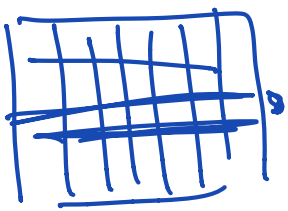
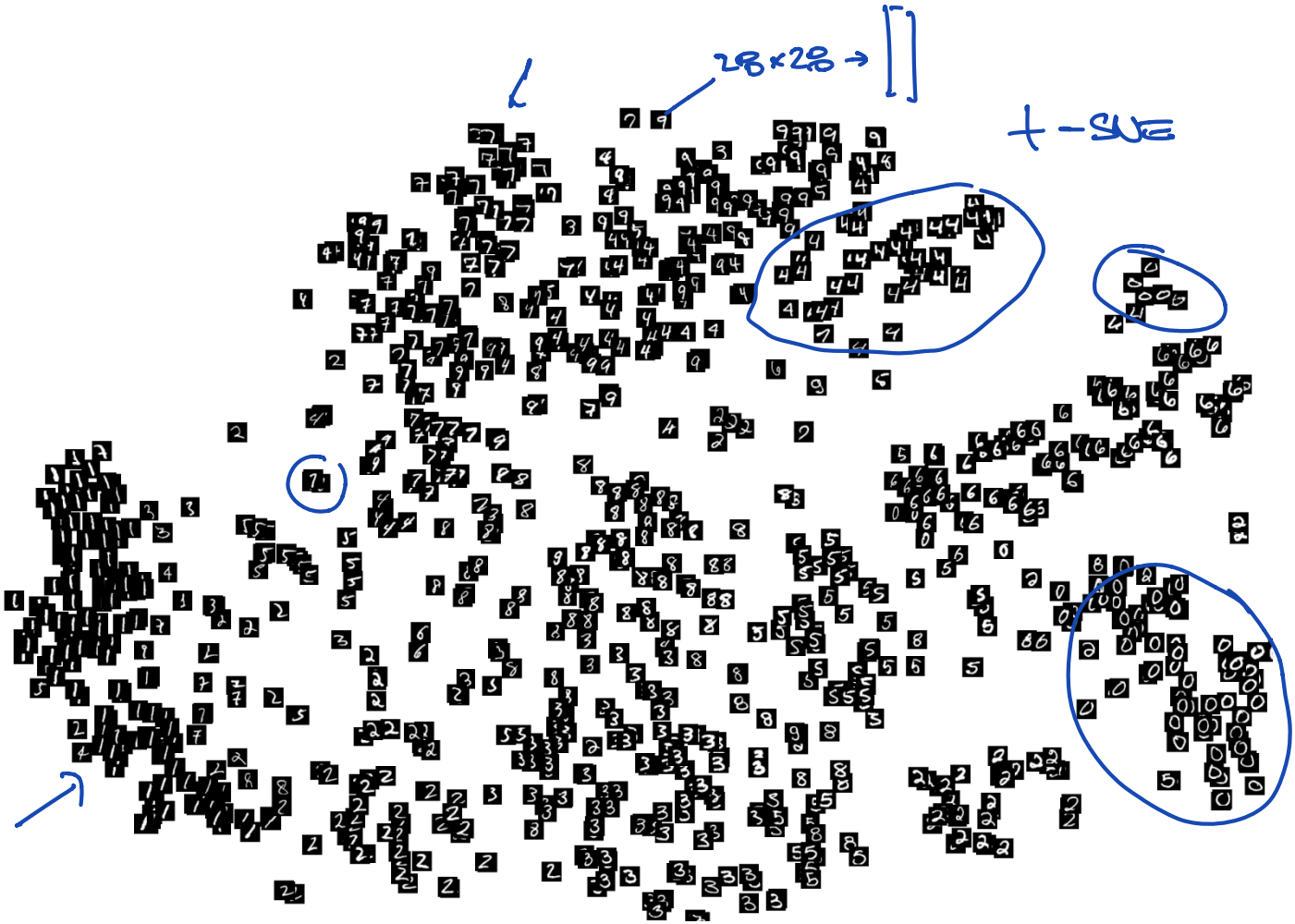
Unsupervised Learning (2)

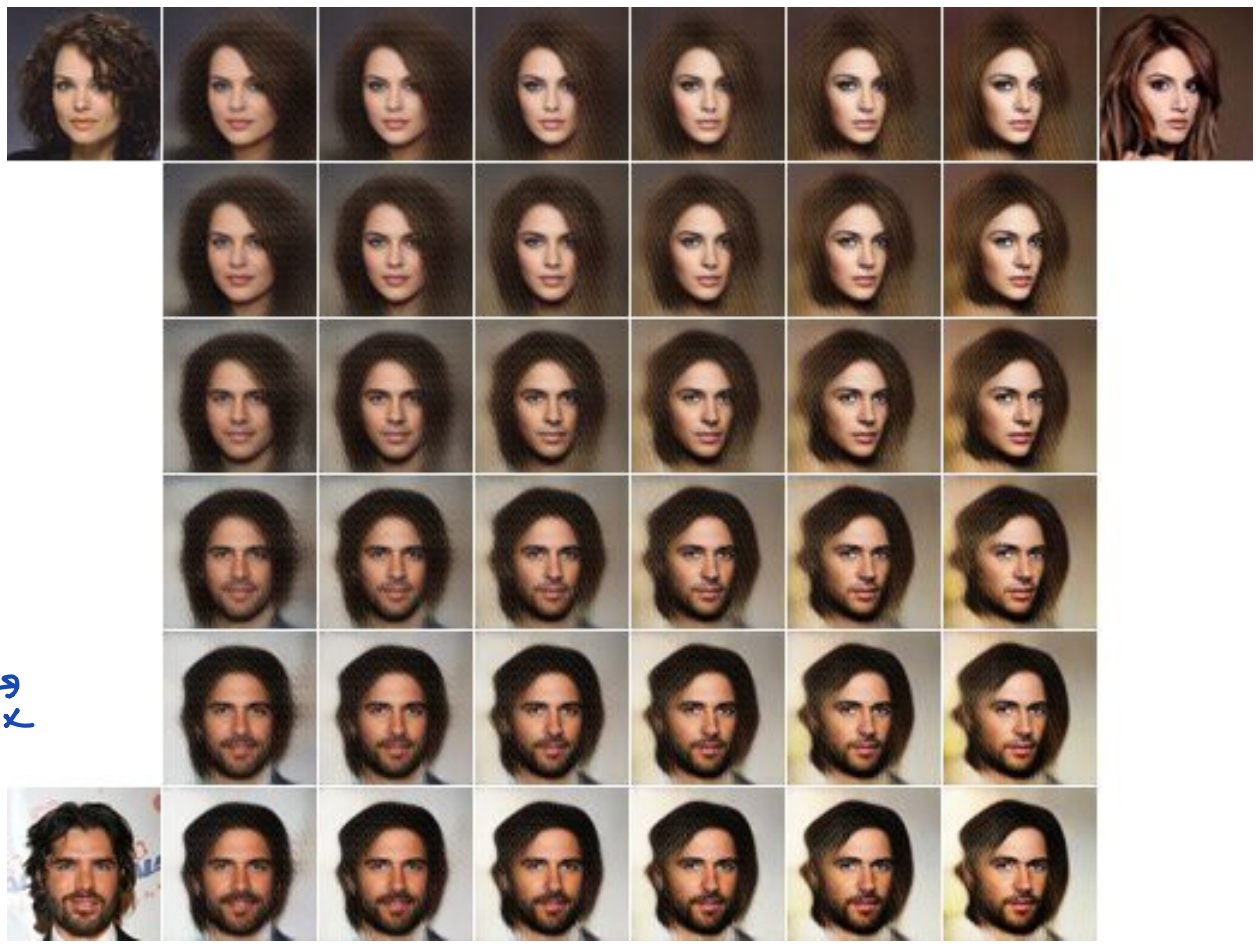
Machine Learning for Data Science 1

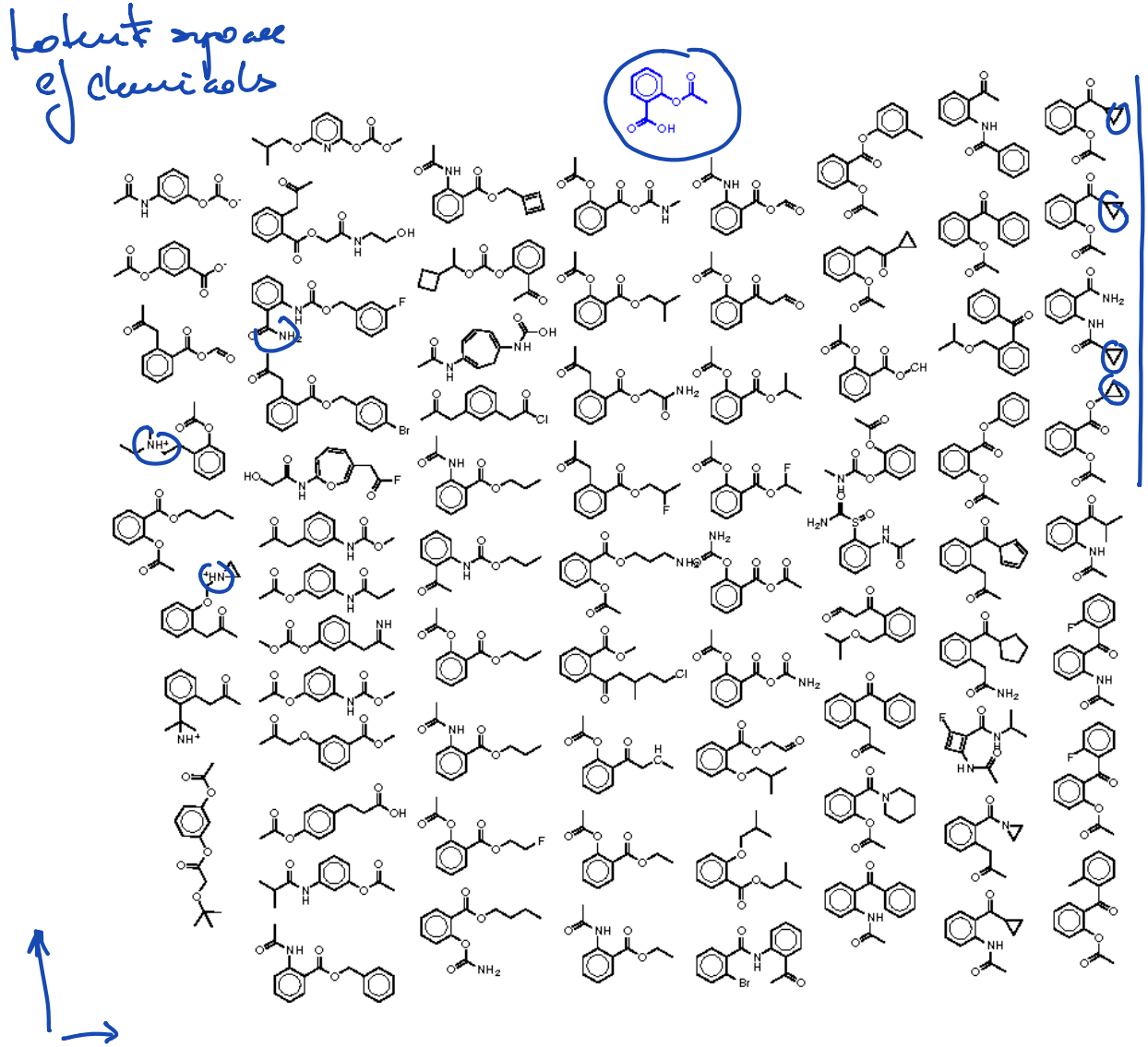
Latent spaces : representation of compressed data in which similar data points (from our view (space) reside together



