UNIVERSITY of LOUISIANA LAFAYETTE

CSCE566-DATA MINING
WEEK 3&4

Matrix Data Mining

Min Shi
min.shi@louisiana.edu

Sep 4, 2024

Outline: Matrix Data Mining

1. Introduction to Matrix Data
2. Principle Component Analysis
3. Support Vector Machines
4. K-Means Clustering
5. Application: Genomics Data Analysis

The definition of matrix

- A matrix is a rectangular array of numbers, symbols, or expressions, arranged in rows and columns.

m-by-n matrix

$a_{i,j}$

n columns — j changes →

m rows — i changes ↓

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots \\ a_{3,1} & a_{3,2} & a_{3,3} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

Types of Matrices

1. Row matrix

$$\begin{pmatrix} 1 & 2 & 3 \end{pmatrix}$$

2. Column matrix

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

3. Rectangular matrix

$$\begin{bmatrix} 1 & 3 & 4 \\ 2 & 5 & 2 \end{bmatrix}$$

4. Square matrix

$$\begin{bmatrix} 1 & 3 & 4 \\ 5 & 2 & 4 \\ 1 & 9 & 6 \end{bmatrix}$$

5. zero matrix

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

6. Diagonal matrix

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}$$

7. Scalar matrix

$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

8. unit matrix

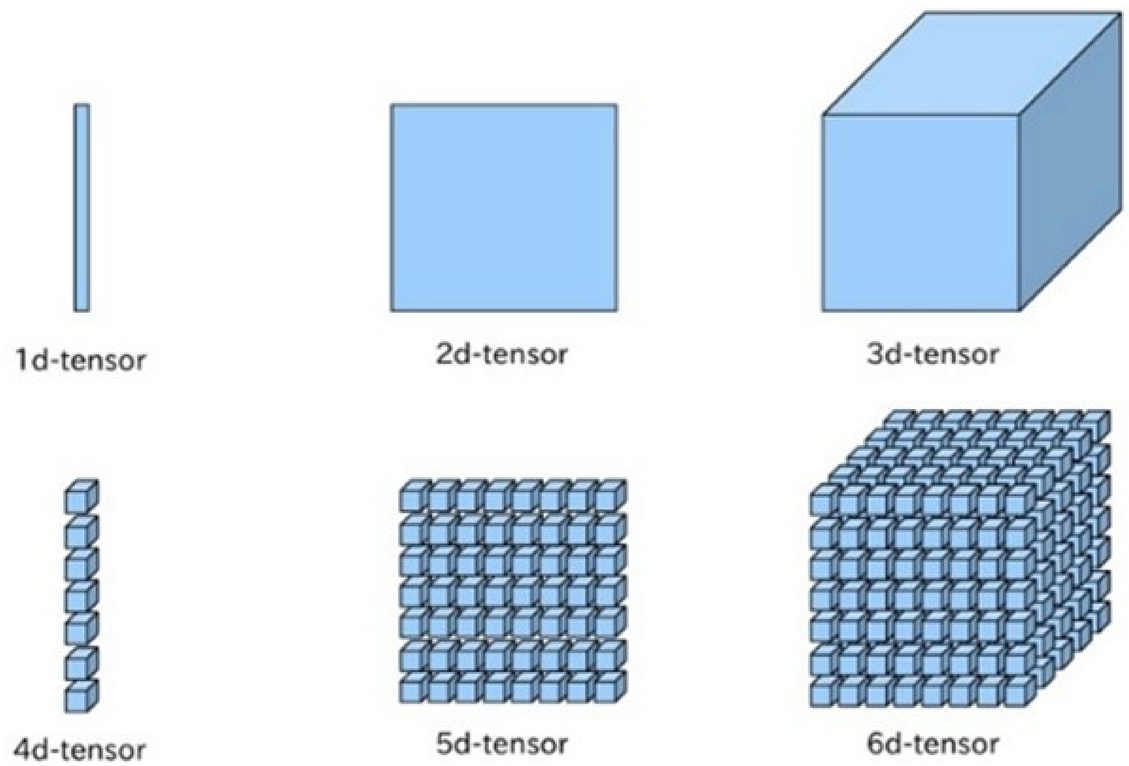
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

9. upper and lower triangular matrix

$$\begin{bmatrix} 1 & 0 & 0 \\ 2 & 4 & 0 \\ 3 & 5 & 6 \end{bmatrix} \quad \begin{bmatrix} 5 & 8 & 3 \\ 0 & 1 & 2 \\ 0 & 0 & 7 \end{bmatrix}$$

The definition of matrix

- A matrix is a special case of tensor:
  - tensors are mathematical objects that generalize **scalars**, **vectors** and **matrices** to higher dimensions
  - a matrix is a two-dimensional (2D) tensor
  - tensor is a common data format used while training a deep learning model



Types of attributes

There are different types of attributes in the matrix data

- **Categorical**
  - Examples: eye color, zip codes, words, rankings (e.g., good, fair, bad), height in {tall, medium, short}
  - Nominal (no order or comparison, e.g., ID numbers, eye color, zip codes) vs Ordinal (order but not comparable, e.g., taste of potato chips on a scale from 1-10, grades, height {tall, medium, short})

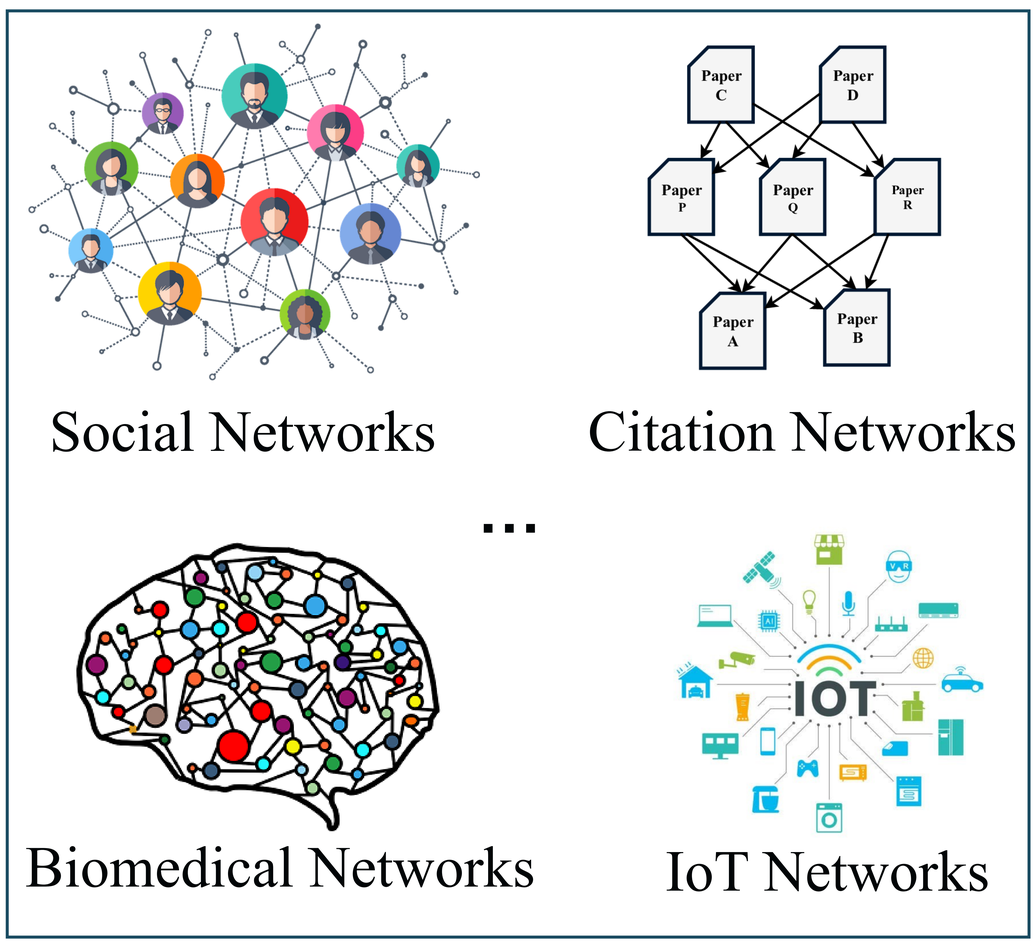
- **Numerical**
  - Examples: dates, temperature, time, length, value, count
  - Discrete (counts) vs Continuous (temperature)
  - Special case: Binary attributes (yes/no, exists/not exists)

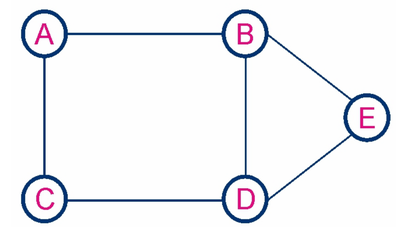
Why matrix data mining?

- Matrices are fundamental in many reasons:
  - Many real-word data naturally take the form of matrices, such as texts, images, social networks, …
  - In deep learning, model weights and inputs are typically represented as matrices.
  - Many mathematical operations and transformations are efficiently carried out using matrices, i.e., multiplication, inversion, and eigenvalue decomposition
  - Facilitate similarity measurement and computing
- Application scenarios:
  - Feature representation and data transformation
  - Image processing, i.e., image data as matrices
  - Graph and network analysis, i.e., adjacency matrices
  - Recommender systems, i.e., collaborative filtering algorithm
  - Natural language processing, i.e., text analysis
  - …

Matrix data mining: networks or graphs

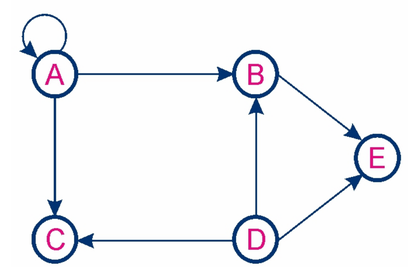
- A network or graph can be represented in a $n \times n$ adjacency matrix:
  - n is the number of nodes within the network
  - entries in the matrix quantify the relationships between nodes





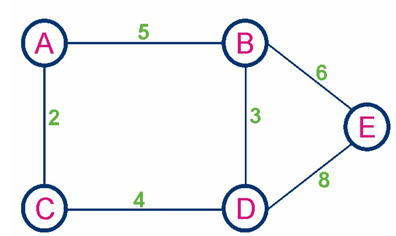
Undirected Graph

	A	B	C	D	E
A	0	1	1	0	0
B	1	0	0	1	1
C	1	0	0	1	0
D	0	1	1	0	1
E	0	1	0	1	0



Directed Graph

	A	B	C	D	E
A	1	1	1	0	0
B	0	0	0	0	1
C	0	0	0	0	0
D	0	1	1	0	1
E	0	0	0	0	0



Weighted Graph

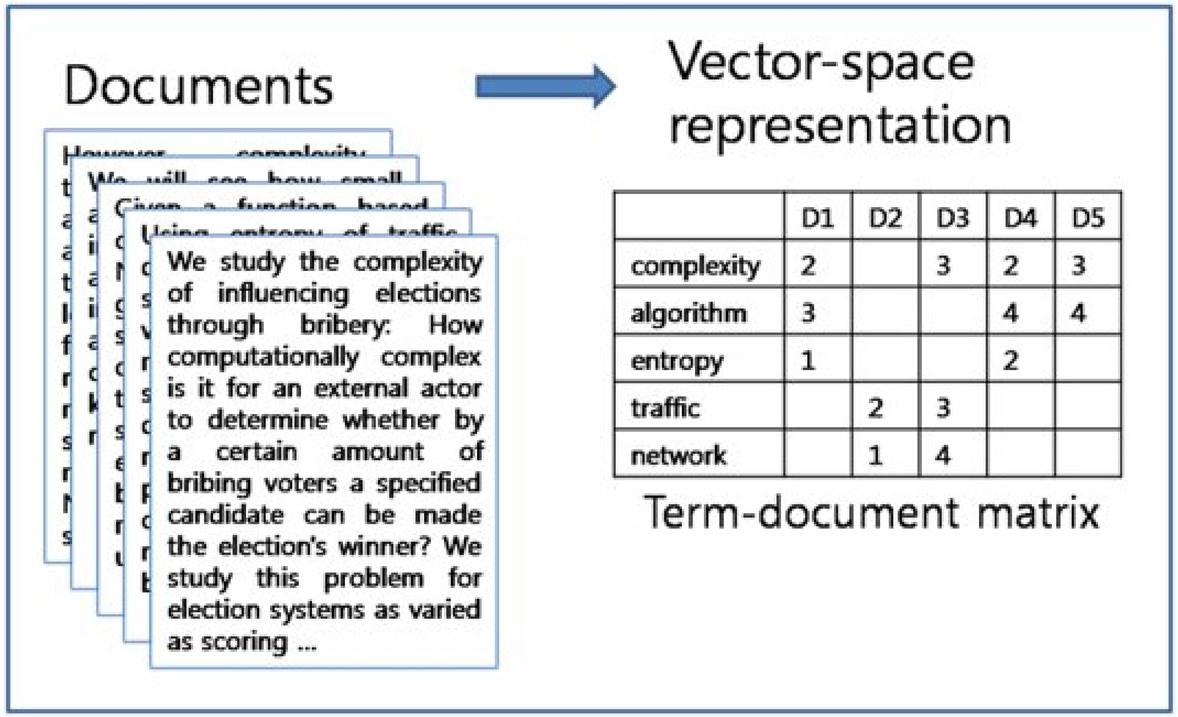
	A	B	C	D	E
A	0	5	2	0	0
B	5	0	0	3	6
C	2	0	0	4	0
D	0	3	4	0	8
E	0	6	0	8	0

Matrix data mining: documents

- A collection of n documents can be represented as a $n \times W$ matrix:
  - W is the number of unique words in the corpus
  - each document is represented as a vector of W words
  - entries indicate the word counts in respective documents

$$C = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix} = \begin{bmatrix} [w_1, w_2, ..., W]_{d_1} \\ [w_1, w_2, ..., W]_{d_2} \\ \vdots \\ [w_1, w_2, ..., W]_{d_n} \end{bmatrix}$$

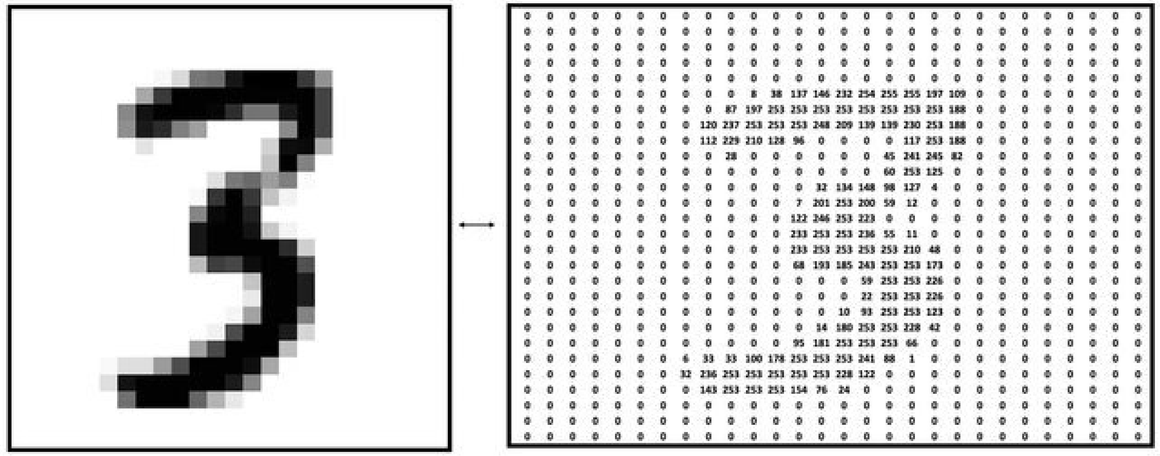
- C: Represents the corpus matrix, where each row corresponds to a different document within the corpus.
- d_i: Denotes the i-th document in the corpus. Here, i ranges from 1 to n, where n is the total number of documents in the corpus.



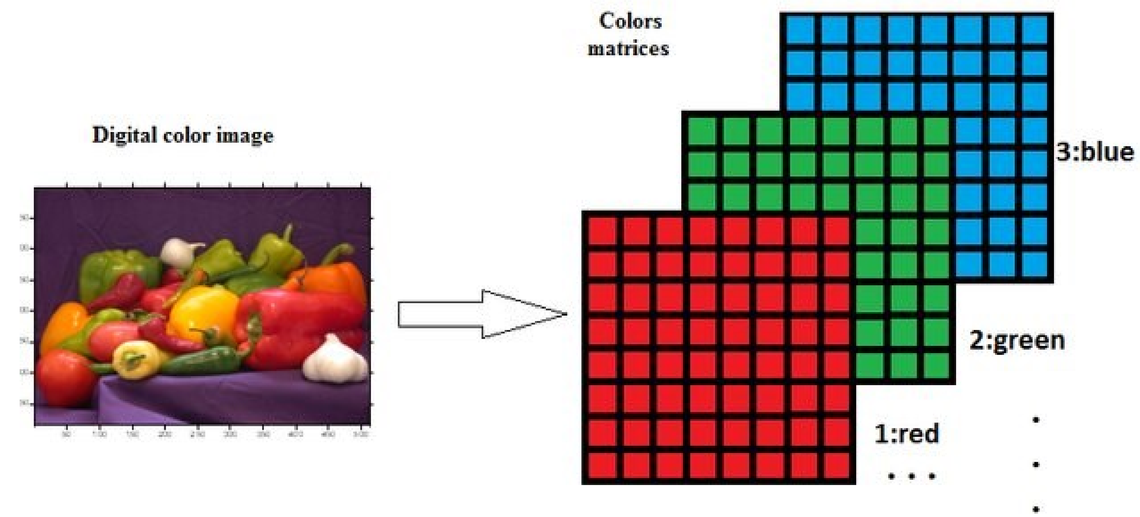
	D1	D2	D3	D4	D5
complexity	2		3	2	3
algorithm	3			4	4
entropy	1			2	
traffic		2	3		
network		1	4		

Matrix data mining: images

- An image can be represented as a $h \times w$ numerical matrix:
  - h is the image height
  - w is the image width
  - entries in the matrix are pixel values (intensities) between 0 and 255
  - gray-scale image is a 2D matrix and color image is a 3D matrix



A gray-scale image is a 2-dimensional matrix



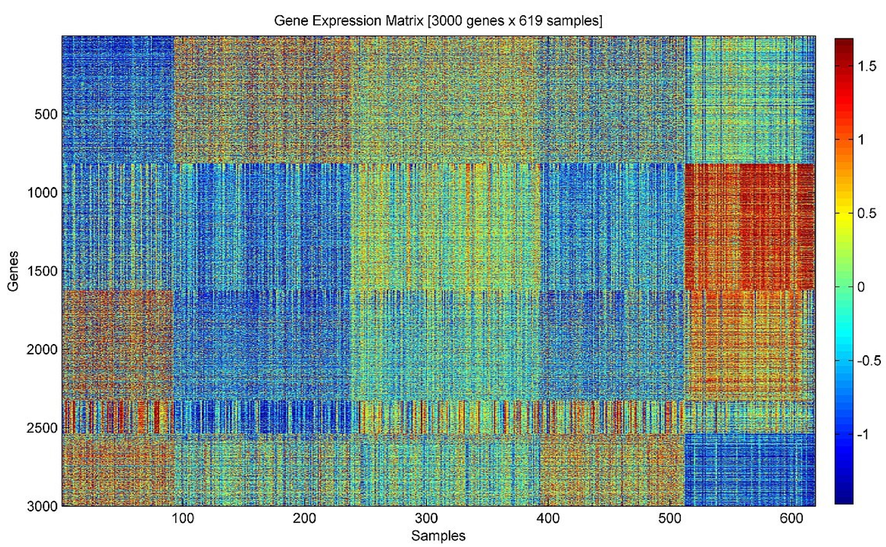
A color image is a 3-dimensional matrix

Matrix data mining: medical diagnostics

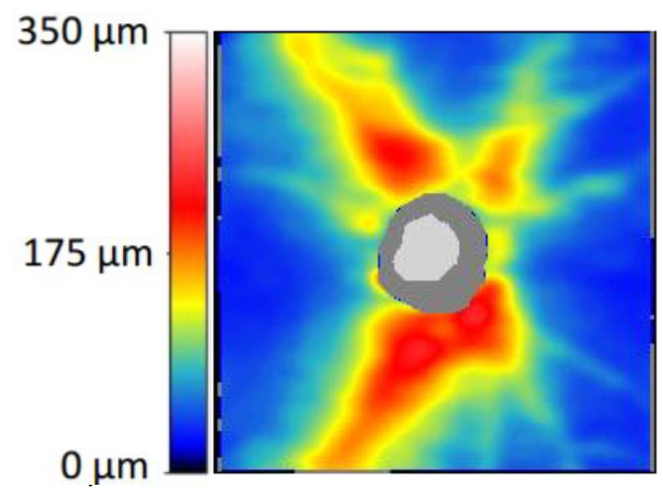
- Medical diagnostics can produce various types of data:
  - electronical health record (EHR)
  - high-dimensional genetics data
  - medical notes, images, and others

age	gender	race	ethnicity	language	maritalstatus	note	glaucoma
56.56	female	black	non-hispanic	english	single	d 3. narrow	yes
53.91	female	white	non-hispanic	english	single	en to ss, cbb	yes
46.3	female	white	non-hispanic	english	single	thought the	no
66.52	male	white	non-hispanic	english	single	l fields, left	yes
82.52	female	black	non-hispanic	english	divorced	perior thinni	yes
41.79	female	white	unknown	english	married or partnered	logist DATE_	yes
43.8	male	white	non-hispanic	english	married or partnered	ses and rnfl	yes
70.21	male	white	non-hispanic	english	single	eing: observa	yes

