Applied Machine Learning

Dimensionality reduction

Isabeau Prémont-Schwarz

School of Computer Science

Learning objectives

What is dimensionality reduction?

What is it good for?

Linear dimensionality reduction:

- Principal Component Analysis
- Relation to Singular Value Decomposition

Motivation

Scenario: we are given high dimensional data and asked to make sense of it!

Real-world data is high-dimensional

- Visualization: we can't visualize beyond 3D
- Compression: processing and storage is costly
- Downstrean analysis, e.g. clustering or classification
  - features may not have any semantics (value of the pixel vs happy/sad)
  - many features may not vary much in our dataset (e.g., background pixels in face images)

Dimensionality reduction: faithfully represent the data in low dimensions

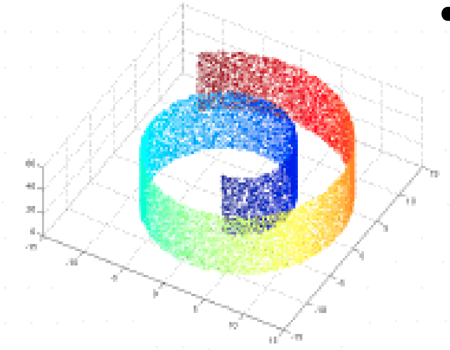
- We can often do this with real-world data *(manifold hypothesis)*
- finding meaningful low-dimensional structures in high-dimensional observations

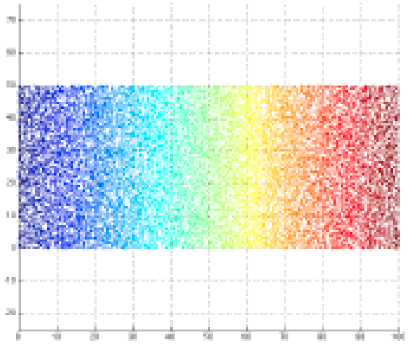
Dimensionality reduction

Dimensionality reduction: faithfully represent the data in low dimensions

- learn a mapping between (coordinates) at low-dimension and high-dimensional data

$x^{(n)} \in \mathbb{R}^3 \quad \leftrightarrows \quad z^{(n)} \in \mathbb{R}^2$





some methods give this mapping in both directions and some only in one direction.

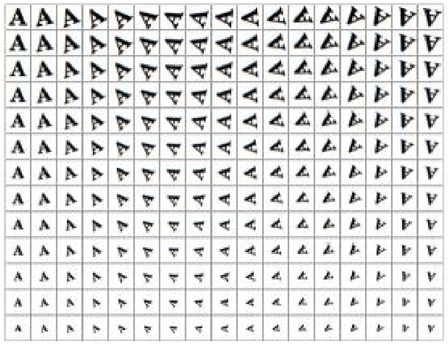
Dimensionality reduction

Dimensionality reduction: faithfully represent the data in low dimensions

- learn a mapping between (coordinates at) low-dimension and high-dimensional data

$x^{(n)} \in \mathbb{R}^{400}$ ⇄ $z^{(n)} \in \mathbb{R}^2$

each image is 20x20



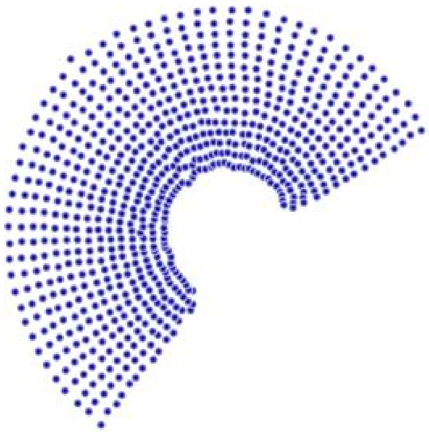
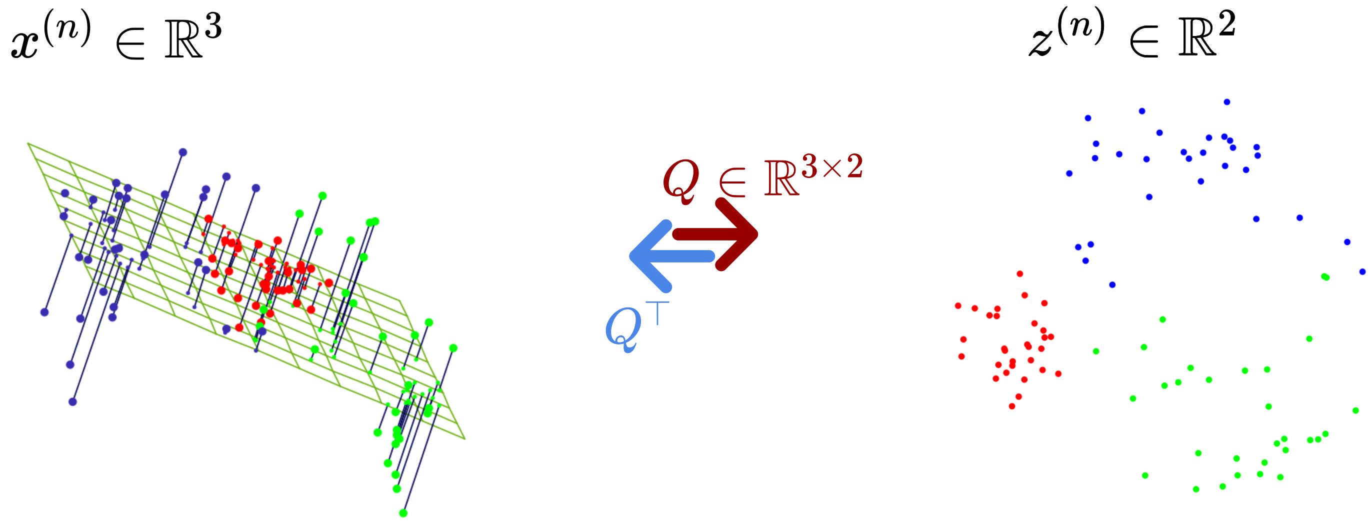


image: wikipedia

Principal Component Analysis (PCA)