10,100
1000 ..


28×28 →
+-SVE



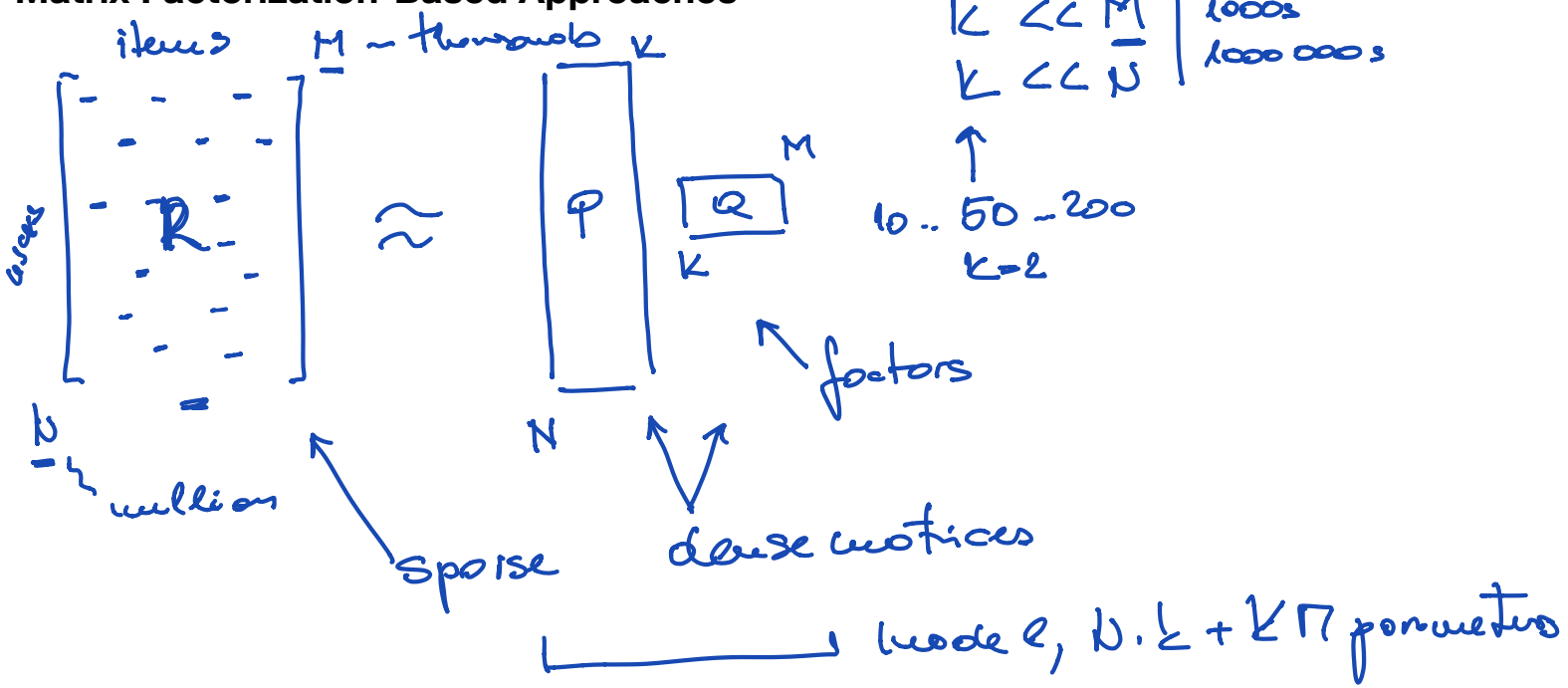

latent space
of chemicals

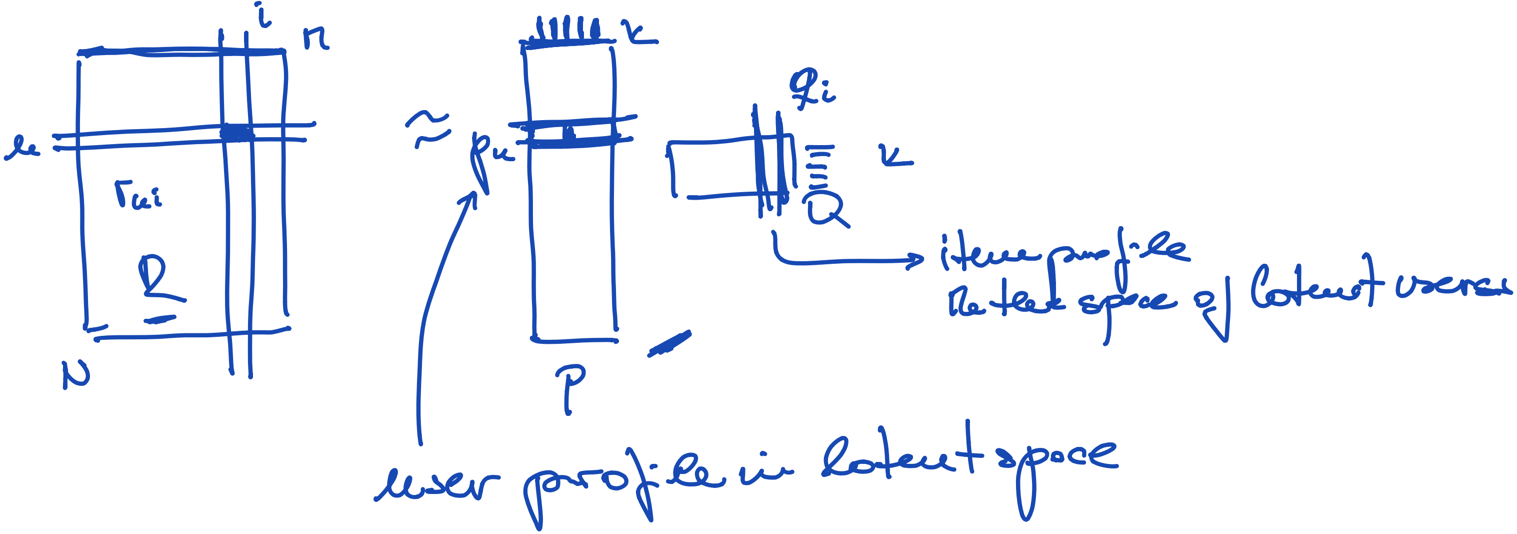
PCA
MDS
t-SNE

dimensionality reduction
construct latent spaces

PCA — projection
MDS, t-SNE — embedding

Sparse Data,
Matrix Factorization-Based Approaches





$$r_{ui} \approx p_u^T q_i \qquad \hat{r}_{ui} \approx r_{ui}$$

$$\hat{r}_{ui} = p_u^T q_i = \sum_{k=1}^{K} p_{uk} \cdot q_{ki}$$

$$e_{ui}^2 = (r_{ui} - \hat{r}_{ui})^2$$

$$= \left(r_{ui} - \sum_{k} p_{uk} \cdot q_{ki}\right)^2 \quad \Big| \text{ minimize}$$

$\rightarrow J$

gradient (stochastic gradient descent)

$$\frac{\partial e_{ui}^2}{\partial p_{uk}} = -2(r_{ui} - \hat{r}_{ui}) q_{ki} \qquad \text{for } k = 1 .. K$$

$$= -2\, e_{ui} \cdot q_{ki}$$

$$\frac{\partial e_{ui}^2}{\partial q_{ki}} = -2(r_{ui} - \hat{r}_{ui}) p_{uk}$$

$$= -2\, e_{ui}\, p_{uk}$$

update rules

$$p_{uk} \leftarrow p_{uk} - \alpha \frac{\partial e_{ui}^2}{\partial p_{uk}} =$$

$$= p_{uk} + \alpha\, e_{ui}\, q_{ki} \quad + \lambda p_{uk}$$

$$q_{ki} \leftarrow q_{ki} + \alpha\, e_{ui}\, p_{uk} \quad + \lambda q_{ki}$$

for $k = 1 .. K$

regularization $\|P\|_F, \|Q\|_F$ $J = e_{ui}^2 + \lambda p_u^T p_u + \lambda q_i^T q_i$

initialization $P, Q : [0, 0.001]$ stopping criteria RMSE

Input: $\mathcal{T}'$: training set, η: learning rate, λ: regularization factor
Output: $\mathbf{P}^*$,$\mathbf{Q}^*$: the user and item feature matrices

