



The Introduction To Artificial Intelligence

Yuni Zeng yunizeng@zstu.edu.cn
2025-2026-1

The Introduction to Artificial Intelligence

- Teaching hours: 32 hours
- Assessment:

Assessment Methods	Assessment Requirements	Assessment Weighting
Homework	4-6 times (50%)	50% of grade
Literature review	Presentation (50%)	
in-class tests	-	10% of grade
Final exam	Open-book examination	40% of grade

- No Copy! No Plagiarize!

The Introduction to Artificial Intelligence

- Literature Review: **25% final grade**
 - 分组: 每组最多5人，分组结果这周内确定。
 - 提交: 报告, 汇报

The Introduction to Artificial Intelligence

- Part I Brief Introduction to AI & Different AI tribes
- Part II Knowledge Representation & Reasoning
- Part III AI GAMES and Searching
- Part IV Model Evaluation and Selection
- Part V Machine Learning



Machine Learning

Supervised
learning

Unsupervised
learning

Reinforcement
learning

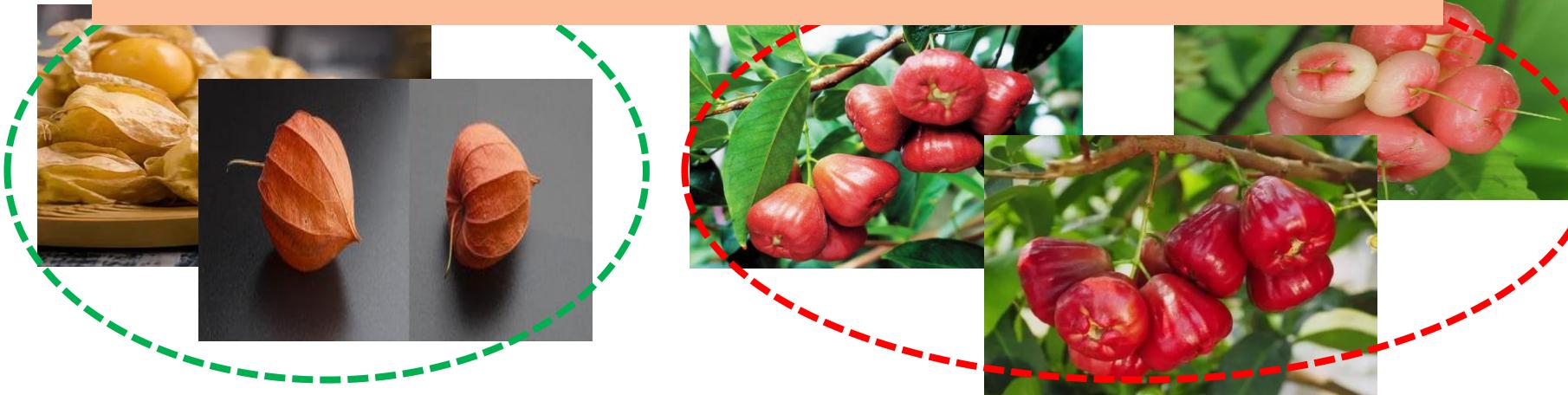
Unsupervised learning

- Clustering
 - K-means method
 - Spectral clustering
- Representation learning

Clustering



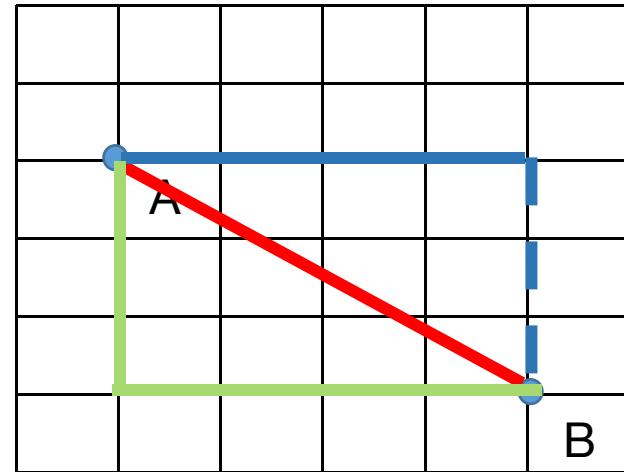
Clustering: Grouping a set of data in such a way that data in the same cluster are more similar to each other than to those in other clusters.



Clustering

□ Distance Metrics

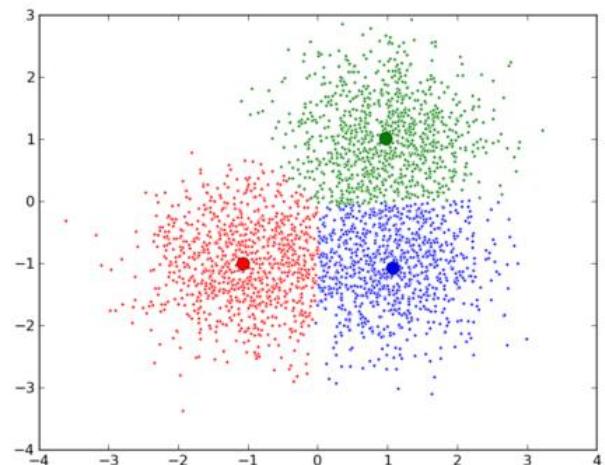
- Euclidean distance
- $d_e(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$
- Sum of squared distance
- $d_q(x, y) = \sum_{i=1}^n (x_i - y_i)^2$
- Manhattan distance
- $d_m(x, y) = \sum_{i=1}^n |x_i - y_i|$
- Chebyshev distance
- $d_c(x, y) = \max_{i=1,\dots,n} |x_i - y_i|$



Clustering

□ K-means Clustering

- K-means clustering is a sort of clustering algorithm, and it is popular for cluster analysis in data mining.
- K-means clustering aims to partition **N** observations into **K** clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster.

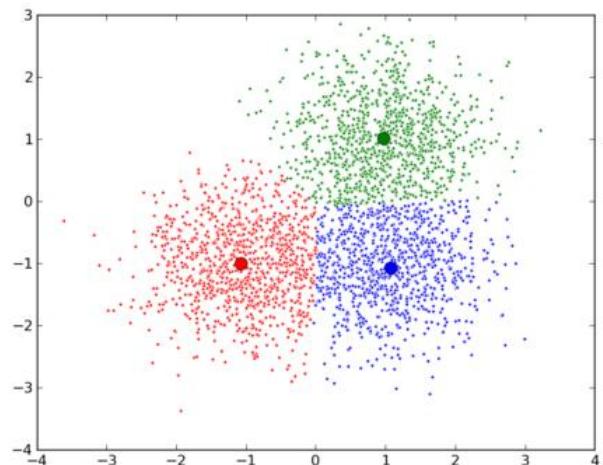


Clustering

□ K-means Clustering

- High intra-clustering similarity
- Low inter-clustering similarity.
- So,

$$object : \sum_{i=1}^N \min_{u_j \in C} \|x_i - u_j\|^2$$



Clustering

□ K-means Clustering

e.g. $k=2$

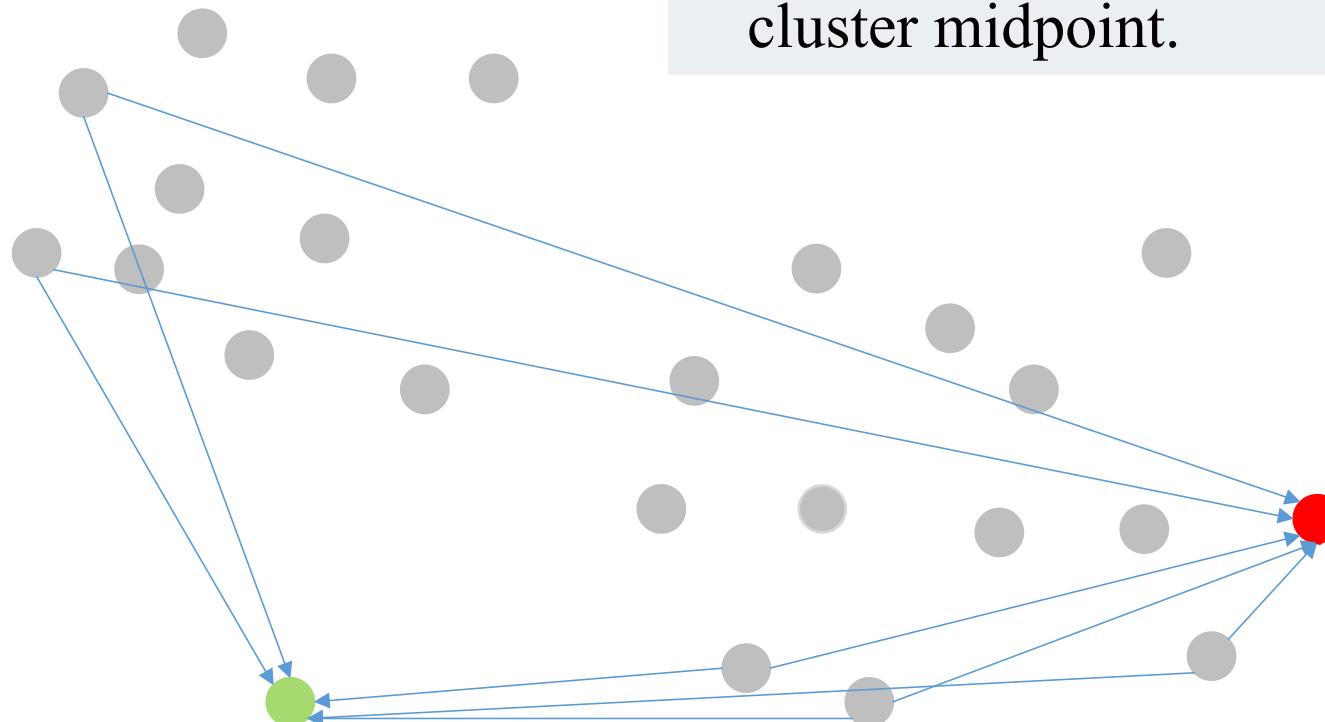
First the k cluster midpoints μ_1, \dots, μ_k are randomly or manually initialized.