PCA is a **linear** dimensionality reduction method



where Q has orthonormal columns $Q^\top Q = I$

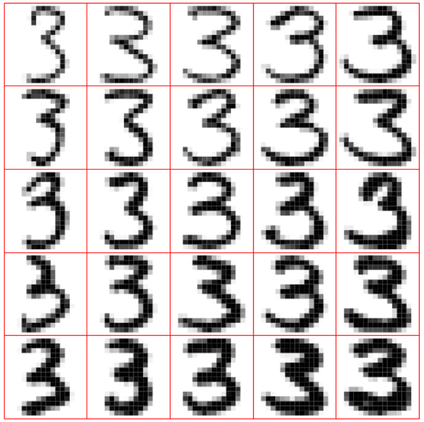
it follows that the pseudo-inverse of Q is $Q^\dagger = (Q^\top Q)^{-1} Q^\top = Q^\top$

PCA: optimization objective

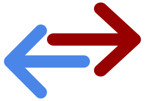
PCA is a **linear** dimensionality reduction method

$x^{(n)} \in \mathbb{R}^{784}$

each image has 28x28=784 pixels

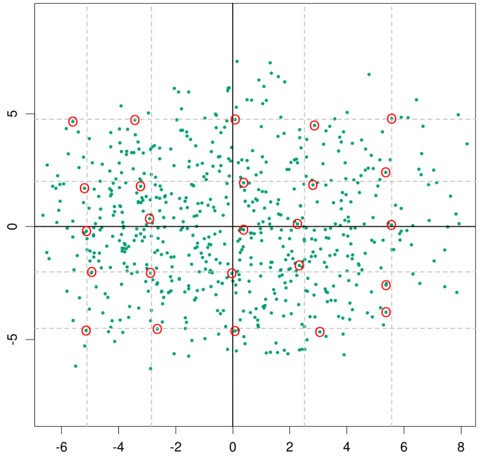


$Q \in \mathbb{R}^{784\times 2}$



$Q^\top$

$z^{(n)} \in \mathbb{R}^2$



faithfulness is measured by the reconstruction error

$$\min_Q \quad \sum_n ||x^{(n)} - \underbrace{x^{(n)\top} Q}_{z^{(n)}} Q^\top||_2^2 \quad s.t. \quad Q^\top Q = I$$

PCA: optimization objective

PCA is a **linear** dimensionality reduction method

faithfulness is measured by the reconstruction error

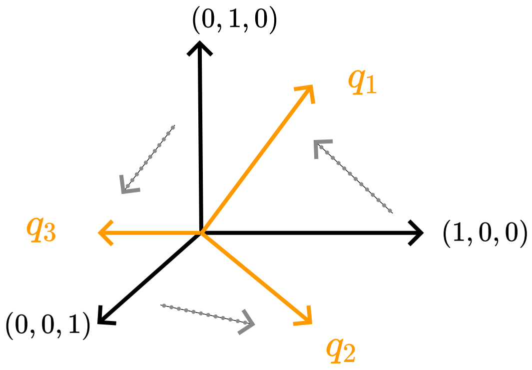
$$\min_Q \quad \sum_n ||x^{(n)} - \underbrace{x^{(n)\top} Q}_{z^{(n)}} Q^\top ||_2^2 \qquad s.t. \qquad Q^\top Q = I$$

strategy: find $D \times D$ matrix Q, and only use D' columns

Since Q is orthogonal we can think of it as a change of coordinates

$$Q = \begin{bmatrix} Q_{1,1}, \ldots, Q_{1,D} \\ \vdots, \ddots, \vdots \\ Q_{D,1}, \ldots, Q_{D,D} \end{bmatrix}$$

$q_1 \qquad q_D$



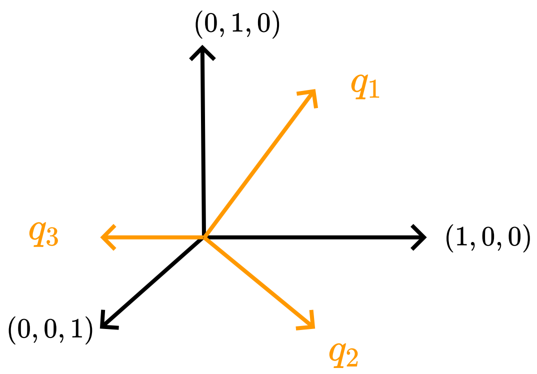
PCA: a change of coordinates

strategy: find $D \times D$ matrix Q, and only use D' columns

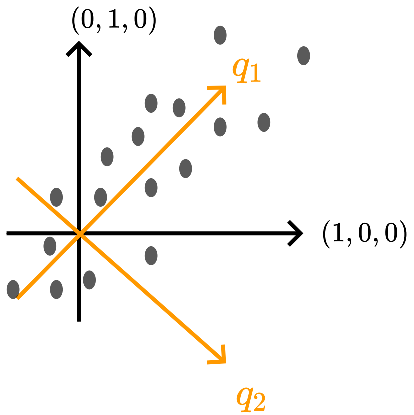
Since Q is orthonormal we can think of it as a change of coordinates

$$Q = \begin{bmatrix} Q_{1,1}, & \dots, & Q_{1,D} \\ \vdots, & \ddots, & \vdots \\ Q_{D,1}, & \dots, & Q_{D,D} \end{bmatrix}$$

q_1 $\quad$ q_D



example $D = 2$



we want to change coordinates such that
coordinates 1,2,...,D' best explain the data for any given D'

PCA preserves variance

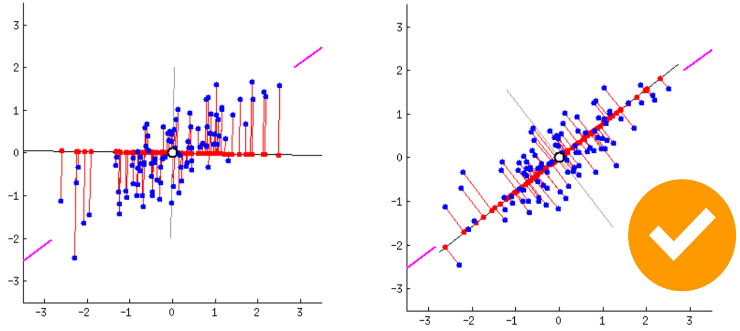
$$Q = \begin{bmatrix} Q_{1,1}, \dots, Q_{1,D} \\ \vdots, \ddots, \vdots \\ Q_{D,1}, \dots, Q_{D,D} \end{bmatrix}$$

q_1

Find a change of coordinate using *orthonormal matrix*

first new coordinate has maximum **variance** *(lowest reconstruction error)*
second coordinate has the next largest variance
...