Takacs et al JMLR 2009

```
1  Partition T' into two sets: T'_I, T'_II (validation set)
2  Initialize P and Q with small random numbers.
3  loop until the terminal condition is met. One epoch:
4      iterate over each (u, i, r_ui) element of T'_I:
5          compute e'_ui;
6          compute the gradient of e'_ui, according to Eq. (6);
7          for each k
8              update p_u, the u-th row of P,
9              and q_i, the i-th column of Q according to Eq. (7);
10     calculate the RMSE on T'_II;
11     if the RMSE on T'_II was better than in any previous epoch:
12         Let P* = P and Q* = Q.
13     terminal condition: RMSE on T'_II does not decrease during two epochs.
14 end
```

Algorithm 1: Training algorithm for RISMF

oct 2006

cinematch

480 000 users

18 000 movies

100 480 000 ratings

10%, beat cinematch

june 2009

march 12, 2010

concluded by Federal Trade Commission

privacy concerns

Non-Negative Matrix Factorization

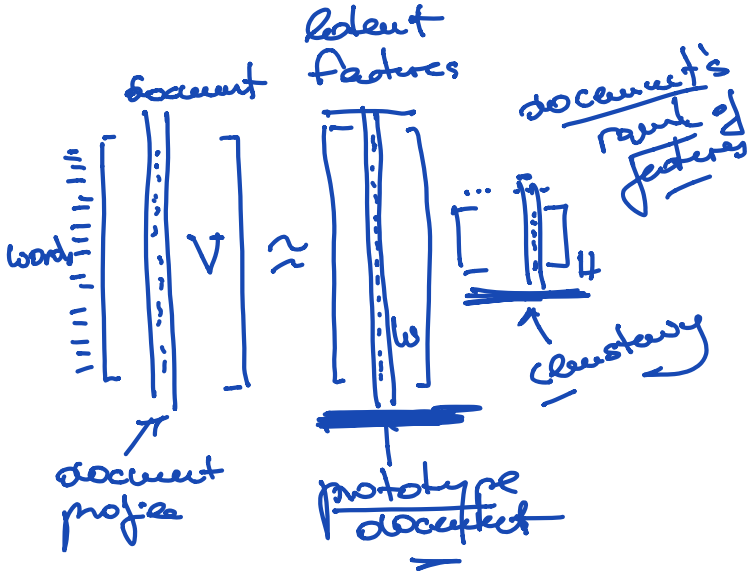
letters to nature

Learning the parts of objects by non-negative matrix factorization

Daniel D. Lee* **& H. Sebastian Seung*†**

** Bell Laboratories, Lucent Technologies, Murray Hill, New Jersey 07974, USA*
† Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, USA

Is perception of the whole based on perception of its parts? There is psychological[1] and physiological[2,3] evidence for parts-based representations in the brain, and certain computational theories of object recognition rely on such representations[4,5]. But little is known about how brains or computers might learn the parts of objects. Here we demonstrate an algorithm for non-negative matrix factorization that is able to learn parts of faces and semantic features of text. This is in contrast to other methods, such as principal components analysis and vector quantization, that learn holistic, not parts-based, representations. Non-negative matrix factorization is distinguished from the other methods by its use of non-negativity constraints. These constraints lead to a parts-based representation because they allow only additive, not subtractive, combinations. When non-negative matrix factorization is implemented as a neural network, parts-based representations emerge by virtue of two properties: the firing rates of neurons are never negative and synaptic strengths do not change sign.



with $\|V - WH\|_F$

subj. to $W \geq 0$, $V \geq 0$

Nature 1999

multiplicative update rules

$V \sim WH$ ← model
W, H ← latent matrices

$$H_{[ij]} \leftarrow H_{[ij]} \cdot \frac{W^T V_{[ij]}}{W^T W H_{[ij]}}$$

$$W_{[ij]} \leftarrow W_{[ij]} \cdot \frac{V H^T_{[ij]}}{W H H^T_{[ij]}}$$

initialize W, H
repeat
update rules
until convergence
validation
metric

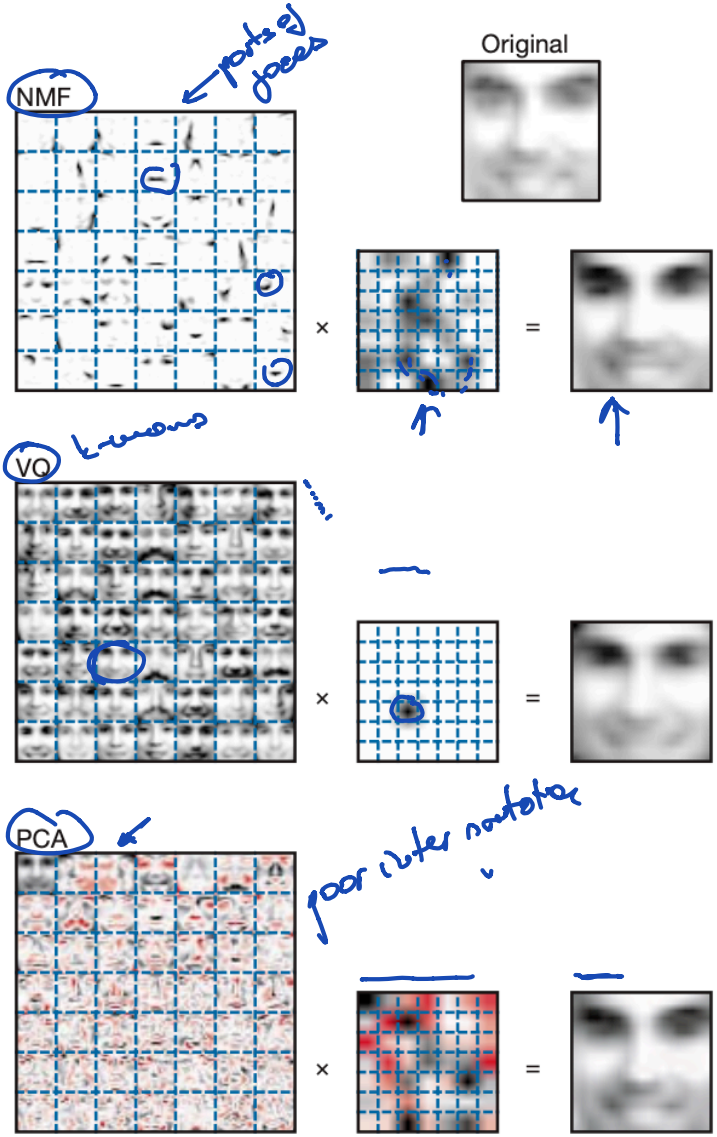
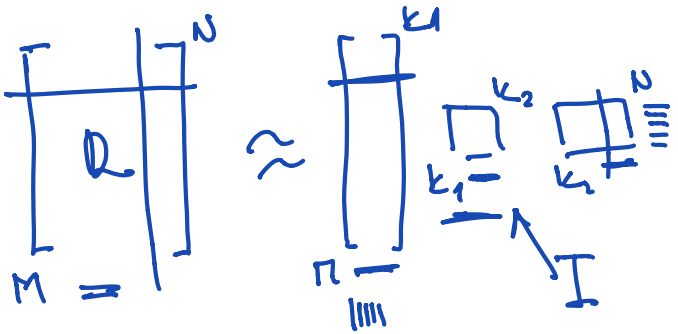


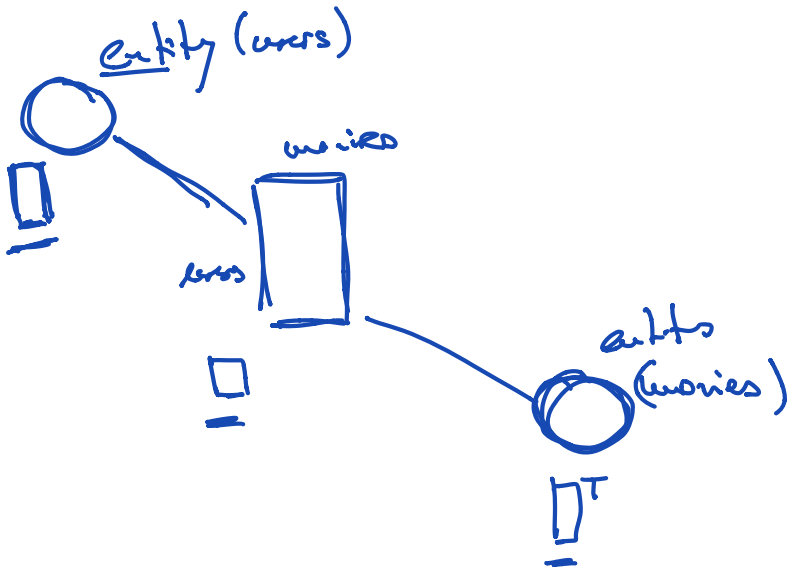
Figure 1 Non-negative matrix factorization (NMF) learns a parts-based representation of faces, whereas vector quantization (VQ) and principal components analysis (PCA) learn holistic representations. The three learning methods were applied to a database of $m = 2{,}429$ facial images, each consisting of $n = 19 \times 19$ pixels, and constituting an $n \times m$ matrix V. All three find approximate factorizations of the form $V \approx WH$, but with three different types of constraints on W and H, as described more fully in the main text and methods. As shown in the 7×7 montages, each method has learned a set of $r = 49$ basis images. Positive values are illustrated with black pixels and negative values with red pixels. A particular instance of a face, shown at top right, is approximately represented by a linear superposition of basis images. The coefficients of the linear superposition are shown next to each montage, in a 7×7 grid, and the resulting superpositions are shown on the other side of the equality sign. Unlike VQ and PCA, NMF learns to represent faces with a set of basis images resembling parts of faces.