Clustering

□ K-means Clustering

e.g. $k=2$



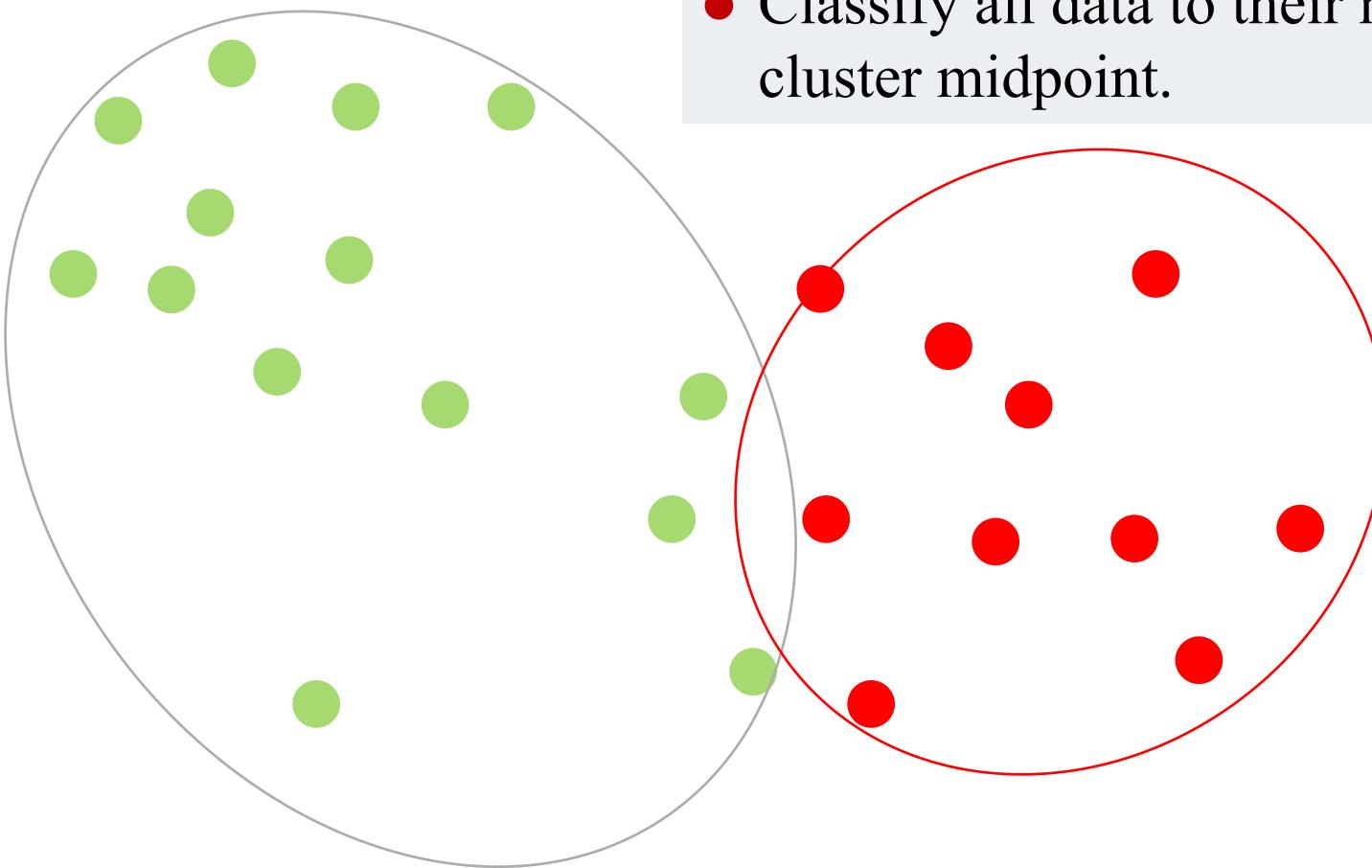
Then the following two steps are
repeatedly carried out:

- Classify all data to their nearest cluster midpoint.

Clustering

□ K-means Clustering

e.g. $k=2$



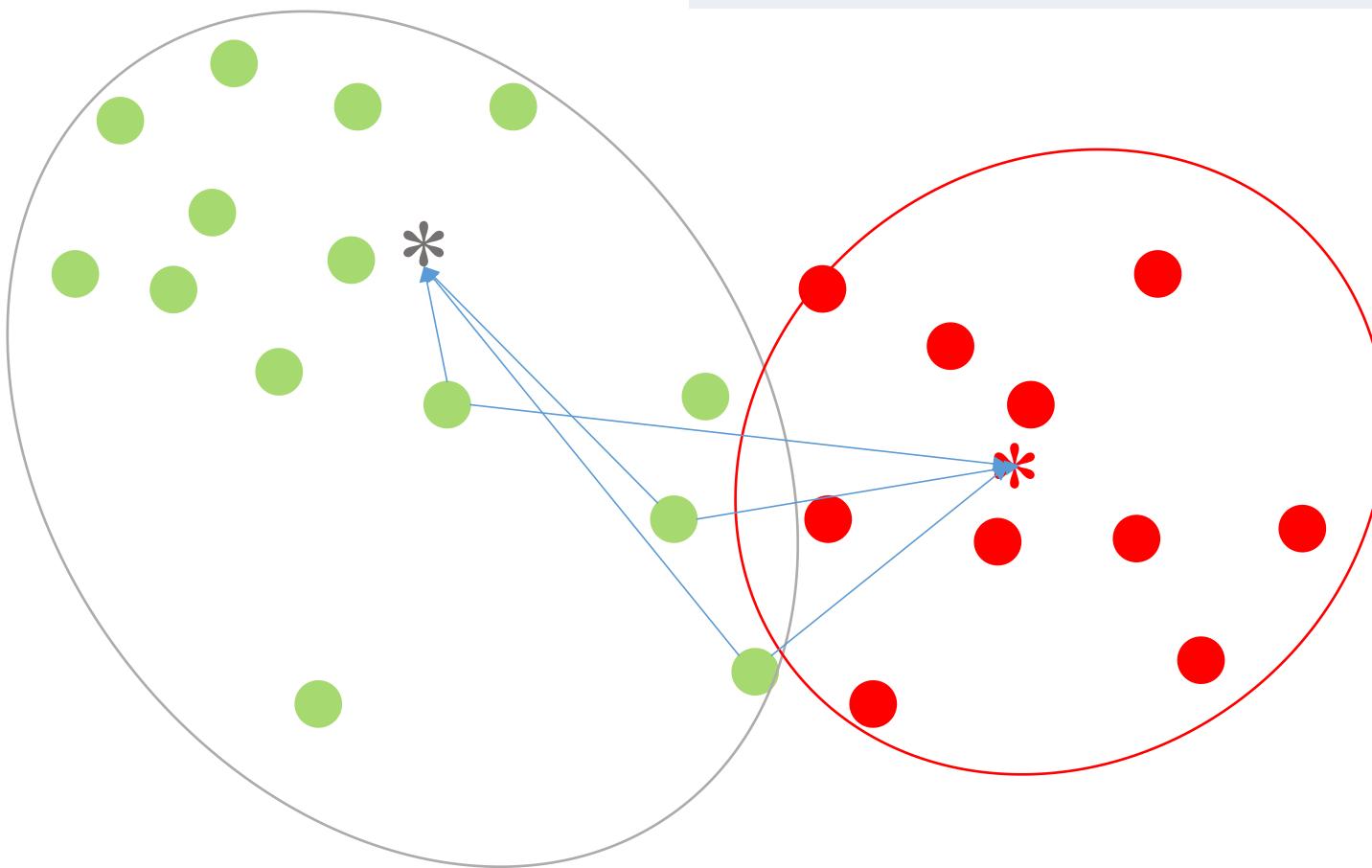
Then the following two steps are **repeatedly** carried out:

- Classify all data to their nearest cluster midpoint.

Clustering

□ K-means Clustering

e.g. $k=2$



Then the following two steps are **repeatedly** carried out:

- Re-compute of the cluster midpoint.

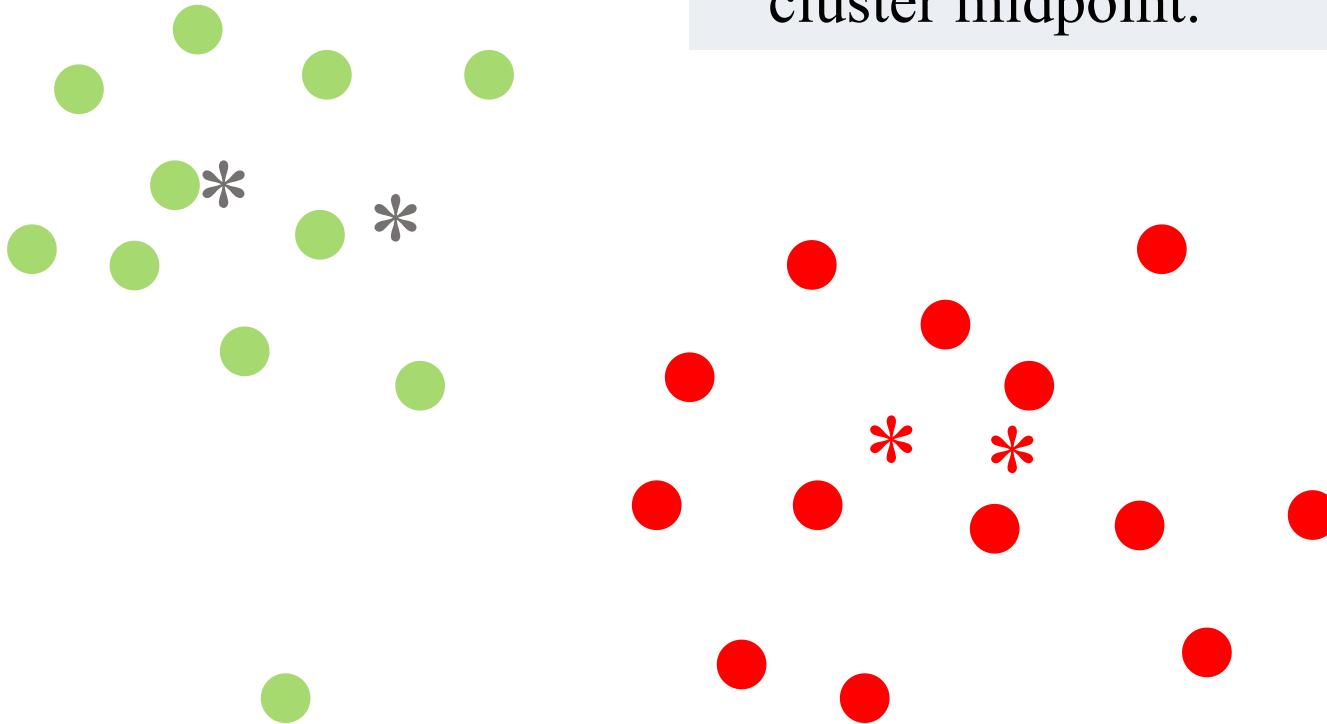
Clustering

□ K-means Clustering

e.g. $k=2$

Then the following two steps are
repeatedly carried out:

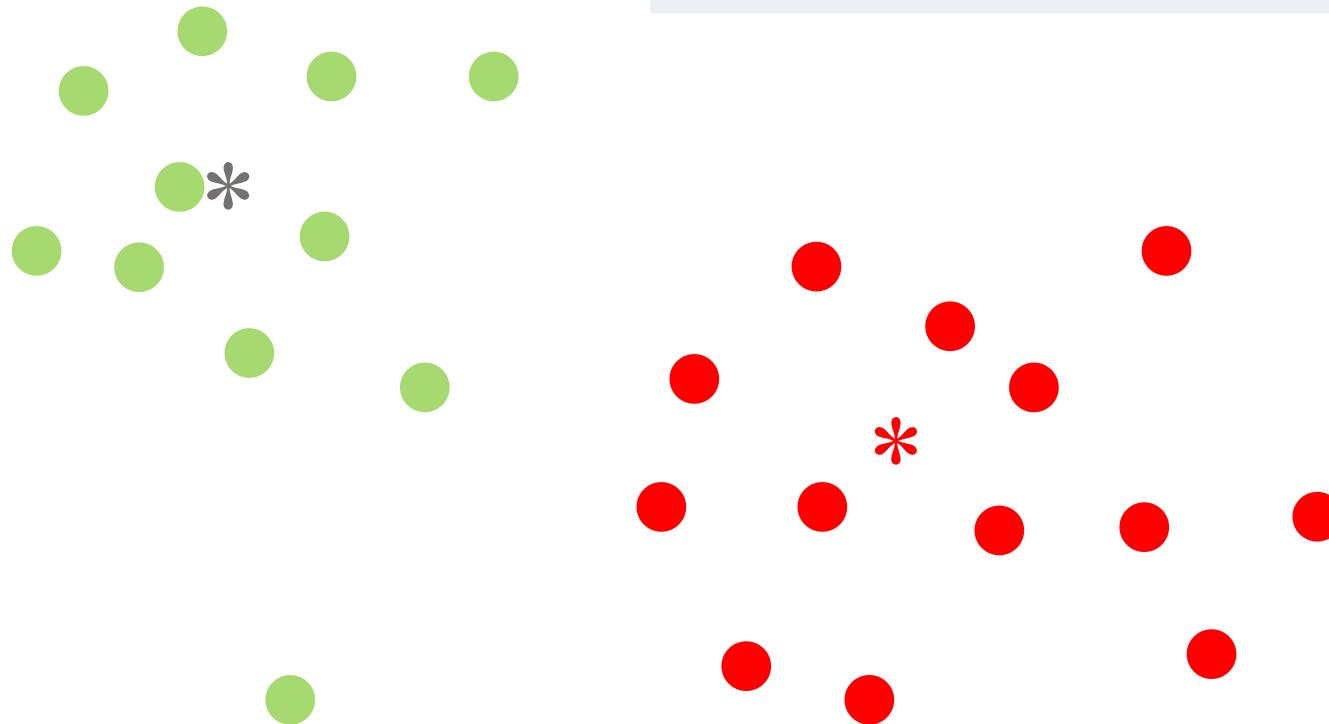
- Classify all data to their nearest cluster midpoint.



Clustering

□ K-means Clustering

e.g. $k=2$



Then the following two steps are
repeatedly carried out:

- Re-compute of the cluster midpoint.

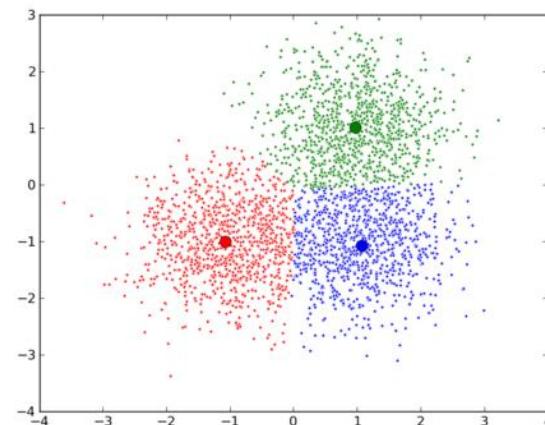
The algorithm converges

Clustering

□ K-means Clustering

The following two steps are **repeatedly** carried out:

- Initialize the midpoints
- Repeat the following 2 steps
 - Classify all data to their nearest cluster midpoint.
 - Re-compute of the cluster midpoint.
- Until the algorithm converges



Clustering

□ K-means Clustering -- Example

Try to cluster these samples X by k-means clustering, when k =2,

$$X = \begin{bmatrix} 0 & 0 & 1 & 5 & 5 \\ 2 & 0 & 0 & 0 & 2 \end{bmatrix}$$

- Initialize the midpoints
- Repeat the following 2 steps
 - Classify all data to their nearest cluster midpoint.
 - Re-compute of the cluster midpoint.
- Until the algorithm converges

Clustering

□ K-means Clustering -- Example

Try to cluster these samples X by k-means clustering, when $k=2$,

$$X = \begin{bmatrix} 0 & 0 & 1 & 5 & 5 \\ 2 & 0 & 0 & 0 & 2 \end{bmatrix}$$

Step 1: First the k cluster midpoints μ_1, \dots, μ_k are randomly or manually initialized.

→ Suppose $m_1^{(0)} =$

$(0, 2)^T$ is the midpoint of cluster $G_1^{(0)}$, $m_2^{(0)} = (0, 0)^T$
is the midpoint of cluster $G_2^{(0)}$

Clustering

□ K-means Clustering -- Example

Try to cluster these samples X by k-means clustering, when $k=2$,

$$X = \begin{bmatrix} 0 & 0 & 1 & 5 & 5 \\ 2 & 0 & 0 & 0 & 2 \end{bmatrix}$$

Step 2: **Classify all data to their nearest cluster midpoint.**

→ Calculate the distance from $x_3 = (1, 0)^T$, $x_4 = (5, 0)^T$, $x_5 = (5, 2)^T$ to the midpoints $m_1^{(0)}$, $m_2^{(0)}$:

$x_3 = (1, 0)^T$, $d(x_3, m_1^{(0)}) = 5$, $d(x_3, m_2^{(0)}) = 1$, So x_3 is $G_2^{(0)}$.

$x_4 = (5, 5)^T$, $d(x_4, m_1^{(0)}) = 29$, $d(x_4, m_2^{(0)}) = 25$, So x_4 is $G_2^{(0)}$.

$x_5 = (5, 2)^T$, $d(x_5, m_1^{(0)}) = 25$, $d(x_5, m_2^{(0)}) = 29$, So x_5 is $G_1^{(0)}$.

Clustering

□ K-means Clustering -- Example

Try to cluster these samples X by k-means clustering, when $k=2$,

$$X = \begin{bmatrix} 0 & 0 & 1 & 5 & 5 \\ 2 & 0 & 0 & 0 & 2 \end{bmatrix}$$

Step 3: New cluster $G_1^{(1)} = \{x_1, x_5\}$ and $G_2^{(1)} = \{x_2, x_3, x_4\}$. So, re-compute of the cluster midpoint.

$$\rightarrow m_1^{(1)} =$$

$(2.5, 2.0)^T$ is the midpoint of cluster $G_1^{(1)}$, $m_2^{(1)} = (2, 0)^T$
is the midpoint of cluster $G_2^{(1)}$

Clustering

□ K-means Clustering -- Example

Try to cluster these samples X by k-means clustering, when $k=2$,

$$X = \begin{bmatrix} 0 & 0 & 1 & 5 & 5 \\ 2 & 0 & 0 & 0 & 2 \end{bmatrix}$$

Step 4: Repeat step 2 and step 3

→ Have new clusters $G_1^{(2)} = \{x_1, x_5\}$ and $G_2^{(2)} = \{x_2, x_3, x_4\}$.

Because the clusters is not change, the clustering stops! The final results is:

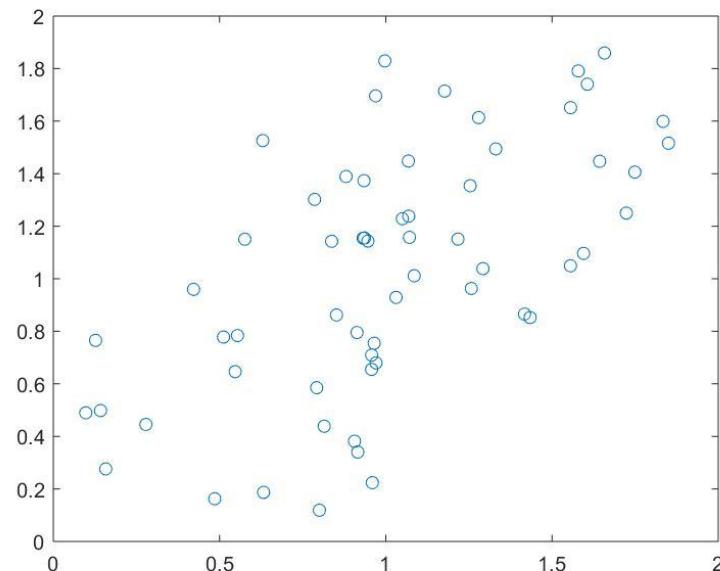
$$G_1^* = \{x_1, x_5\} \text{ and } G_2^* = \{x_2, x_3, x_4\}.$$

