教材《机器学习与大语言模型》配套电子教案

机器学习

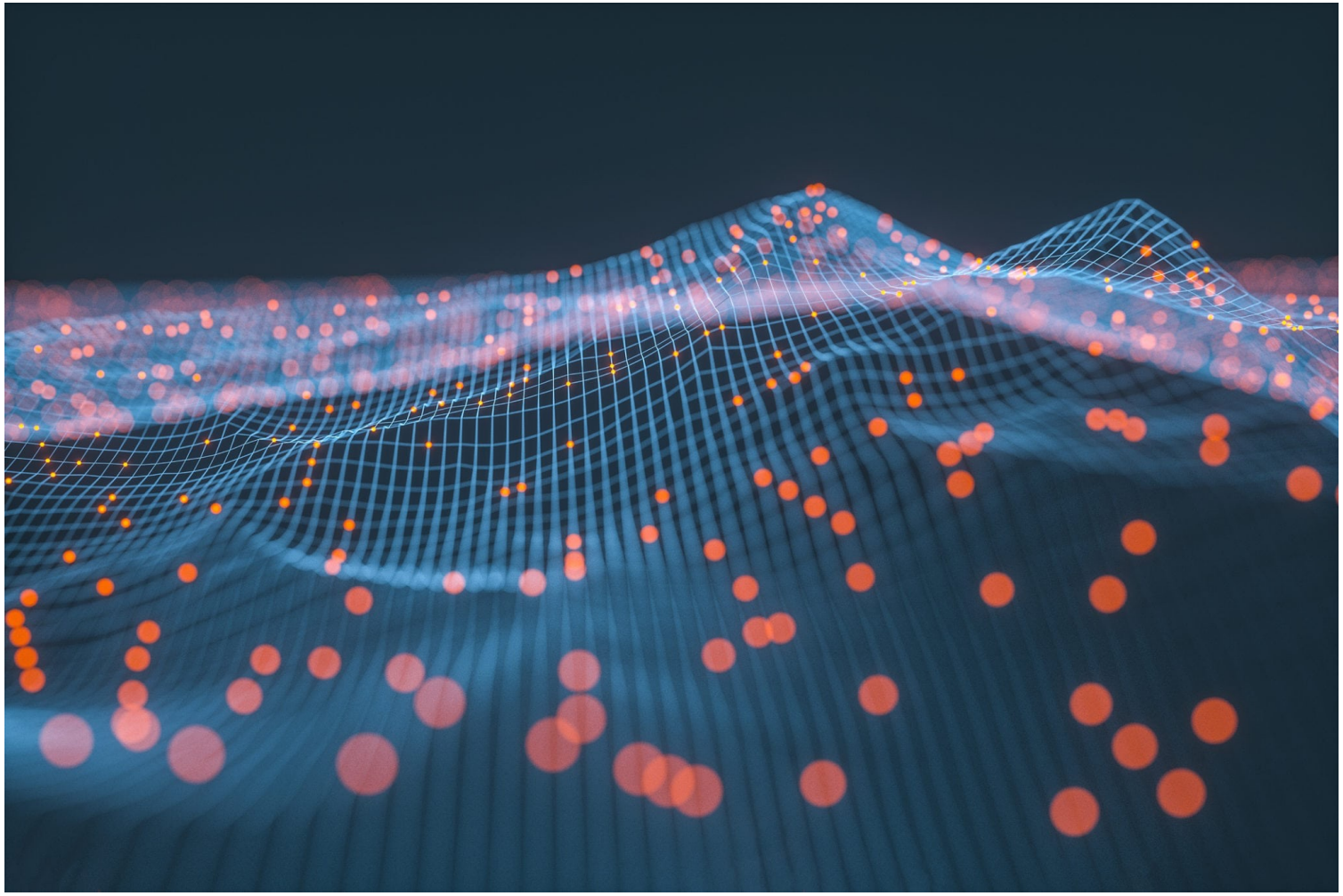
第十五讲: 降维

顾小东
上海交通大学计算机学院

Unsupervised Learning Again

Any other patterns beyond data clusters?

Big Data, are they all useful?



Today

Clustering
(Learning data similarities)

- K-means
- Gaussion Mixture

Generative
(density estimation)

- GAN
- VAE
- Diffusion

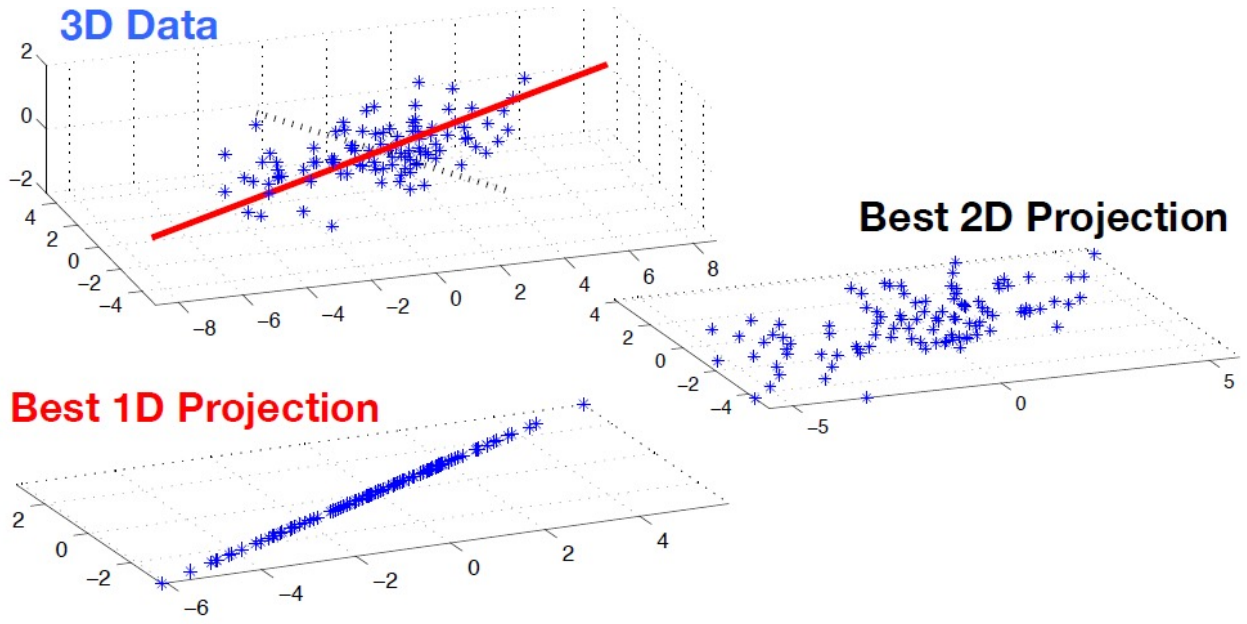
Dimensional Reduction
(data compression)