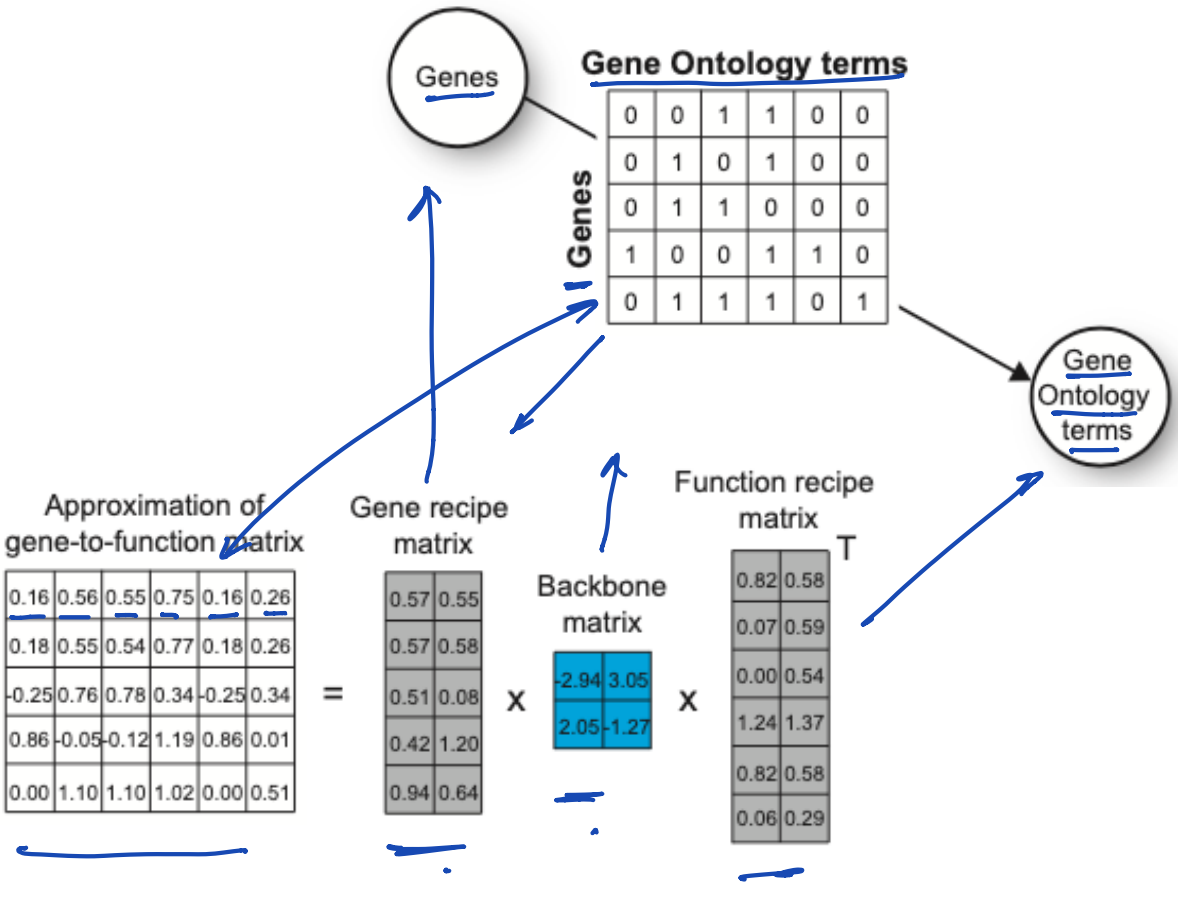
Matrix Tri-Factorization

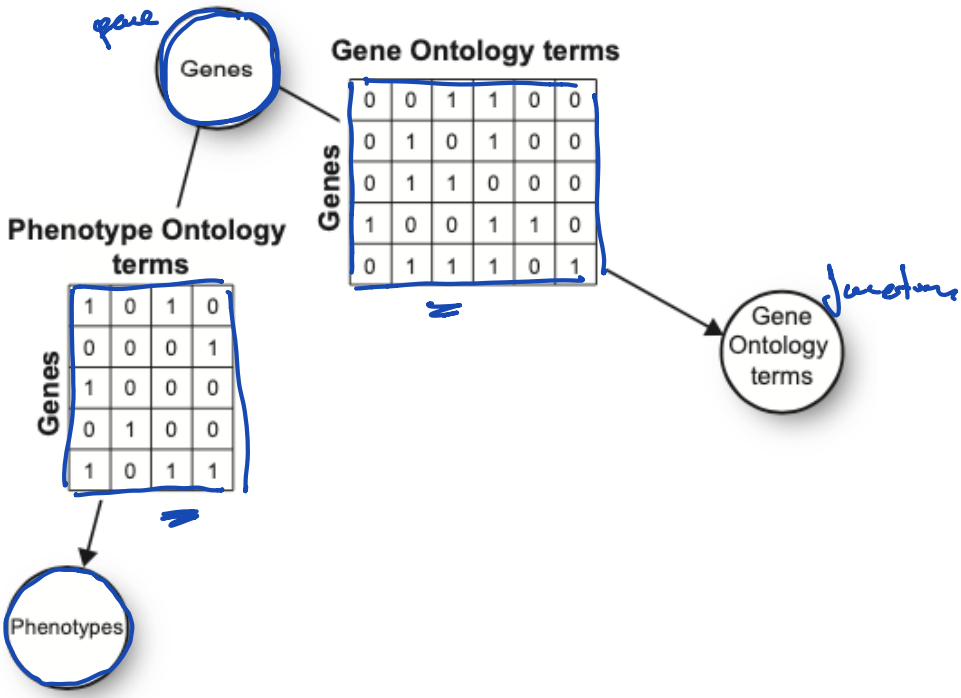


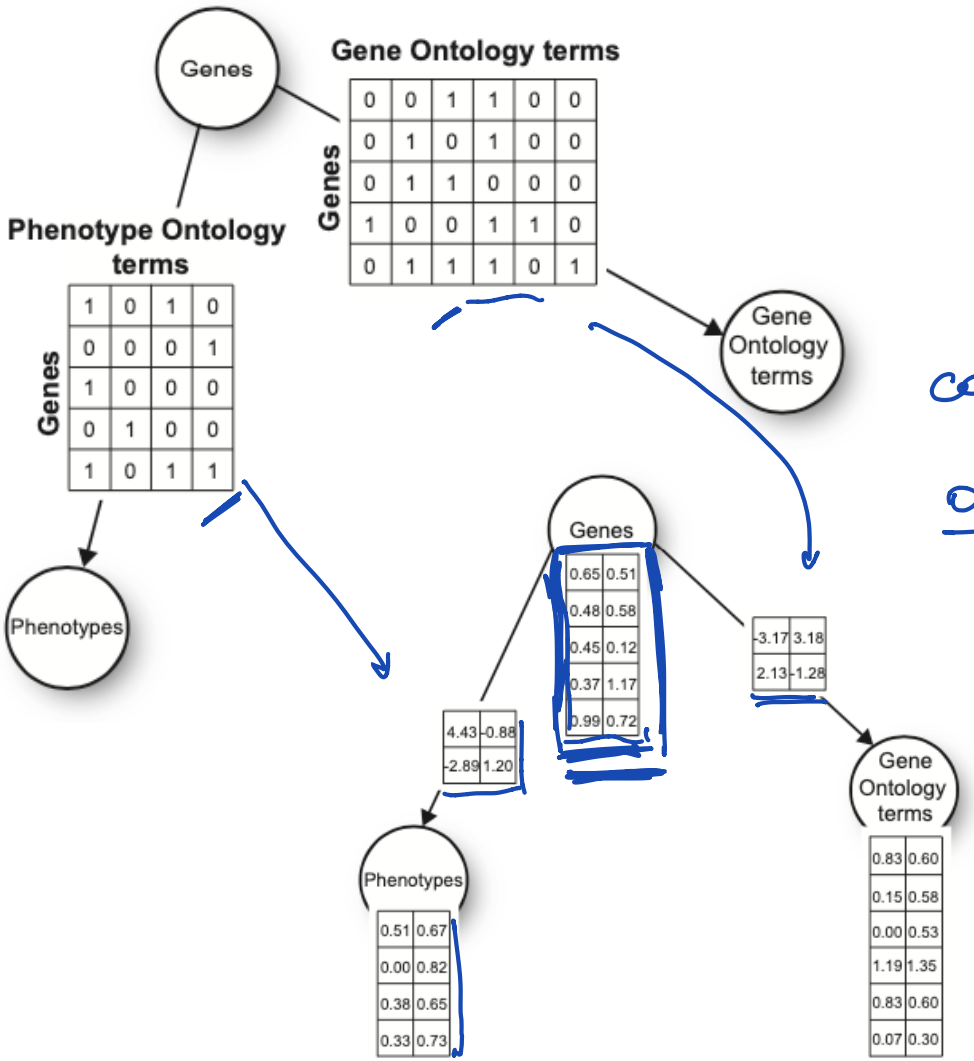
3 latent factors

$K_1 = K_2$



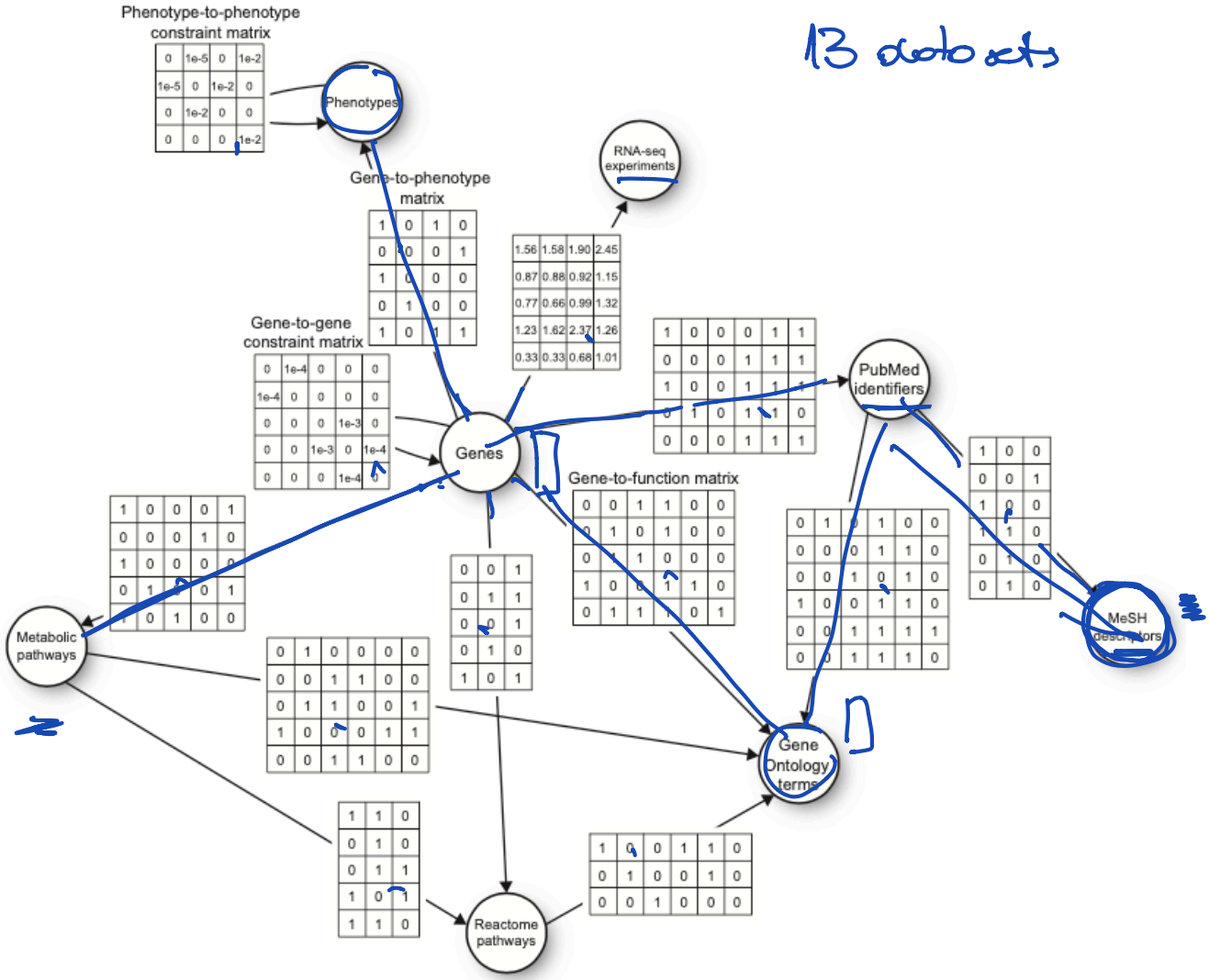

Genes
Gene Ontology terms
Genes
0 0 1 1 0 0
0 1 0 1 0 0
0 1 1 0 0 0
1 0 0 1 1 0
0 1 1 1 0 1
Gene Ontology terms
Approximation of gene-to-function matrix
0.16 0.56 0.55 0.75 0.16 0.26
0.18 0.55 0.54 0.77 0.18 0.26
-0.25 0.76 0.78 0.34 -0.25 0.34
0.86 -0.05 -0.12 1.19 0.86 0.01
0.00 1.10 1.10 1.02 0.00 0.51
=
Gene recipe matrix
0.57 0.55
0.57 0.58
0.51 0.08
0.42 1.20
0.94 0.64
x
Backbone matrix
-2.94 3.05
2.05 -1.27
x
Function recipe matrix
T
0.82 0.58
0.07 0.59
0.00 0.54
1.24 1.37
0.82 0.58
0.06 0.29


Genes
Gene Ontology terms
Genes
0 0 1 1 0 0
0 1 0 1 0 0
0 1 1 0 0 0
1 0 0 1 1 0
0 1 1 1 0 1
Phenotype Ontology terms
Genes
1 0 1 0
0 0 0 1
1 0 0 0
0 1 0 0
1 0 1 1
Gene Ontology terms
Phenotypes

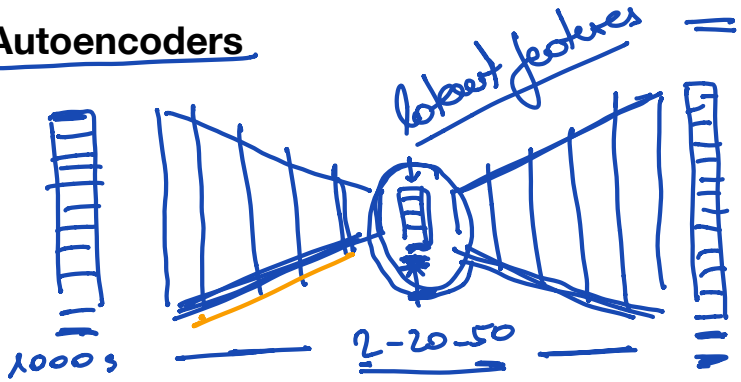


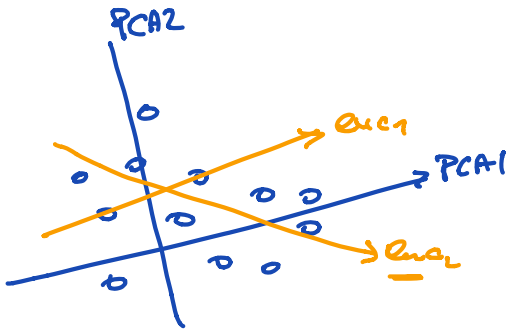
collective matr. factorization

data fusion


13 datasets
Phenotype-to-phenotype constraint matrix