Clustering

□ K-means Clustering --- Example

X										
2x60 double										
1	2	3	4	5	6	7	8	9	10	
1	0.8147	0.9058	0.1270	0.9134	0.6324	0.0975	0.2785	0.5469	0.9575	0.9649
2	0.4387	0.3816	0.7655	0.7952	0.1869	0.4898	0.4456	0.6463	0.7094	0.7547

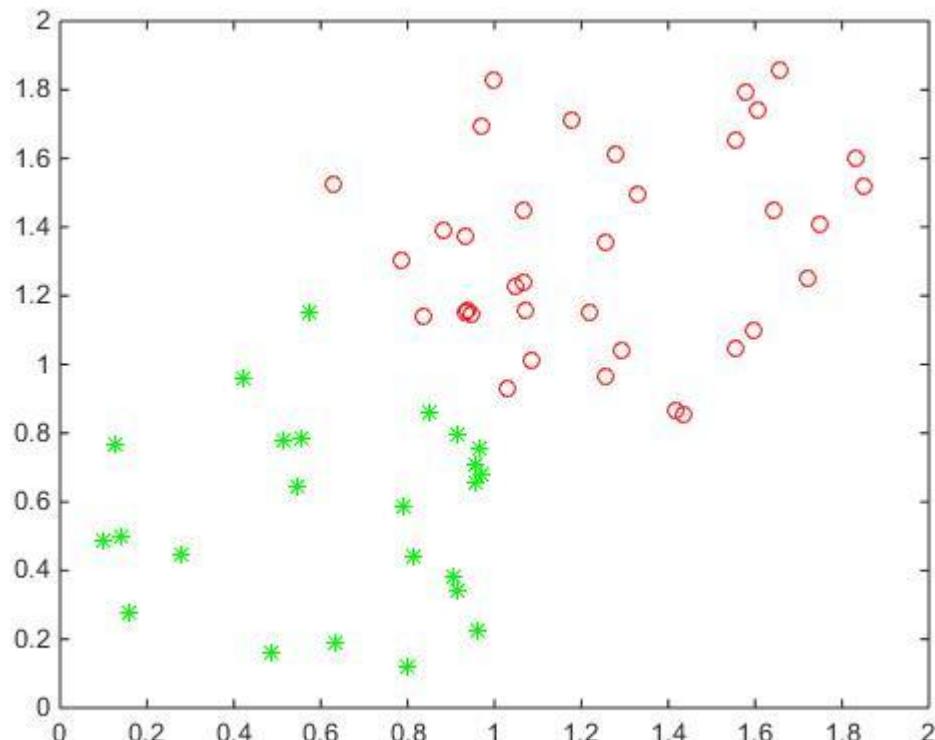
`plot(X(1,:),X(2,:),'o')`



Clustering

□ K-means Clustering --- Example

`IDX = kmeans(X,2);
..... % plot the results`



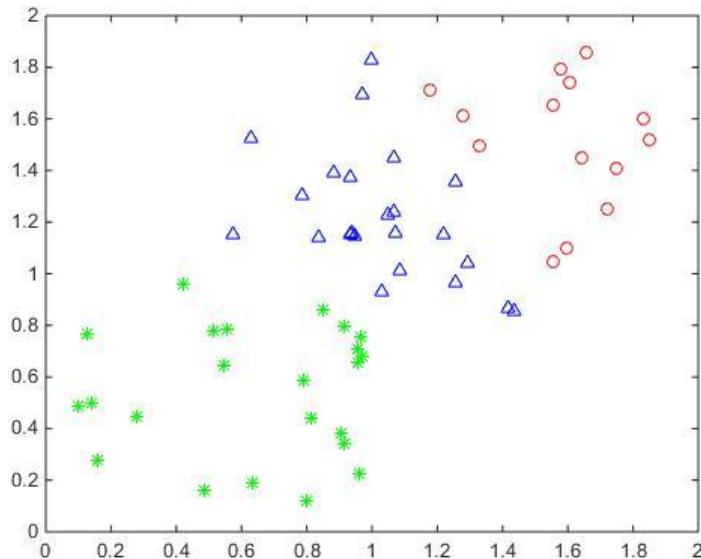
Clustering

□ K-means Clustering --- Example

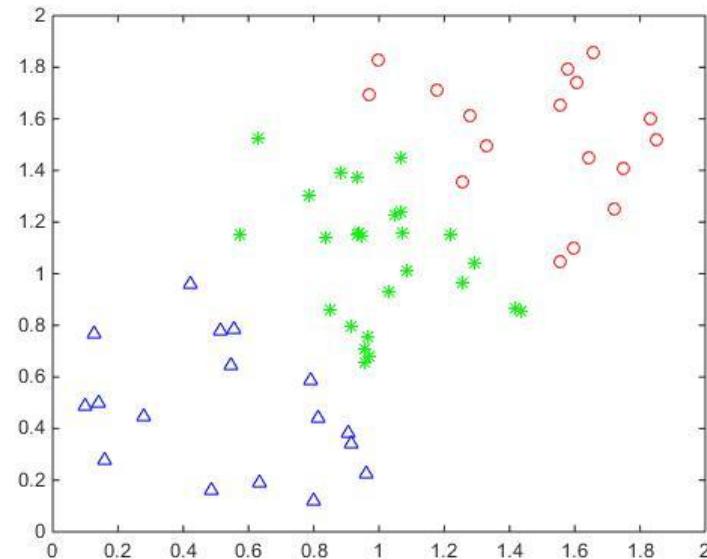
```
IDX = kmeans(X,3);  
..... % plot the results
```

The results are
different!
Why?

Result 1

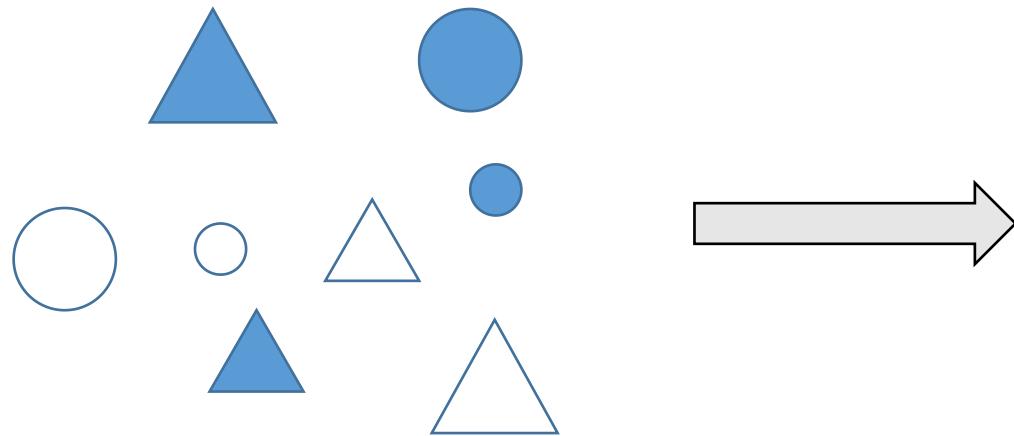


Result 2



Clustering

□ K-means Clustering -Example



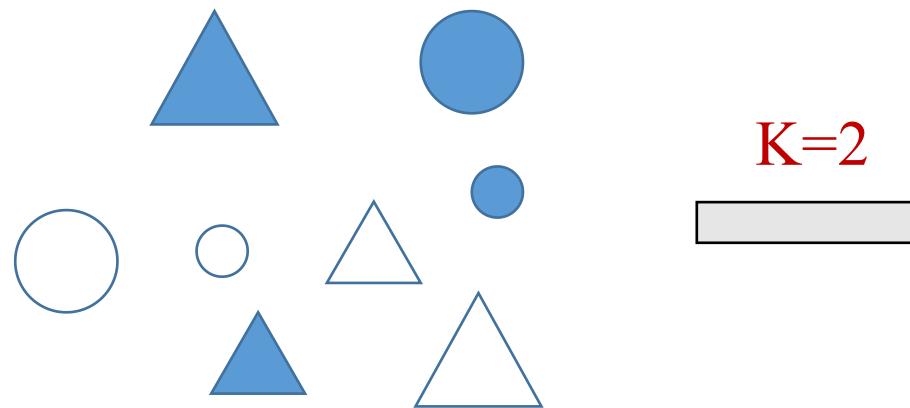
Clustering

K=2?

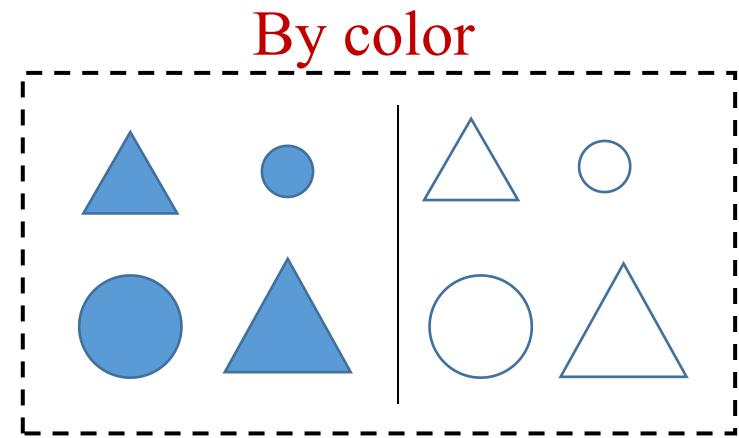
K=4?