- Principal component analysis
- Auto-encoder

Dimensionality Reduction

Reduce the number of dimensions (attributes) of the data while preserving the intrinsic characteristics of the data

Try to identify some "hidden" aspects that determines the characteristics of the data

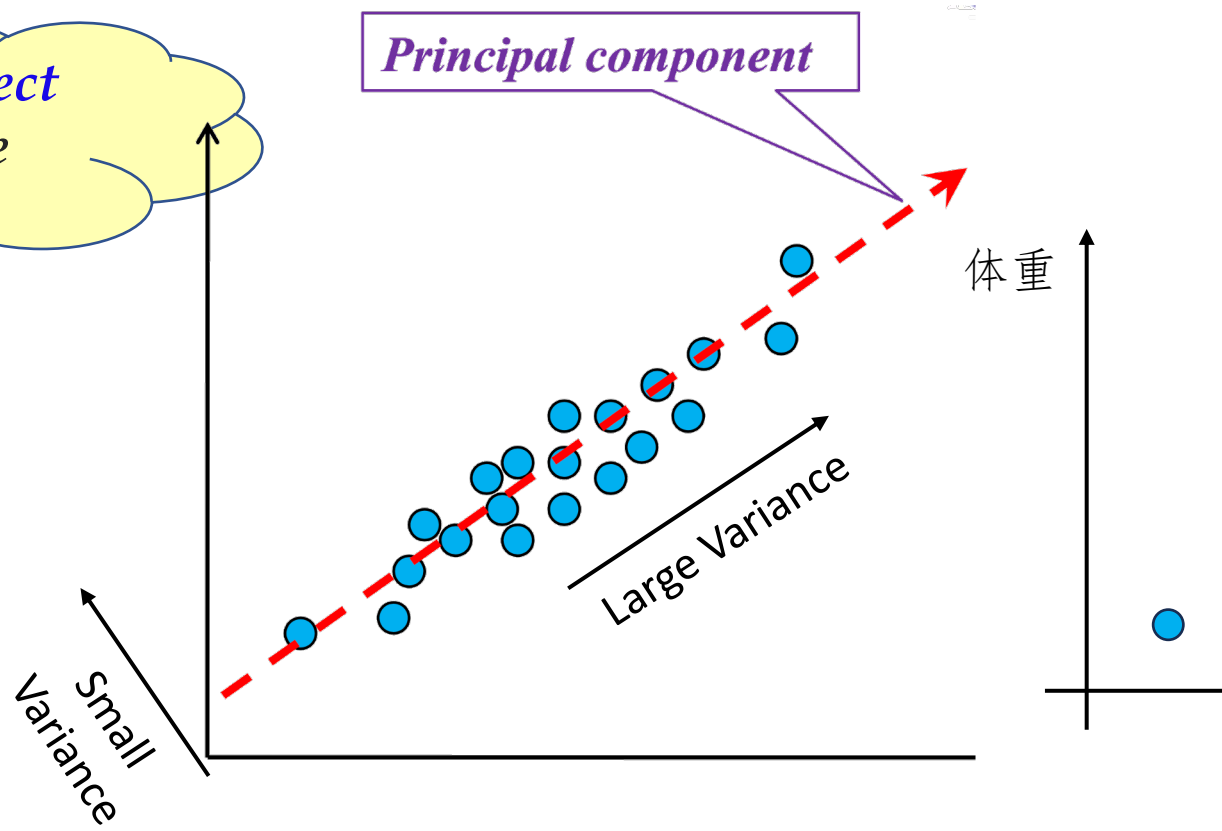


Overview of Methods

- Linear Method
  - **Principal component analysis** (PCA)
  - Linear discriminate analysis (LDA)
  - Factor analysis
  - ...

- Non-linear Method
  - KPCA, KLDA
  - Multidimensional scaling (MDS)
  - t-SNE
  - **Autoencoder**
  - ...

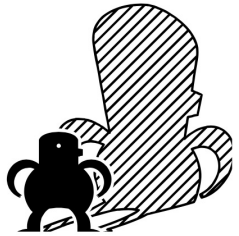
Principal Component Analysis (PCA)



Principal component = the spreading tendency

= dimensionality with the maximum variance

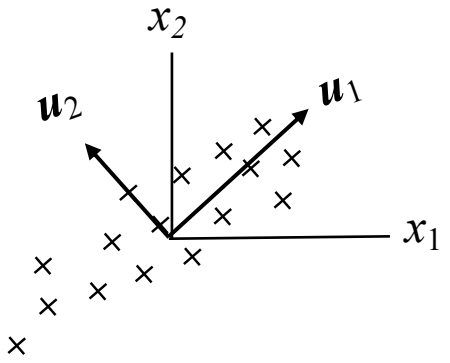
Problem Formulation



Can simply subtract each x_i by the sample mean.