Phenotypes
Gene-to-phenotype matrix
RNA-seq experiments
Gene-to-gene constraint matrix
Genes
PubMed identifiers
Gene-to-function matrix
Metabolic pathways
MeSH descriptors
Gene Ontology terms
Reactome pathways

Autoencoders





x $\quad e(x)$ $\quad d(e(x))$

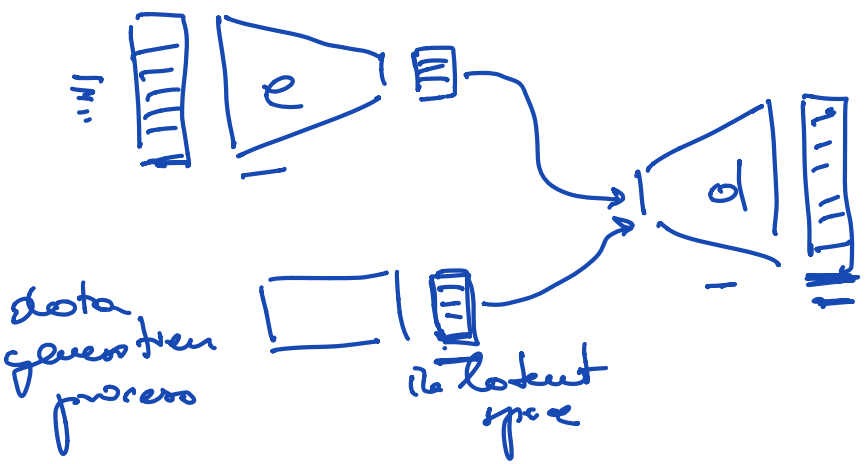
$\mathbb{R}^N$ $\quad \mathbb{R}^m$ $\quad \mathbb{R}^N$ $\quad m \ll N$

encoder $\quad$ decoder

$$\min \| X - d(e(X)) \|_F$$

Variational Autoencoders

- no regularity of latent space
- generate real data



- training is regularized
- latent space has "nice" properties in terms of density
  - ↳ enables generative process

autoencoder (deterministic)

$x \xrightarrow{\text{enc}} z = e(x) \xrightarrow{\text{dec}} d(z)$

input

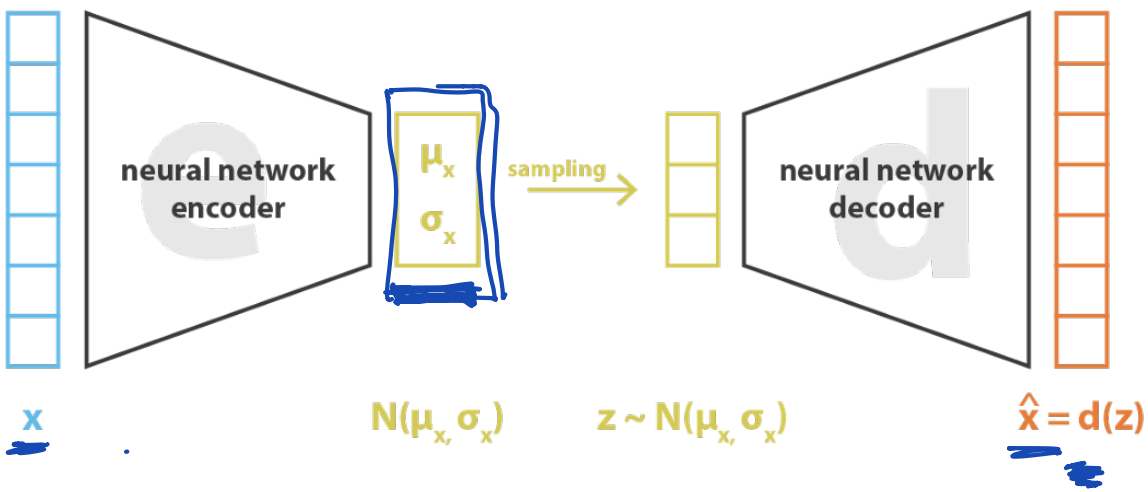
variational autoencoder (probabilistic)

$x \xrightarrow{\text{enc}} p(z|x) \xrightarrow{\text{sample}} z \sim p(z|x) \xrightarrow{\text{decod}} \mathbb{E}[z]$

input — latent distribution — sampled latent representation — input reconstruction

in practice: normal distributions

encoder is trained to return the mean and cov. matr.