along which one of these directions the data has a higher variance? more spread out?



this direction is the vector q_1

projection is given by $\frac{x^{(n)\top} q_1}{||q_1||_2} = x^{(n)\top} q_1$

projection of the whole dataset is $X q_1 = z_1$

$z_1^\top = [z_1^{(1)}, z_1^{(2)}, \dots, z_1^{(N)}]$

PCA preserves variance

Find a change of coordinate using *orthonormal matrix*

first new coordinate has maximum variance

projection of the whole dataset is $z_1 = X q_1$

$Var(z_1) = \frac{1}{N} \sum_n (z_1^{(n)} - 0)^2$

assuming features have zero mean, maximize the variance of the projection: $\frac{1}{N} z_1^\top z_1$

$$\max_{q_1} \frac{1}{N} z_1^\top z_1 = \max_{q_1} \frac{1}{N} q_1^\top X^\top X q_1 = \max_{q_1} q_1 \Sigma q_1^\top$$

dxd **covariance matrix**

$$\Sigma = \frac{1}{N} X^\top X = \frac{1}{N} \sum_n (x^{(n)} - 0)(x^{(n)} - 0)^\top$$

because the mean is zero

$\Sigma_{i,j}$ is the sample covariance of feature i and j

$$\Sigma_{i,j} = \mathrm{Cov}[X_{:,i}, X_{:,j}] = \frac{1}{N} \sum_n x_i^{(n)} x_j^{(n)}$$

Covariance matrix

variance of a random variable $\text{Var}(x) = \mathbb{E}[(x - \mathbb{E}[x])^2] \; = \mathbb{E}[x^2] - \mathbb{E}[x]^2$

covariance of two random variable $\text{Cov}(x, y) = \mathbb{E}[(x - \mathbb{E}[x])(y - \mathbb{E}[y])] \; = \mathbb{E}[xy] - \mathbb{E}[x]\mathbb{E}[y]$

for $x \in \mathbb{R}^D$ we have covariance matrix

$\text{Cov}(x_1, x_1) = \text{Var}(x_1)$ $\quad$ $\text{Cov}(x_1, x_D)$

$$\Sigma = \begin{bmatrix} \Sigma_{1,1} & \cdots & \Sigma_{1,D} \\ \vdots & \ddots & \vdots \\ \Sigma_{D,1} & \cdots & \Sigma_{D,D} \end{bmatrix} = \mathbb{E}[(x - \mathbb{E}[x])(x - \mathbb{E}[x])^\top] \; = \mathbb{E}[xx^\top] - \mathbb{E}[x]\mathbb{E}[x]^\top$$

$D \times 1$ $\quad$ $1 \times D$ $\quad$ $D \times D$ $\quad$ $D \times D$

given a dataset $\mathcal{D} = \{x^{(1)}, \ldots, x^{(N)}\}$ sample covariance matrix

Σ^{MLE}

$$\hat{\Sigma} = \mathbb{E}_\mathcal{D}[(x - \mathbb{E}_\mathcal{D}[x])(x - \mathbb{E}_\mathcal{D}[x])^\top]$$

the empirical estimate

$x - \left(\frac{1}{N}\sum_{x \in \mathcal{D}} x\right)$

Correlation and dependence

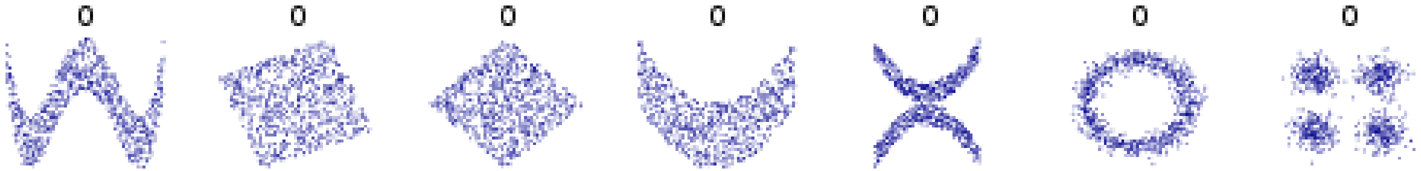
correlation is normalized covariance

$$\mathrm{Corr}(x_i, x_j) = \frac{\mathrm{Cov}(x_i, x_j)}{\sqrt{\mathrm{Var}(x_i)\mathrm{Var}(x_j)}} \in [-1, +1]$$

two variables that are independent are uncorrelated as well

$$p(x_i, x_j) = p(x_i)p(x_j) \rightarrow \mathbb{E}[x_i x_j] = \mathbb{E}[x_i]\mathbb{E}[x_j] \rightarrow \mathrm{Cov}(x_i x_j) = 0$$

the inverse is generally not true (zero correlation doesn't mean independence)



in each example above correlation between two coordinates is zero, but they are not independent

image from wikipedia

Decomposing the covariance matrix

covariance matrix is symmetric positive semi definite

- symmetric
  - $\Sigma_{d,d'} = \mathrm{Cov}(x_d, x_{d'}) = \mathrm{Cov}(x_{d'}, x_d) = \Sigma_{d',d}$
- positive semi definite
  - for any $y \in \mathbb{R}^D$ we have $y^\top \Sigma y = (y^\top \mathbb{E}[(x - \mathbb{E}[x])(x - \mathbb{E}[x])^\top] y) = \mathrm{Var}(y^\top x) \geq 0$

any symmetric positive semi-definite matrix can be decomposed as

$$\Sigma = Q \Lambda Q^\top$$

Spectral Decomposition

diagonal $D \times D$

orthogonal $QQ^\top = Q^\top Q = I$ *(rotation and reflection)*

PCA with Eigenvalue decomposition

find a change of coordinate using *an orthogonal matrix*

first new coordinate has maximum variance

$$\max_{q_1} \; q_1 \Sigma q_1^\top \qquad s.t. \quad ||q_1|| = 1$$

covariance matrix is **symmetric** and **positive semi-definite**

$$(X^\top X)^\top = X^\top X \qquad a^\top \Sigma a = \frac{1}{N} a^\top X^\top X a = \frac{1}{N} ||Xa||_2^2 \geq 0 \quad \forall a$$

any symmetric matrix has the following decomposition

$$\Sigma = Q \Lambda Q^\top$$

(as we see shortly using Q here is not a co-incidence)