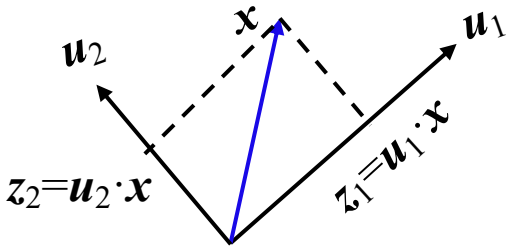
- Data representations

$$\boldsymbol{X} = [x_1, \ldots, x_N],\ x_i \in \mathbb{R}^d, \text{ where } \sum_{i=1}^{N} x_i = 0$$



- Projection of a data point

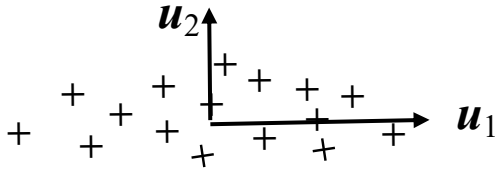
$$z_j = \boldsymbol{u}_j^{\mathrm{T}} x \qquad \boldsymbol{Z} = \boldsymbol{U}^{\mathrm{T}} \boldsymbol{X}$$



- Variance in any projected dimension

$$var(\boldsymbol{u}^{\mathrm{T}}\boldsymbol{X}) = \tfrac{1}{N} (\boldsymbol{u}^{\mathrm{T}}\boldsymbol{X})(\boldsymbol{u}^{\mathrm{T}}\boldsymbol{X})^{\mathrm{T}}$$
$$= \tfrac{1}{N} \boldsymbol{u}^{\mathrm{T}}\mathbf{X}\mathbf{X}^{\mathrm{T}}\boldsymbol{u}$$
$$= \boldsymbol{u}^{\mathrm{T}}\boldsymbol{S}\boldsymbol{u}$$

$S = \mathrm{cov}(X) = \tfrac{1}{N} XX^{\mathrm{T}}$



PCA = Solving an Optimization Problem

- In summary, PCA aims to solve a constrained optimization problem:

$$\max_{u} \; \boldsymbol{u}^{T}\boldsymbol{S}\boldsymbol{u}$$

$$\text{s.t.} \; \boldsymbol{u}^{T}\boldsymbol{u} = \mathbf{1}$$

- This is a quadratic programming (QP) problem, which is one type of convex optimization problem.
- S is positive **semi-definite**.

$$\begin{aligned} x^T S x &= x^T E[(X-\mu)^T(X-\mu)]x \\ &= E[x^T(X-\mu)^T(X-\mu)x] \\ &= E[((X-\mu)x)^T((X-\mu)x)] \\ &= E[\|(X-\mu)x\|^2] \\ &= \|(X-\mu)x\|^2 \\ &\geqslant 0 \end{aligned}$$

Solving PCA

- Lagrangian: $\mathcal{L} = \boldsymbol{u}^{\mathrm{T}}\boldsymbol{S}\boldsymbol{u} - \lambda(\boldsymbol{u}^{\mathrm{T}}\boldsymbol{u} - 1)$

$$\nabla_{\boldsymbol{u}}\mathcal{L} = 0 \Rightarrow \boldsymbol{S}\boldsymbol{u} = \lambda\boldsymbol{u}$$

- Solving PCA subjects to **eigen decomposition on S**:

  $\boldsymbol{u}$: eigen vector, λ: eigen value

- Since $\boldsymbol{S}$ is positive semi-definite, we have

$$\boldsymbol{S} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^{\mathrm{T}} = [\boldsymbol{u}_1,\ldots,\boldsymbol{u}_{\mathrm{d}}]\mathrm{diag}(\lambda_1,\lambda_2,\ldots,\lambda_{\mathrm{d}})[\boldsymbol{u}_1,\ldots,\boldsymbol{u}_{\mathrm{d}}]^{\mathrm{T}}$$

The Idea

- Find k orthogonal principal components

  correspond to the k eigen vectors with k **largest** eigen values

$$S = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^{T} \approx \boldsymbol{U}_{1:k}\boldsymbol{\Sigma}_{1:k}\boldsymbol{U}^{T}_{1:k}$$

- Project each input vector x into this subspace

$$z_j = \boldsymbol{u}_j^{T}x \qquad \boldsymbol{z} = \boldsymbol{U}_{1:k}^{T}\boldsymbol{x}$$

Intuitions:

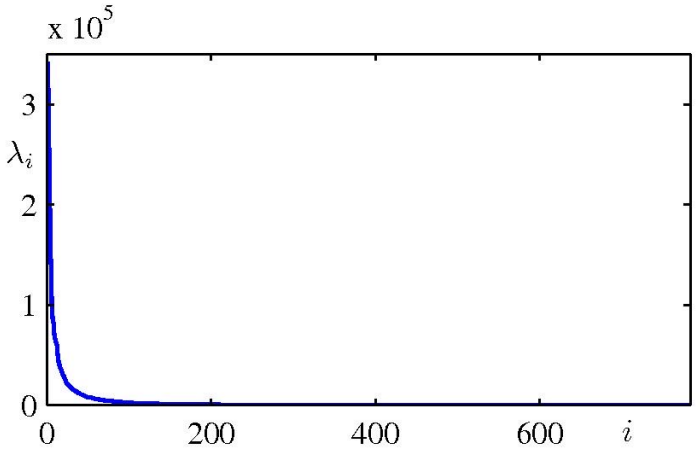
– *Eigen value is the variance of the data projected to the corresponding eigen vector*

– *Eigen vectors are essentially orthogonal*

Determine "k"

- **What to discard**: the eigen vectors with small eigen values
- Limit the loss of information within an acceptable interval (usually 5%)

$$\text{info loss} = \frac{\lambda_{k+1} + \ldots + \lambda_d}{\lambda_1 + \lambda_2 + \ldots + \lambda_d}$$



Variance preserved at *i*-th eigenvalue

The Algorithm

1. Shifting data set to have zeros mean

2. Compute the covariance matrix S

3. Conduct eigen decomposition on S, and rank the eigen vectors according to their eigen values

4. Determine the number of dimensions *k* of the new space

5. Select the first *k* eigen vectors with d largest eigen values as the basis of the new space

Applications of PCA

- Data visualization
- Preprocessing
- Modeling – prior for new data
- Compression

Application: Face Recognition

- The space of face images are extremely high-dimensional
  - 24x24 image = 576 dimensions
  - slow and lots of storage

But very few 576-dimensional vectors are valid face images

How to effectively model the subspace of face images?