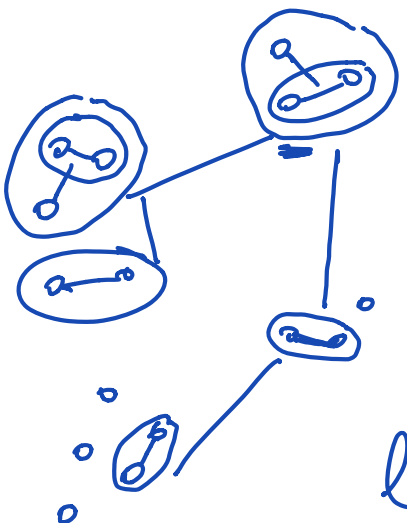
$$\text{loss} = \|x - \hat{x}\|^2 + KL[N(\mu_x, \sigma_x), N(0, I)] = \|x - d(z)\|^2 + KL[N(\mu_x, \sigma_x), N(0, I)]$$

Clustering

grouping of objects

objects in the same group are more similar to each other than to objects in other groups

Connectivity-Based Clustering → hierarchical clustering



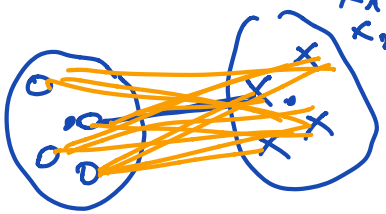
similarity

distance

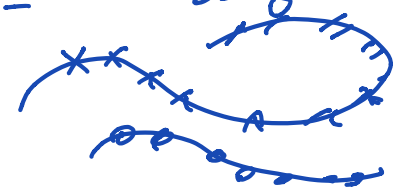
distances between the objects

linkage

$D(c_1, c_2) = \min D(x_1, x_2)$ $x_1 \in c_1$, $x_2 \in c_2$



Single linkage



$D(c_1, c_2) = \max D(x_1, x_2)$ $x_1 \in c_1$, $x_2 \in c_2$

Complete linkage

$D(c_1, c_2) = \frac{1}{|c_1||c_2|} \sum_{x_1 \in c_1} \sum_{x_2 \in c_2} D(x_1, x_2)$ Average linkage

$D(c_1, c_2) = \sum_{x \in c_1 \cup c_2} D(x, \mu_{c_1 \cup c_2})$

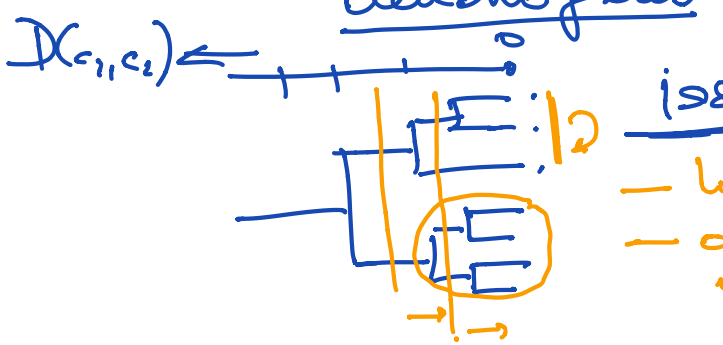
Ward linkage

minimizes the variance within the clusters

Hierarchical clustering

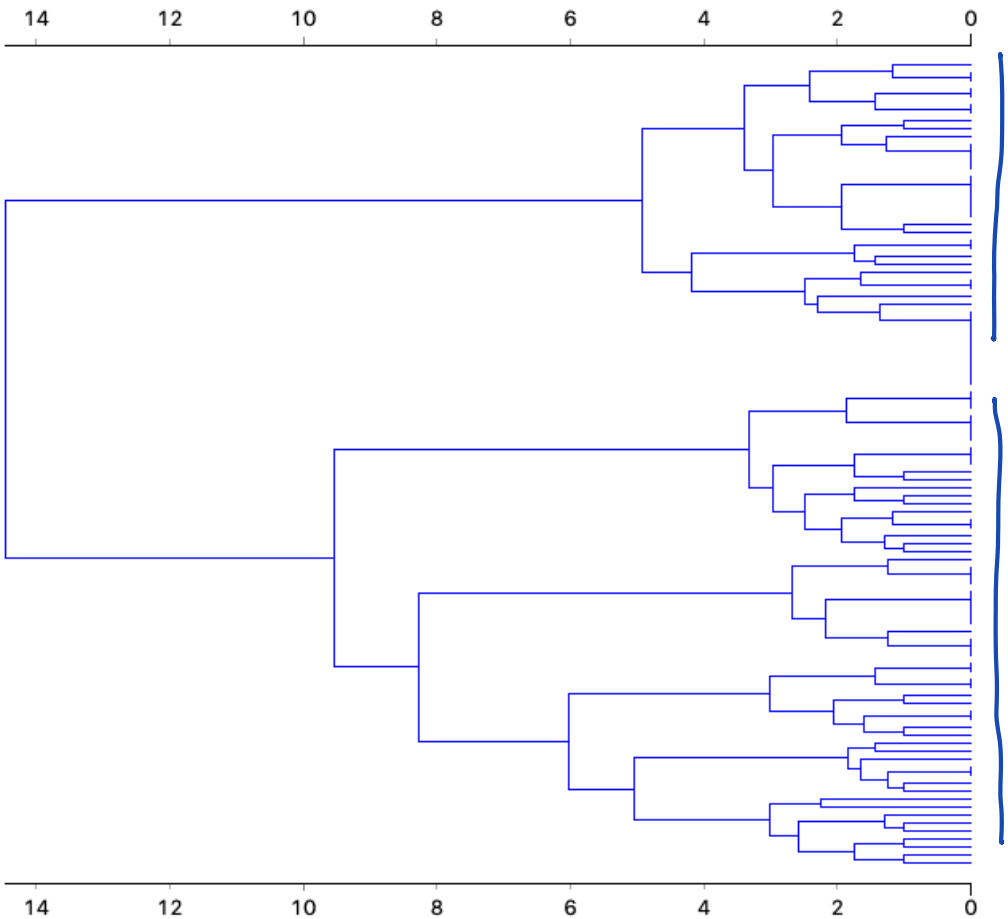
- each object is its own cluster
- merge two closest clusters
- repeat, until just 1 cluster