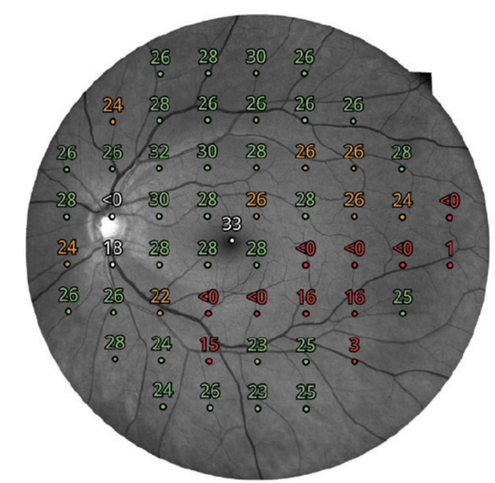
HER is organized in a matrix



Gene expression matrix



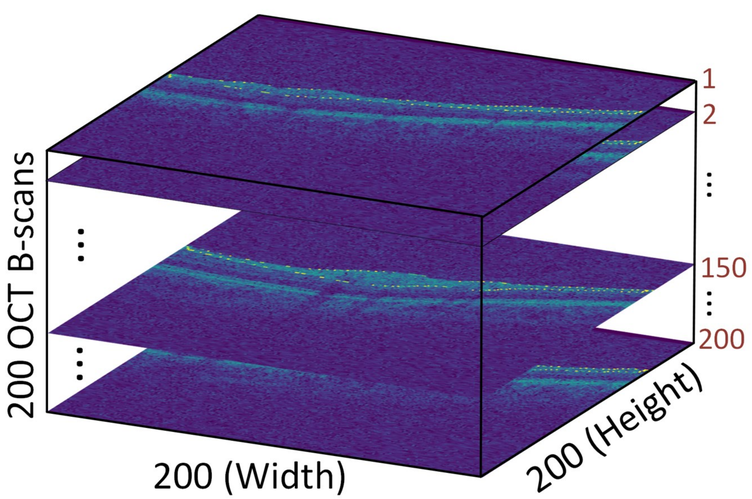
Retinal nerve fiver layer thickness map is a matrix



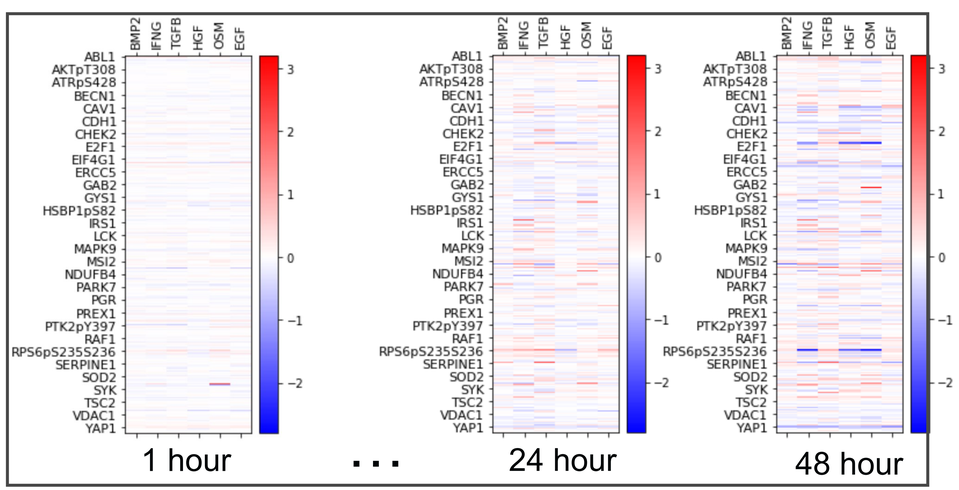
Visual field map is a matrix

Matrix data mining: time-series data

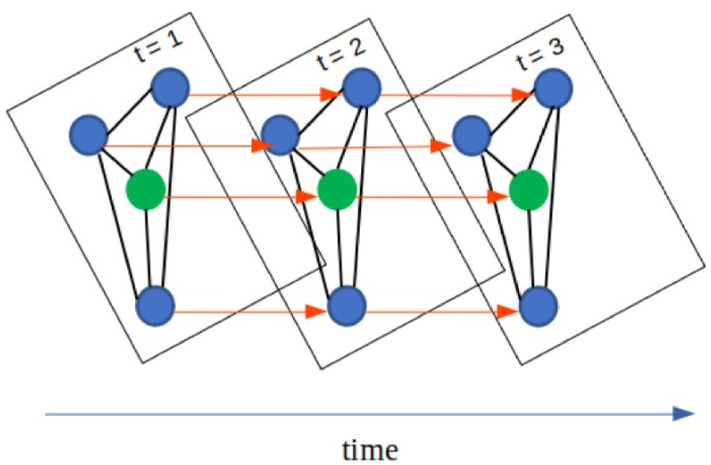
- Time-series data can be represented as a series of sequentially ordered matrices:
  - data are evolving over time
  - the data can be represented as a matrix at every time step
  - a series of ordered matrices can be viewed as a tensor



OCT B-Scans



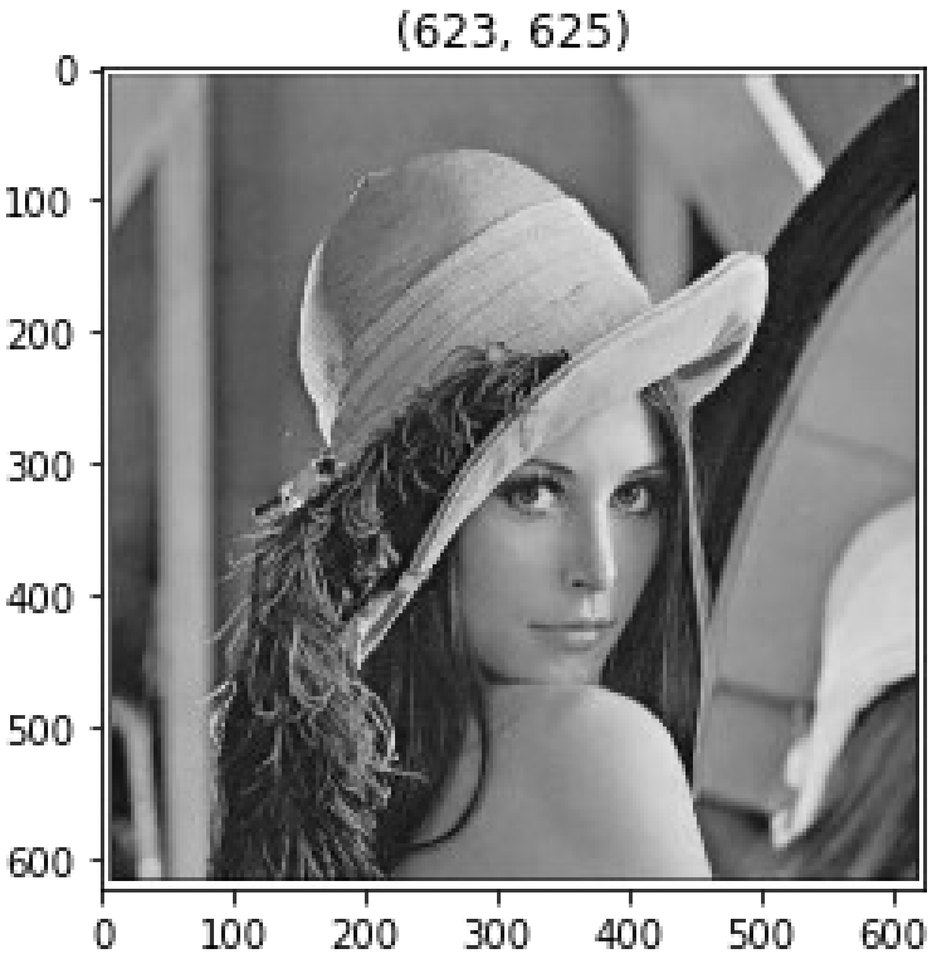
Dynamic gene expression



Dynamic graphs

Matrix operations

Addition, Multiplication, Transpose, Flipping, Rotation, …

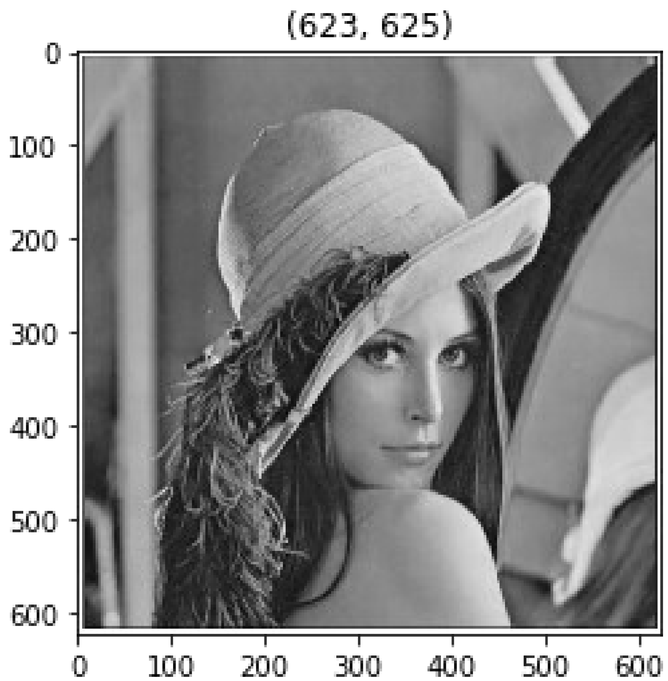



```python
import numpy as np
import cv2
import matplotlib.pyplot as plt
```

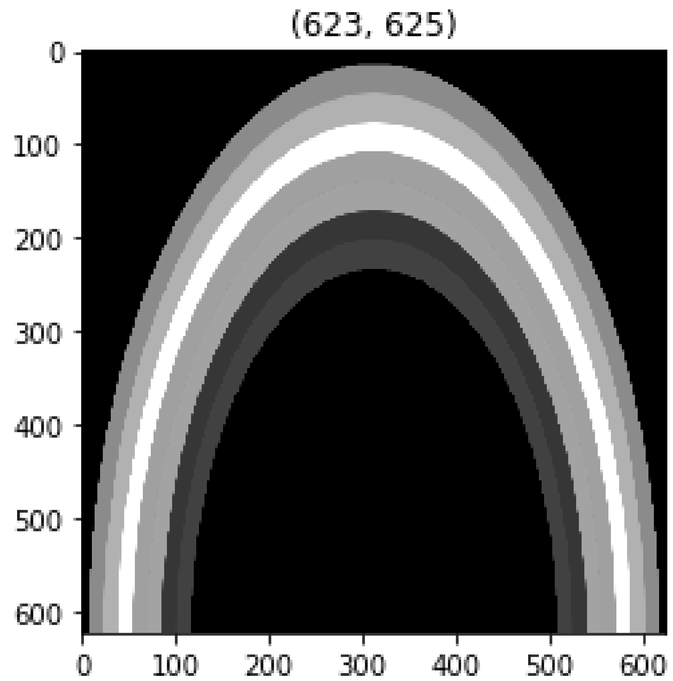
```python
img = cv2.imread('example.jpg', cv2.IMREAD_GRAYSCALE)
```

```
array([[255, 253, 255, ..., 255, 252, 255],
       [255, 255, 255, ..., 255, 254, 255],
       [254, 255, 255, ..., 250, 255, 255],
       ...,
       [254, 255, 255, ..., 255, 255, 255],
       [255, 251, 255, ..., 245, 248, 255],
       [255, 247, 251, ..., 255, 255, 255]], dtype=uint8)
```

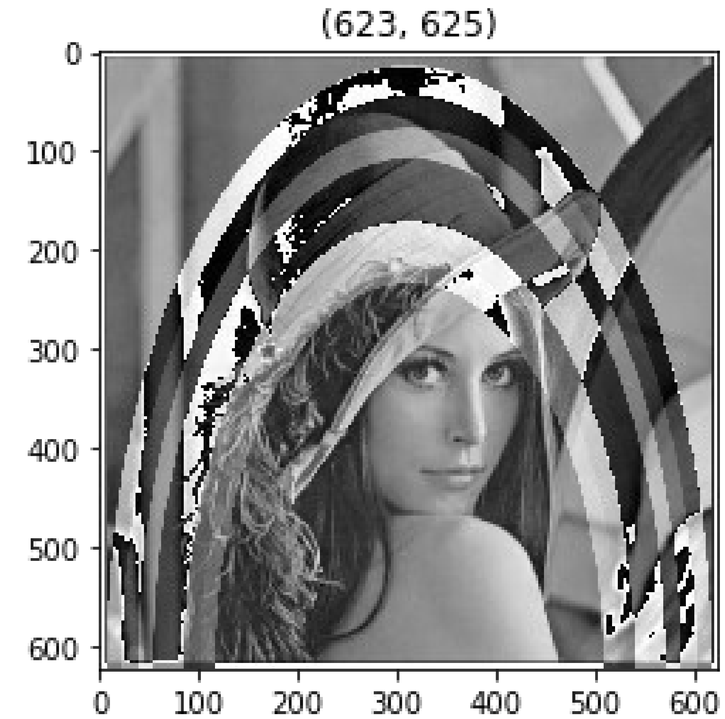
Addition



+



=

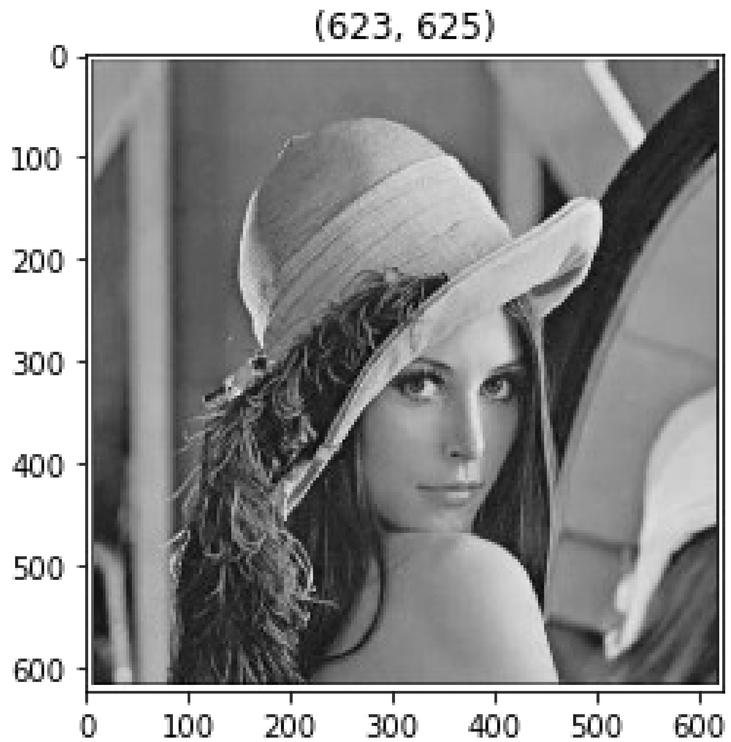



```
[[255, 253, 255, ..., 255, 252, 255],
 [255, 255, 255, ..., 255, 254, 255],
 [254, 255, 255, ..., 250, 255, 255],
 ...,
 [254, 255, 255, ..., 255, 255, 255],
 [255, 251, 255, ..., 245, 248, 255],
 [255, 247, 251, ..., 255, 255, 255]]
```

```
[[0, 0, 0, ..., 0, 0, 0],
 [0, 0, 0, ..., 0, 0, 0],
 [0, 0, 0, ..., 0, 0, 0],
 ...,
 [0, 0, 0, ..., 0, 0, 0],
 [0, 0, 0, ..., 0, 0, 0],
 [0, 0, 0, ..., 0, 0, 0]]
```

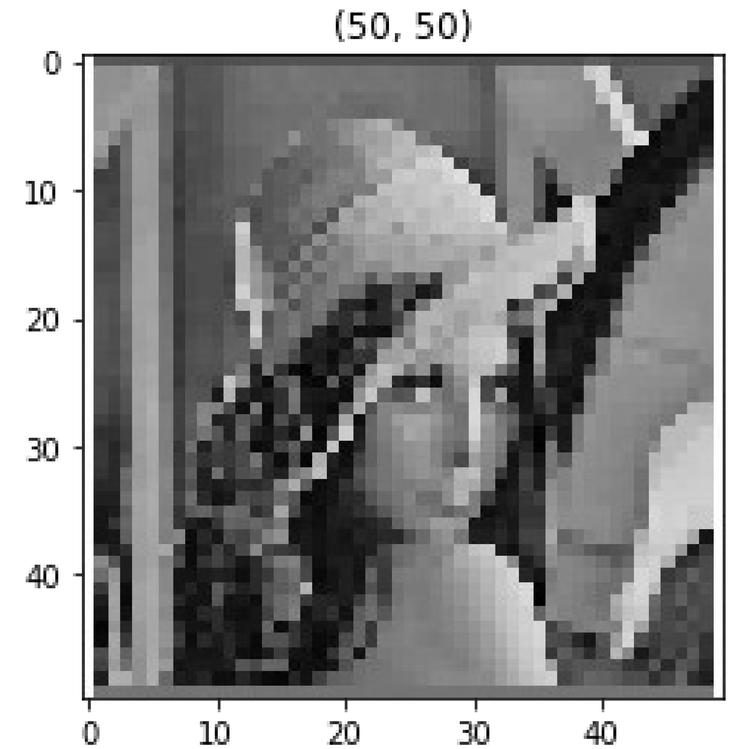
```
[[255, 253, 255, ..., 255, 252, 255],
 [255, 255, 255, ..., 255, 254, 255],
 [254, 255, 255, ..., 250, 255, 255],
 ...,
 [254, 255, 255, ..., 255, 255, 255],
 [255, 251, 255, ..., 245, 248, 255],
 [255, 247, 251, ..., 255, 255, 255]]
```

Resize




```
cv2.resize(img, (50, 50))
```

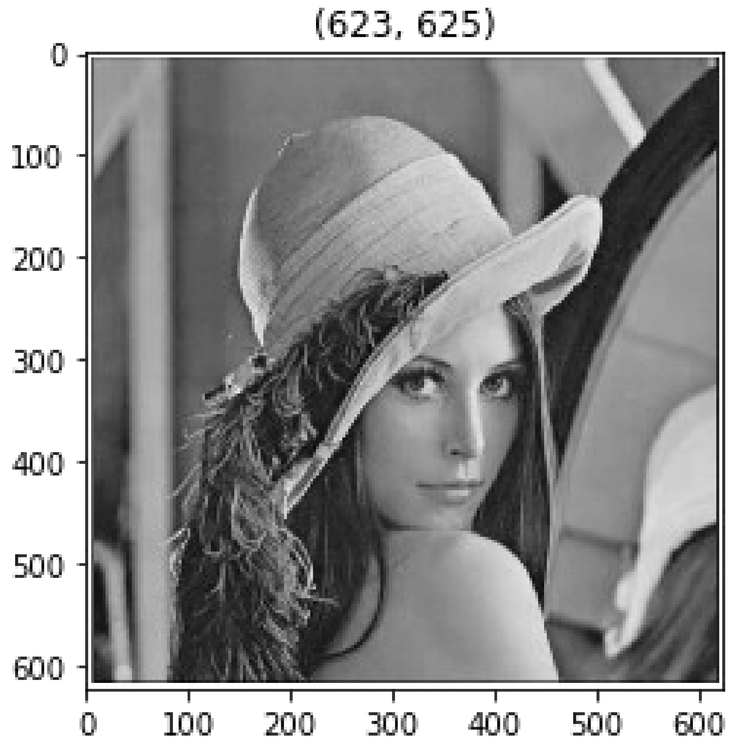
Downscale




```
[[255, 253, 255, ..., 255, 252, 255],
 [255, 255, 255, ..., 255, 254, 255],
 [254, 255, 255, ..., 250, 255, 255],
 ...,
 [254, 255, 255, ..., 255, 255, 255],
 [255, 251, 255, ..., 245, 248, 255],
 [255, 247, 251, ..., 255, 255, 255]]
```

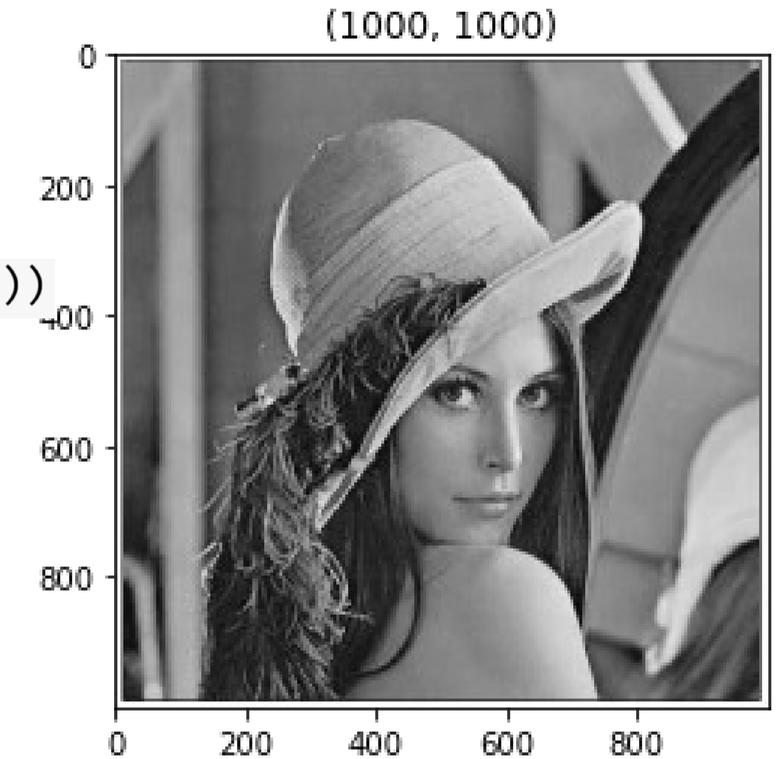
```
[[251, 106, 105, ..., 101, 101, 252],
 [254, 155, 157, ..., 124, 105, 255],
 [254, 156, 157, ...,  89,  47, 255],
 ...,
 [254,  87, 166, ...,  98,  63, 254],
 [254,  53, 183, ...,  69,  55, 255],
 [250, 130, 132, ..., 130, 130, 252]]
```