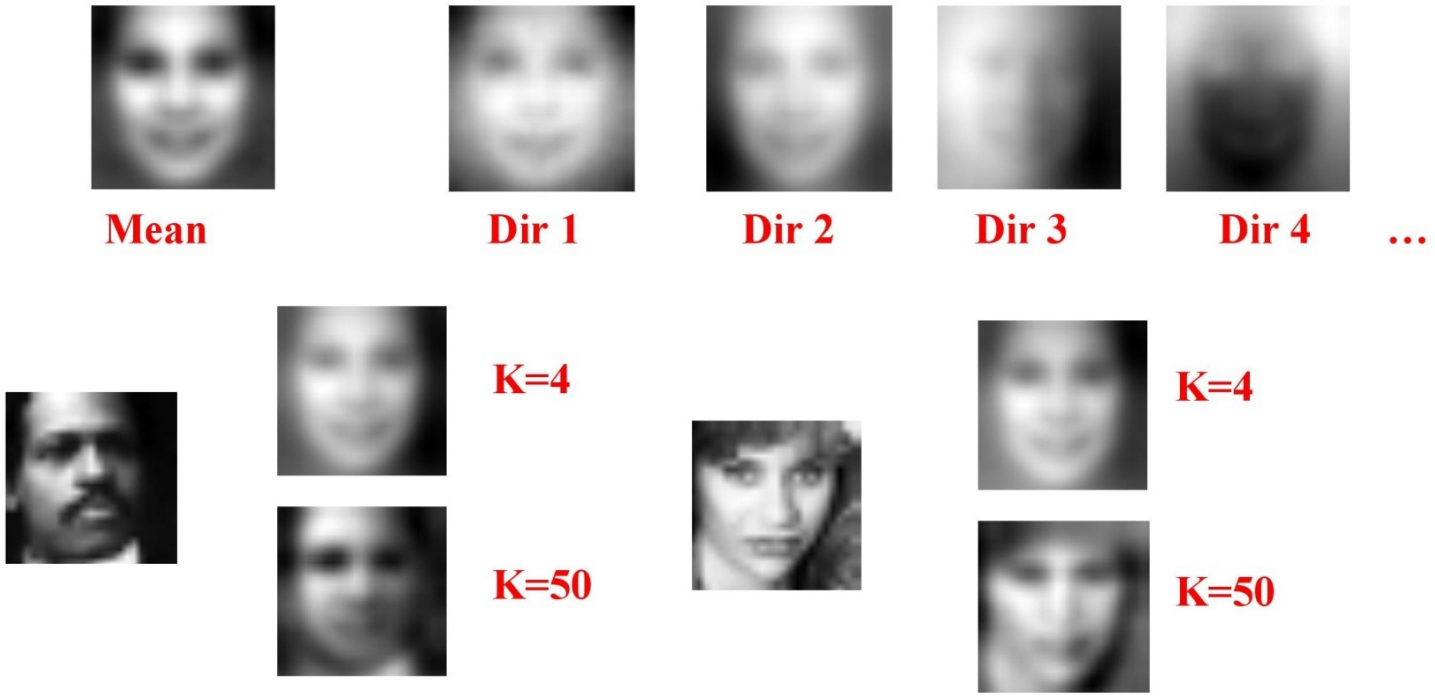
Adapted from Derek Hoiem

"Eigen-Faces"

Example: Viola Jones data set

– 24×24 images of faces = 576 dimensional measurements

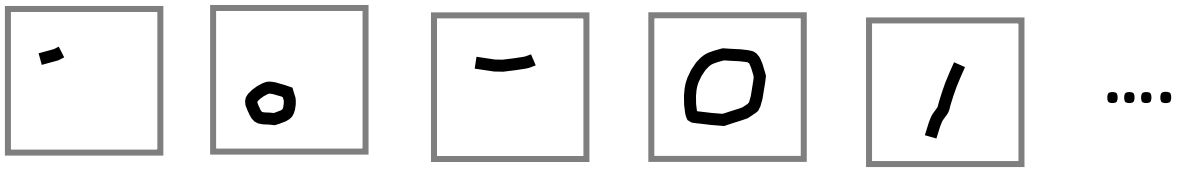
– Take the first-K PCA components



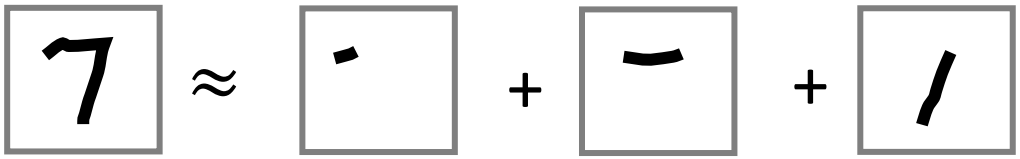
Slide credit: Alexander Ihler

PCA – Another Perspective

Basic components of handwriting digits:



u_1 u_2 u_3 u_4 u_5



$x \approx u_1 + u_3 + u_5 \Rightarrow \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ \vdots \end{bmatrix}$