Clustering

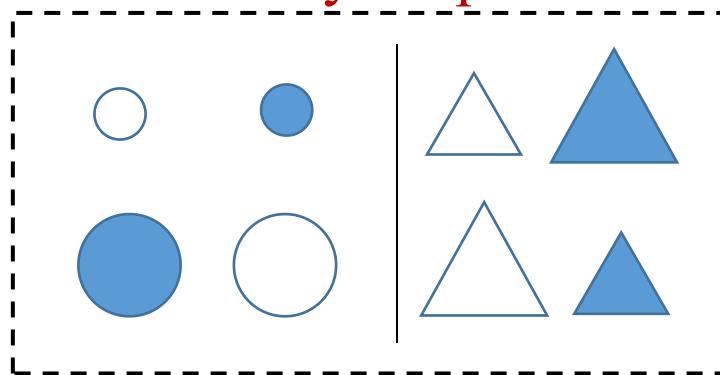
□ K-means Clustering -Example



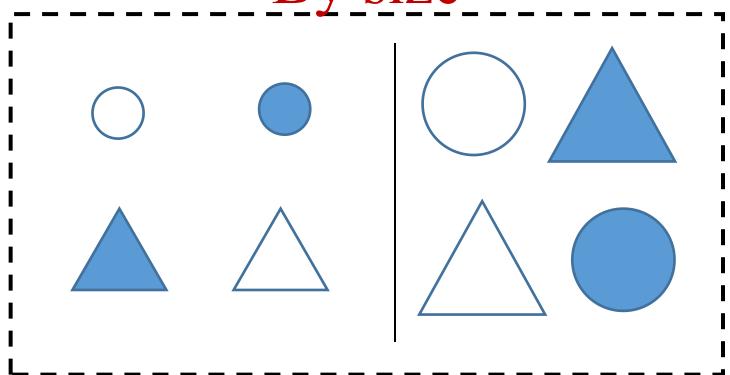
K=2
→



By shape

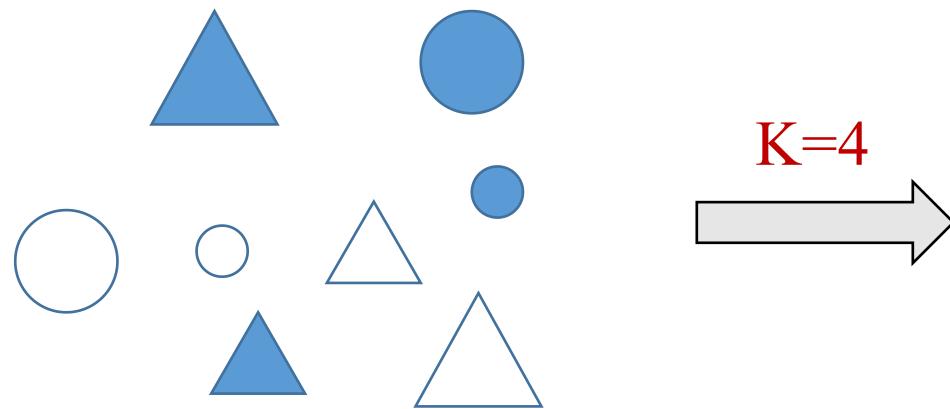


By size

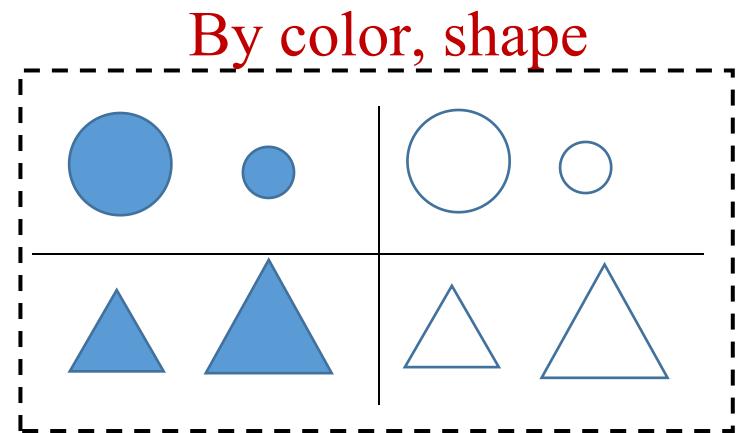


Clustering

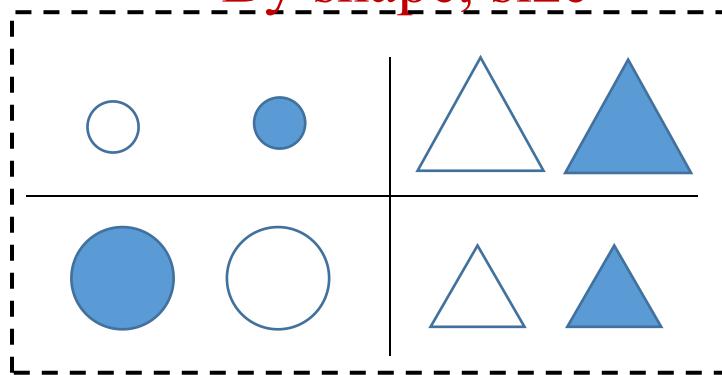
□ K-means Clustering -Example



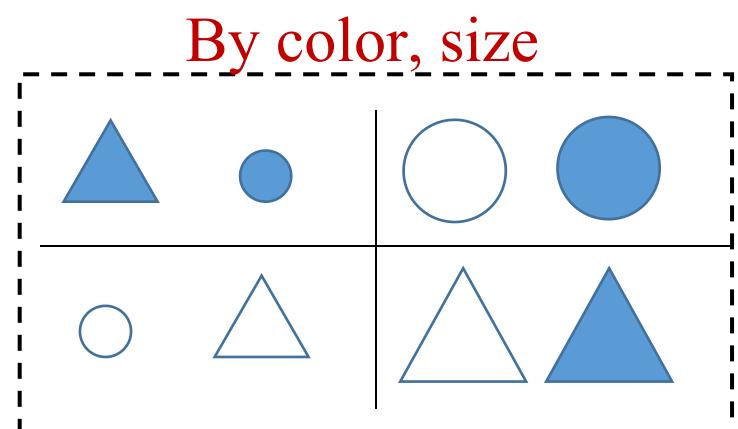
K=4



By shape, size



By color, size



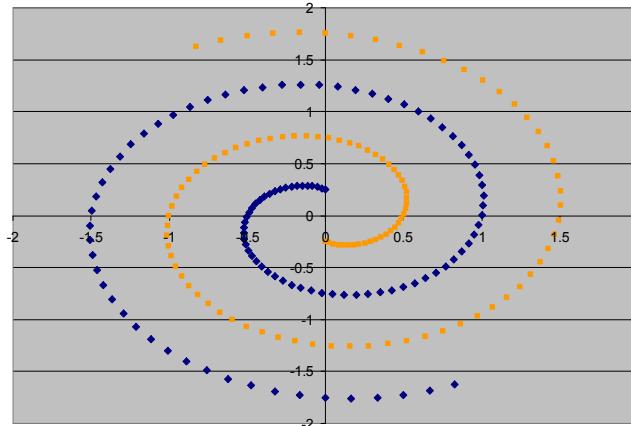
Clustering

□ K-means Clustering --- Disadvantage

- 1. Time complexity:
 - $O(NKT)$, where N is the number of data, K is the number of clusters, and T is the number of iterations.
- 2. Sensitive to noise
- 3. Different results are obtained with different initial centers.

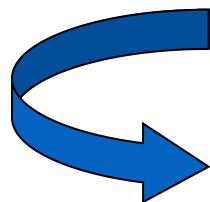
Clustering

□ K-means Clustering --- Example

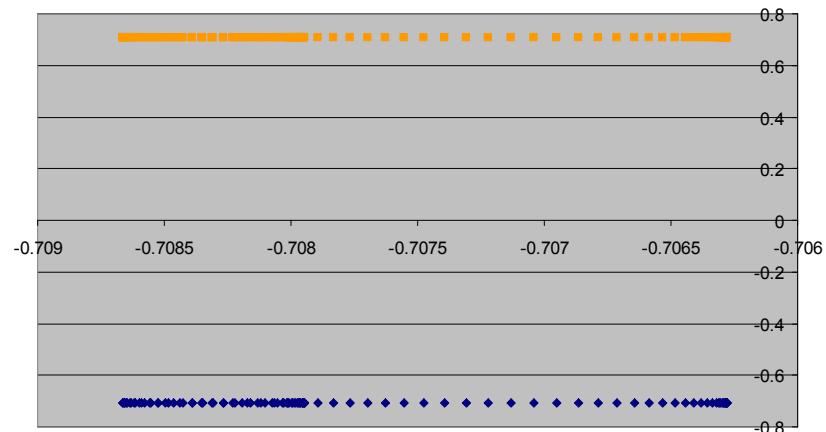


Dataset exhibits complex cluster shapes

⇒ K-means performs very poorly in this space due bias toward dense spherical clusters.



In the embedded space given by two leading eigenvectors, clusters are trivial to separate.



Clustering

□ Spectral Clustering

- Algorithms that cluster points using eigenvectors of matrices derived from the data.
- Obtain data **representation** in the low-dimensional space that can be easily clustered.
- Variety of methods that use the eigenvectors differently
- Disadvantage: difficult to understand....

Clustering

□ Spectral Clustering

- Three basic stages:
 1. Pre-processing
 - Construct a **matrix representation** of the dataset.
 2. Decomposition
 - Compute **eigenvalues and eigenvectors** of the matrix.
 - Map each point to a lower-dimensional representation based on one or more eigenvectors.
 3. Grouping
 - Assign points to two or more **clusters** (e.g. by **k means method**), based on the new representation.

Clustering

□ Spectral Clustering

1 . Pre-processing

- Construct a **matrix representation** of the dataset.
(Build Laplacian matrix L)

2. Decomposition

- Find eigenvalues Λ and eigenvectors X of the matrix L
 - Map vertices to corresponding components of smallest two eigenvalues.

	X_1	X_2	X_3	X_4	X_5	X_6
X_1	1.5	-0.8	-0.6	0	-0.1	0
X_2	-0.8	1.6	-0.8	0	0	0
X_3	-0.6	0.8	1.6	-0.2	0	0
X_4	0	0	-0.2	1.7	-0.8	-0.7
X_5	0.1	0	0.4 0.2	0.1 0.4	0.8 0.4	-0.2 0.8
X_6	0	0	0.4 0.2	0.1 -0.2	0.0 0.2	-0.4 1.5
2.3				0.4 -0.4	0.9 0.2	-0.4 -0.6
2.5				0.4 -0.7	-0.4 -0.8	-0.6 -0.2
3.0				0.4 -0.7	-0.2 0.5	0.8 0.9

It is easier to divide these six points into two clusters using this new representation.