Resize




```
cv2.resize(img, (1000, 1000))
```

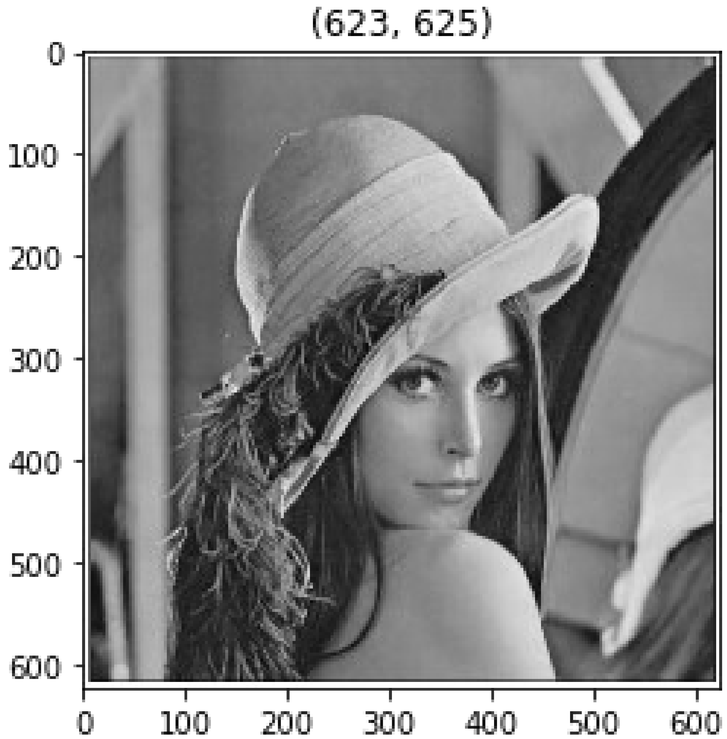
Upscale



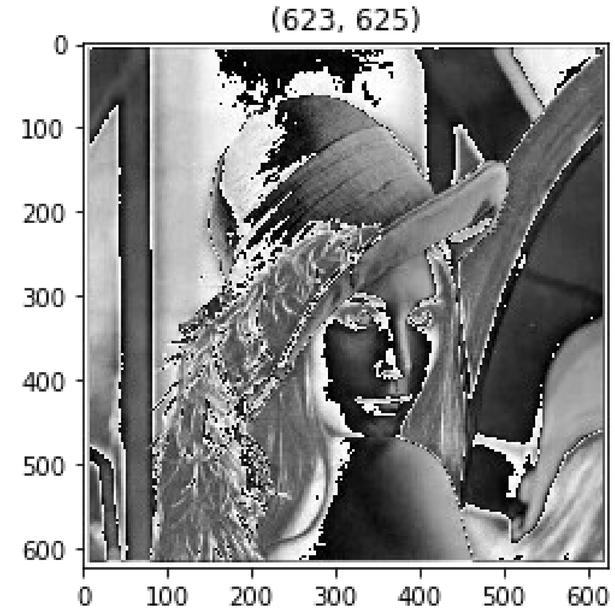

```
[[255, 253, 255, ..., 255, 252, 255],
 [255, 255, 255, ..., 255, 254, 255],
 [254, 255, 255, ..., 250, 255, 255],
 ...,
 [254, 255, 255, ..., 255, 255, 255],
 [255, 251, 255, ..., 245, 248, 255],
 [255, 247, 251, ..., 255, 255, 255]]
```

```
[[255, 254, 253, ..., 252, 254, 255],
 [255, 254, 254, ..., 253, 254, 255],
 [255, 255, 255, ..., 254, 254, 255],
 ...,
 [255, 253, 251, ..., 248, 252, 255],
 [255, 252, 249, ..., 252, 253, 255],
 [255, 251, 247, ..., 255, 255, 255]]
```

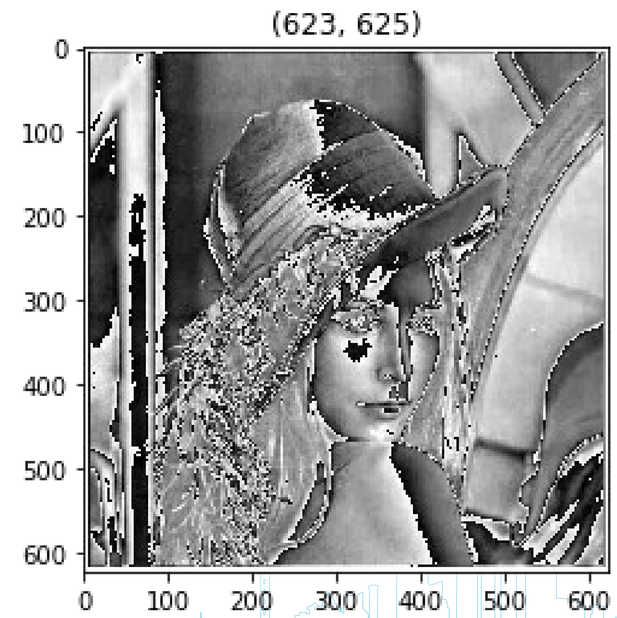
Multiplication



$\times 1$

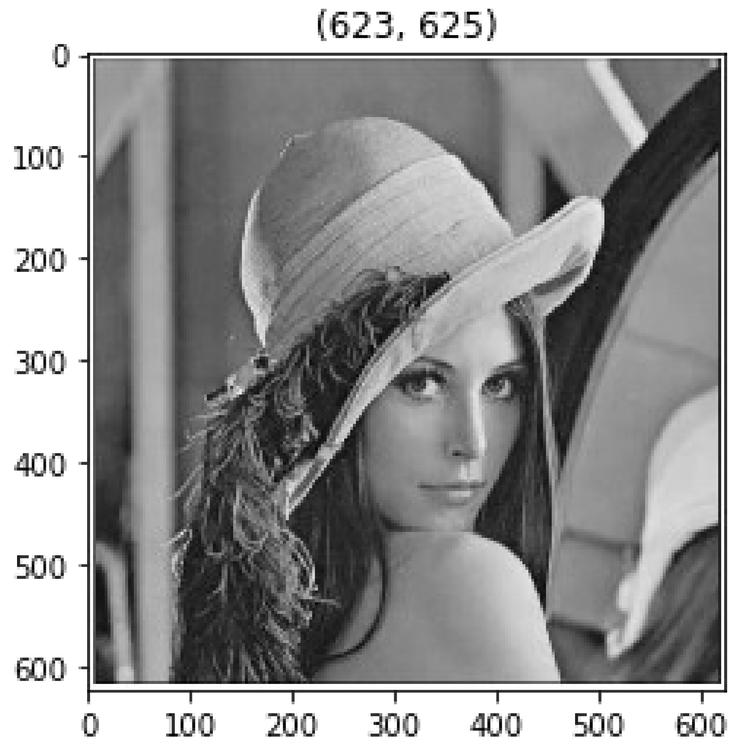


$\times 2$

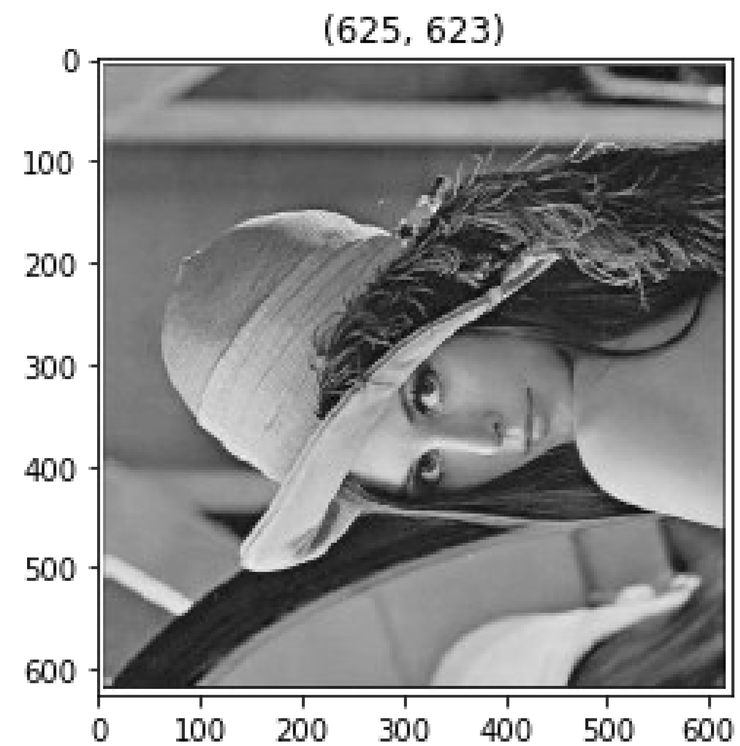



```
[[255, 253, 255, ..., 255, 252, 255],
 [255, 255, 255, ..., 255, 254, 255],
 [254, 255, 255, ..., 250, 255, 255],
 ...,
 [254, 255, 255, ..., 255, 255, 255],
 [255, 251, 255, ..., 245, 248, 255],
 [255, 247, 251, ..., 255, 255, 255]]
```

Transpose

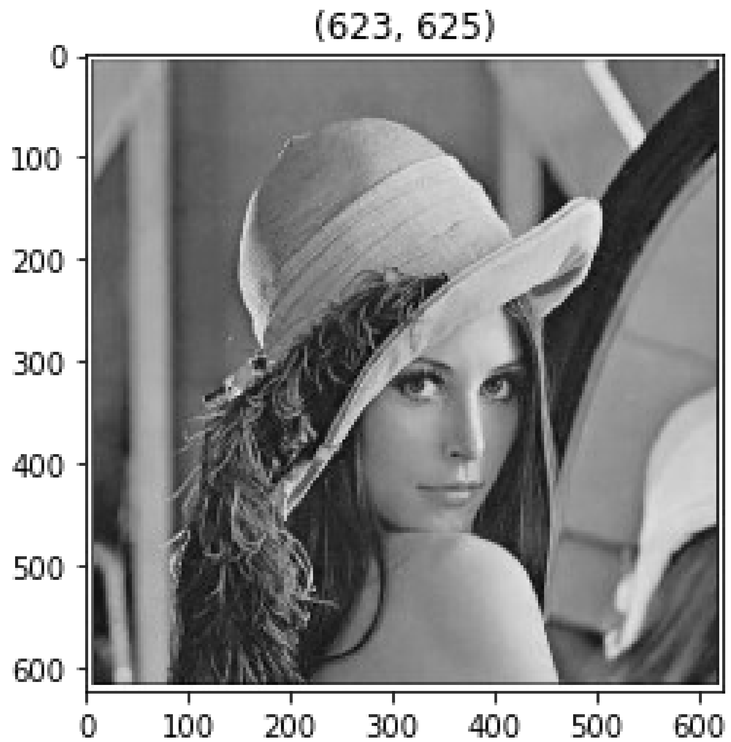



```
[[255, 253, 255, ..., 255, 252, 255],
 [255, 255, 255, ..., 255, 254, 255],
 [254, 255, 255, ..., 250, 255, 255],
 ...,
 [254, 255, 255, ..., 255, 255, 255],
 [255, 251, 255, ..., 245, 248, 255],
 [255, 247, 251, ..., 255, 255, 255]]
```

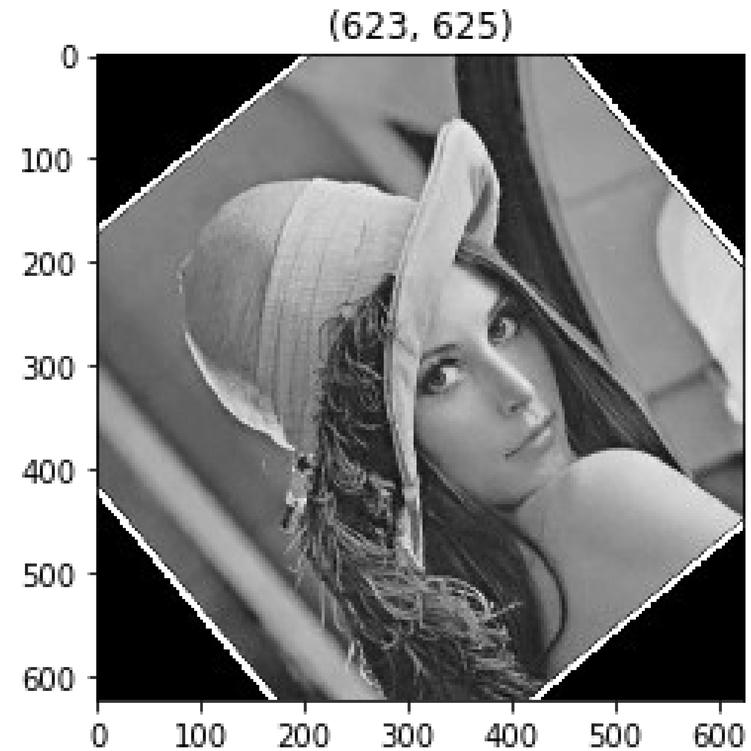



```
[[255, 255, 254, ..., 254, 255, 255],
 [253, 255, 255, ..., 255, 251, 247],
 [255, 255, 255, ..., 255, 255, 251],
 ...,
 [255, 255, 250, ..., 255, 245, 255],
 [252, 254, 255, ..., 255, 248, 255],
 [255, 255, 255, ..., 255, 255, 255]]
```

Rotation




```
rotate(img, 40, reshape=False,
       mode='constant', cval=0)
```



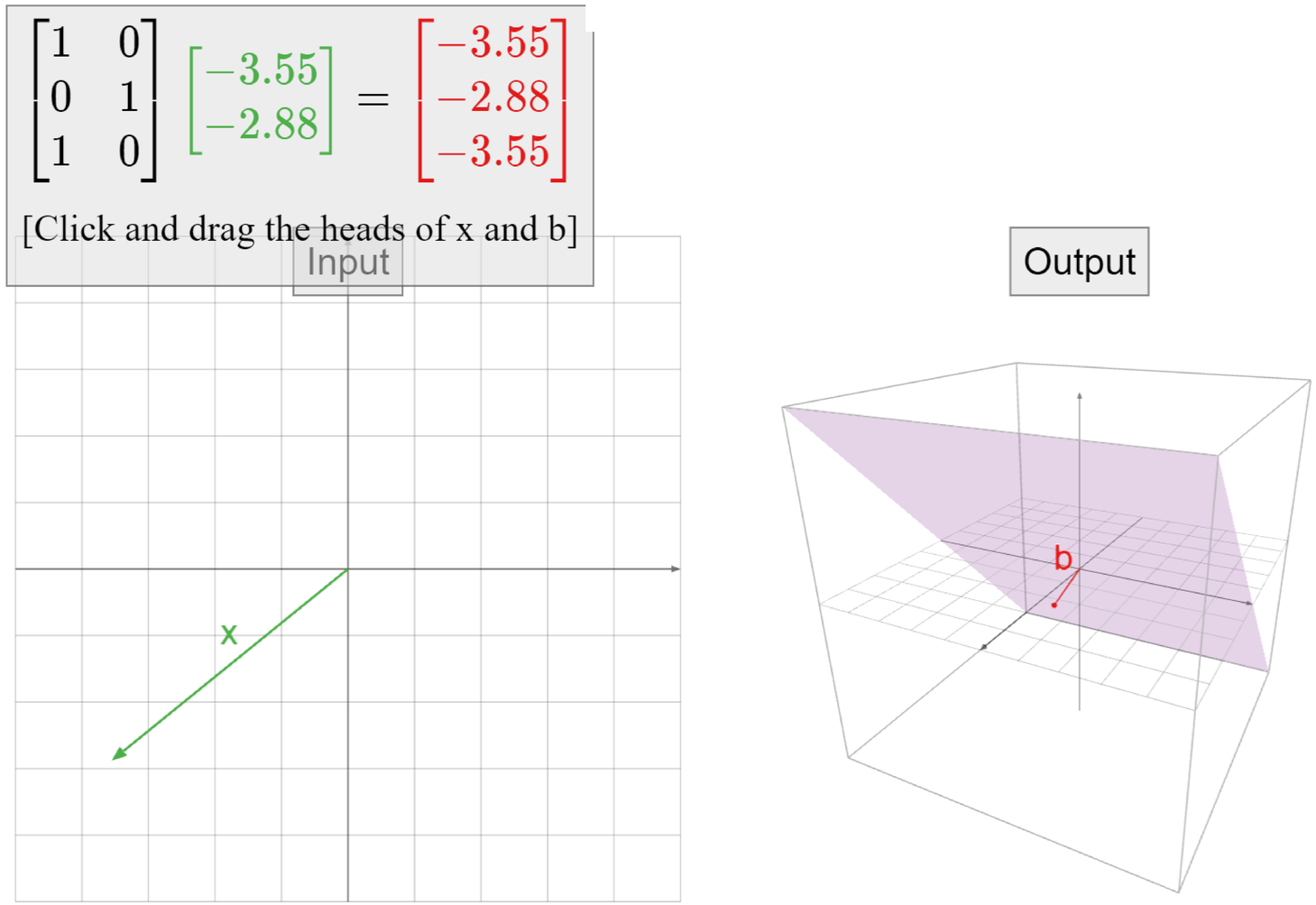

```
[[255, 253, 255, ..., 255, 252, 255],
 [255, 255, 255, ..., 255, 254, 255],
 [254, 255, 255, ..., 250, 255, 255],
 ...,
 [254, 255, 255, ..., 255, 255, 255],
 [255, 251, 255, ..., 245, 248, 255],
 [255, 247, 251, ..., 255, 255, 255]]
```

```
[[0, 0, 0, ..., 0, 0, 0],
 [0, 0, 0, ..., 0, 0, 0],
 [0, 0, 0, ..., 0, 0, 0],
 ...,
 [0, 0, 0, ..., 0, 0, 0],
 [0, 0, 0, ..., 0, 0, 0],
 [0, 0, 0, ..., 0, 0, 0]]
```

Transformation

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{pmatrix},$$

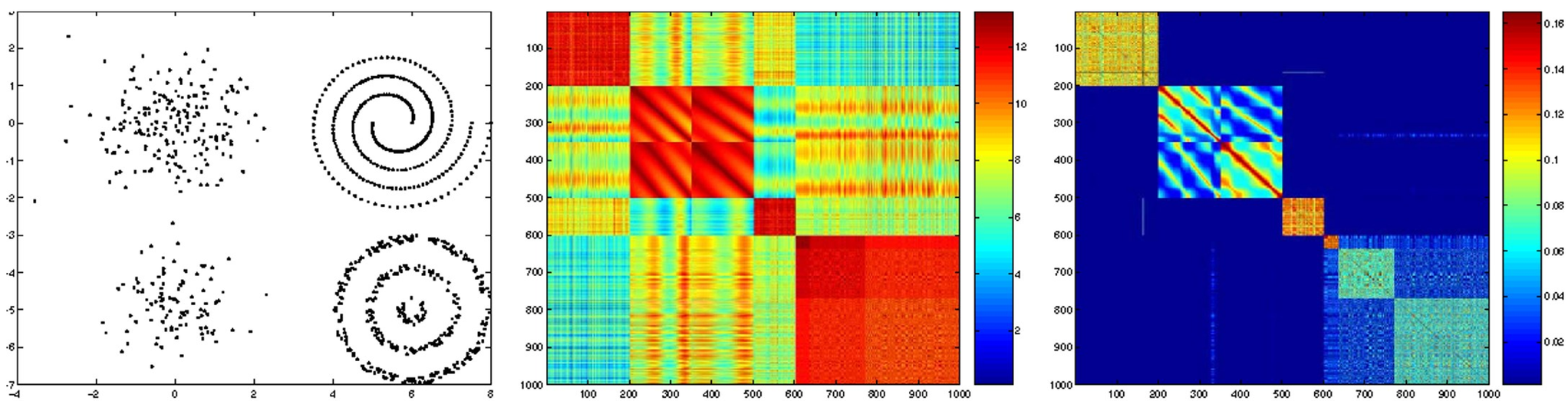
and define $T(x) = Ax$. The domain of T is $\mathbf{R}^2$, and the codomain is $\mathbf{R}^3$. The range of T is the column space; since A has two columns which are not collinear, the range is a plane in $\mathbf{R}^3$.



https://textbooks.math.gatech.edu/ila/matrix-transformations.html

Matrix data mining tasks

- Common tasks in matrix data mining:
  - Dimension Reduction, e.g., visualization and reduce complexity
  - Clustering, e.g., identify hidden patterns or groupings.
  - Classification, e.g., spam detection, image recognition
  - Anomaly Detection, e.g., identify out-of-distribution data points
  - Time Series Analysis, e.g., evolving matrix data

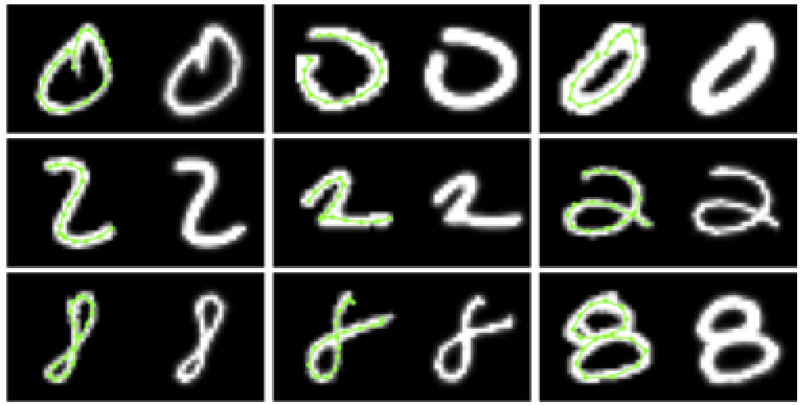


Outline: Matrix Data Mining

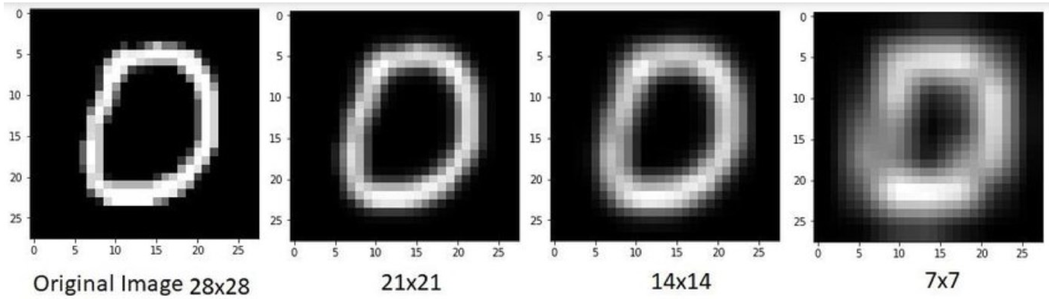
1. Introduction to Matrix Data
2. Principle Component Analysis
3. Support Vector Machines
4. K-Means Clustering
5. Application: Genomics Data Analysis

Dimensionality reduction

Much of the real-word data is high-dimensional yet contains redundant information.



