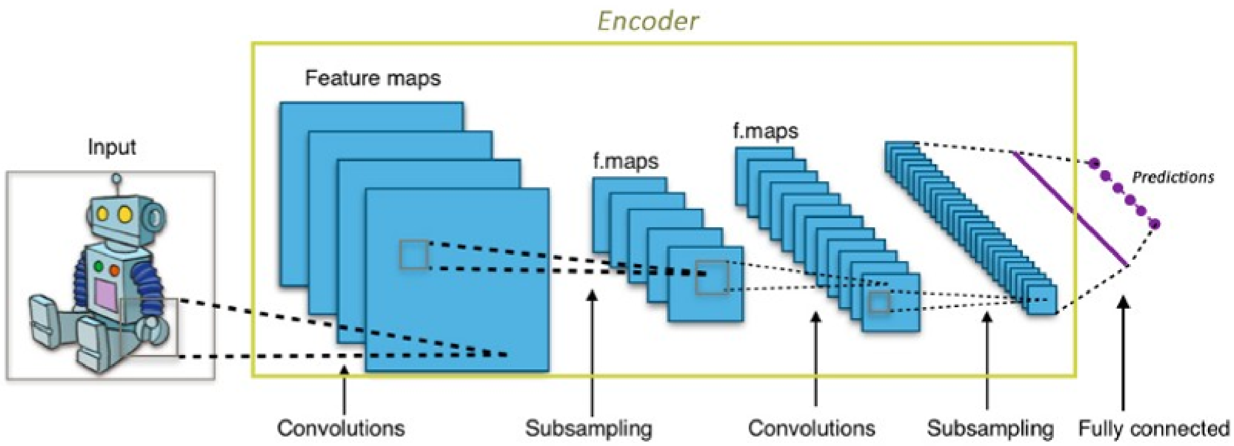
- Invert a (Convolutional) Neural Network
- Use adversarial training

L. Iocchi, F. Patrizi, V. Ntouskos 15. Dimensionality reduction 42 / 1

GANs

Encoder (CNN)

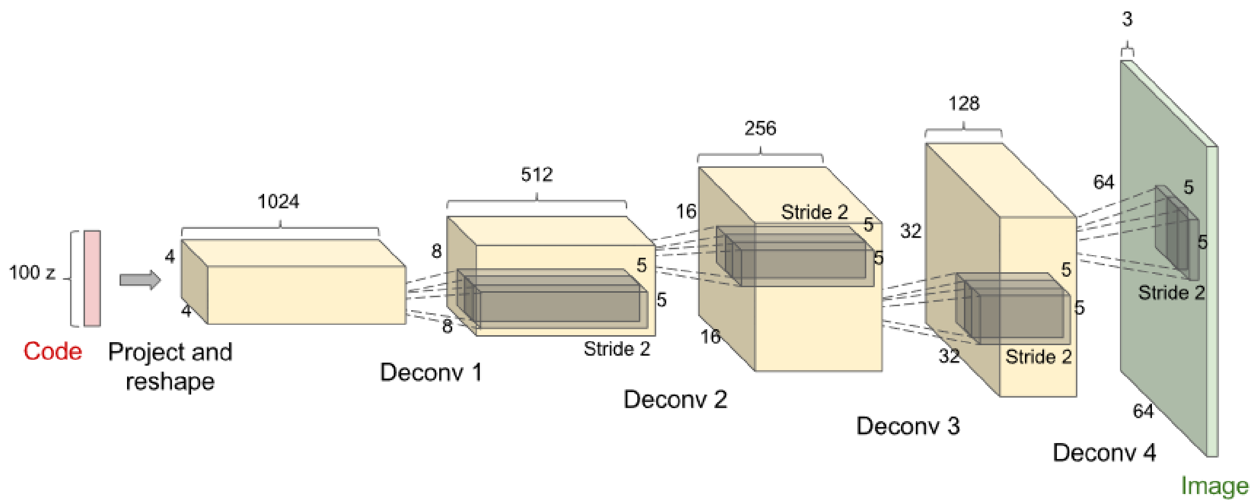
- processes an image and produces a vector/*code*