$QQ^\top = Q^\top Q = I$ dxd orthogonal matrix
each column is an eigenvector

diagonal and sorted $(\lambda_1 > \lambda_2 > \lambda_3 > \ldots)$
corresponding eigenvalues are on the diagonal
positive semi-definiteness means these are non-negative

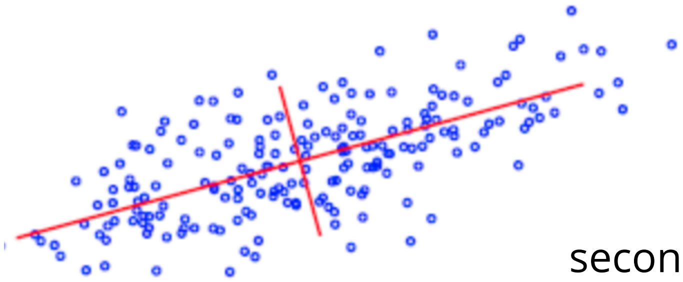
PCA: Principal Component Analysis

find a change of coordinate using *an orthogonal matrix*

first new coordinate has maximum variance

$$q_1^* = \arg\max_{q_1} \; q_1^\top \Sigma q_1 \qquad s.t. \quad ||q_1|| = 1$$

$\max_{q_1} \; q_1^\top Q \Lambda Q^\top q_1 = \lambda_1$ using eigenvalue decomposition



maximizing direction is the eigenvector with the largest eigenvalue (first column of Q)

$q_1 = Q_{:,1}$ first principal direction

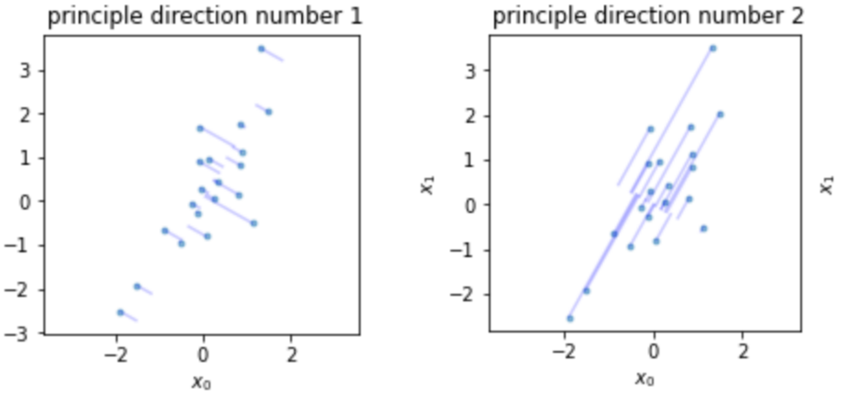
second eigenvector gives the $q_2 = Q_{:,2}$ second principal direction

$\vdots$

so for PCA we need to find the eigenvectors of the covariance matrix

Reducing dimensionality

projection into the principal direction q_i is given by Xq_i



think of the projection XQ as a change of coordinates

we can use the first D' coordinates $Z = XQ_{:,:D'}$
to reduce the dimensionality while capturing a lot of the variance in the data

we can project back into original coordinates using $\underbrace{\tilde{X}}_{\text{reconstruction}} = ZQ_{:,:D'}^{\top}$

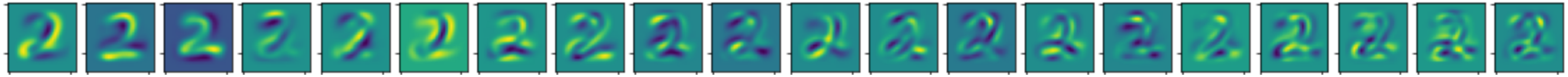
Example: digits dataset

let's only work with digit 2! $x^{(n)} \in \mathbb{R}^{784}$

$x^{(1)}$ $x^{(2)}$ ...

center the data and form the covariance matrix Σ 784×784

find the **eigenvectors** of the covariance matrix, the principal directions

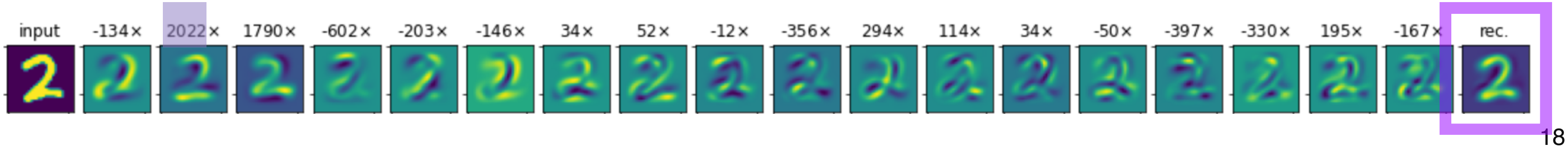


q_1 q_2 $\dots$ q_{20}

use the first 20 directions to reduce dimensionality from 784 to 20!

PC coefficient $x^\top q_i$ (the new coordinates)