- MNIST: $28 \times 28 = 784$ pixel values



	Matrix Data			
r1	1	0	3	-1
r2	0	1	1	2
r3	2	0	6	-2
r4	1	1	4	1

Rank = 2

Linear Independent Rows

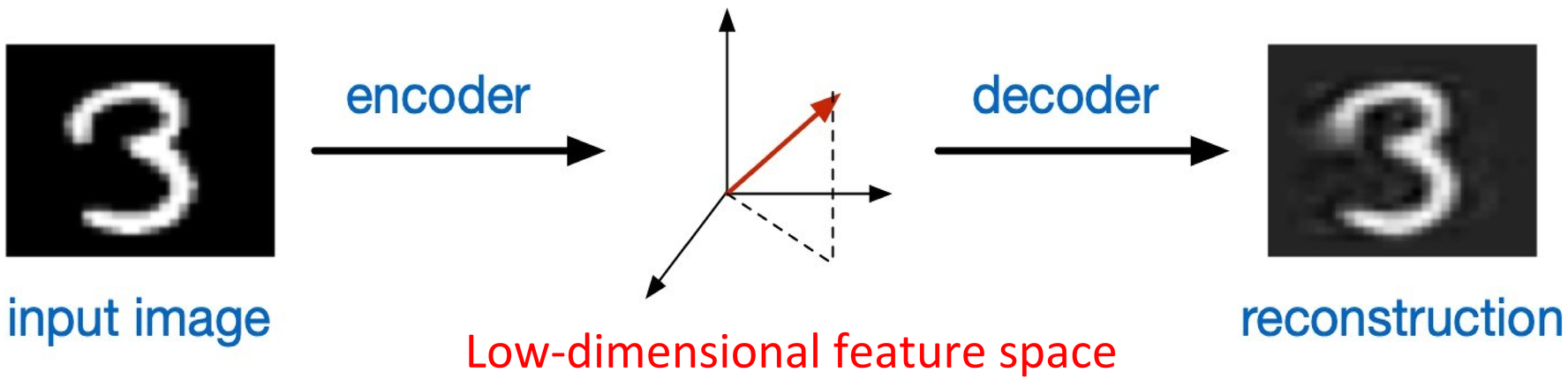
	Matrix Data			
r1	1	0	3	-1
r2	0	1	1	2
r3	2	0	6	-2
r4	1	1	4	1

Redundant Information

Dimensionality reduction

	Matrix Data			
r1	1	0	3	-1
r2	0	1	1	2
r3	2	0	6	-2
r4	1	1	4	1

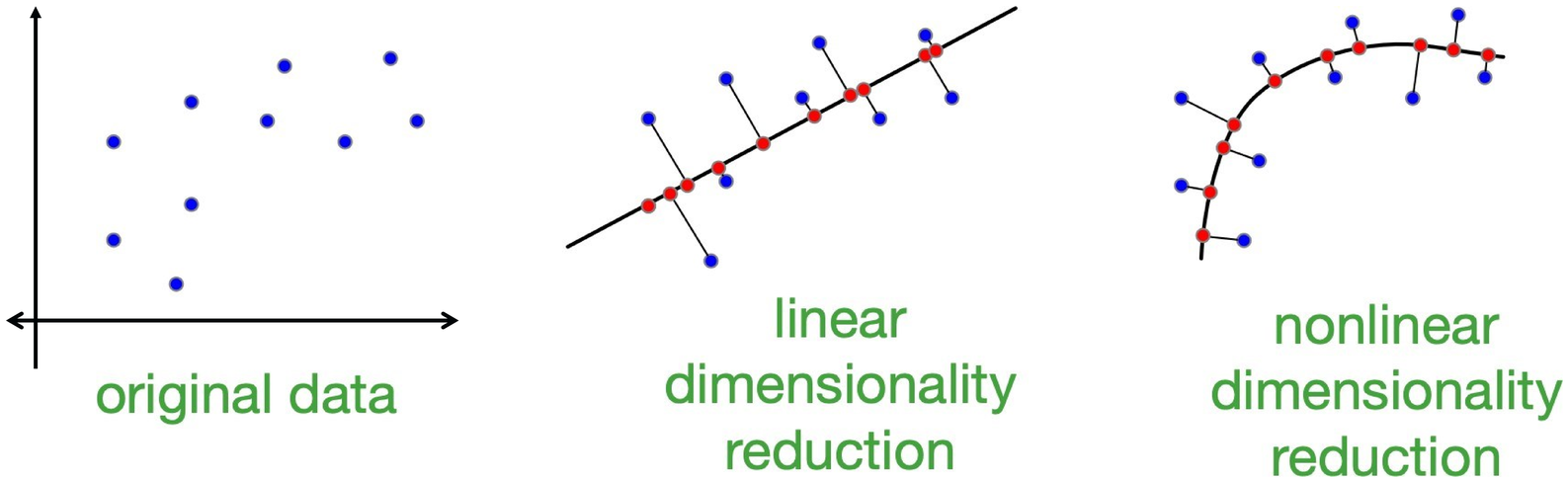
Dimensionality reduction



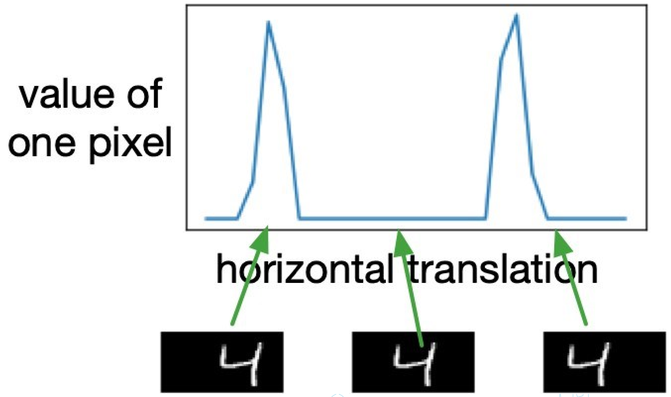
- In dimensionality reduction, we try to learn a mapping to a lower dimensional space that preserves as much information as possible about the input.
- Motivations
  - Data compression
  - Save computation/memory
  - Reduce overfitting
  - Visualize in 2 dimensions

Dimensionality reduction

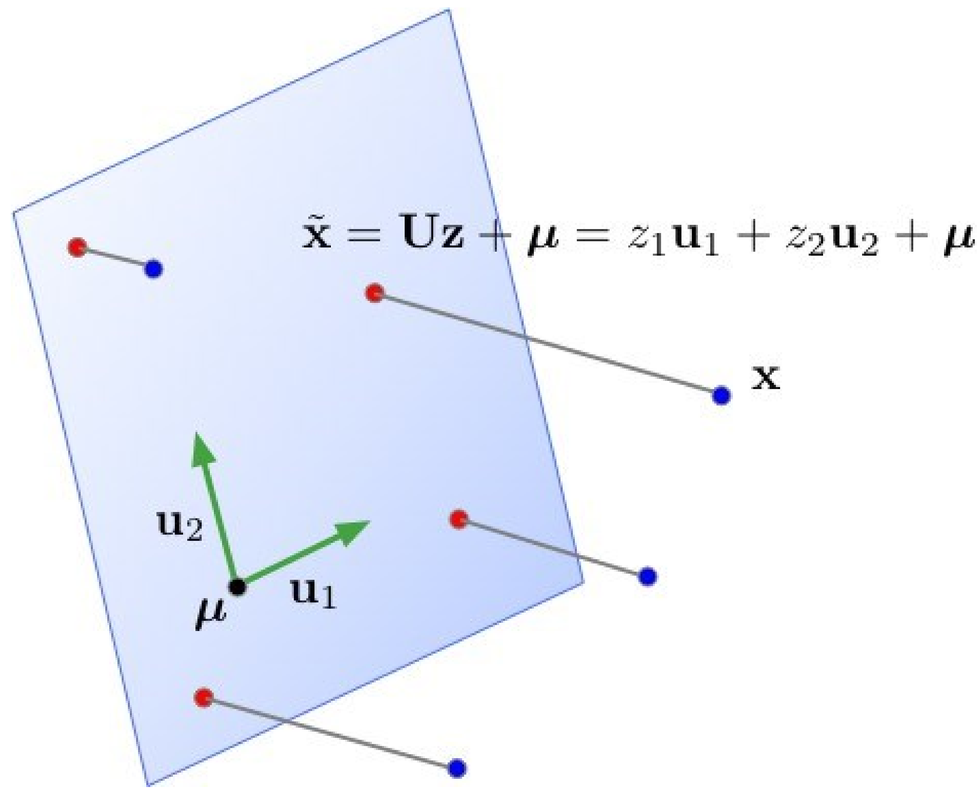
Can be linear or nonlinear:



- Linear dimensionality reduction methods (e.g. PCA) are much simpler, and easier to get to work.
- But many kinds of transformations behave nonlinearly in image space (e.g. translation of an image).



Projection onto a subspace

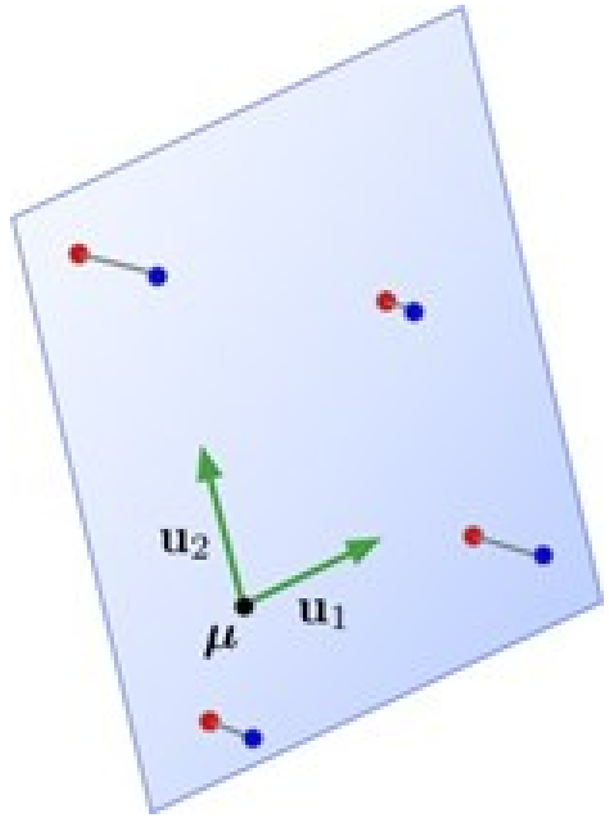


$$\mathbf{z} = \mathbf{U}^T(\mathbf{x} - \boldsymbol{\mu})$$

- Here, the columns of **U** form an orthonormal basis for a subspace S.
- The projection of a point **x** onto S is the point $\tilde{\mathbf{x}} \in S$ closest to **x**. In machine learning, $\tilde{\mathbf{x}}$ is also called the reconstruction of **x**.
- **z** is its low-dimensional representation.

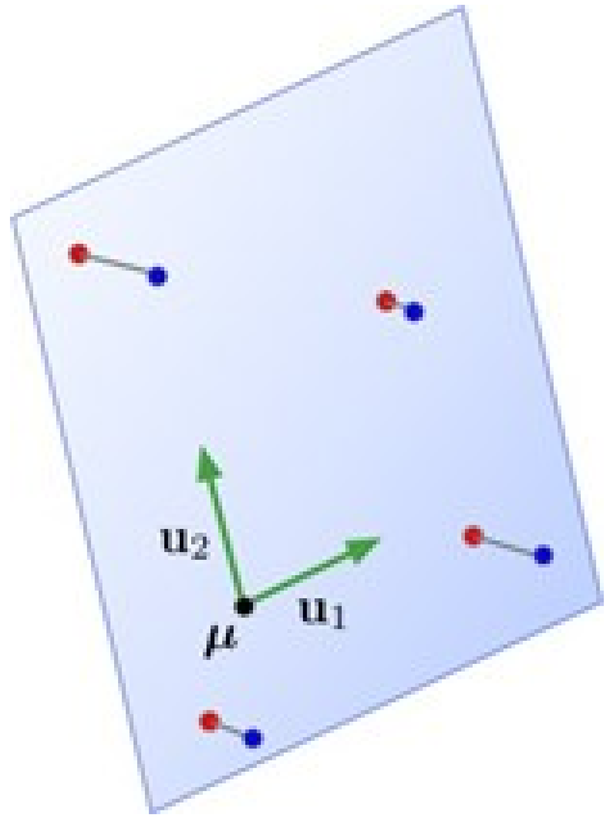
Projection onto a subspace

- If we have a K -dimensional subspace in a D-dimensional input space, then $\mathbf{x} \in \mathrm{R}^D$ and $\mathbf{z} \in \mathrm{R}^K$.
- If the data points **x** all lie close to the subspace, then we can approximate distances, dot products, etc. in terms of these same operations on the code vectors **z**.
- If $K \ll D$, then it's much cheaper to work with **z** than **x**.



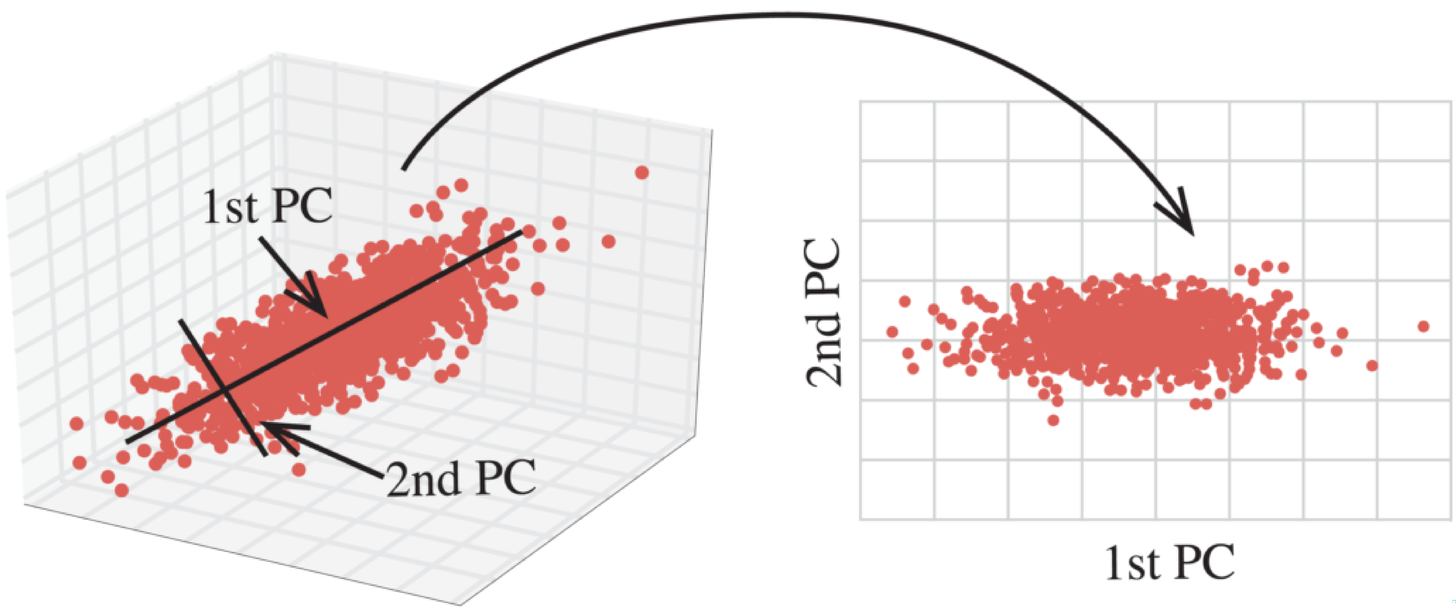
Projection onto a subspace

- If we have a K -dimensional subspace in a D-dimensional input space, then $\mathbf{x} \in \mathrm{R}^D$ and $\mathbf{z} \in \mathrm{R}^K$.
- If the data points **x** all lie close to the subspace, then we can approximate distances, dot products, etc. in terms of these same operations on the code vectors **z**.
- If $K \ll D$, then it's much cheaper to work with **z** than **x**.



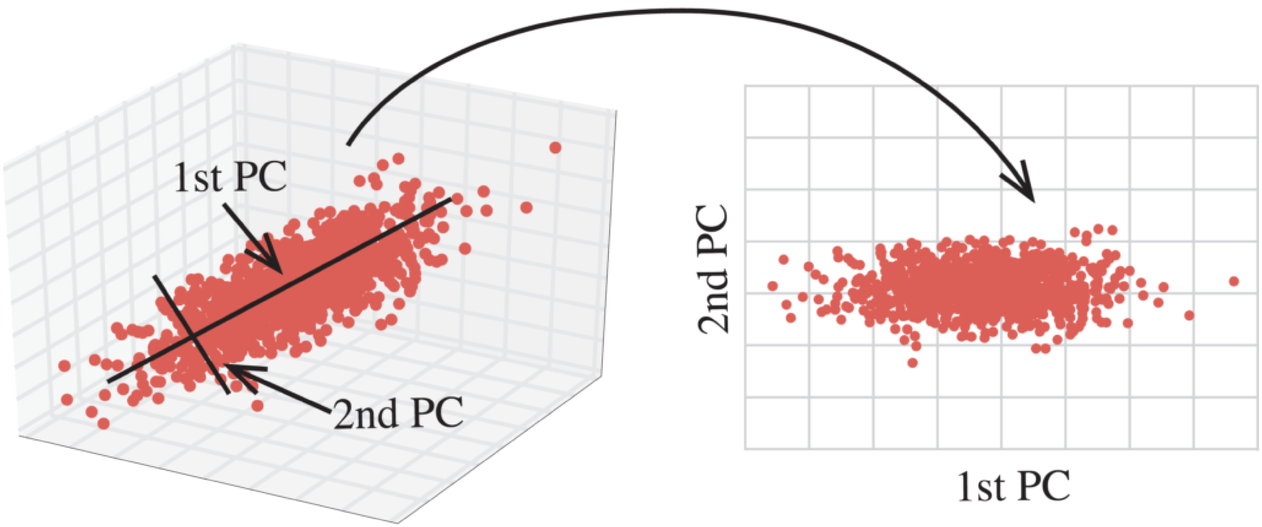
Principle component analysis (PCA)

- PCA aims to reduce the dimensionality of datasets while preserving the crucial information.
  - Image compression, genomic data analysis, etc.
- PCA does this by transforming the original variables into a set of new, uncorrelated variables called **principal components**.
- Choosing a subspace to maximize the projected variance, or minimize the reconstruction error.