GANs

Decoder

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



GANs

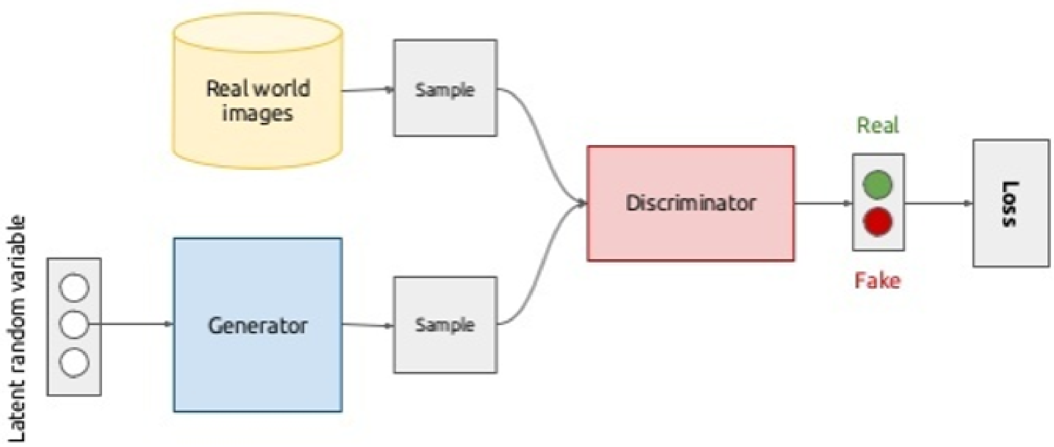
Problem
How to train the decoder to produce meaningful data?

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

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)

VAEs

Goal

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

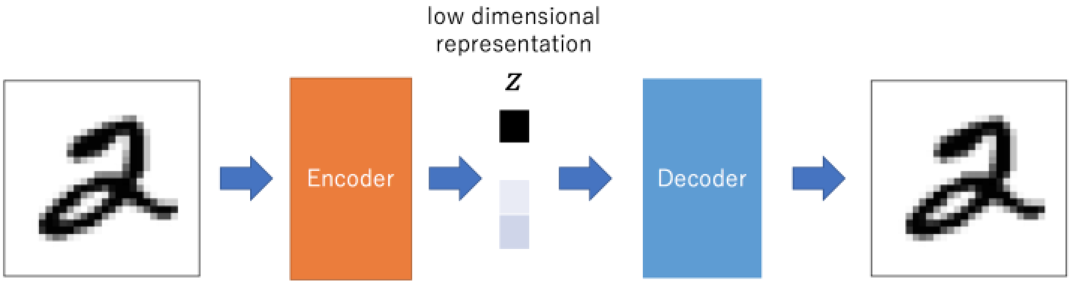
Examples

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

VAEs

What is an auto-encoder?

- a combination of an encoder and a decoder
- trained based on reconstruction loss
- provides low-dimensional representation