$$x \approx z_1 u_1 + z_2 u_2 + \ldots + z_k u_k$$

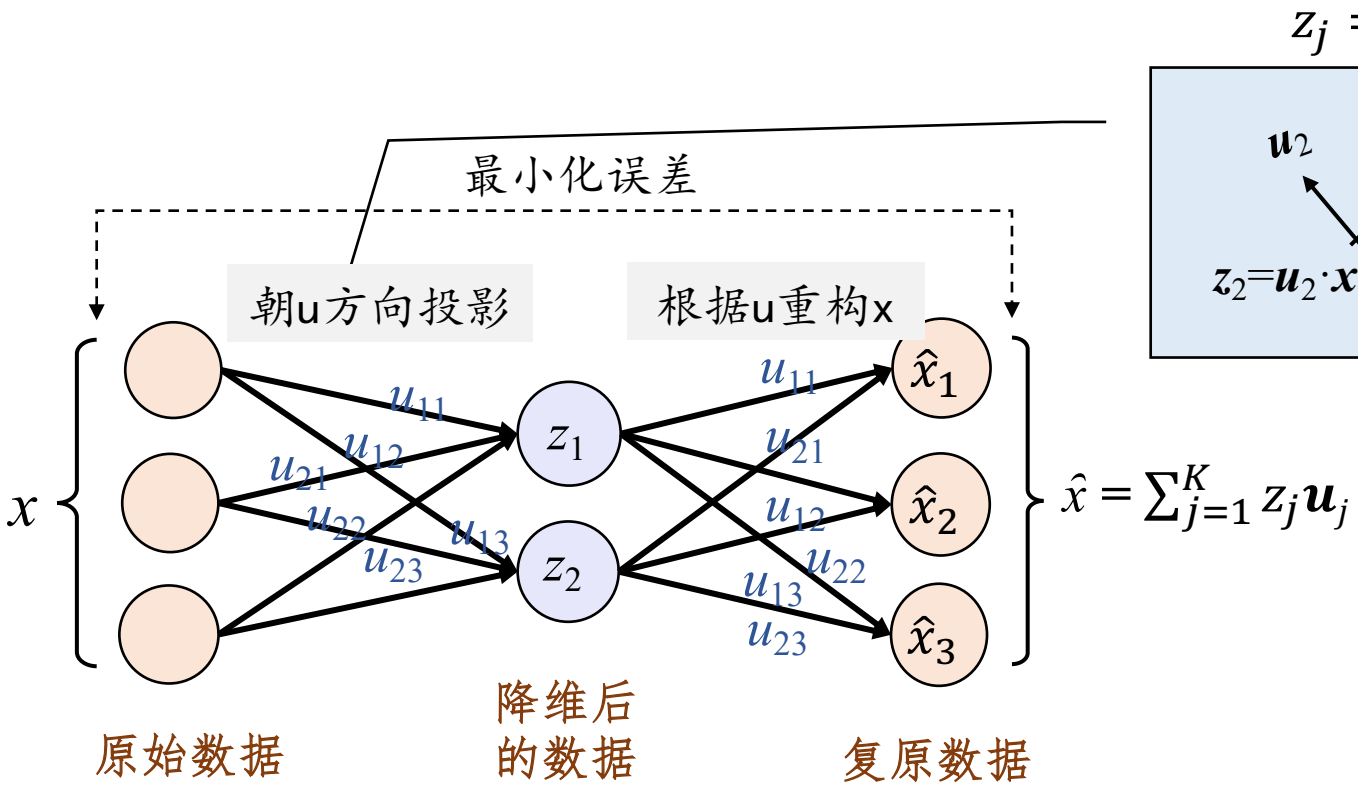
Image pixels

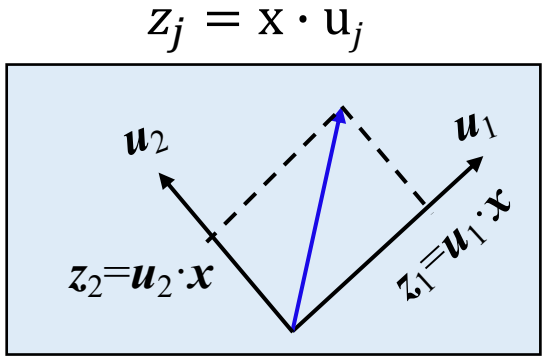
components

sample mean

$\begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_k \end{bmatrix}$ Compressed representation

PCA = Neural Network

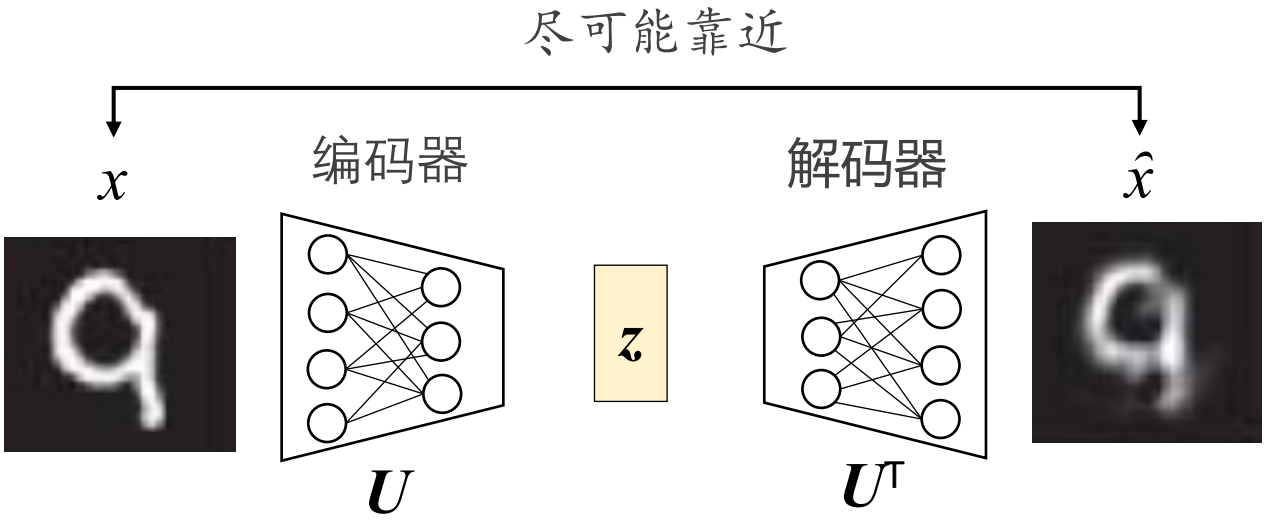




PCA = a neural network with one hidden layer (linear activation)