Clustering



0000000000000000
1111111111111111
2222222222222222
3333333333333333
4444444444444444
5555555555555555
6666666666666666
7777777777777777
8888888888888888
9999999999999999

Clustering

□ Spectral Clustering ---Applications: Motion segmentation

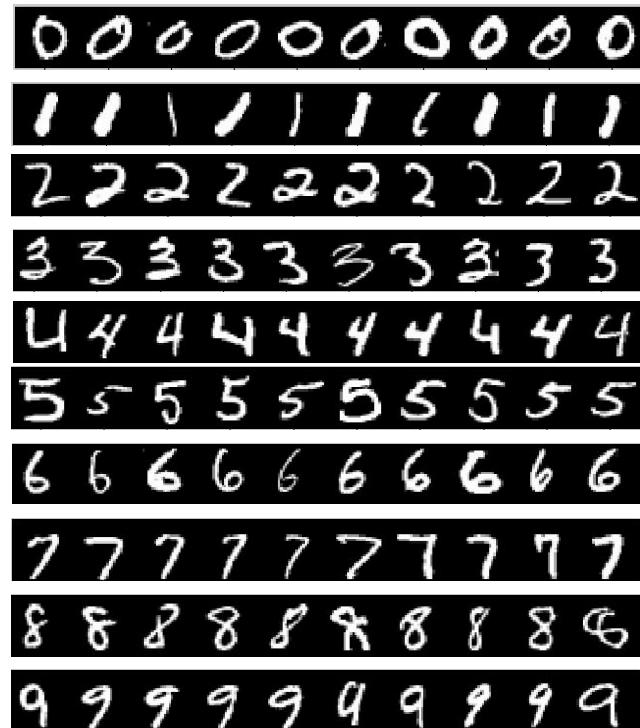


Unsupervised learning

- Clustering
 - K-means method
 - Spectral clustering
- Representation learning

Representation Learning

- Representation = **re** + presentation



Raw data



Coding

Representation

Unsupervised learning

- Clustering
 - K-means method
 - Spectral clustering
- Representation learning
 - *Linear coding*
 - PCA
 - Applications

Representation Learning

□ Linear coding

representation

A: “dictionary”

$$y = Ax \in \mathbb{R}^m$$
$$\begin{matrix} m \times 1 \\ y \end{matrix} = \begin{matrix} m \times n^A \\ A \end{matrix} \begin{matrix} n \times 1 \\ x \end{matrix}$$

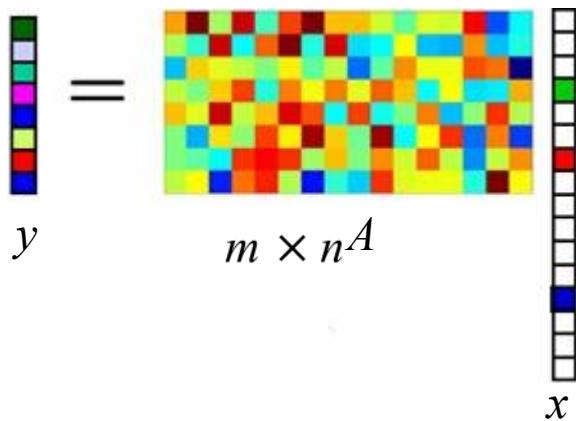
e.g.

$$y = \begin{bmatrix} 0.75 \\ 0.27 \\ 0.3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0.5 & 0.1 & 0 & 0 \\ 0.2 & 0.1 & 0 & 0.1 & 1 & 0 \\ 0 & 0.1 & 1 & 0.2 & 0.1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0.6 \\ 0 \\ 0 \\ 1.5 \\ 0 \\ 0 \end{bmatrix}$$

Representation Learning

- Linear coding representation

$$y = Ax \in R^m$$

$$\begin{matrix} m \times 1 \\ y \end{matrix} = \begin{matrix} m \times n \\ A \end{matrix} \begin{matrix} n \times 1 \\ x \end{matrix}$$


Considering the solution to the linear equation:

If $m > n \Rightarrow$ **overdetermined** \Rightarrow no solution / unique solution

If $m < n \Rightarrow$ **underdetermined** \Rightarrow no solution / infinite solutions

Unsupervised learning

- Clustering
 - K-means method
 - Spectral clustering
- Representation learning
 - Linear coding
 - PCA
 - Applications

Representation Learning

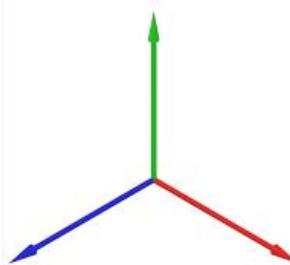
- Linear Feature Extraction

- Given the original d -dimension feature space $X = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m) \in \mathbb{R}^{d \times m}$
- Get the reduced d' -dimension feature space $Z = (\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_m) \in \mathbb{R}^{d' \times m}$ after transformation ($d' < d$)
- Transformation process:

$$Z = \mathbf{W}^T X$$

Where $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{d'}) \in \mathbb{R}^{d \times d'}$ is the transformation matrix, $\mathbf{w}_i \in \mathbb{R}^{d \times 1}$, and $Z \in \mathbb{R}^{d' \times m}$ is the coordinate expression of sample X in low dimension space.

- If $\mathbf{w}_i^T \mathbf{w}_j = 0 (i \neq j)$, then \mathbf{w}_i is **orthogonal** to \mathbf{w}_j (\mathbf{w}_i is independent from \mathbf{w}_j), the new coordinate system $\{\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{d'}\}$ is orthogonal, and \mathbf{W} is the orthogonal matrix.



Representation Learning

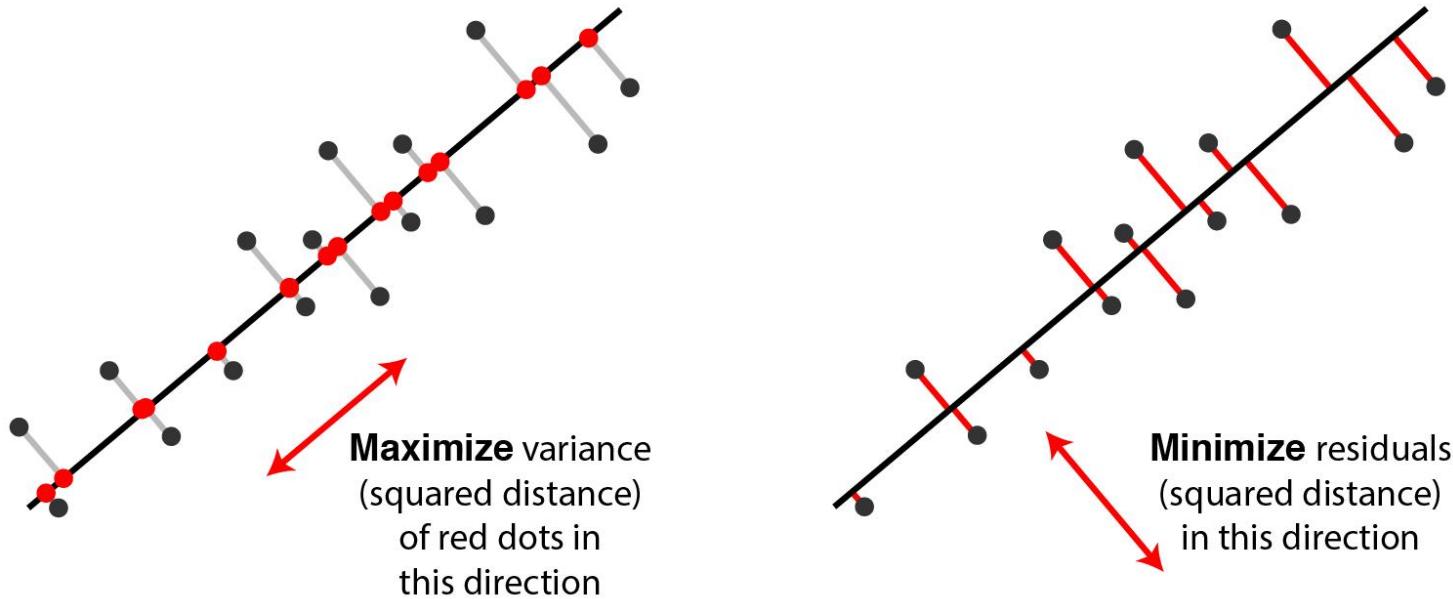
□ Principal Component Analysis (PCA)

- Linear combination of original features: use the orthogonal transform matrix \mathbf{W} to transform the d relevant features into d' ($d' < d$) irrelevant features. These d' irrelevant features are called principal components for classification.
- Use principal components to approximate the original sample.
- Realize the dimension reduction by replace original sample using few principal components.

Representation Learning

□ Principal Component Analysis (PCA)

- **Minimize reconstruction error (residual):** the sample $\tilde{\mathbf{x}}$ reconstructed from the reduced (projected) space is close enough to the original sample \mathbf{x} .
- **Maximum class separability (variance):** ensure that projected data from different classes can be separated well.