using 20 numbers we can represent each image with a good accuracy



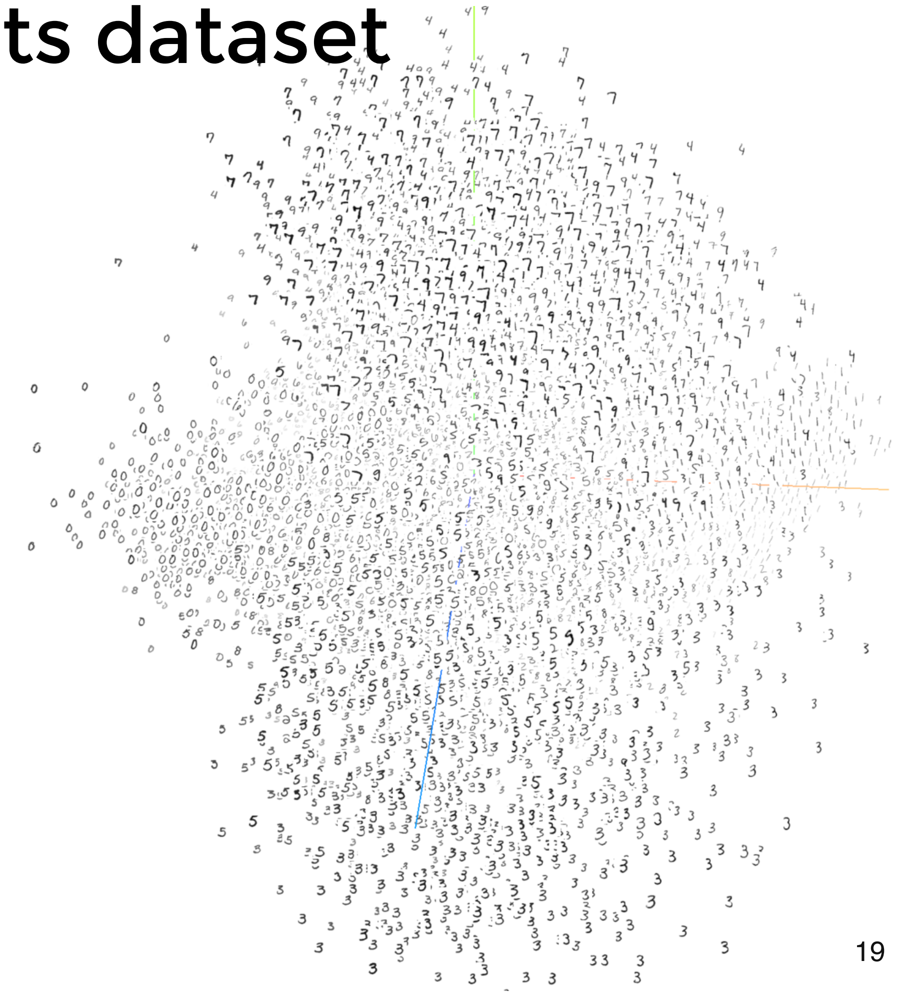
example: digits dataset

3D embedding of MNIST digits
(https://projector.tensorflow.org/)

$x^{(n)} \in \mathbb{R}^{784}$

the embedding 3D coordinates are

$$Xq_1, Xq_2, Xq_3$$

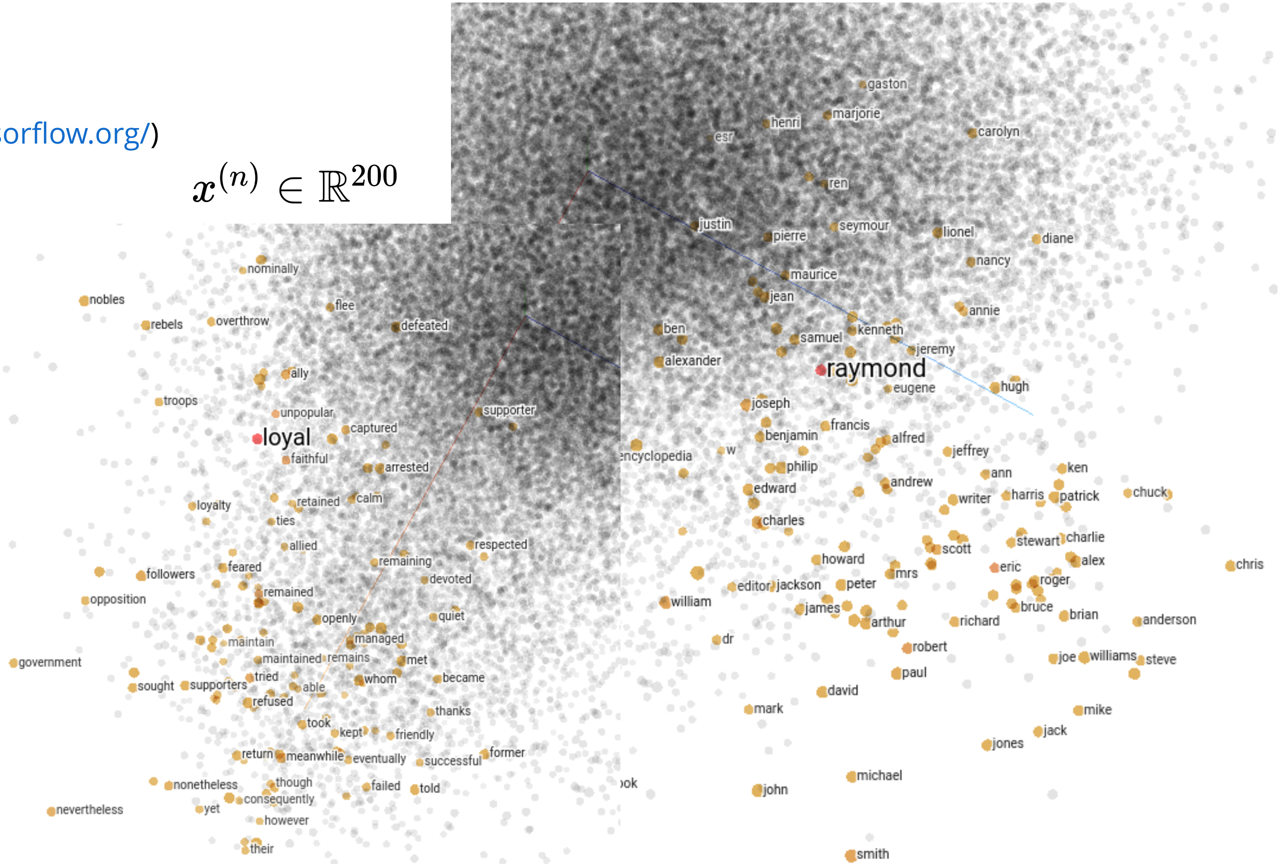


example: text dataset

3D embedding of Word2Vec embeddings (https://projector.tensorflow.org/)

it is common to use dimensionality reduction to visualize and inspect results of other representation learning methods

$x^{(n)} \in \mathbb{R}^{200}$



example: face dataset

eigenfaces for face recognition

read more here

$x^{(n)} \in \mathbb{R}^{64\times 64}$

$x^{(n)}$

$z^{(n)} = {x^{(n)}}^\top Q_{:,:250}$

Figure #9: n_components=250

$q_1, q_2, \ldots q_{15}$

Figure #6: Bunch of ghost shaped images. Look at them in the eyes.