PCA = Auto-encoder

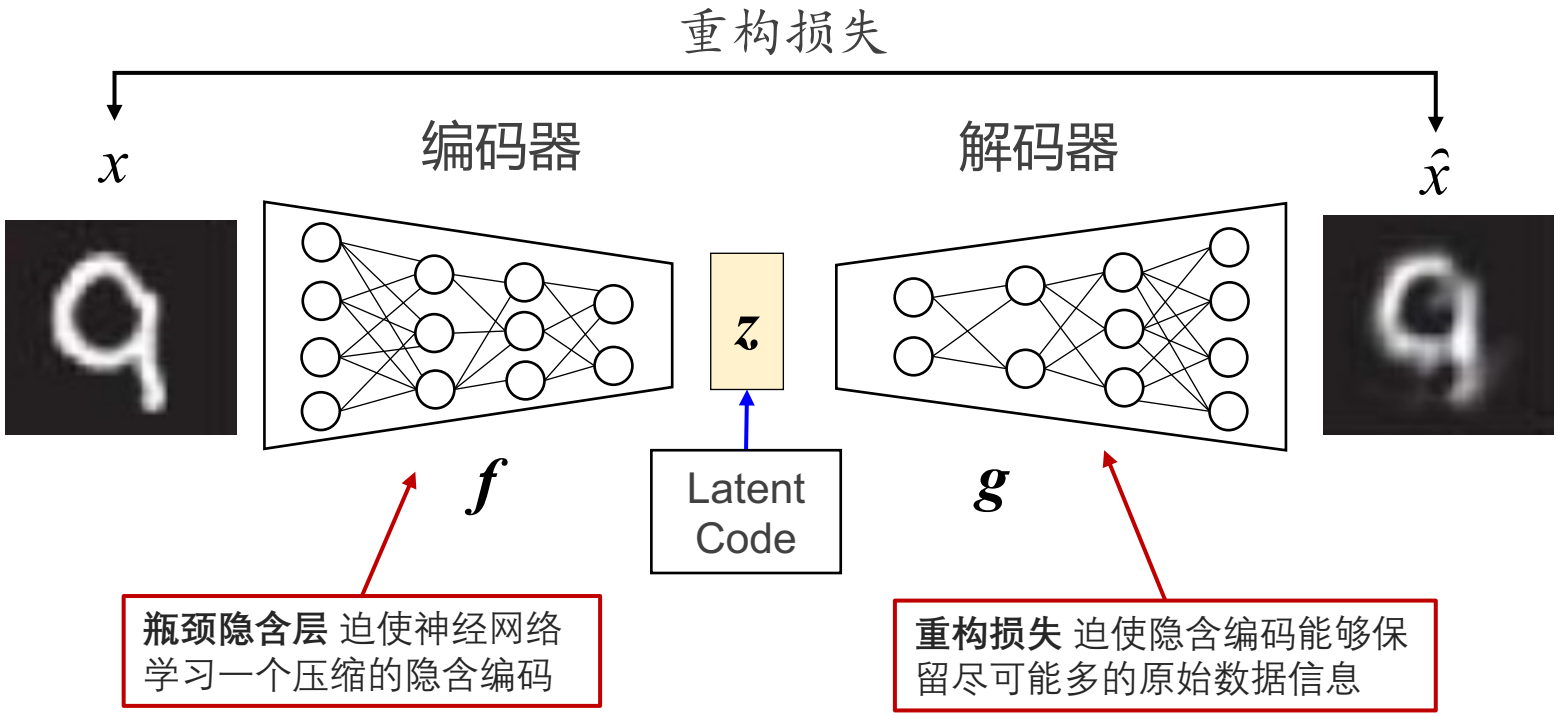
- **Auto-encoder**: an encoder-decoder **neural network** whose output tries to reconstruct the input.



Learns a **lower-dimensional** feature representation (z) from unlabeled training data (x).

Deep Auto-encoder

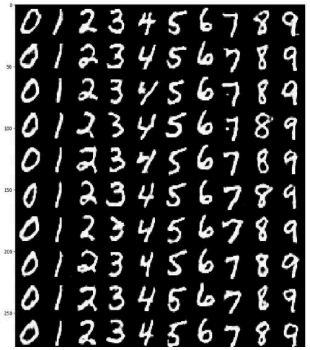
- Of course, the auto-encoder can be deep

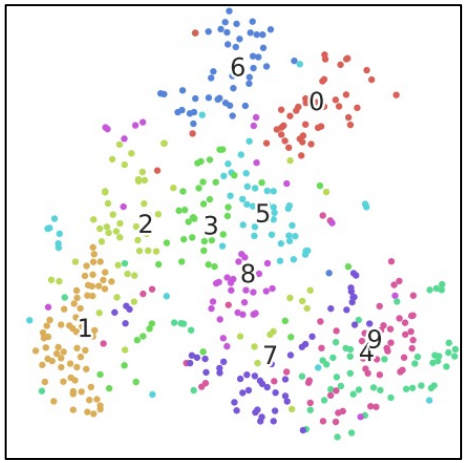


Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." *Science* 313.5786 (2006): 504-507