Principle component analysis (PCA)

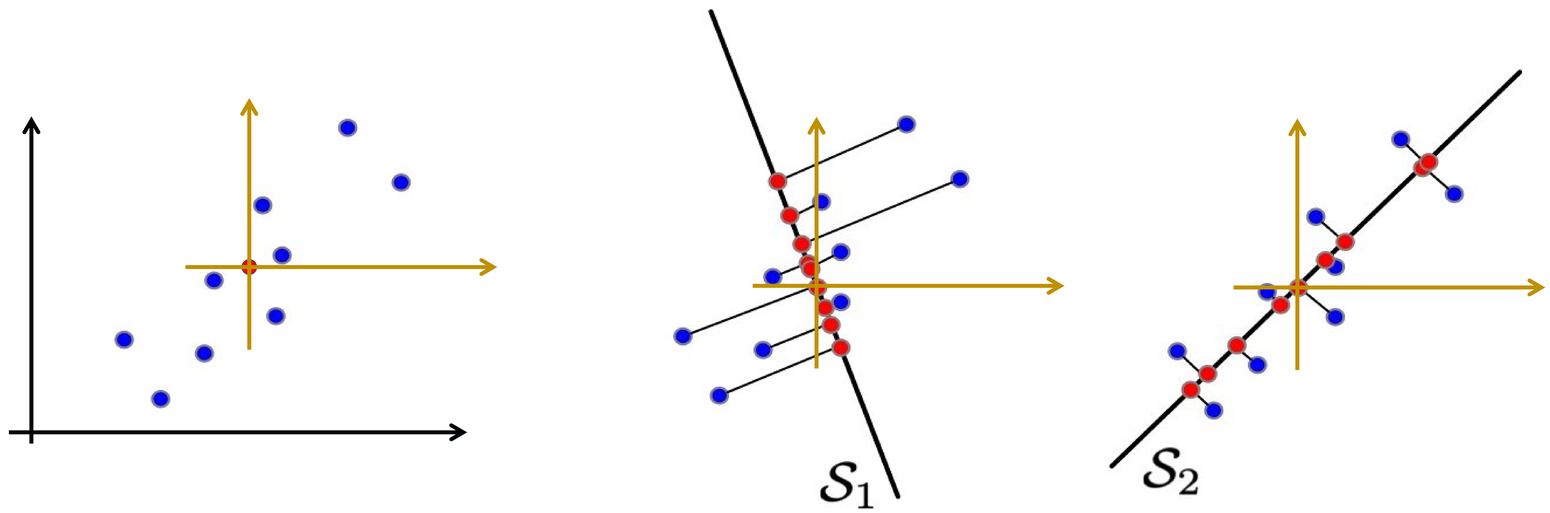
Given feature vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$, compress the data points into low dimensional representation $\mathbf{y}_1, \ldots, \mathbf{y}_n \in \mathbb{R}^K$ where $K << d$



Learn a subspace

Choose a subspace to **maximize** the projected variance, or **minimize** the reconstruction error.

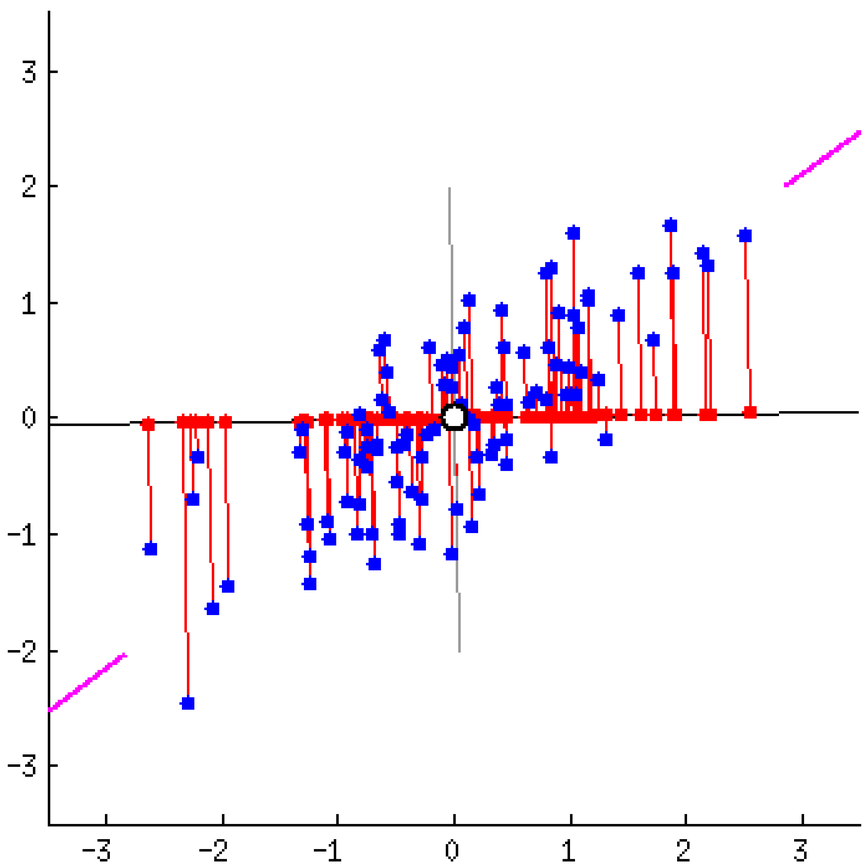
- Which of the following subspaces is a better representation of the dataset?



- On average, the data points are closer to S2 than to S1.
- The projections onto S2 are more spread out than the projections onto S1.

Learn a subspace

Choose a subspace to **maximize** the projected variance, or **minimize** the reconstruction error.



Minimize the reconstruction error

$$\min \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{x}^{(i)} - \tilde{\mathbf{x}}^{(i)}\|^2$$

Maximize the variance of the code vectors

$$\max \sum_j \mathrm{Var}(z_j) = \frac{1}{N} \sum_j \sum_i (z_j^{(i)} - \bar{z}_j)^2$$

$$= \frac{1}{N} \sum_i \|\mathbf{z}^{(i)} - \bar{\mathbf{z}}\|^2$$

$$= \frac{1}{N} \sum_i \|\mathbf{z}^{(i)}\|^2$$

Exercise: show $\bar{\mathbf{z}} = 0$

Principle component analysis (PCA)

Co-variance matrix:

$$\mathbf{\Sigma} = \begin{pmatrix} \mathrm{Var}(X_1) & \mathrm{Cov}(X_1, X_2) & \mathrm{Cov}(X_1, X_3) \\ \mathrm{Cov}(X_2, X_1) & \mathrm{Var}(X_2) & \mathrm{Cov}(X_2, X_3) \\ \mathrm{Cov}(X_3, X_1) & \mathrm{Cov}(X_3, X_2) & \mathrm{Var}(X_3) \end{pmatrix}$$

data value of X

mean value of X

data value of Y

$$Cov(x, y) = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{n}$$

mean value of Y

Recall:

- Spectral Decomposition: a symmetric matrix **A** has a full set of eigenvectors, which can be chosen to be orthogonal. This gives a decomposition

$$\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T,$$

where **Q** is orthogonal and **Λ** is diagonal. The columns of **Q** are eigenvectors, and the diagonal entries λ_j of **Λ** are the corresponding eigenvalues.

- I.e., symmetric matrices are diagonal in some basis.
- A symmetric matrix **A** is positive semidefinite iff each $\lambda_j \geq 0$.

Principle component analysis (PCA)

Step 1: Standardize the Dataset

- **Normalization**: Ensure that each feature has zero mean and unit variance.

Step 2: Compute the Covariance Matrix

- **Covariance Calculation**: Calculate the covariance matrix to examine the correlations between different features in the data.

Step 3: Eigen Decomposition

- **Compute Eigenvalues and Eigenvectors**

Step 4: Sort Eigenvalues and Eigenvectors

- **Sorting**: Sort the eigenvectors by decreasing eigenvalues.

Step 5: Select Principal Components

- **Component Selection**: Choose the top k eigenvectors as the principal components, where k is the number of dimensions you want to keep.

Step 6: Transform the Original Matrix

- **Projection**: Project the original (standardized) data onto the selected principal components to transform your data into a new subspace with reduced dimensions.

$$X_{\text{pca}} = X \times \text{selected_vectors}$$

Example

f1	f2	f3	f4
1	2	3	4
5	5	6	7
1	4	2	3
5	3	2	1
8	1	2	2

Raw data

$$x_{new} = \frac{x - \mu}{\sigma}$$

Standardization formula

	f1	f2	f3	f4
μ =	4	3	3	3.4
σ =	3	1.58114	1.73205	2.30217

Normalization

f1	f2	f3	f4
-1	-0.63246	0	0.26062
0.33333	1.26491	1.73205	1.56374
-1	0.63246	-0.57735	-0.17375
0.33333	0	-0.57735	-1.04249
1.33333	-1.26491	-0.57735	-0.60812

Normalized data

	f1	f2	f3	f4
f1	0.8 - λ	-0.25298	0.03849	-0.14479
f2	-0.25298	0.8- λ	0.51121	0.4945
f3	0.03849	0.51121	0.8 - λ	0.75236
f4	-0.14479	0.4945	0.75236	0.8 - λ

= 0

Av = λv; Av-λv =0;

(A-λI)v = 0

	f1	f2	f3	f4
f1	0.8	-0.25298	0.03849	-0.14479
f2	-0.25298	0.8	0.51121	0.4945
f3	0.03849	0.51121	0.8	0.75236
f4	-0.14479	0.4945	0.75236	0.8

Co-variance matrix A

$$\lambda = 2.51579324\ ,\ 1.0652885\ ,\ 0.39388704\ ,\ 0.02503121$$

$$\begin{pmatrix} 0.800000-\lambda & -(0.252982) & 0.038490 & -(0.144791) \\ -(0.252982) & 0.800000-\lambda & 0.511208 & 0.494498 \\ 0.038490 & 0.511208 & 0.800000-\lambda & 0.752355 \\ -(0.144791) & 0.494498 & 0.752355 & 0.800000-\lambda \end{pmatrix} \times \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix} = 0$$

e1	e2	e3	e4
0.161960	-0.917059	-0.307071	0.196162
-0.524048	0.206922	-0.817319	0.120610
-0.585896	-0.320539	0.188250	-0.720099
-0.596547	-0.115935	0.449733	0.654547

eigenvectors(4 * 4 matrix)

Example

$\lambda = 2.51579324\ ,\ 1.0652885\ ,\ 0.39388704\ ,\ 0.02503121$

e1	e2	e3	e4
0.161960	-0.917059	-0.307071	0.196162
-0.524048	0.206922	-0.817319	0.120610
-0.585896	-0.320539	0.188250	-0.720099
-0.596547	-0.115935	0.449733	0.654547

eigenvectors(4 * 4 matrix)

e1	e2
0.161960	-0.917059
-0.524048	0.206922
-0.585896	-0.320539
-0.596547	-0.115935

Top 2 eigenvectors(4*2 matrix)

f1	f2	f3	f4
-1	-0.63246	0	0.26062
0.33333	1.26491	1.73205	1.56374
-1	0.63246	-0.57735	-0.17375
0.33333	0	-0.57735	-1.04249
1.33333	-1.26491	-0.57735	-0.60812

Normalized data

Feature matrix * top k eigenvectors = Transformed Data

f1	f2	f3	f4
-1.000000	-0.632456	0.000000	0.260623
0.333333	1.264911	1.732051	1.563740
-1.000000	0.632456	-0.577350	-0.173749
0.333333	0.000000	-0.577350	-1.042493
1.333333	-1.264911	-0.577350	-0.608121

(5,4)

*

e1	e2
0.161960	-0.917059
-0.524048	0.206922
-0.585896	-0.320539
-0.596547	-0.115935

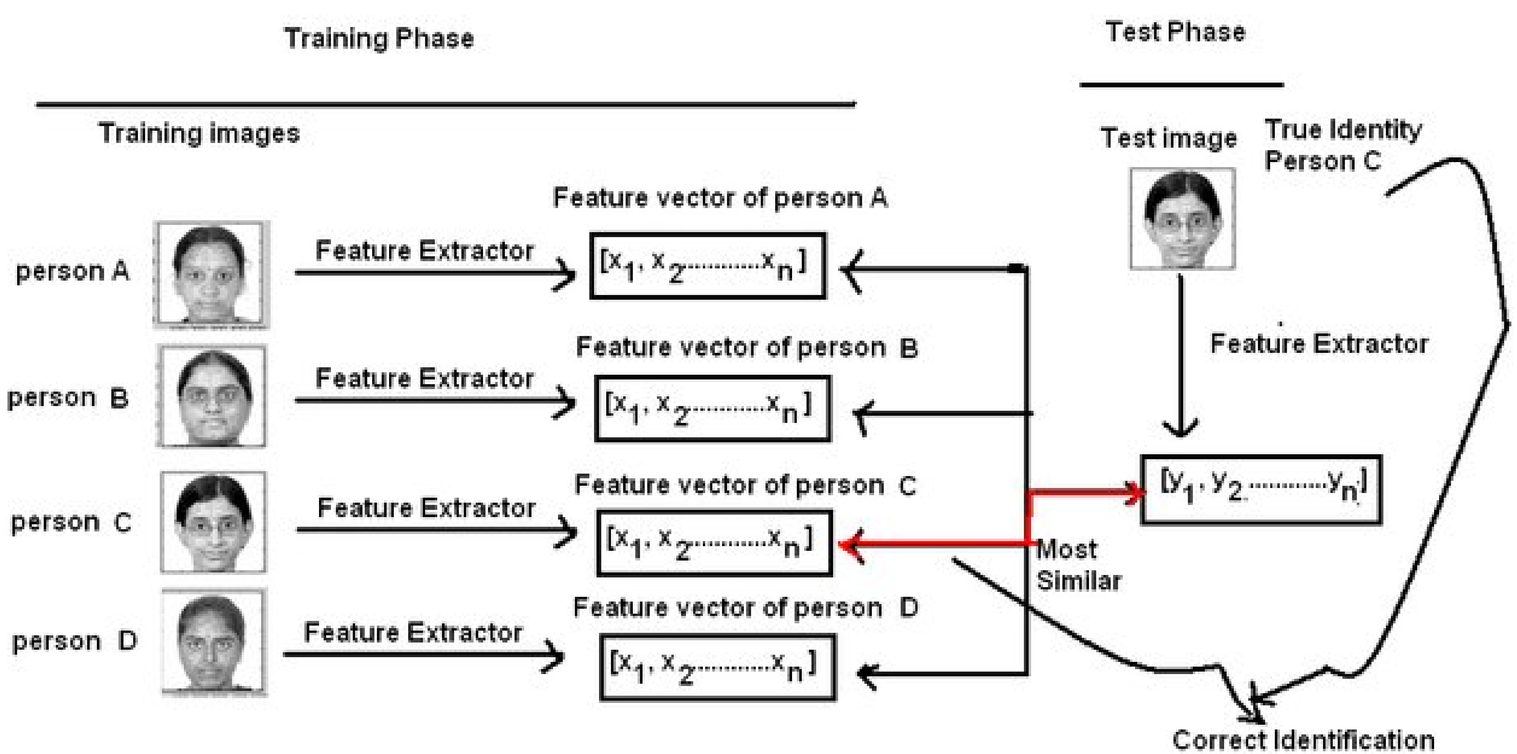
(4,2)

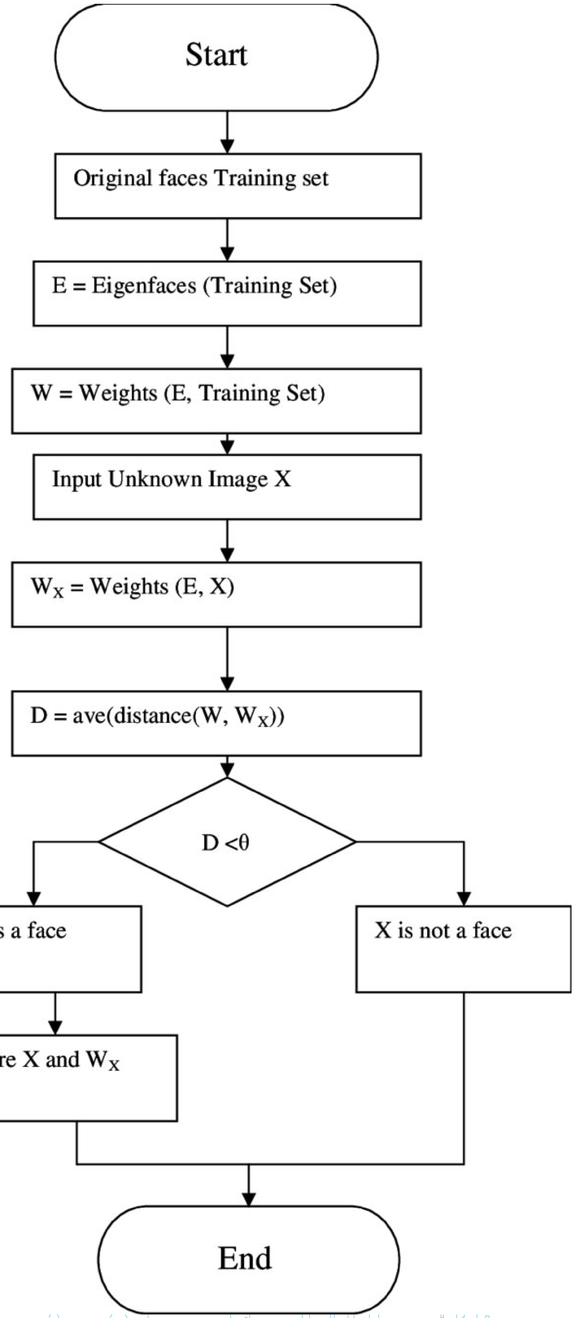
=

nf1	nf2
0.014003	0.755975
-2.556534	-0.780432
-0.051480	1.253135
1.014150	0.000239
1.579861	-1.228917

(5,2)

Application: Face Recognition



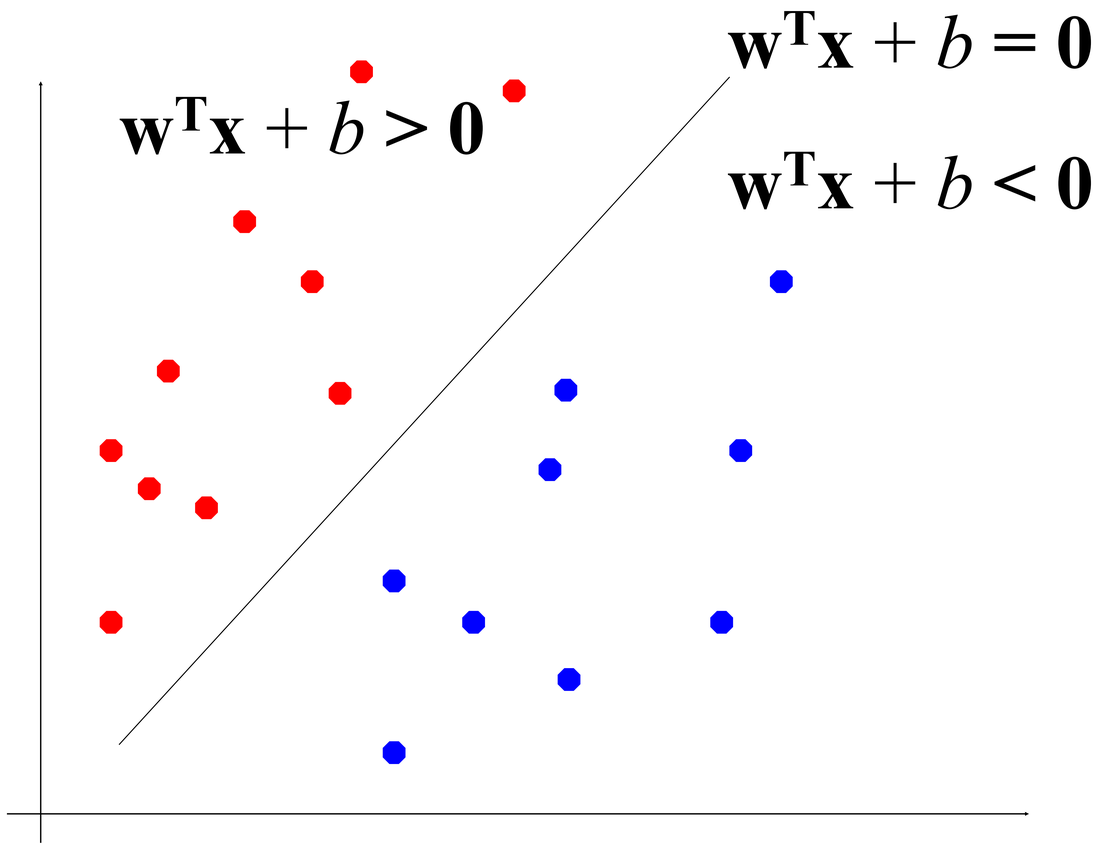


Outline: Matrix Data Mining

1. Introduction to Matrix Data
2. Matrix Decomposition
3. Support Vector Machines
4. K-Means Clustering
5. Application: Genomics Data Analysis