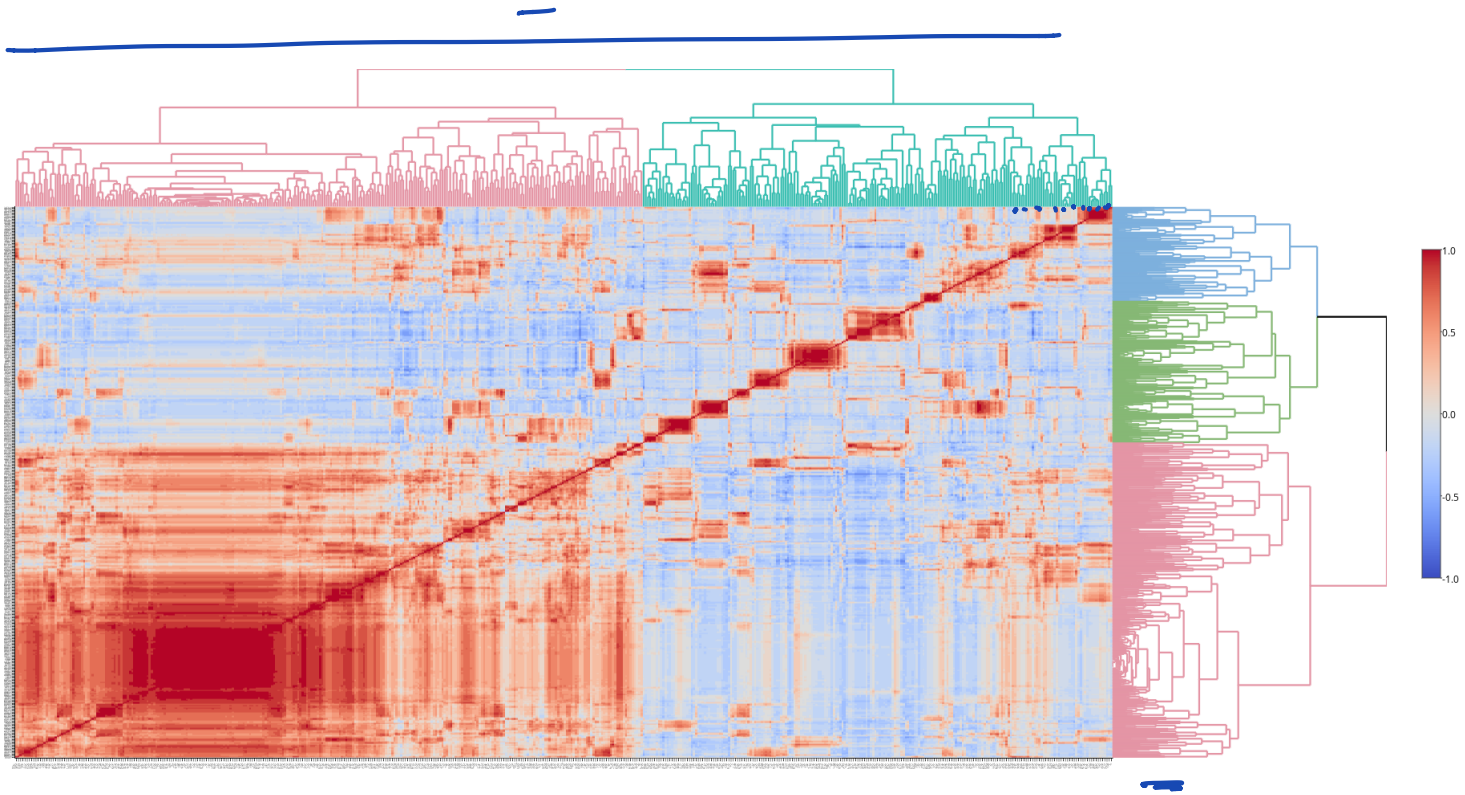
dendrogram

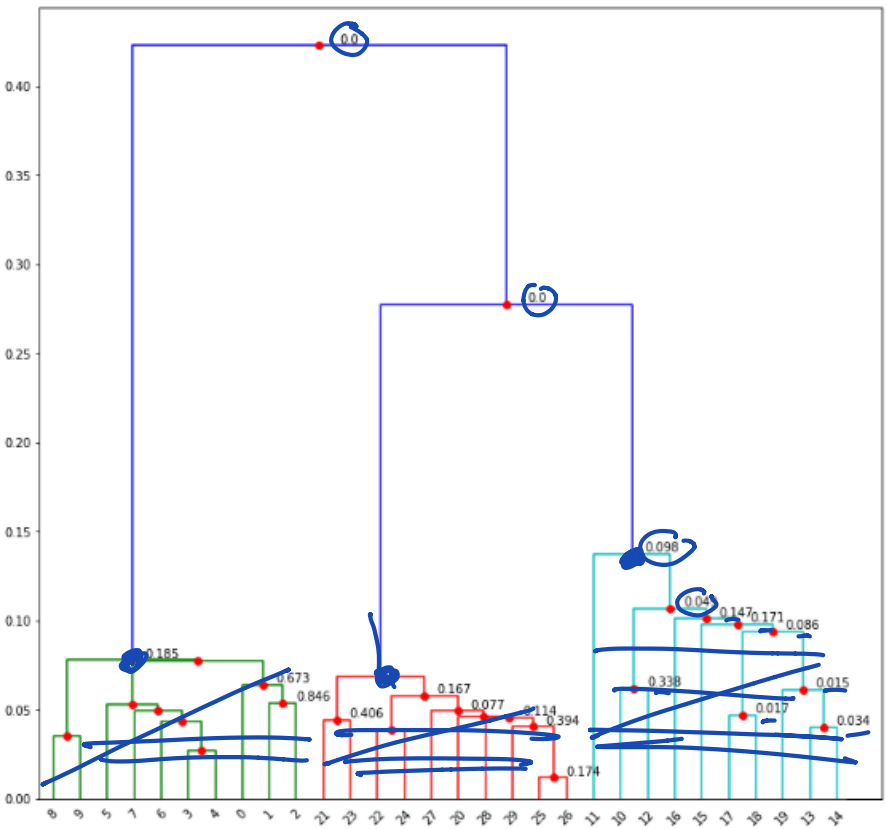


issues

- how many clusters
- are these clusters "significant"
- branch ordering


14
12
10
8
6
4
2
0
14
12
10
8
6
4
2
0


1.0
0.5
0.0
-0.5
-1.0


0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0.0
0.0
0.098
0.04
0.147
0.171
0.086
0.185
0.673
0.846
0.167
0.406
0.077
0.114
0.394
0.174
0.338
0.015
0.017
0.034
8 9 5 7 6 3 4 0 1 2 21 23 22 24 27 20 28 29 25 26 11 10 12 16 15 17 18 19 13 14

Centroid-Based Clustering

k-means clustering

For k clusters, find cluster members so that to minimize the distance to the cluster prototype (centroid).

NP-hard.

↳ k-means clustering

- choose k centroids, $m^{(i)}$ (randomly, heuristics)
- repeat
  - assign objects to the nearest centroid
  - move centroid to the center of the assigned objects

until no change in cluster assignments

goals

low complexity, k is low

speed, fast convergence

problems

- depends on initialization
  K-MEANS++
- Do ...
- K defined

- clusters
  needs to be
  "optimal"

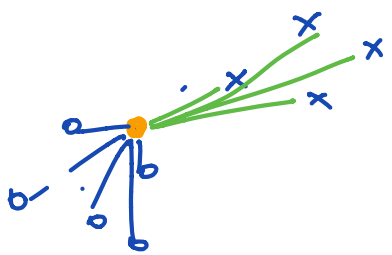
how many clusters do we have?

Assessment of Cluster Quality

complexity $\longleftrightarrow$ cluster quality

λ

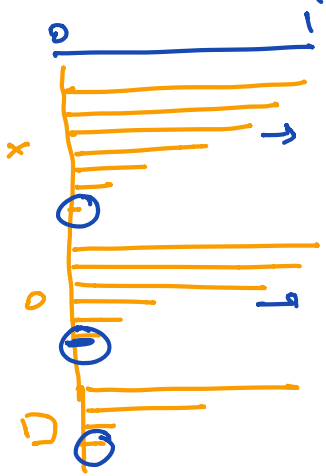
the silhouette



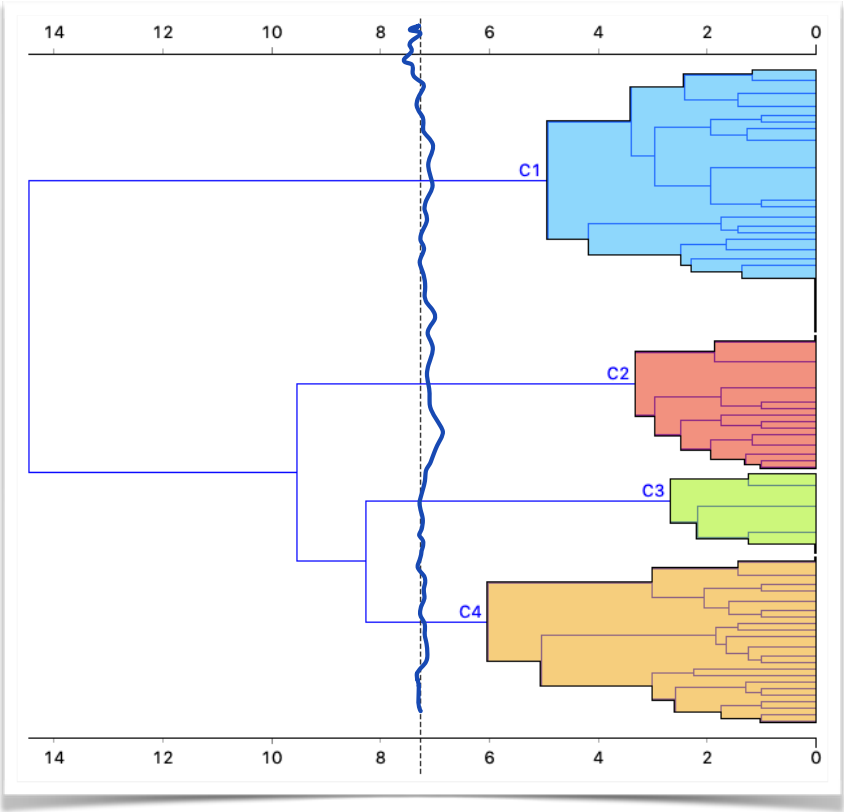
a : within cluster av. dist

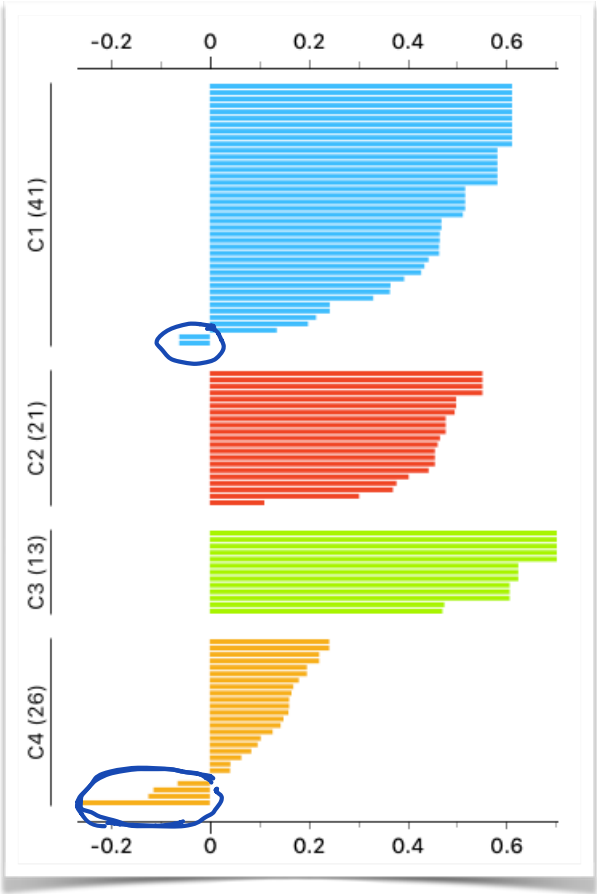
b : dist. to closest cluster

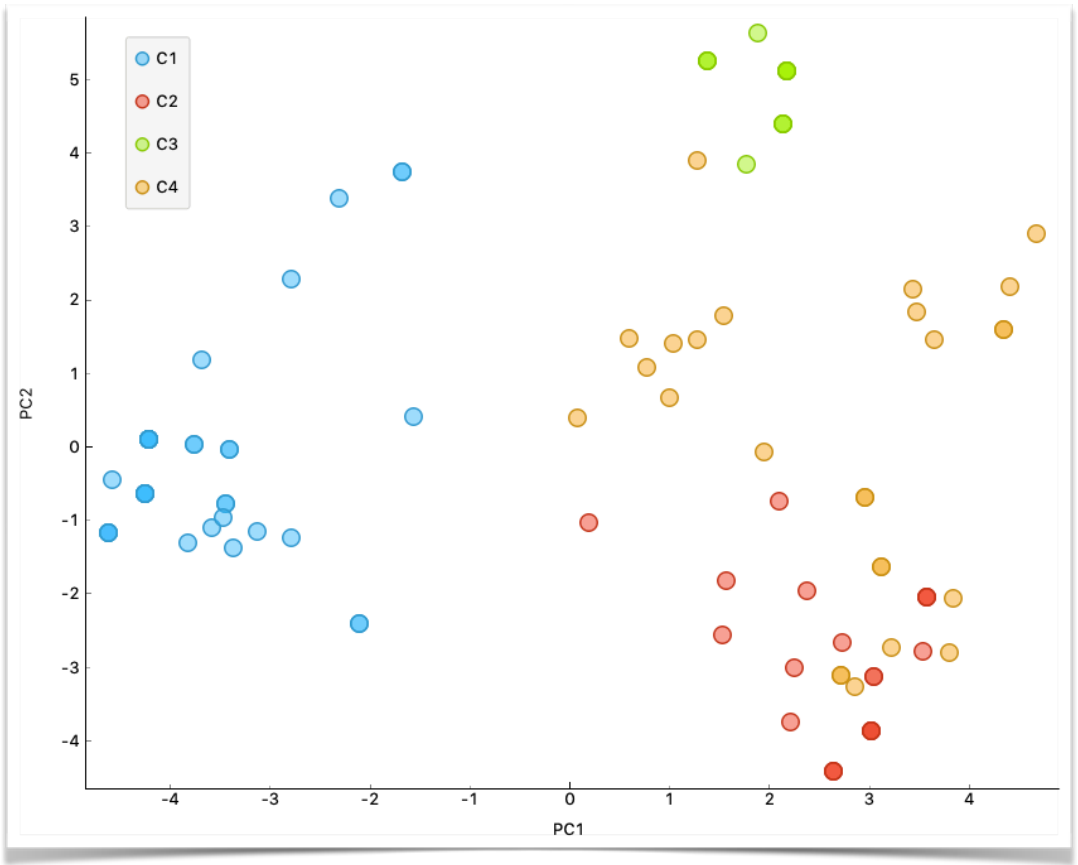
$$s = \frac{b - a}{\max(b, a)} \in [-1, 1] \quad [0, 1]$$