Auto-encoder vs PCA

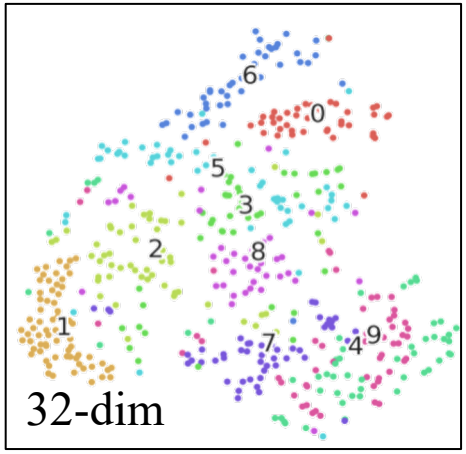
Example



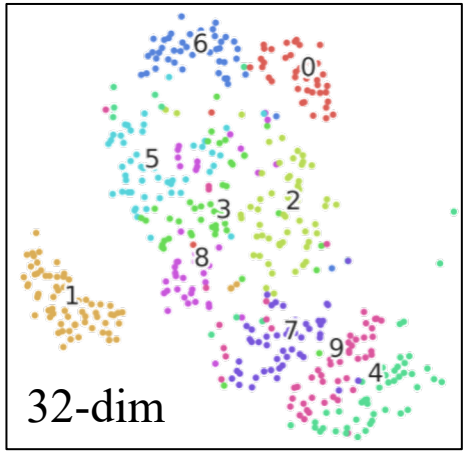


Pixels
(tSNE)