mean face used for centring the data

there is another way to do PCA

without using the covariance matrix

Singular Value Decomposition (SVD)

any N x D real matrix has the following decomposition

$$\underset{N \times D}{X} = \underset{N \times N}{U}\,\underset{N \times D}{S}\,\underset{D \times D}{V}^{\top}$$

orthogonal

$$\begin{bmatrix} | & & | \\ | & \cdots & | \\ u_1 & \cdots & u_N \\ | & \cdots & | \\ | & & | \end{bmatrix}$$

rectangular diagonal

$$\begin{bmatrix} s_1 & & \\ & s_2 & \\ & & \ddots \\ & & \end{bmatrix}$$

orthogonal

$$\begin{bmatrix} | & \cdots & | \\ v_1 & \cdots & v_N \\ | & \cdots & | \end{bmatrix}^{\top}$$

$u_i^{\top} u_j = 0 \forall i \neq j$

$\{u_i\}$ *left singular vectors*

$s_i \geq 0$

singular values

$v_i^{\top} v_j = 0 \forall i \neq j$

right singular vectors

compressed SVD

assuming $N > D$ we can ignore

- the last (N-D) columns of U
- last (N-D) rows of S

why?

similarly if $D > N$ we can compress V, S

$$\underset{N \times D}{X} = \underset{N \times D}{U}\,\underset{D \times D}{S}\,\underset{D \times D}{V}^{\top}$$

Singular value & eigenvalue decomposition

recall that for PCA we used the eigenvalue decomposition of $\Sigma = \frac{1}{N} X^\top X$

how does it relate to SVD?

$$X^\top X = (USV^\top)^\top (USV^\top) = VS^\top U^\top USV^\top = VS^2V^\top$$

compare to $\frac{1}{N} X^\top X = Q \Lambda Q^\top$

$$(X^\top X)^{-1} = VS^{-2}V^\top$$

→ **eigenvectors** of Σ are **right singular vectors** of X $Q = V$

for PCA we could use SVD

- this is the standard computation which works directly with data matrix instead of the covariance matrix optional

Picking the number of PCs

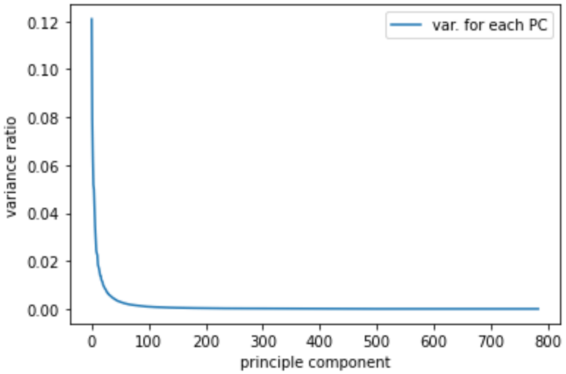
number of PCs in PCA is a hyper-parameter, how should we choose this?

each new principle direction explains some variance in the data $a_d = \frac{1}{N}\sum_n {z_d^{(n)}}^2$

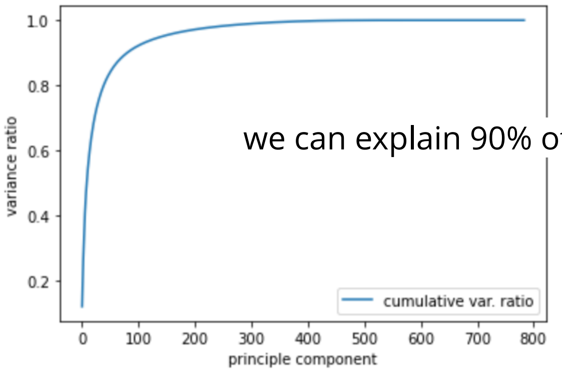
such that we have $a_1 \geq a_2 \geq \ldots \geq a_D$ (by definition of PCA)

we can divide by total variance to get a ratio $r_i = \frac{a_i}{\sum_d a_d}$

example for our digits example we get



sum of variance ratios up to a PC



we can explain 90% of variance in the data using 100 PCs

first few principal directions explain most of the variance in the data!

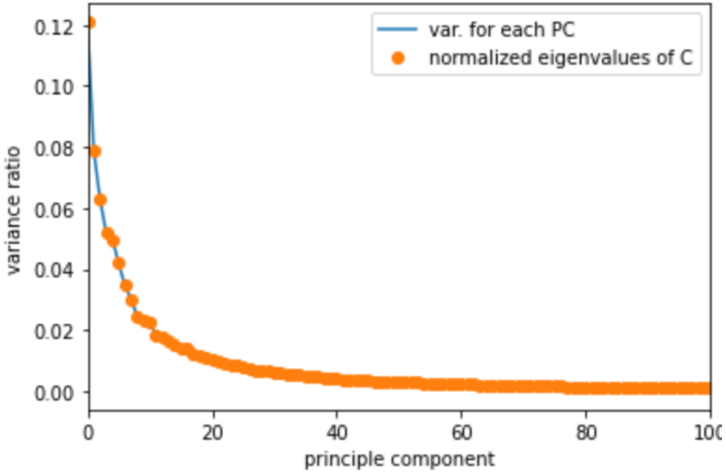
Picking the number of PCs

recall that for picking the principal direction we maximized the variance of the PC

$$\max_{\substack{q \\ ||q||=1}} \frac{1}{N} q X^\top X q^\top \quad = \max_{\substack{q \\ ||q||=1}} q \Sigma q^\top \quad = \max_{\substack{q_1 \\ ||q||=1}} q^\top Q \Lambda Q^\top q = \lambda_1$$

so the variance ratios are also given by $r_i = \frac{\lambda_i}{\sum_d \lambda_d}$

so we can also use eigenvalues to pick the number of PCs



digits **example**:

two estimates of variance ratios do match

Matrix factorization

PCA and SVD perform matrix factorization

$$X \approx (\underbrace{XQ}_{N \times D'})\underbrace{Q^\top}_{D' \times D}$$

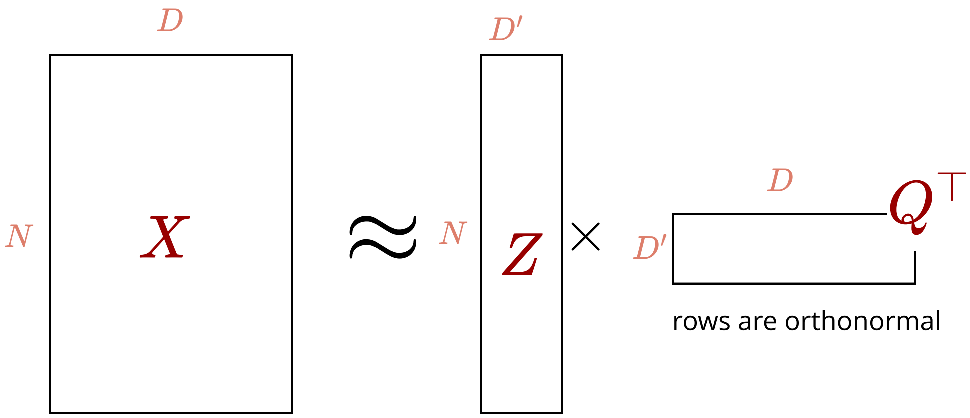
$Q^\top$: rows of this matrix are principal components

factor matrix

Z ($N \times D'$): this is the matrix of low-dimensional features

pc coefficients

factor loading matrix

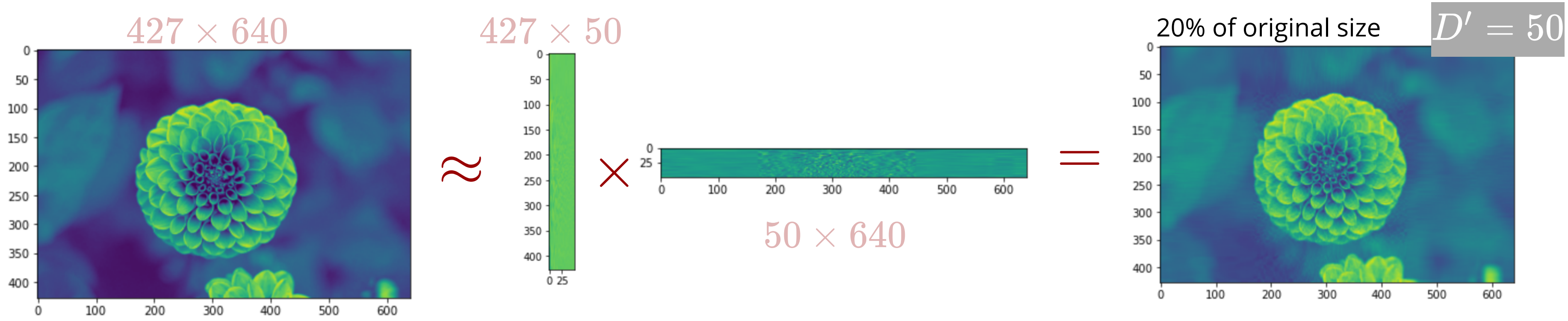


this gives a row-rank approximation to our original matrix X

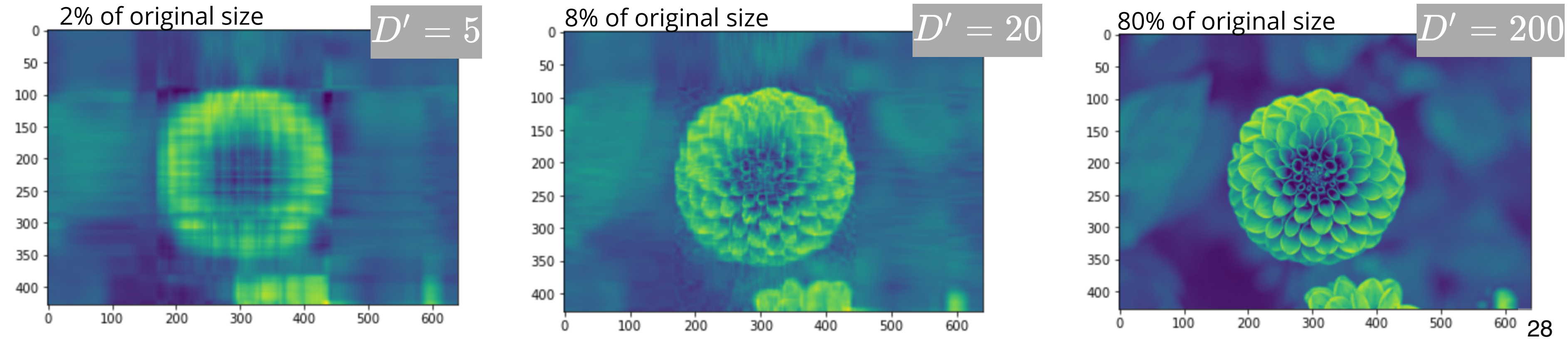
- we can use this to compress the matrix
- we can find give a "smooth" reconstruction of X (remove noise or fill missing values)

example

Matrix factorization



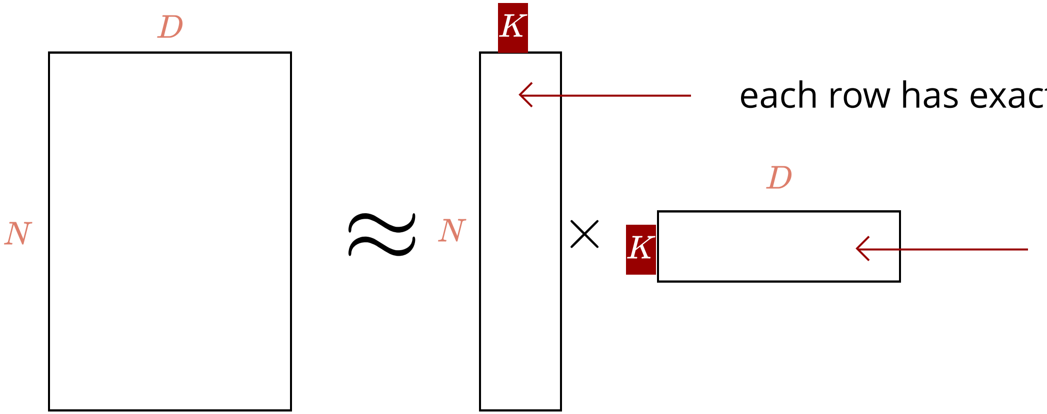
changing the rank D' gives different amount of compression