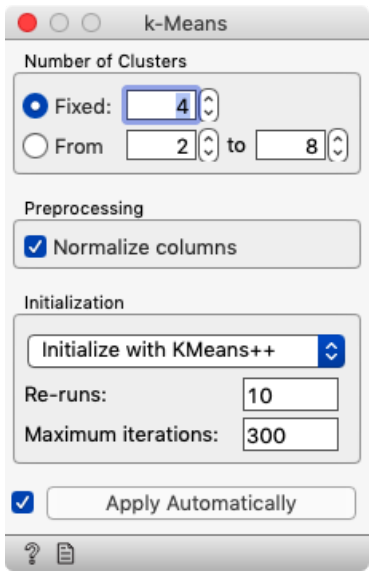


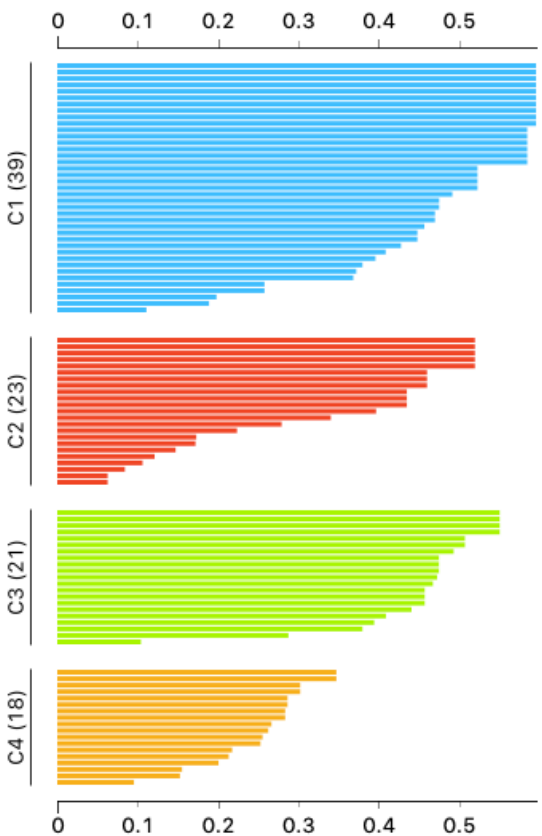
find k so that $\bar{s}$ is maximized


14
12
10
8
6
4
2
0
C1
C2
C3
C4


-0.2
0
0.2
0.4
0.6
C1 (41)
C2 (21)
C3 (13)
C4 (26)


C1
C2
C3
C4
PC2
PC1
5
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4


k-Means
Number of Clusters
Fixed: 4
From 2 to 8
Preprocessing
Normalize columns
Initialization
Initialize with KMeans++
Re-runs: 10
Maximum iterations: 300
Apply Automatically


0
0.1
0.2
0.3
0.4
0.5
C1 (39)
C2 (23)
C3 (21)
C4 (18)

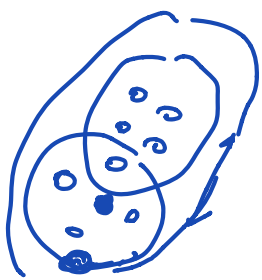
Density-Based Clustering

DBSCAN

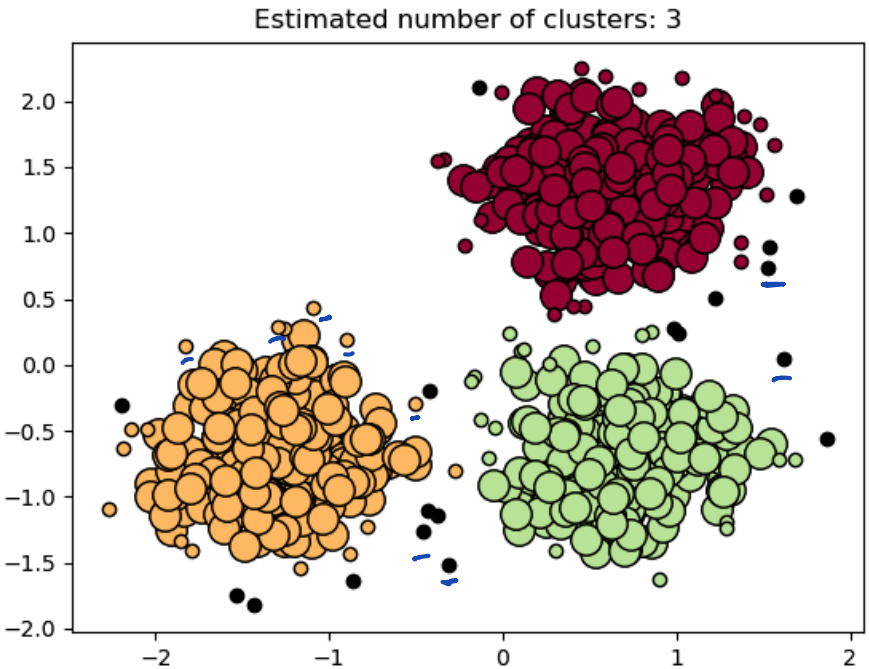
minPts : minimal. number of objects in the core

ε : radius of the core

- find the cores
- merge overlapping cores



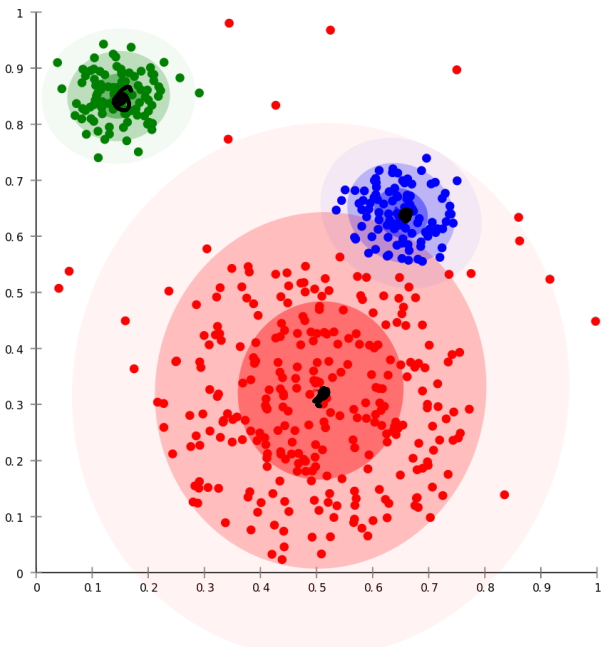
- not all points are clustered
- some outliers of clusters
- find dense regions


Estimated number of clusters: 3
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2
−1
0
1
2


DBSCAN
k-means

Distribution-Based Clustering

Probabilistic clustering

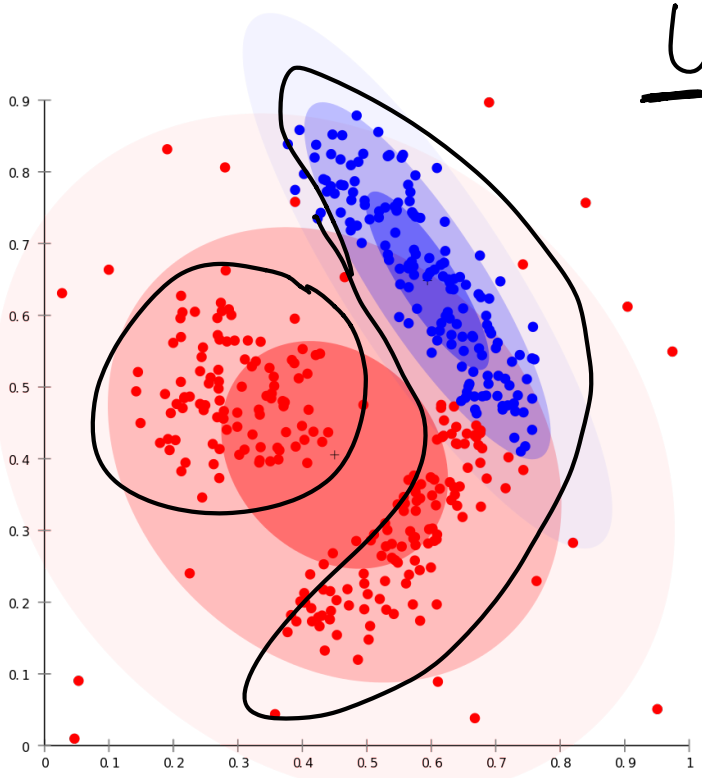


crisp clustering

fixed number of Gaussian distributions

expectation maximization

UMAP



Interpretation