Linear separators

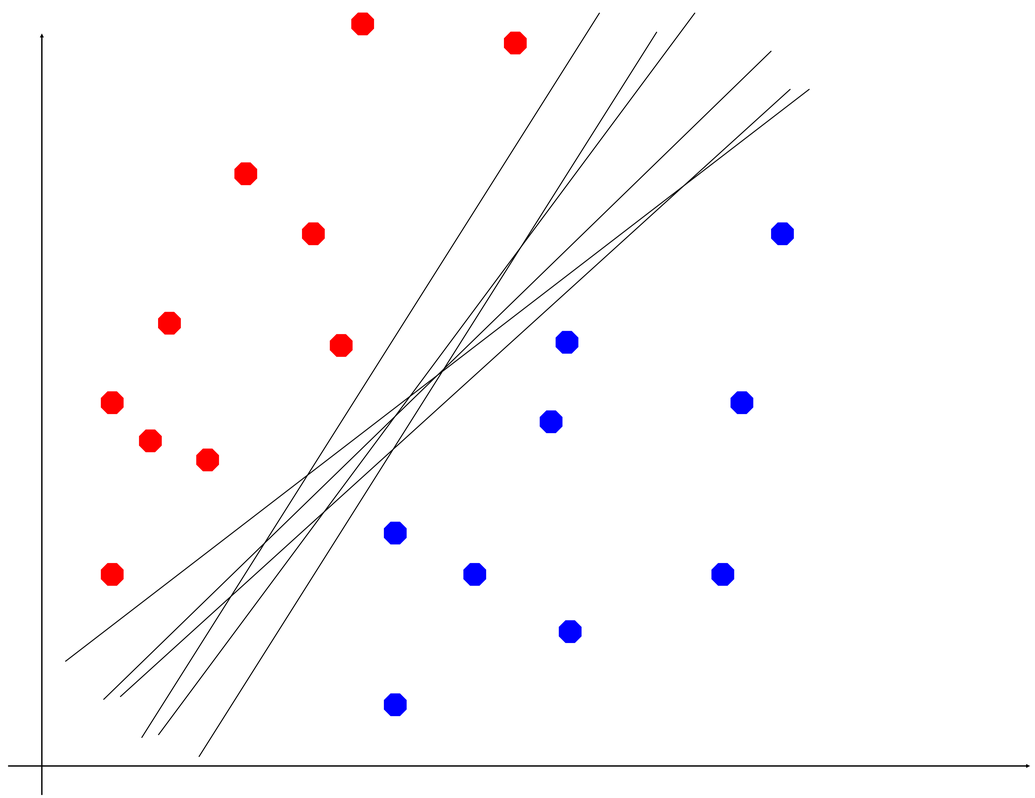
- **Linear Separators:** Binary classification can be viewed as the task of separating classes in feature space:



$$f(\mathbf{x}) = \text{sign}(\mathbf{w}^{\mathbf{T}}\mathbf{x} + b)$$

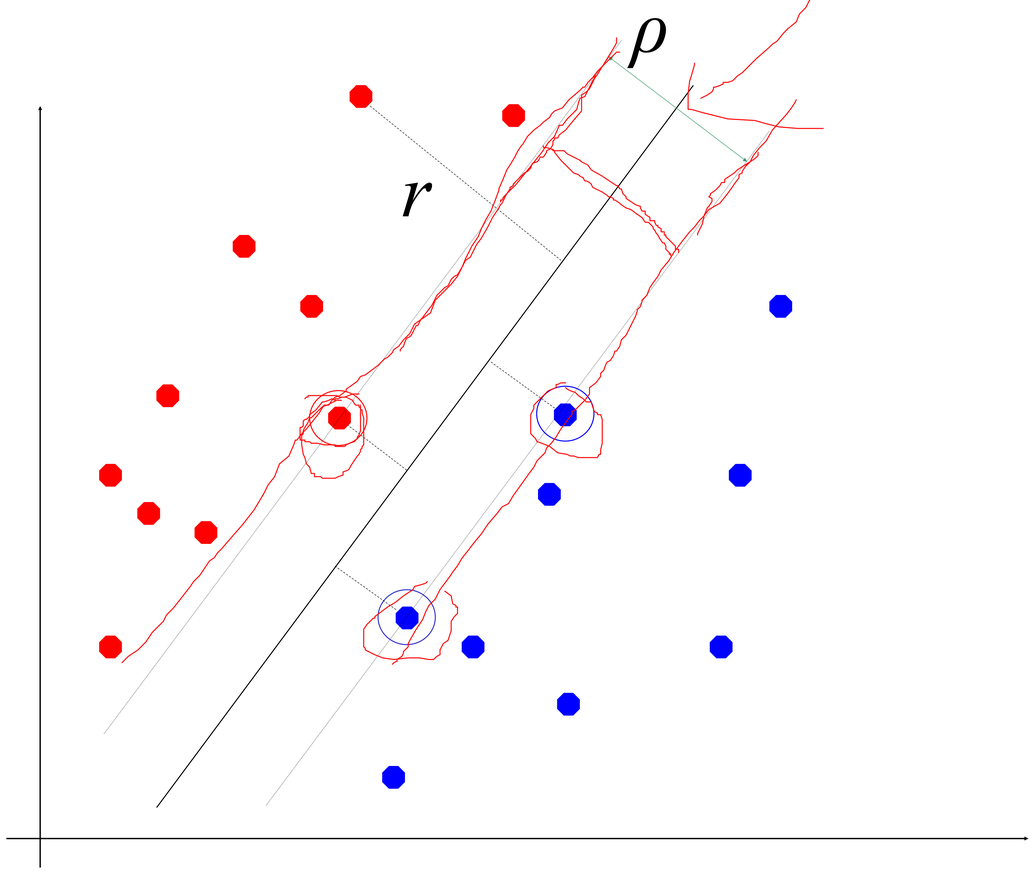
Linear separators

- **Linear Separators:** Which of the linear separators is optimal?



Classification margin

- Distance from example $\mathbf{x}_i$ to the separator is $r = \dfrac{\mathbf{w}^T \mathbf{x}_i + b}{\|\mathbf{w}\|}$
- Examples closest to the hyperplane are ***support vectors***.
- ***Margin*** ρ of the separator is the distance between support vectors.



Classification margin

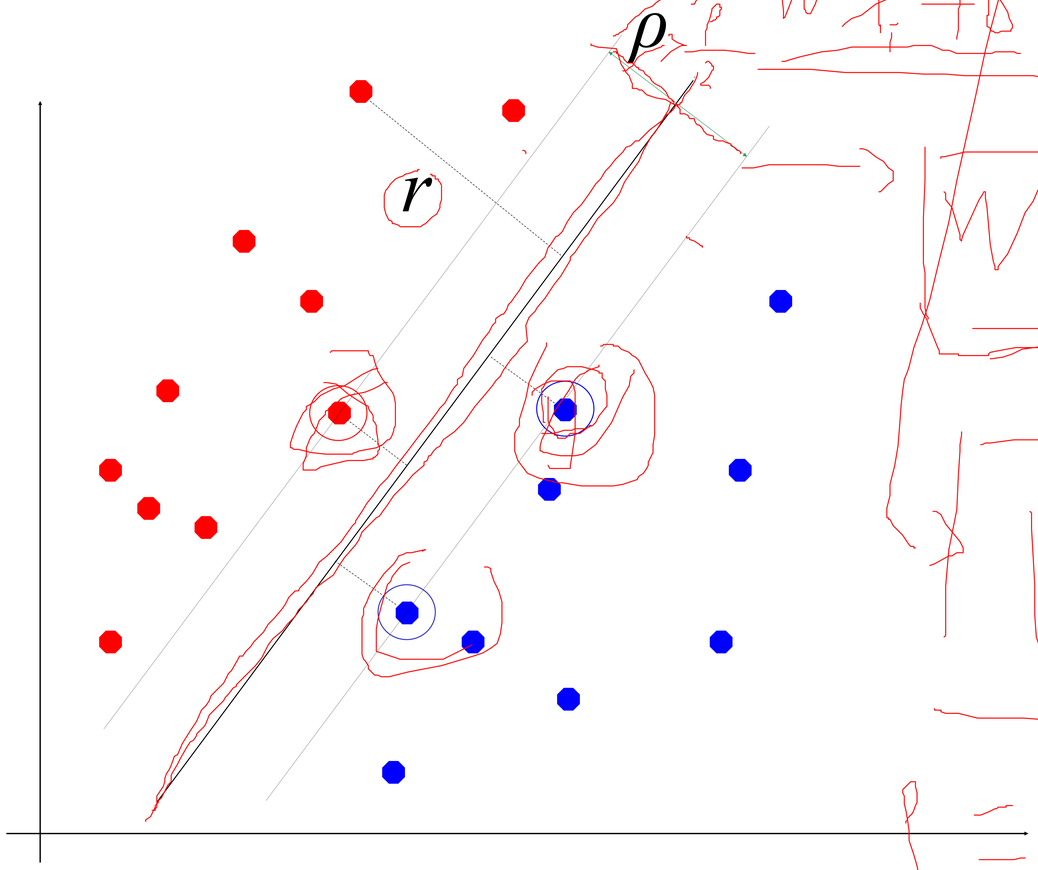
- Distance from example $\mathbf{x}_i$ to the separator is $r = \dfrac{\mathbf{w}^T\mathbf{x}_i + b}{\|\mathbf{w}\|}$ $= \dfrac{\rho}{\|w\|}$
- Examples closest to the hyperplane are ***support vectors***.
- ***Margin*** ρ of the separator is the distance between support vectors.

$w^T x_i + b + \frac{\rho}{2} \geq 0$

$w^T x_i + b - \frac{\rho}{2} \leq 0$

$r = \frac{1}{\|w\|}$

$\rho = 2r$



Linear SVM mathematically

- Let training set $\{(\mathbf{x}_i, y_i)\}_{i=1..n}$, $\mathbf{x}_i \in \mathbf{R}^d$, $y_i \in \{-1, 1\}$ be separated by a hyperplane with margin ρ. Then for each training example $(\mathbf{x}_i, y_i)$:

$$\begin{array}{ll} \mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b \leq -\rho/2 & \text{if } y_i = -1 \\ \mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b \geq \rho/2 & \text{if } y_i = 1 \end{array} \quad \Leftrightarrow \quad y_i(\mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b) \geq \rho/2$$

- For every support vector $\mathbf{x}_s$ the above inequality is an equality. After rescaling $\mathbf{w}$ and b by $\rho/2$ in the equality, we obtain that distance between each $\mathbf{x}_s$ and the hyperplane is

$$r = \frac{y_s(\mathbf{w}^T\mathbf{x}_s + b)}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|}$$

- Then the margin can be expressed through (rescaled) $\mathbf{w}$ and b as:

$$\rho = 2r = \frac{2}{\|\mathbf{w}\|}$$

Linear SVM mathematically

- Then we can formulate the *quadratic optimization problem:*

Find $\mathbf{w}$ and b such that

$$\rho = \frac{2}{\|\mathbf{w}\|} \text{ is maximized}$$

and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$

Which can be reformulated as:

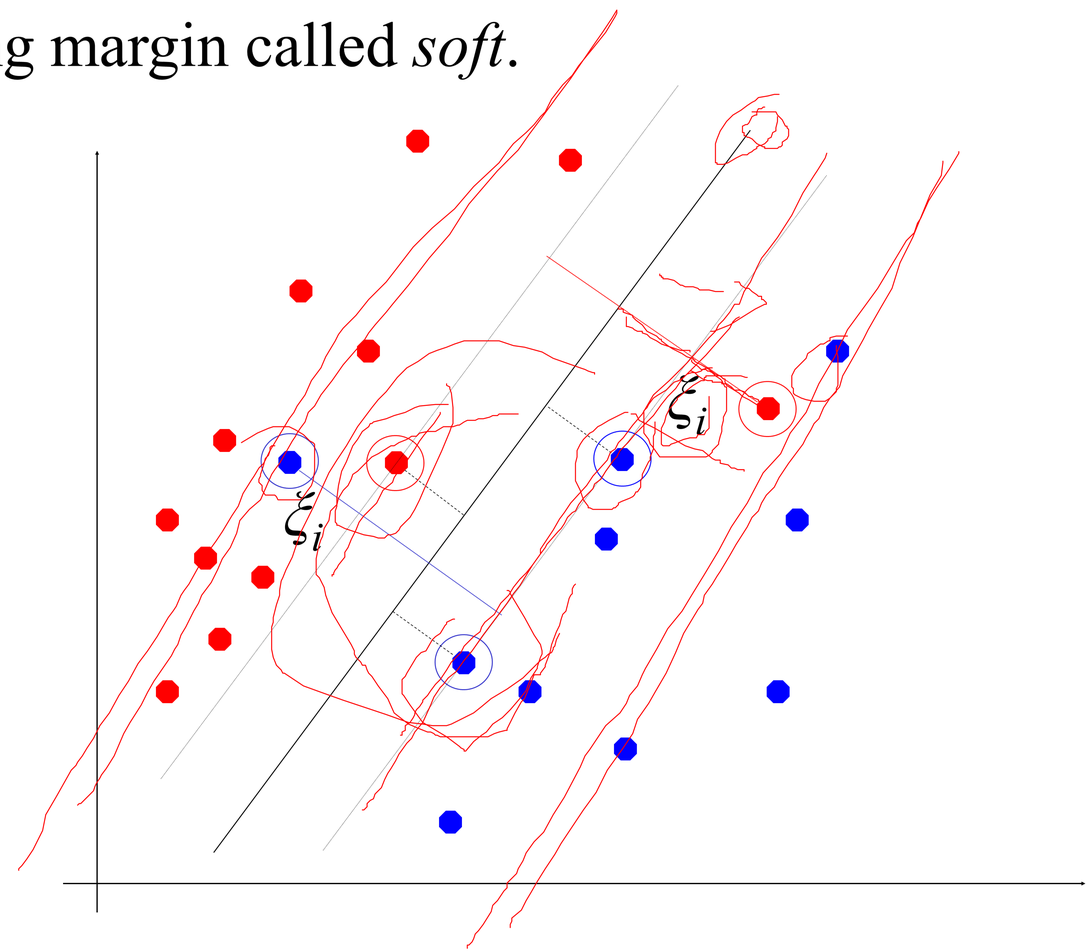
Find $\mathbf{w}$ and b such that

$$\mathbf{\Phi}(\mathbf{w}) = \|\mathbf{w}\|^2 = \mathbf{w}^T\mathbf{w} \text{ is minimized}$$

and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $y_i\,(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$

Soft margin classification

- What if the training set is not linearly separable?
- *Slack variables* ξ_i can be added to allow misclassification of difficult or noisy examples, resulting margin called *soft*.



Soft margin classification mathematically

- The old formulation:

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi}(\mathbf{w}) = \mathbf{w}^T\mathbf{w}$ is minimized
and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$

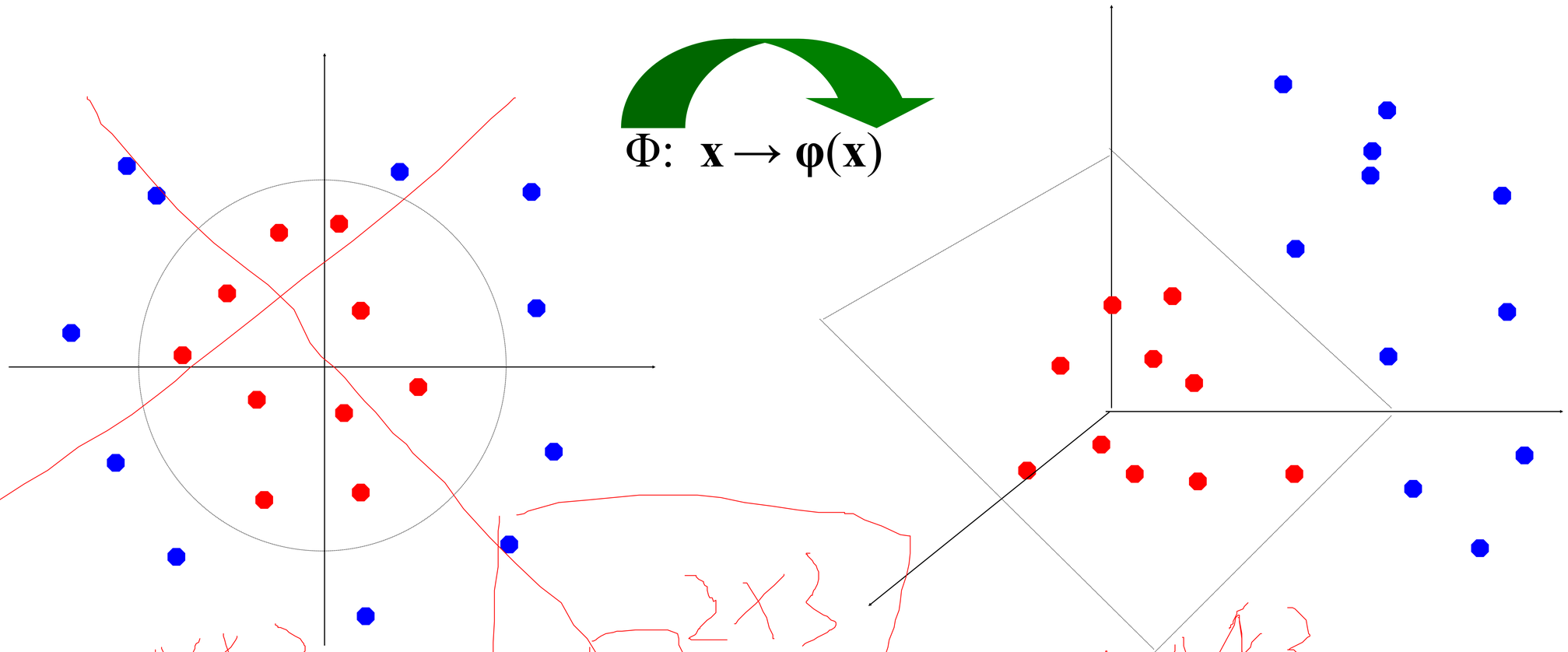
- Modified formulation incorporates slack variables:

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi}(\mathbf{w}) = \mathbf{w}^T\mathbf{w} + C\Sigma\xi_i$ is minimized
and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i$, $\xi_i \geq 0$

- Parameter C can be viewed as a way to control overfitting: it "trades off" the relative importance of maximizing the margin and fitting the training data.

Non-linear SVMs: feature spaces

- General idea: the original feature space can always be mapped to some higher-dimensional feature space where the training set is separable:



$A^{N\times 2} \times B^{2\times 3} = A^{N\times 3}$

Examples of Kernel Functions

- Linear: $K(\mathbf{x}_i,\mathbf{x}_j)= \mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j$
  - Mapping Φ: $\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x})$, where $\boldsymbol{\varphi}(\mathbf{x})$ is $\mathbf{x}$ itself

- Polynomial of power p: $K(\mathbf{x}_i,\mathbf{x}_j)= (1+ \mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j)^p$
  - Mapping Φ: $\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x})$, where $\boldsymbol{\varphi}(\mathbf{x})$ has $\binom{d+p}{p}$ dimensions

- Gaussian (radial-basis function): $K(\mathbf{x}_i,\mathbf{x}_j) = e^{-\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{2\sigma^2}}$
  - Mapping Φ: $\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x})$, where $\boldsymbol{\varphi}(\mathbf{x})$ is *infinite-dimensional*: every point is mapped to *a function* (a Gaussian); combination of functions for support vectors is the separator.

- Higher-dimensional space still has *intrinsic* dimensionality d (the mapping is not *onto*), but linear separators in it correspond to *non-linear* separators in original space.

Outline: Matrix Data Mining

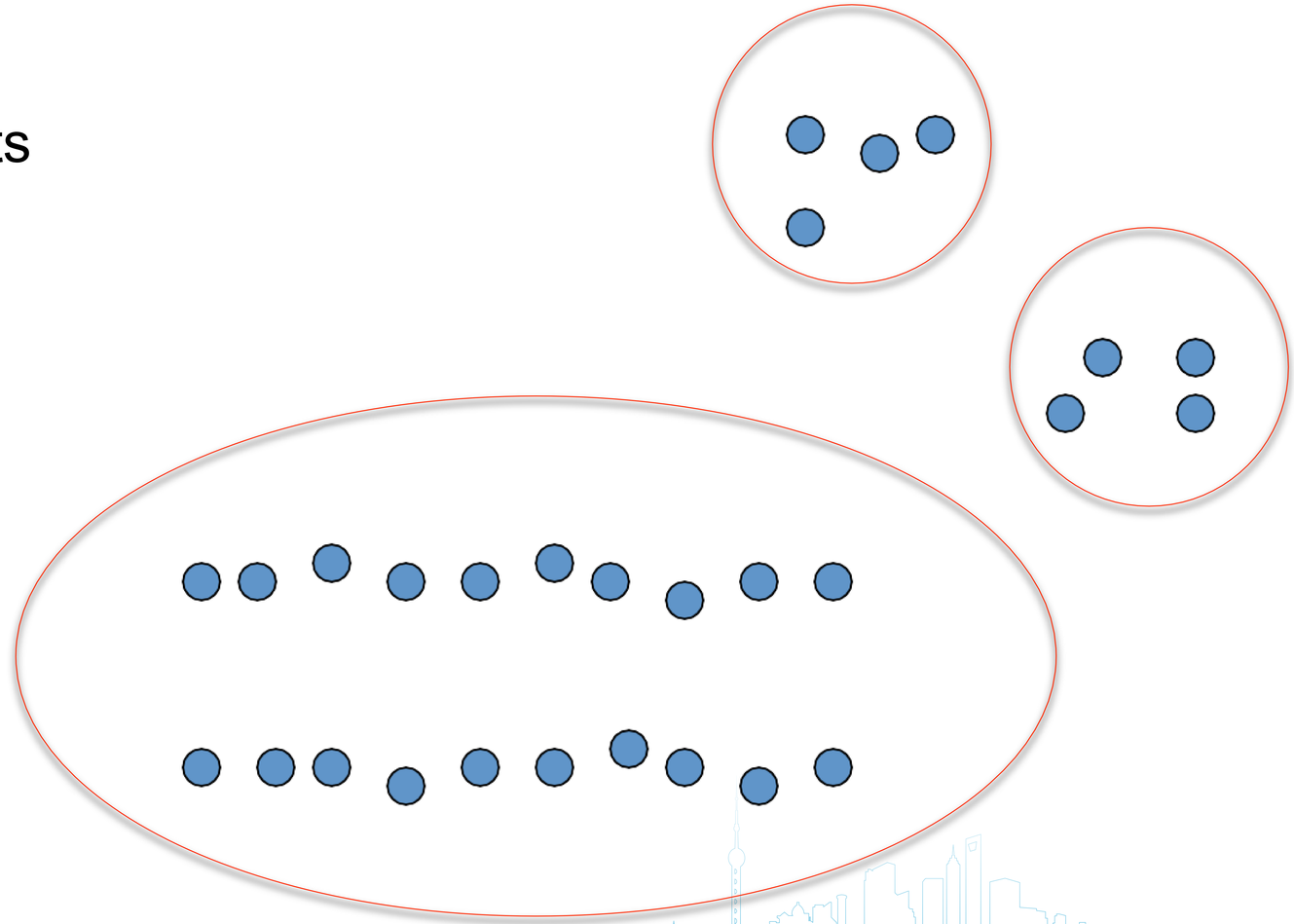
1. Introduction to Matrix Data
2. Principle Component Analysis
3. Support Vector Machines
4. K-Means Clustering
5. Application: Genomics Data Analysis