Matrix factorization

relationship to **K-means**

K-means also can be seen as matrix factorization



each row has exactly one nonzero (*responsibilities*) , e.g., [0,1,0,0,0]

each row is a cluster center μ_k

matrix product simply equates each row of X with one row of the factor matrix

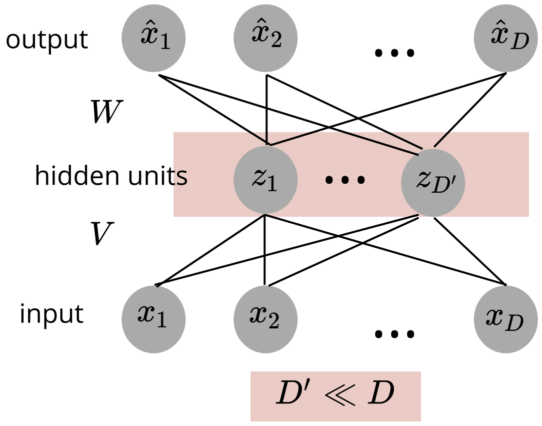
- instead of principal components → cluster centers
- factor loading matrix → one nonzero per row of Z (each node belongs to one cluster)

Autoencoders

a feed-forward neural net which predicts its input

- can be trained with reconstruction loss
  - e.g. mean squared error: $\sum_n ||x^{(n)} - \hat{x}^{(n)}||_2^2$

dimensionality reduction with **a bottleneck layer**
much smaller than input



Autoencoders

a feed-forward neural net which predicts its input

- can be trained with reconstruction loss
  - e.g. reconstruction loss: $||x - \psi(\phi(x))||_2^2$ (where $\psi(\phi(x)) = \hat{x}$)

dimensionality reduction with **a bottleneck layer** (much smaller than input)

- optimal weights for **linear** autoencoder are the principal components
- **nonlinear** dimensionality reduction if activations are not all linear
  - projecting the data on a non-linear manifold
  - deep autoencoders are very powerful

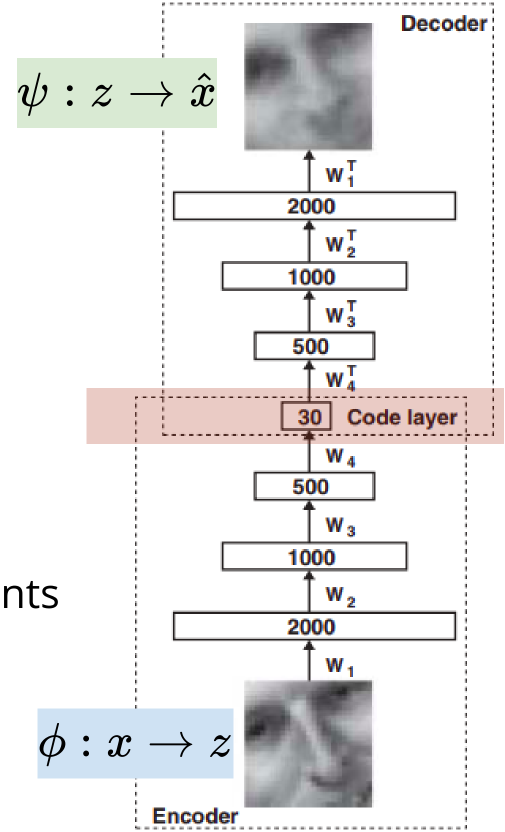
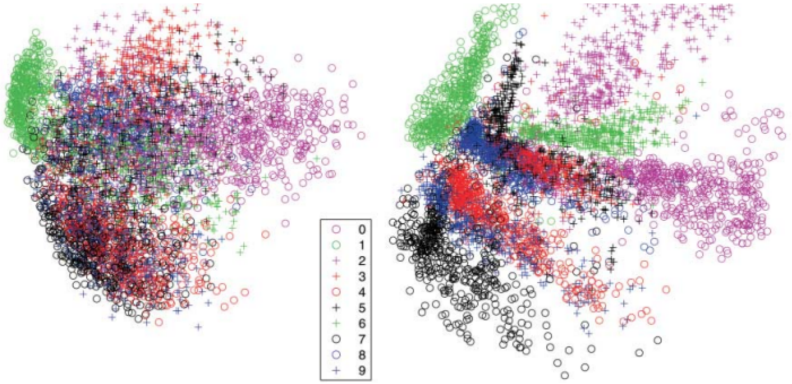


image form:
https://www.cs.toronto.edu/~hinton/science.pdf

Autoencoders: example

MNIST digits



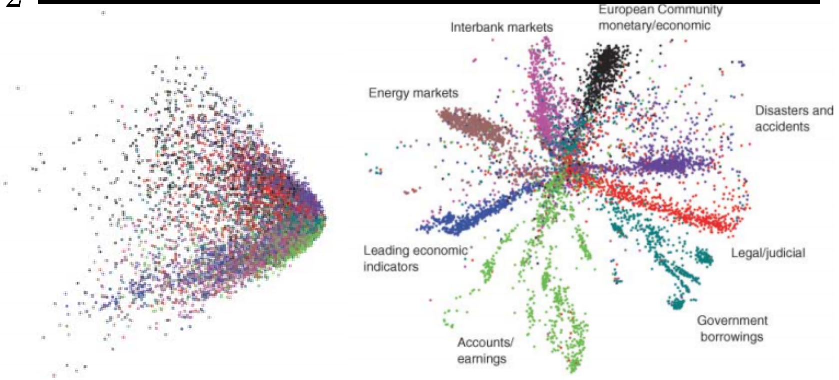
$\sum_n ||x^{(n)} - x^{(n)\top} QQ^\top||_2^2$

$s.t. \quad Q^\top Q = I$

PCA v.s. Autoencoder

$\sum_n ||x^{(n)} - \psi(\phi(x^{(n)}))||_2^2$

newswire stories



read the paper here

Summary

Dimensionality reduction helps us:

- visualize our data
- compress it
- simplify the computational need of further analysis (clustering, supervised learning etc.)
- also can be used for anomaly detection (not discussed)

PCA is a linear dimensionality reduction method

- projects the data to a linear space (spanned by D' principal directions)
  - directions are eigenvectors of the covariance matrix
  - the projection has maximum variance (minimum reconstruction error)
  - eigenvalues tell us about the contribution of each new principal direction
- PCA using Singular Value Decomposition
- Model selection for PCA
- PCA as matrix factorization and its relationship to k-means
- practical note: don't forget to subtract the mean!