Representation Learning

□ Principal Component Analysis (PCA)

- Minimize reconstruction error

- Data standardization. Centralize the original data (subtract mean vector). That is, set $\mathbf{x}_i = \mathbf{x}_i - \frac{1}{m} \sum_{i=1}^m \mathbf{x}_i$, then $\sum_{i=1}^m \mathbf{x}_i = 0$, $\mathbf{x}_i \in \mathbb{R}^{d \times 1}$.
- Assume transformation matrix $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{d'}) \in \mathbb{R}^{d \times d'}$, $\mathbf{w}_i \in \mathbb{R}^{d \times 1}$ is the standard orthogonal basis vector. That is, $\|\mathbf{w}_i\|_2 = 1$, $\mathbf{W}^T \mathbf{W} = \mathbf{I}$, $\mathbf{w}_i^T \mathbf{w}_j = 0$ ($i \neq j$).
- The projection of \mathbf{x}_i in low dimension coordinate system is $\mathbf{z}_i = \mathbf{W}^T \mathbf{x}_i$, $\mathbf{z}_i = (z_{i1}; z_{i2}; \dots; z_{id'}) \in \mathbb{R}^{d' \times 1}$. Where $z_{ij} = \mathbf{w}_j^T \mathbf{x}_i$ is the jth-coordinate of \mathbf{x}_i in low dimension space.
- Reconstruct \mathbf{x}_i by \mathbf{z}_i , then $\tilde{\mathbf{x}}_i = \sum_{j=1}^{d'} z_{ij} \mathbf{w}_j$

Minimize reconstruction error : $\tilde{\mathbf{x}}_i$, \mathbf{x}_i

Representation Learning

□ Principal Component Analysis (PCA)

- Minimize reconstruction error

➤ For the entire training set, the distance between original sample \mathbf{x}_i and reconstructed sample $\tilde{\mathbf{x}}_i$ is

$$\sum_{i=1}^m \|\tilde{\mathbf{x}}_i - \mathbf{x}_i\|_2^2 = - \sum_{i=1}^m \mathbf{z}_i^T \mathbf{z}_i + \sum_{i=1}^m \mathbf{x}_i^T \mathbf{x}_i$$
$$\propto - \text{tr} \left(\mathbf{W}^T \left(\sum_{i=1}^m \mathbf{x}_i \mathbf{x}_i^T \right) \mathbf{W} \right)$$

$$\text{tr}(A) = \sum_{i=1}^n a_{ii}$$

➤ Take minimizing the reconstruction distance as the objective function, since $\sum_{i=1}^m \mathbf{x}_i \mathbf{x}_i^T$ is the covariance matrix $\mathbf{X}\mathbf{X}^T$, then the objective function is

$$\min_{\mathbf{W}} - \text{tr}(\mathbf{W}^T \mathbf{X} \mathbf{X}^T \mathbf{W})$$
$$\text{s. t. } \mathbf{W}^T \mathbf{W} = \mathbf{I}$$

Representation Learning

□ Principal Component Analysis (PCA)

- Proof

$$\begin{aligned}\sum_{i=1}^m \|\hat{\mathbf{x}}_i - \mathbf{x}_i\|_2^2 &= \sum_{i=1}^m \left\| \sum_{j=1}^{d'} z_{ij} \mathbf{w}_j - \mathbf{x}_i \right\|_2^2 = \sum_{i=1}^m \|\mathbf{Wz}_i - \mathbf{x}_i\|_2^2 \\ &= \sum_{i=1}^m (\mathbf{Wz}_i - \mathbf{x}_i)^T (\mathbf{Wz}_i - \mathbf{x}_i) \\ &= - \sum_{i=1}^m \mathbf{z}_i^T \mathbf{z}_i + \sum_{i=1}^m \mathbf{x}_i^T \mathbf{x}_i \\ &= - \sum_{i=1}^m \text{tr}(\mathbf{z}_i \mathbf{z}_i^T) + \text{const} \\ &= - \text{tr} \left(\sum_{i=1}^m \mathbf{z}_i \mathbf{z}_i^T \right) + \text{const} \\ &= - \text{tr} \left(\sum_{i=1}^m \mathbf{W}^T \mathbf{x}_i \mathbf{x}_i^T \mathbf{W} \right) + \text{const} \\ &\approx - \text{tr} \left(\mathbf{W}^T \left(\sum_{i=1}^m \mathbf{x}_i \mathbf{x}_i^T \right) \mathbf{W} \right) + \text{const} \\ &\propto - \text{tr} \left(\mathbf{W}^T \left(\sum_{i=1}^m \mathbf{x}_i \mathbf{x}_i^T \right) \mathbf{W} \right)\end{aligned}$$

Representation Learning

□ Principal Component Analysis (PCA)

- Maximum class separability

$$\max_{\mathbf{W}} \text{tr}(\mathbf{W}^T \mathbf{X} \mathbf{X}^T \mathbf{W})$$
$$\text{s. t. } \mathbf{W}^T \mathbf{W} = \mathbf{I}$$



$$\min_{\mathbf{W}} -\text{tr}(\mathbf{W}^T \mathbf{X} \mathbf{X}^T \mathbf{W})$$
$$\text{s. t. } \mathbf{W}^T \mathbf{W} = \mathbf{I}$$

Representation Learning

□ Principal Component Analysis (PCA)

Solve PCA to get $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{d'})$

$$\min_{\mathbf{W}} -\text{tr}(\mathbf{W}^T \mathbf{X} \mathbf{X}^T \mathbf{W})$$

$$\text{s. t. } \mathbf{W}^T \mathbf{W} = \mathbf{I}$$

Where $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m) \in \mathbb{R}^{d \times m}$, $\mathbf{W} = \{\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{d'}\} \in \mathbb{R}^{d \times d'}$, $\mathbf{I} \in \mathbb{R}^{d' \times d'}$

- **S1.** Define the Lagrange function using Lagrange multiplier matrix $\Lambda = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_{d'}) \in \mathbb{R}^{d' \times d'}$, Λ is the diagonal matrix.

$$L(\mathbf{W}, \Lambda) = -\text{tr}(\mathbf{W}^T \mathbf{X} \mathbf{X}^T \mathbf{W}) + \text{tr}(\Lambda^T (\mathbf{W}^T \mathbf{W} - \mathbf{I}))$$

- **S2.** Set the partial of $L(\mathbf{W}, \Lambda)$ on \mathbf{W} as 0.

$$\frac{\partial L(\mathbf{W}, \Lambda)}{\partial \mathbf{W}} = -2\mathbf{X} \mathbf{X}^T \mathbf{W} + 2\mathbf{W} \Lambda = \mathbf{0}$$

$$\boxed{\mathbf{X} \mathbf{X}^T \mathbf{W} = \mathbf{W} \Lambda}$$
$$\mathbf{X} \mathbf{X}^T \mathbf{w}_i = \lambda_i \mathbf{w}_i$$

The definition of eigenvalue and eigenvector for matrix.
Then $\mathbf{X} \mathbf{X}^T = \mathbf{W} \Lambda \mathbf{W}^T$

Solve eigenvalue Λ and corresponding eigenvector \mathbf{W} for matrix $\mathbf{X} \mathbf{X}^T$

Representation Learning

□ Principal Component Analysis (PCA)

Solve PCA to get $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{d'})$

$$\begin{aligned} & \min_{\mathbf{W}} -\text{tr}(\mathbf{W}^T \mathbf{X} \mathbf{X}^T \mathbf{W}) \\ & \text{s. t. } \mathbf{W}^T \mathbf{W} = \mathbf{I} \end{aligned}$$

Where $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m) \in \mathbb{R}^{d \times m}$, $\mathbf{W} = \{\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{d'}\} \in \mathbb{R}^{d \times d'}$, $\mathbf{I} \in \mathbb{R}^{d' \times d'}$

- **S3.** Substitute $\mathbf{X} \mathbf{X}^T \mathbf{w}_i = \lambda_i \mathbf{w}_i$ in the objective function

$$\begin{aligned} \min_{\mathbf{W}} -\text{tr}(\mathbf{W}^T \mathbf{X} \mathbf{X}^T \mathbf{W}) &= \max_{\mathbf{W}} \text{tr}(\mathbf{W}^T \mathbf{X} \mathbf{X}^T \mathbf{W}) = \max_{\mathbf{W}} \sum_{i=1}^{d'} \mathbf{w}_i^T \mathbf{X} \mathbf{X}^T \mathbf{w}_i \\ &= \max_{\mathbf{W}} \sum_{i=1}^{d'} \mathbf{w}_i^T \lambda_i \mathbf{w}_i = \max_{\mathbf{W}} \sum_{i=1}^{d'} \lambda_i \mathbf{w}_i^T \mathbf{w}_i = \max_{\mathbf{W}} \sum_{i=1}^{d'} \lambda_i \end{aligned}$$

- **S4.** We will get the optimal solution by letting $\lambda_1, \lambda_2, \dots, \lambda_{d'}$ and $\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{d'}$ be the top maximum d' eigenvalue and corresponding eigenvector.
- **S5.** Solve $\lambda_1, \lambda_2, \dots, \lambda_{d'}$ and $\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{d'}$ by eigenvalue decomposing the covariance matrix $\mathbf{X} \mathbf{X}^T$. Rank the eigenvalue $\lambda_1 > \lambda_2 > \dots > \lambda_d$, and take the top maximum d' eigenvalues and its corresponding eigenvectors. And compose the transformation matrix $\mathbf{W} = \{\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{d'}\}$, which is the solution of PCA.

Representation Learning

□ Principal Component Analysis (PCA)

- d' can be specified by user.
- Or specified the minimum d' by setting the threshold t (80% or 90%), which satisfies the following inequivalent relation

$$\frac{\sum_{i=1}^{d'} \lambda_i}{\sum_{i=1}^d \lambda_i} \geq t$$

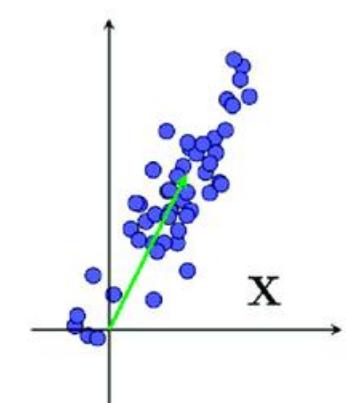
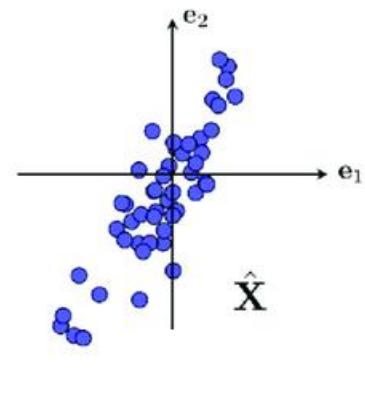
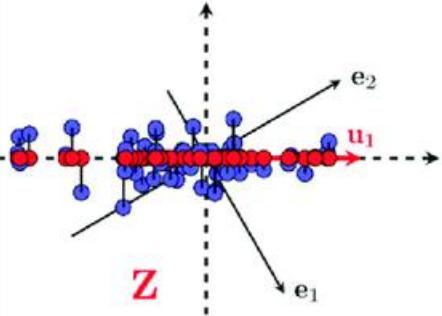
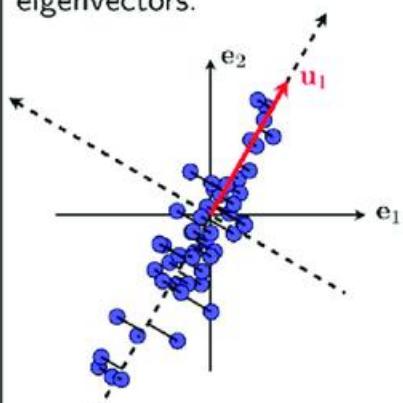
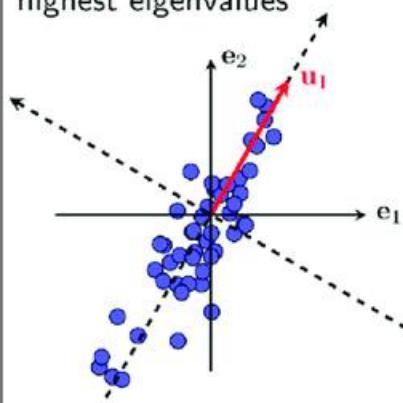
Representation Learning