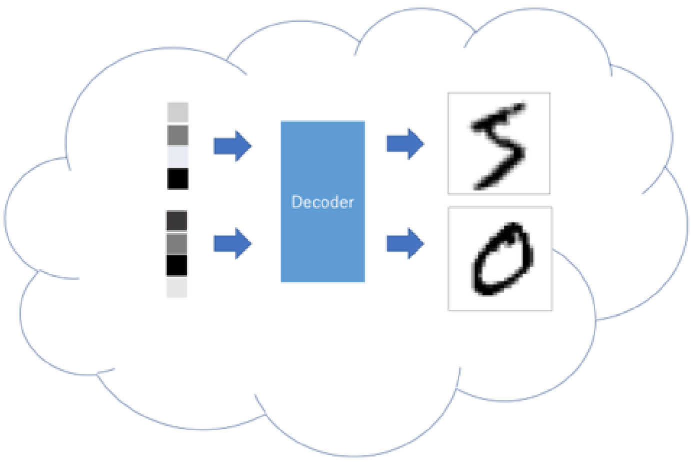


VAEs

Goal

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

AE problem: we don't know if the vectors are valid or not

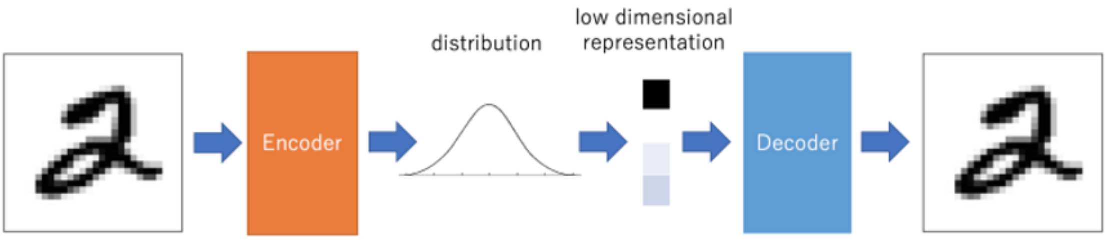


VAEs

Main idea

Encoder produces a *distribution* instead of a vector

Decoder operates on samples from this distribution



GANs

Roles

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)

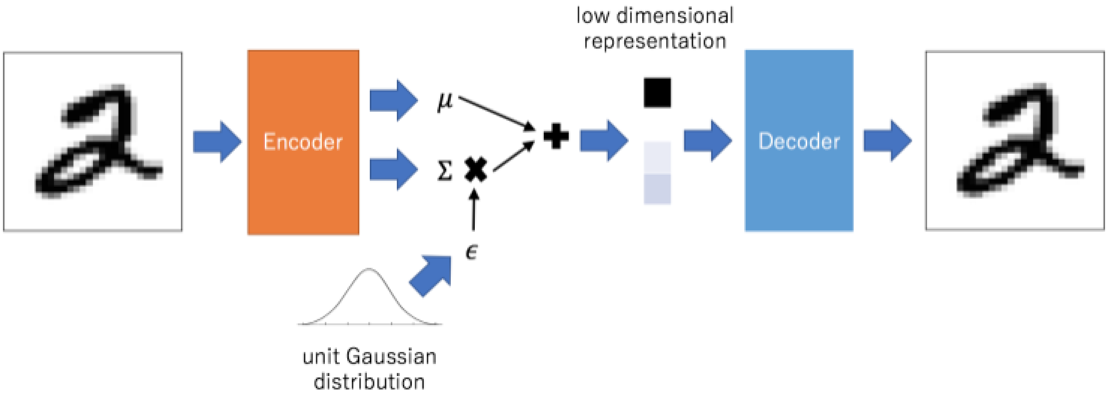
VAEs

One more problem:

- Sampling operation is not differentiable

Solution:

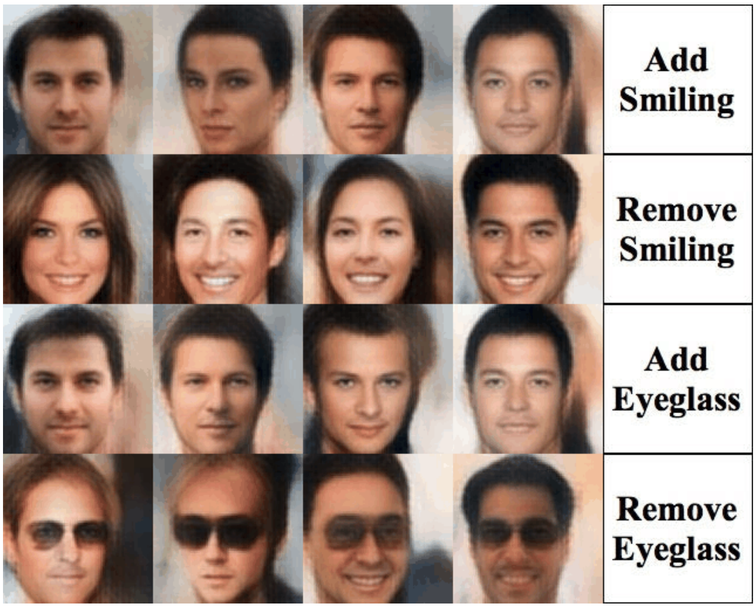
- re-parametrization



Sapienza University of Rome, Italy - Machine Learning (2019/2020)