Clustering

Clustering:

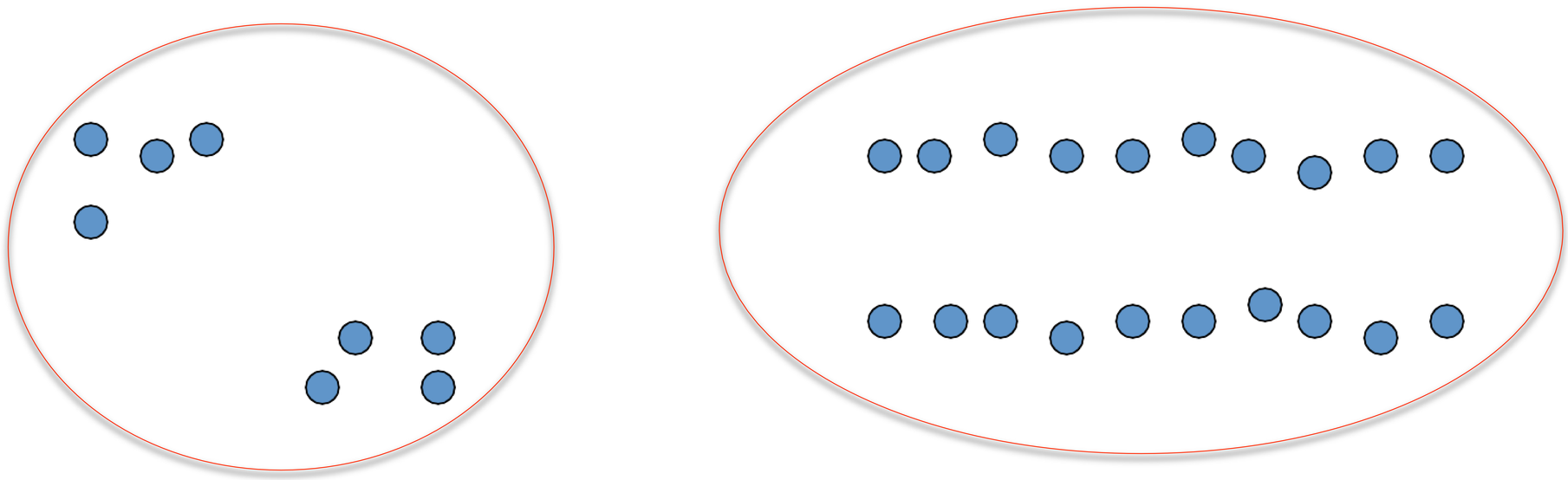
- Unsupervised learning
- Requires data, but no labels
- Detect patterns e.g. in
  - Group emails or search results
  - Customer shopping patterns
  - Regions of images
- Useful when don't know what you're looking for
- But: can get gibberish

- Basic idea: group together similar instances
- Example: 2D point patterns



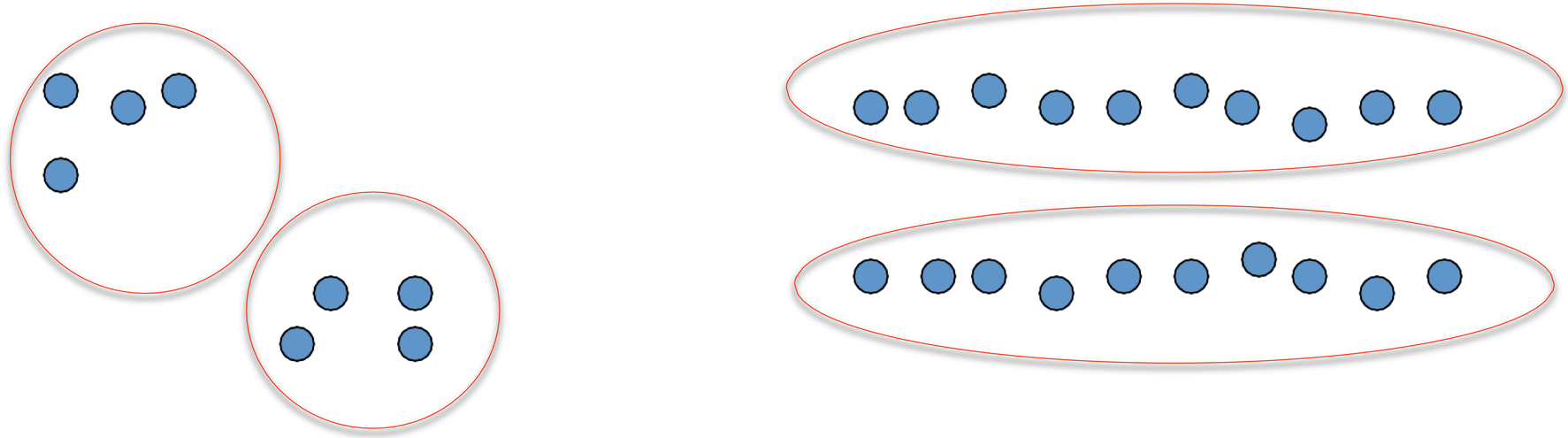
Clustering

- Basic idea: group together similar instances
- Example: 2D point patterns



Clustering

- Basic idea: group together similar instances
- Example: 2D point patterns



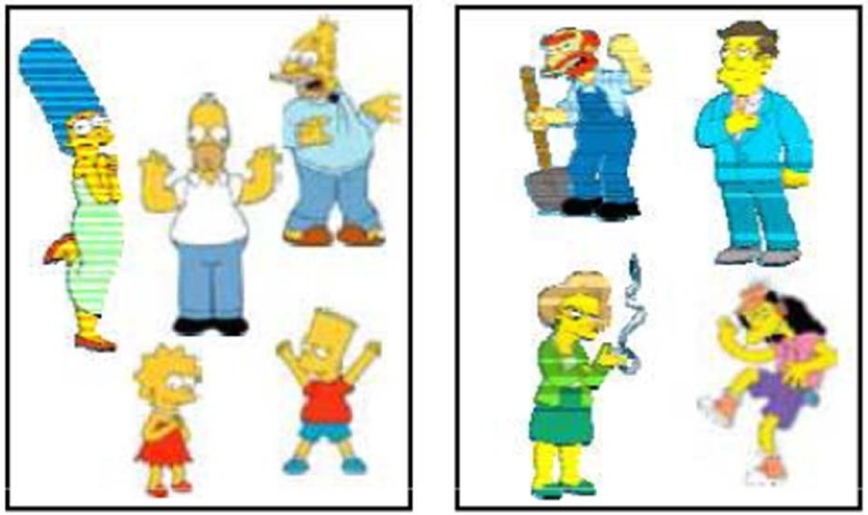
- What could "similar" mean?
  - One option: small Euclidean distance (squared)

$$\mathrm{dist}(\vec{x}, \vec{y}) = ||\vec{x} - \vec{y}||_2^2$$

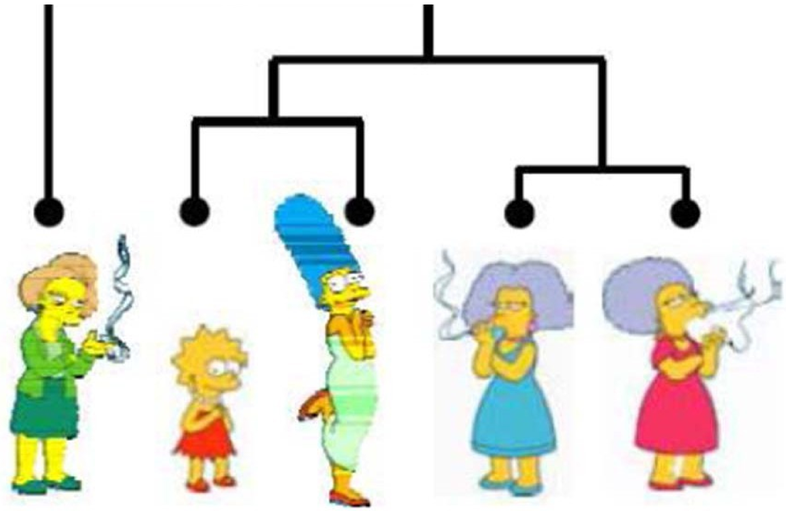
  - Clustering results are crucially dependent on the measure of similarity (or distance) between "points" to be clustered

Clustering algorithms

- Partition algorithms (Flat)
  - K–means
  - Mixture of Gaussian
  - Spectral Clustering

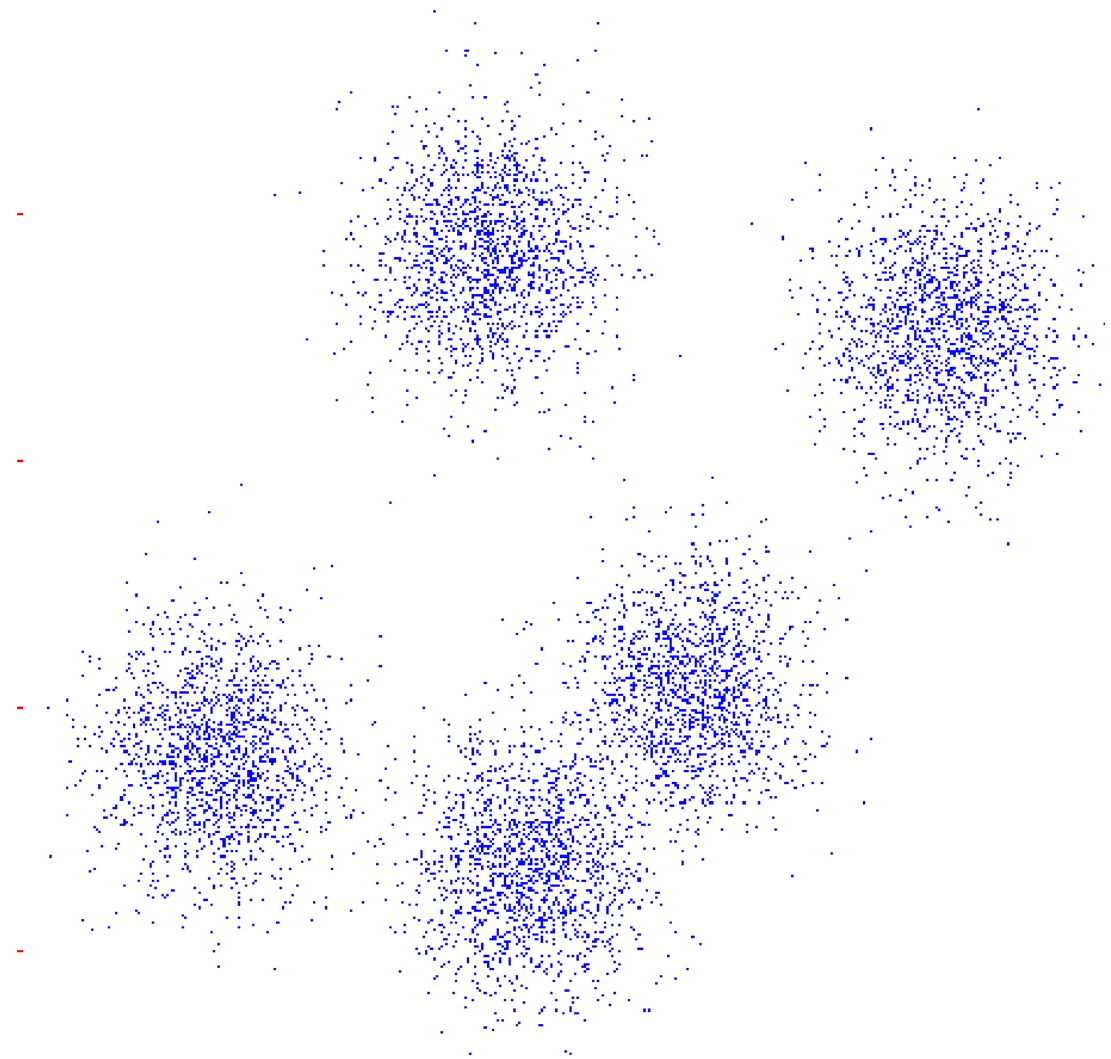


- Hierarchical algorithms
  - Bottom up – agglomerative
  - Top down – divisive



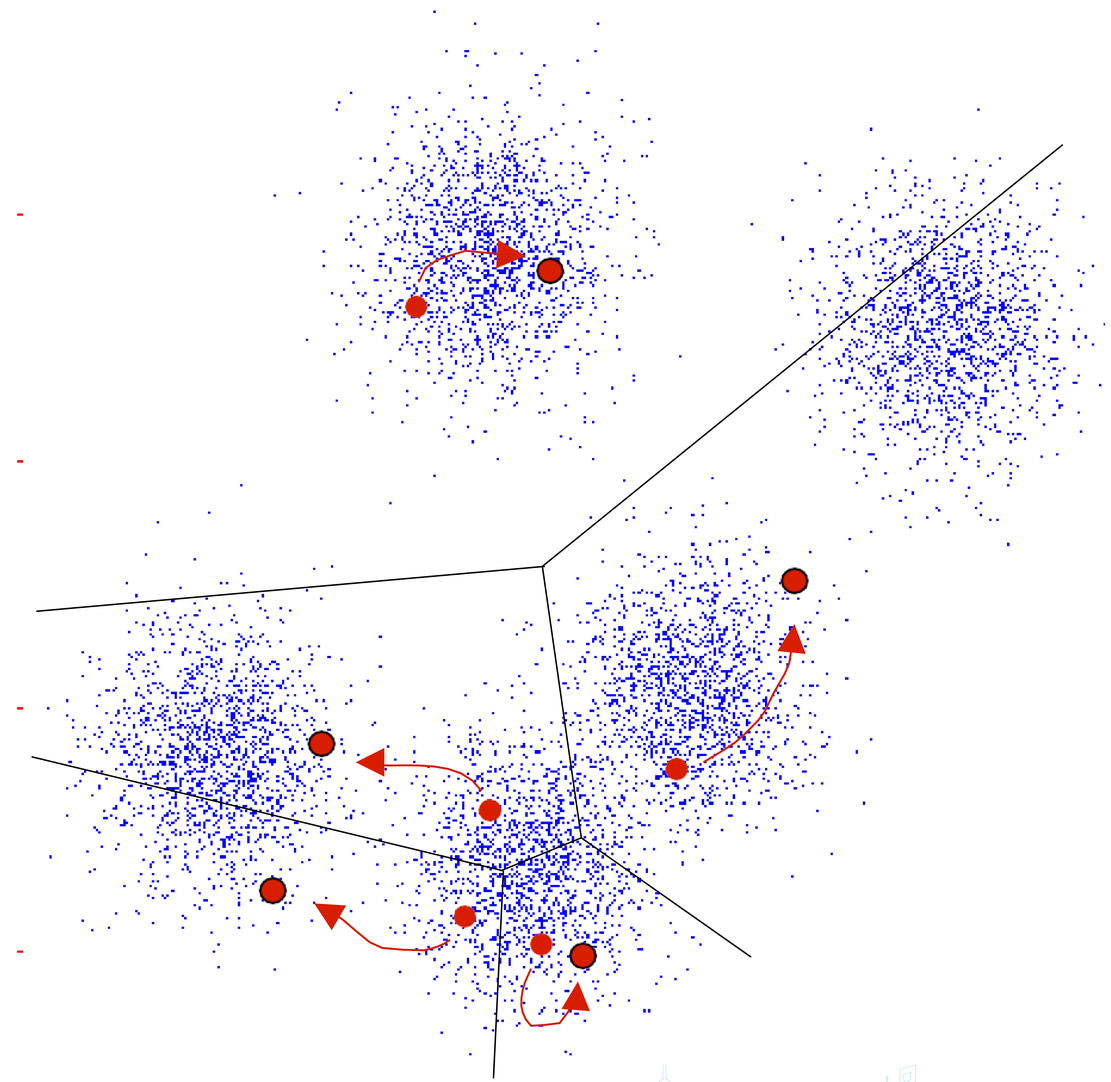
K-Means

- An iterative clustering algorithm
  - Initialize: Pick *K* random points as cluster centers
  - Alternate:
    1. Assign data points to closest cluster center
    2. Change the cluster center to the average of its assigned points
  - Stop when no points' assignments change

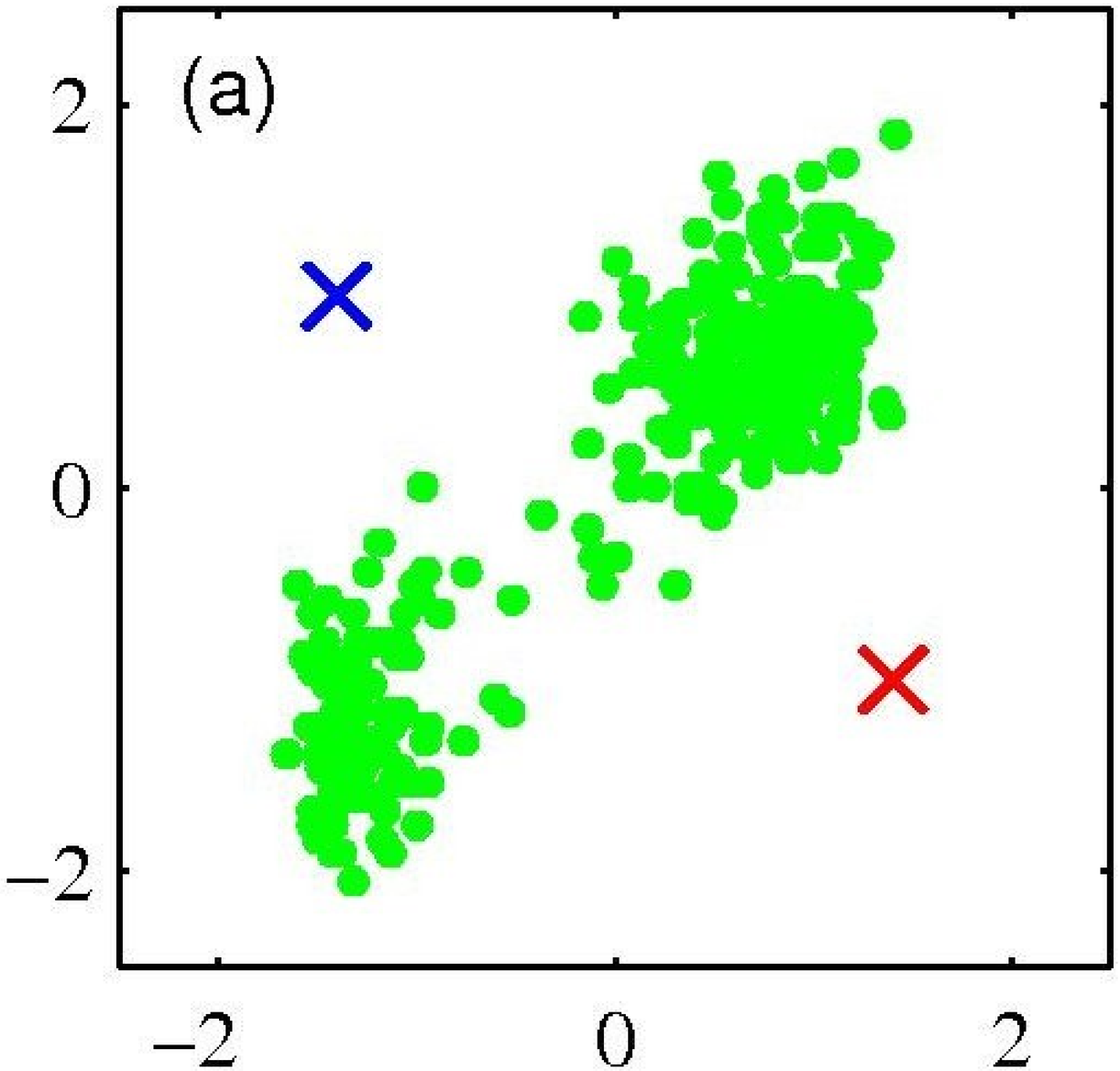


K-Means

- An iterative clustering algorithm
  - Initialize: Pick *K* random points as cluster centers
  - Alternate:
    1. Assign data points to closest cluster center
    2. Change the cluster center to the average of its assigned points
  - Stop when no points' assignments change