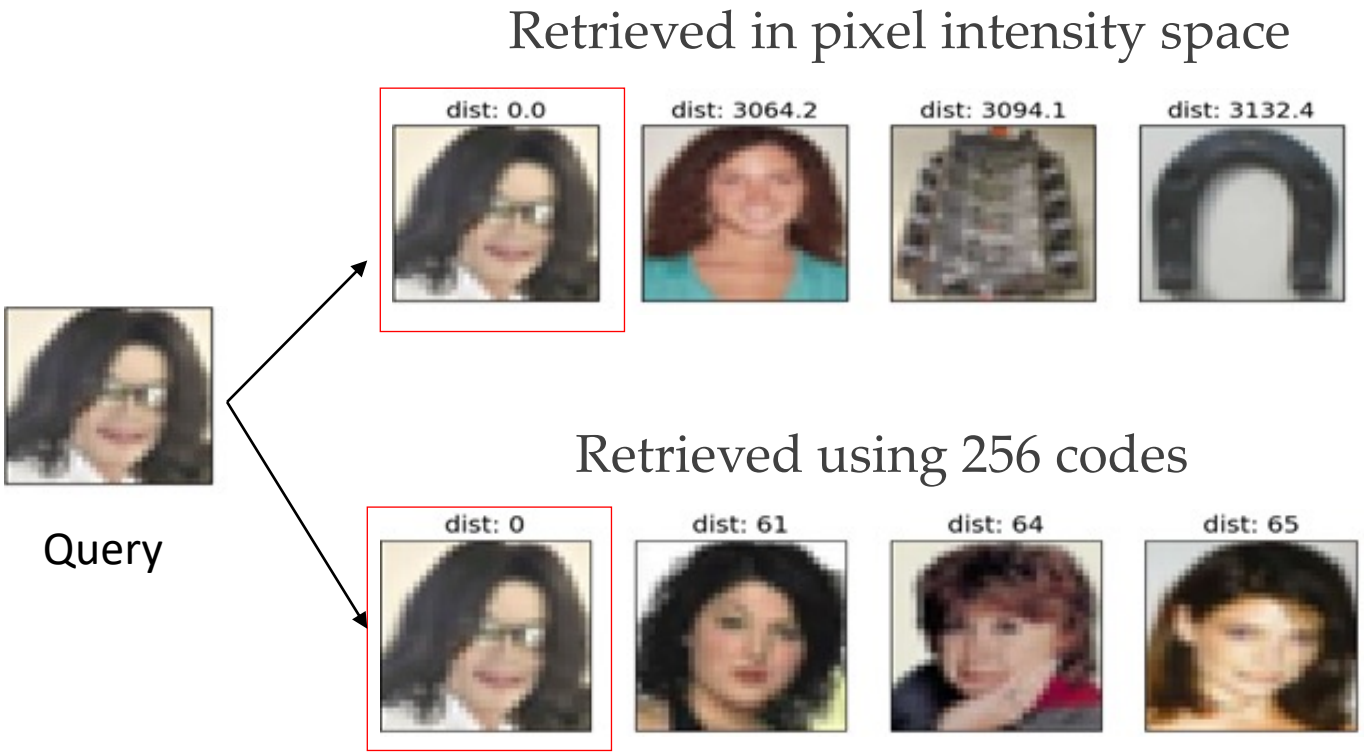
PCA



NN Auto-encoder

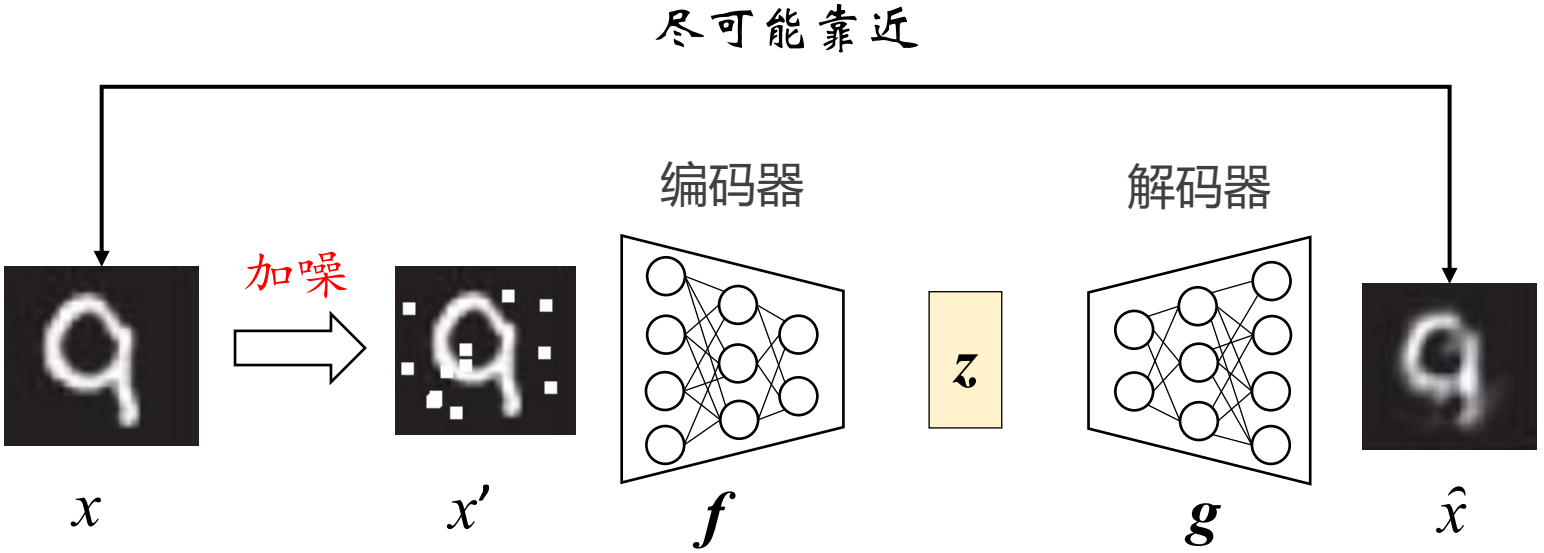


Applications – Image Search



Denoising Auto-encoder

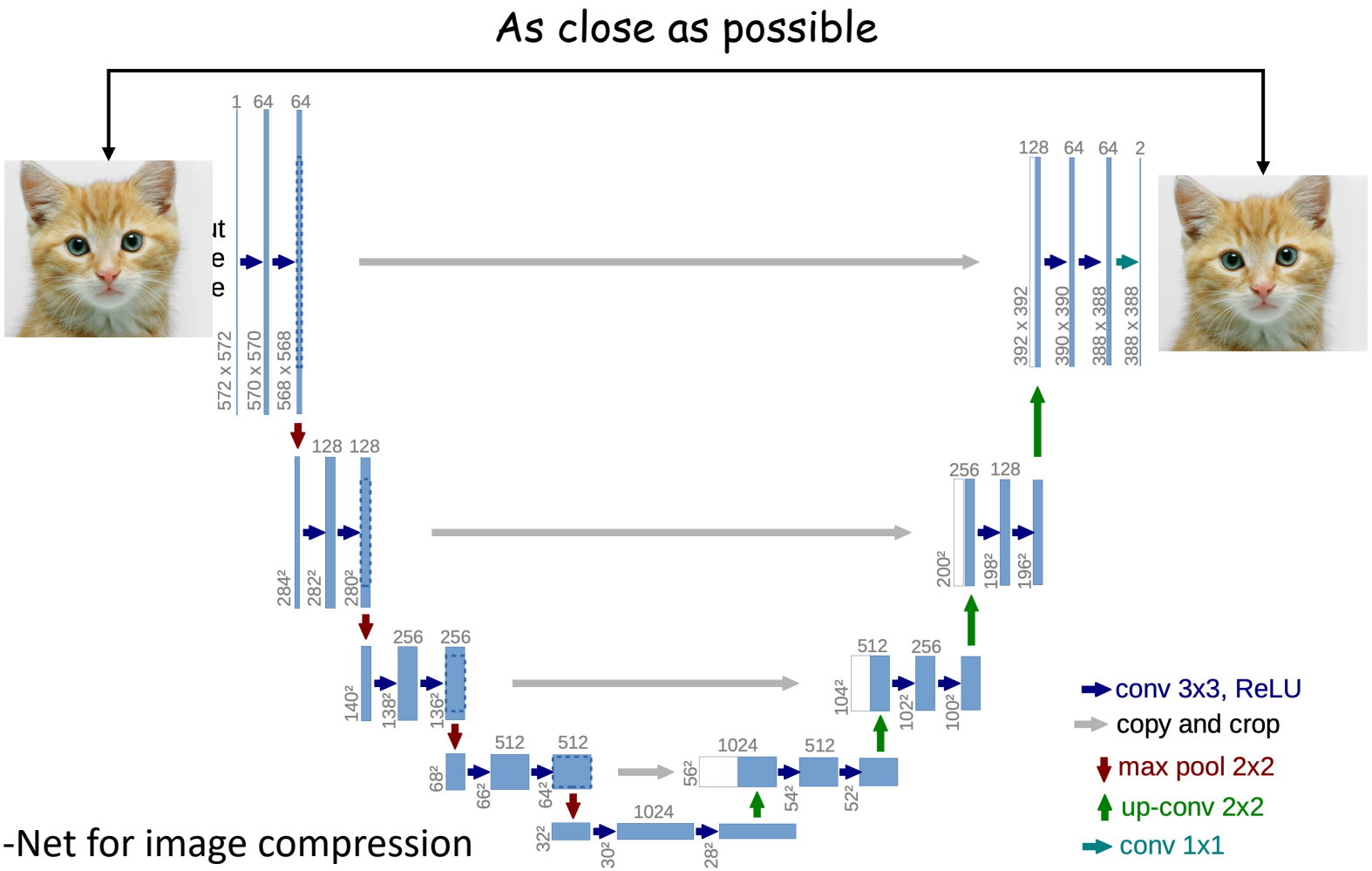
- An autoencoder that receives a **corrupted** data point as input and is trained to predict the original data point.



Applications: remove noise from data.

Vincent, Pascal, et al. "Extracting and composing robust features with denoising autoencoders." *ICML,* 2008.

Auto-encoder with CNN?



U-Net for image compression

What's Next?

Deep Generative Models

Learning to generate data