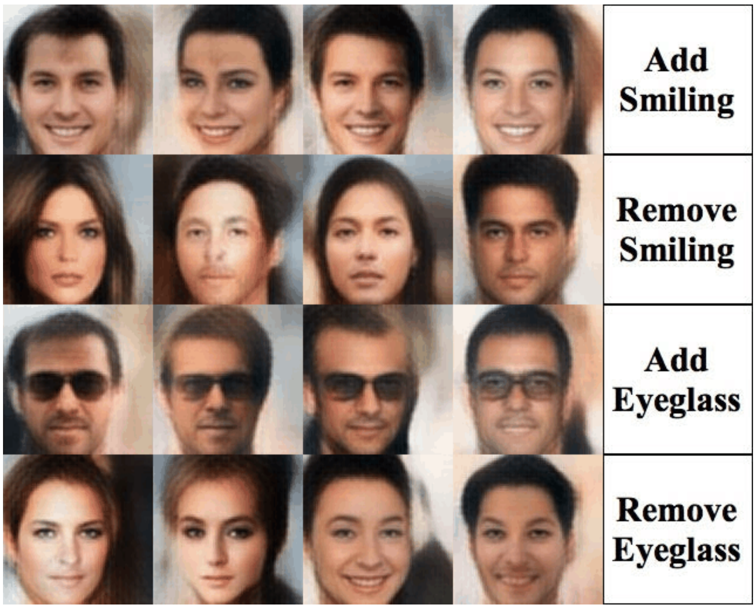
VAEs

Example: faces



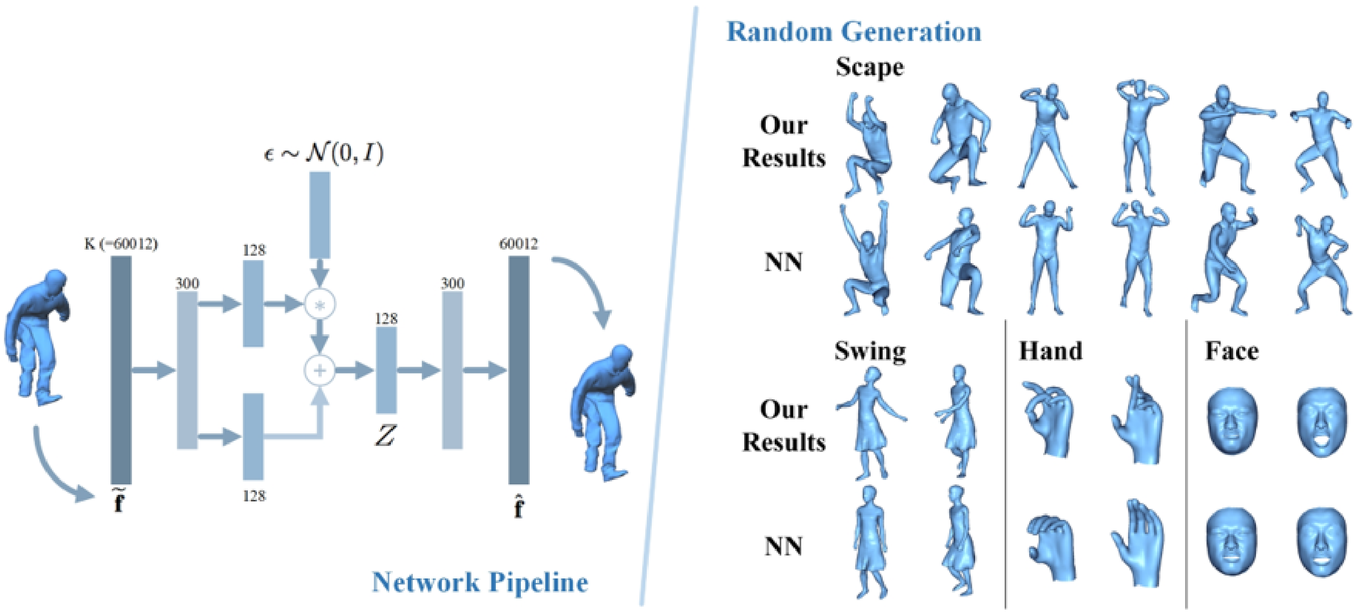
VAEs

Example: faces



VAEs

Example: 3D Mesh deformation



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