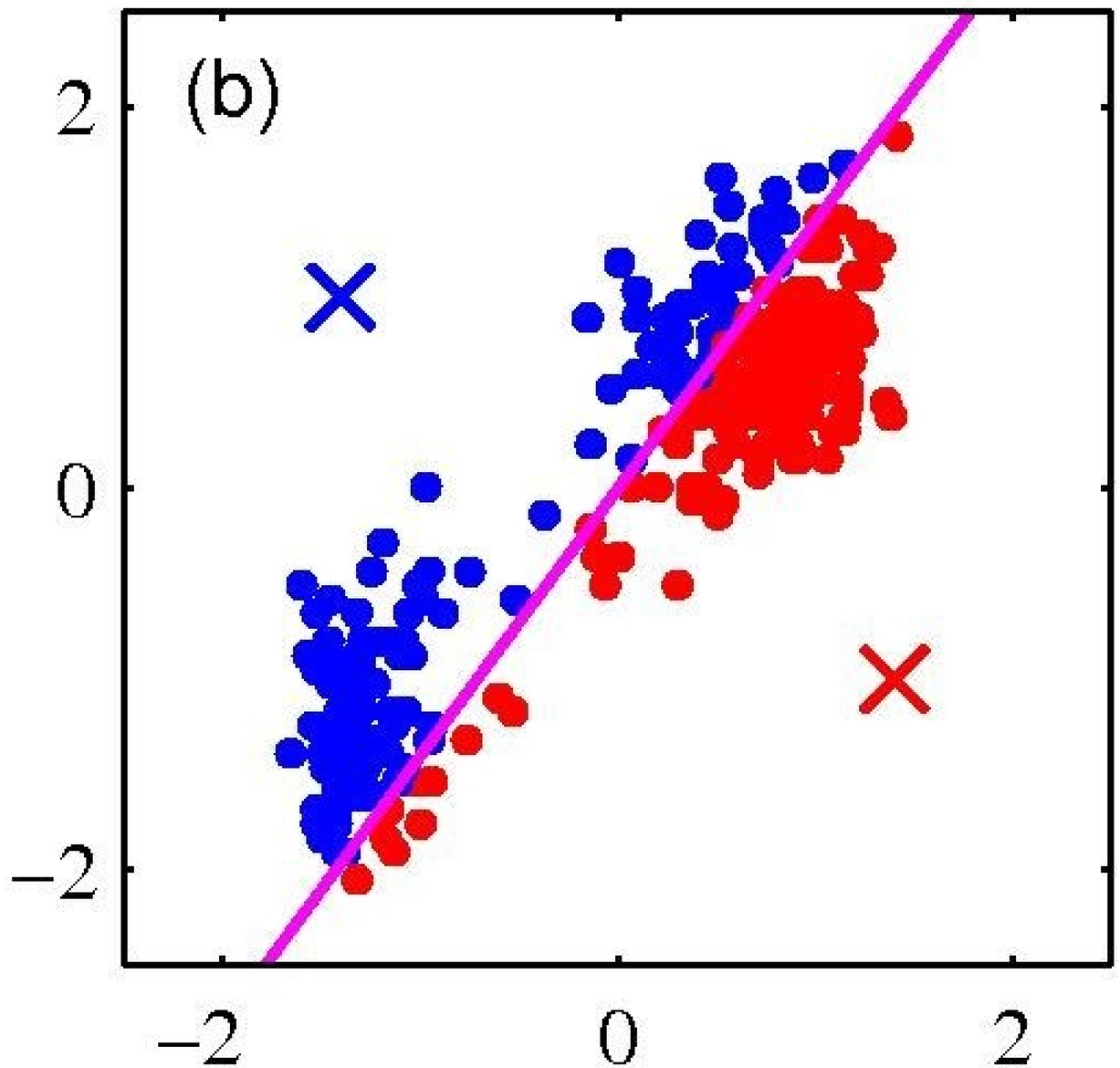
K-Means



- Pick *K* random points as cluster centers (means)

Shown here for *K*=2

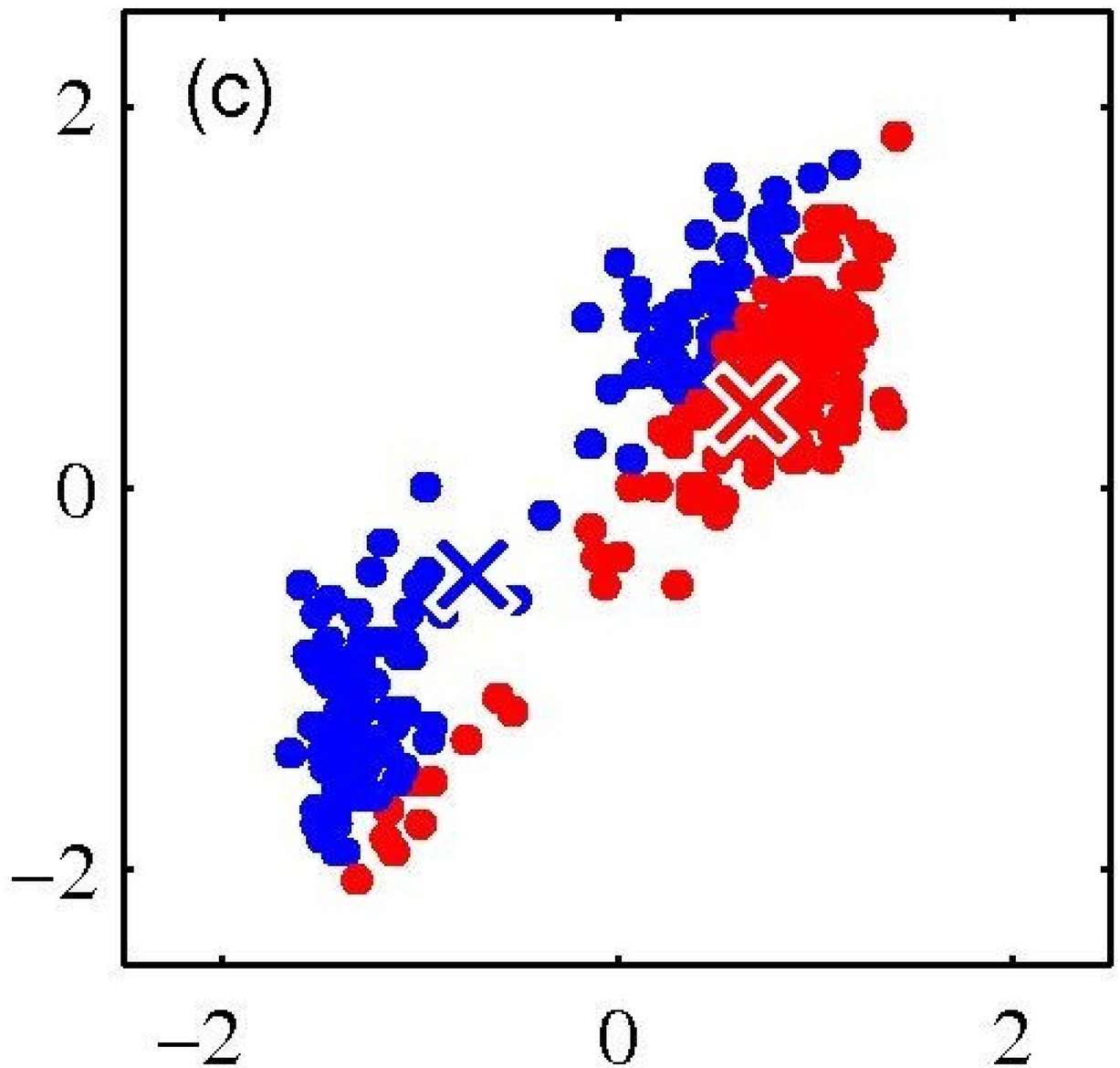
K-Means clustering: Example



Iterative Step 1

- Assign data points to closest cluster center

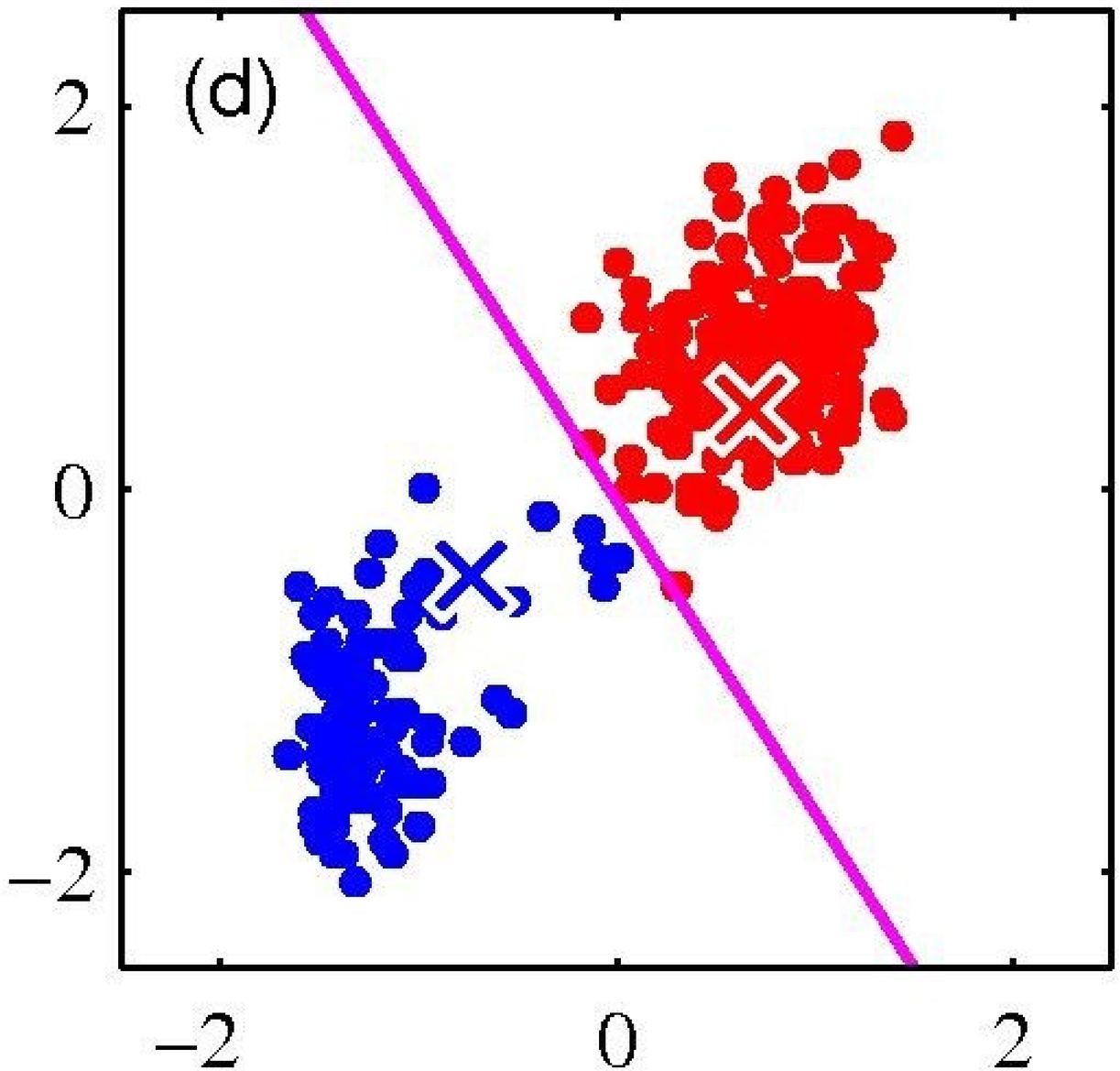
K-Means clustering: Example



Iterative Step 2

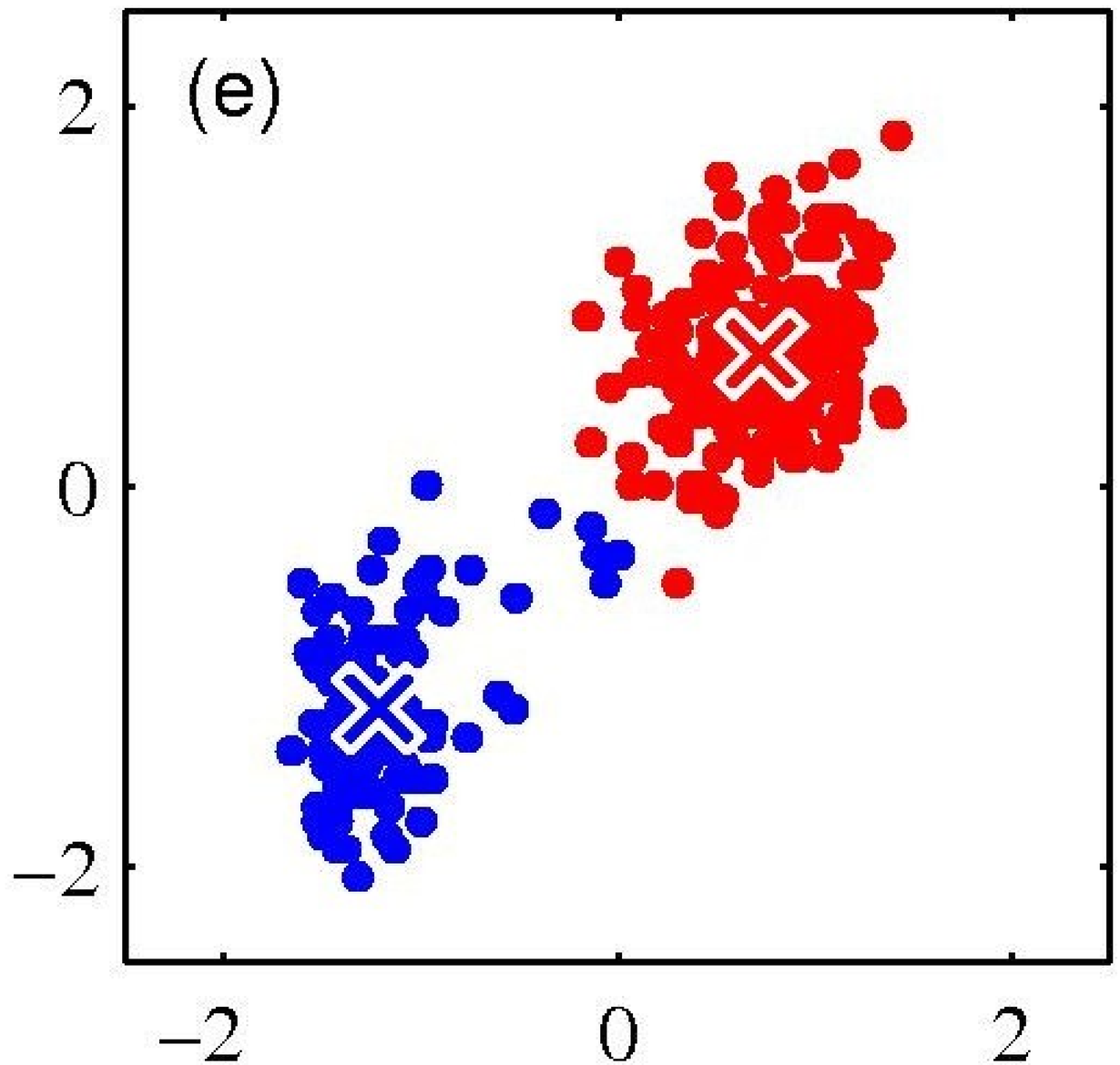
- Change the cluster center to the average of the assigned points

K-Means clustering: Example

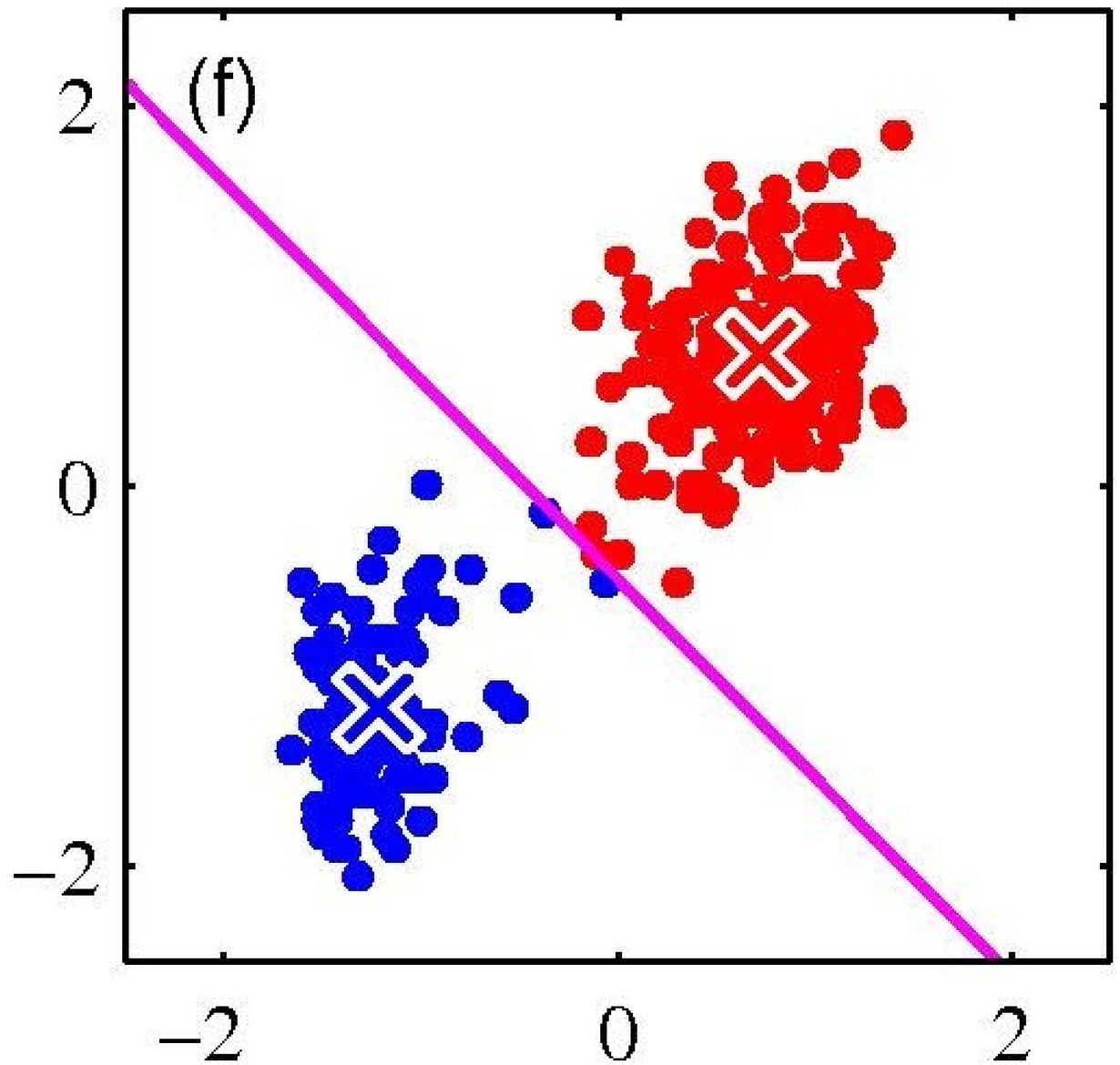


- Repeat until convergence

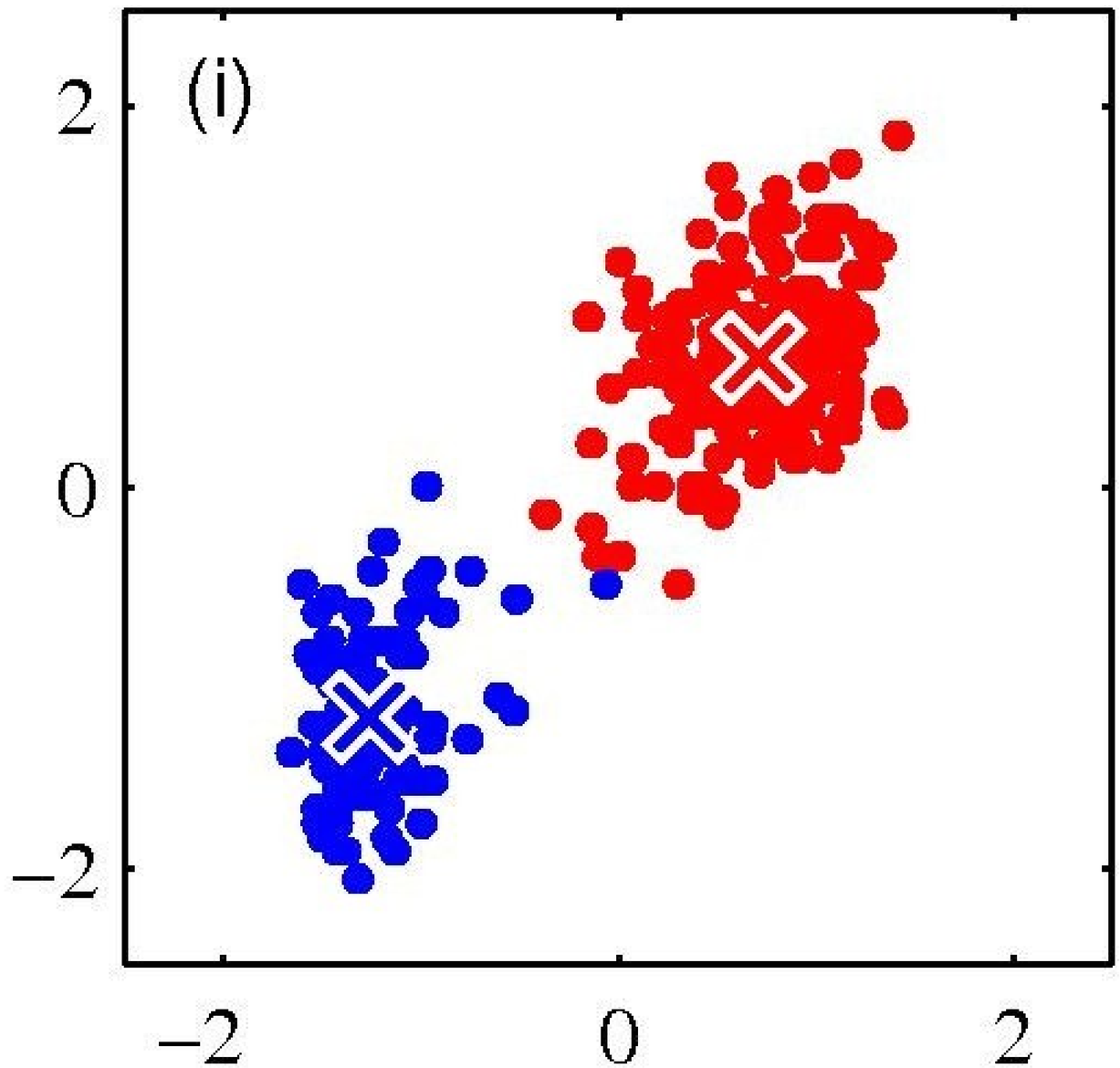
K-Means clustering: Example



K-Means clustering: Example



K-Means clustering: Example



Email: min.shi@louisiana.edu