□ Principal Component Analysis (PCA)

- Process to implement dimension reduction using PCA
- **S1:** Standardization. $\mathbf{x}_i = \mathbf{x}_i - \frac{1}{m} \sum_{i=1}^m \mathbf{x}_i$
- **S2:** Compute the covariance matrix $\frac{1}{m} \mathbf{X} \mathbf{X}^T$
- **S3:** Eigenvalue decompose the matrix $\frac{1}{m} \mathbf{X} \mathbf{X}^T$
- **S4:** Take the top maximum d' eigenvalues and corresponding eigenvectors $\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{d'}$
- **S5:** Get the transformation matrix $\mathbf{W} = \{\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{d'}\}$, and output the reduced feature vectors \mathbf{z}_i for original samples.

Representation Learning

□ Principal Component Analysis (PCA)

1. Find mean vector 	2. Subtract mean 	3. Compute covariance matrix: $S = \frac{1}{N} \hat{X} \hat{X}^T$
7. Obtain projected points in low dimension. 	6. Project data to selected eigenvectors. 	5. Pick K eigenvectors w. highest eigenvalues 

Representation Learning

□ Principal Component Analysis (PCA)

- For PCA, only need to retain mean vector and \mathbf{W} to project new sample on the low dimension space by the vector minus (standardization) and matrix-vector multiplication (projection).
- Discarding $(d - d')$ eigenvectors will lead to information loss. However, dimension reduction can increase the density of samples. Since the discarded eigenvalues and eigenvectors are always related to noise, which achieves de-noising for samples.

Representation Learning

□ Principal Component Analysis (PCA)

Example

- Problem: Given the original sample set $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5) = \begin{pmatrix} -1 & -1 & 0 & 2 & 0 \\ -2 & 0 & 0 & 1 & 1 \end{pmatrix}$. Reduce sample from 2-D to 1-D using PCA.

-
- Process to implement dimension reduction using PCA
 - **S1:** Standardization. $\mathbf{x}_i = \mathbf{x}_i - \frac{1}{m} \sum_{i=1}^m \mathbf{x}_i$
 - **S2:** Compute the covariance matrix $\frac{1}{m} \mathbf{X} \mathbf{X}^T$
 - **S3:** Eigenvalue decompose the matrix $\frac{1}{m} \mathbf{X} \mathbf{X}^T$
 - **S4:** Take the top maximum d' eigenvalues and corresponding eigenvectors $\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{d'}$
 - **S5:** Get the transformation matrix $\mathbf{W} = \{\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{d'}\}$, and output the reduced feature vectors \mathbf{z}_i for original samples.

Representation Learning

Answer:

- S1. Standardization. The mean vector $\frac{1}{m} \sum_{i=1}^m \mathbf{x}_i = 0$
- S2. Compute covariance matrix $\mathbf{A} = \frac{1}{5} \mathbf{X} \mathbf{X}^T = \frac{1}{5} \begin{pmatrix} 6 & 4 \\ 4 & 6 \end{pmatrix}$
- S3. Eigenvalue decompose.

(1) Solve Eigenvalue by $|\lambda I - \mathbf{A}| = 0$

$$|\lambda I - \mathbf{A}| = \begin{vmatrix} \lambda - \frac{6}{5} & -\frac{4}{5} \\ -\frac{4}{5} & \lambda - \frac{6}{5} \end{vmatrix} = (\lambda - \frac{6}{5})^2 - \frac{16}{25} = (\lambda - 2)(\lambda - \frac{2}{5}) = 0$$
$$\lambda_1 = 2, \quad \lambda_2 = \frac{2}{5}$$

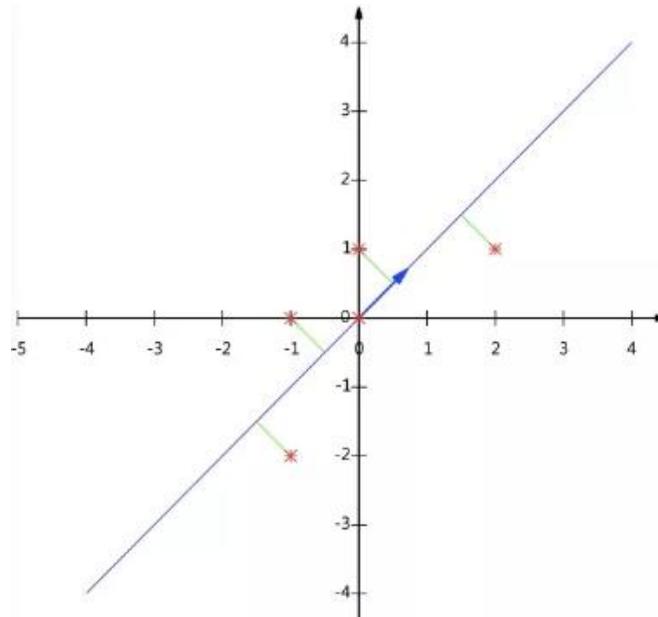
(2) Solve Eigenvector by $(\lambda I - \mathbf{A}) \mathbf{w} = \mathbf{0}$

$$\lambda_1 = 2 \rightarrow \mathbf{w}_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad \lambda_2 = \frac{2}{5} \rightarrow \mathbf{w}_2 = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$

Representation Learning

- S4. Rank the eigenvalues, and select top $d' = 1$ eigenvalue λ_1 and its corresponding eigenvector \mathbf{w}_1 . Standardization eigenvector to get $\mathbf{w}_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$
- S5. Dimension reduction.

$$Z = W^T X = \left(-\frac{3}{\sqrt{2}}, -\frac{1}{\sqrt{2}}, 0, \frac{3}{\sqrt{2}}, \frac{1}{\sqrt{2}} \right)$$



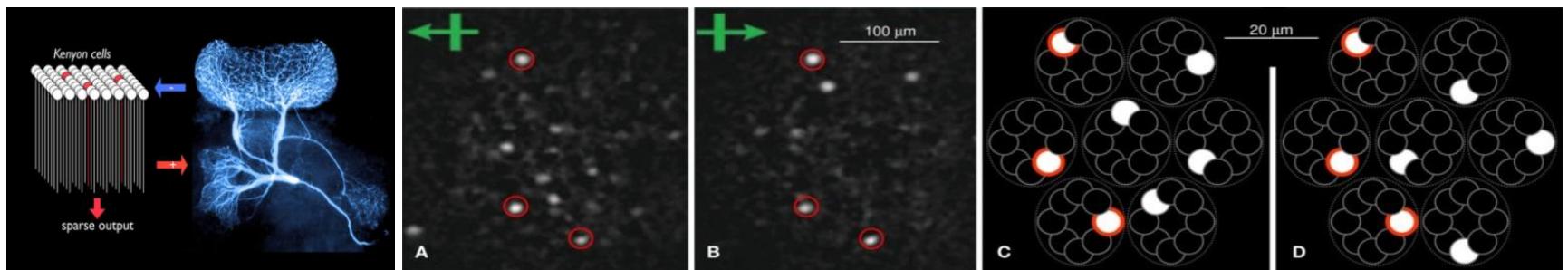
Unsupervised learning

- Clustering
 - K-means method
 - Spectral clustering
- Representation learning
 - Linear coding
 - PCA
 - Others
 - Applications

Representation Learning

□ Sparse coding

- Information are coded sparsely in our brain.
- Simulation of sparse coding mechanism in brain



Representation Learning

Sparse coding

$$y = Ax \in R^m$$

$$\begin{matrix} m \times 1 \\ y \end{matrix} = \begin{matrix} m \times n \\ A \end{matrix} \begin{matrix} n \times 1 \\ x \end{matrix}$$

Objective function

210

IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 31, NO. 2, FEBRUARY 2009

Robust Face Recognition via Sparse Representation

John Wright, *Student Member, IEEE*, Allen Y. Yang, *Member, IEEE*,
Arvind Ganesh, *Student Member, IEEE*, S. Shankar Sastry, *Fellow, IEEE*, and
Yi Ma, *Senior Member, IEEE*

Abstract—We consider the problem of automatically recognizing human faces from frontal views with varying expression and illumination, as well as occlusion and disguise. We cast the recognition problem as one of classifying among multiple linear regression models and argue that new theory from sparse signal representation offers the key to addressing this problem. Based on a sparse representation computed by ℓ_1 -minimization, we propose a general classification algorithm for (image-based) object recognition. This new framework provides new insights into two crucial issues in face recognition: *feature extraction* and *robustness to occlusion*. For feature extraction, we show that if sparsity in the recognition problem is properly harnessed, the choice of features is no longer critical. What is critical, however, is whether the number of features is sufficiently large and whether the sparse representation is correctly computed. Unconventional features such as downsampled images and random projections perform just as well as conventional features such as Eigenfaces and Laplacianfaces, as long as the dimension of the feature space surpasses certain threshold, predicted by the theory of sparse representation. This framework can handle errors due to occlusion and corruption uniformly by exploiting the fact that these errors are often sparse with respect to the standard (pixel) basis. The theory of sparse representation helps predict how much occlusion the recognition algorithm can handle and how to choose the training images to maximize robustness to occlusion. We conduct extensive experiments on publicly available databases to verify the efficacy of the proposed algorithm and corroborate the above claims.

$$\min_x \|x\|_1, \text{ s.t. } \|y - Ax\|_2^2 \leq \epsilon$$

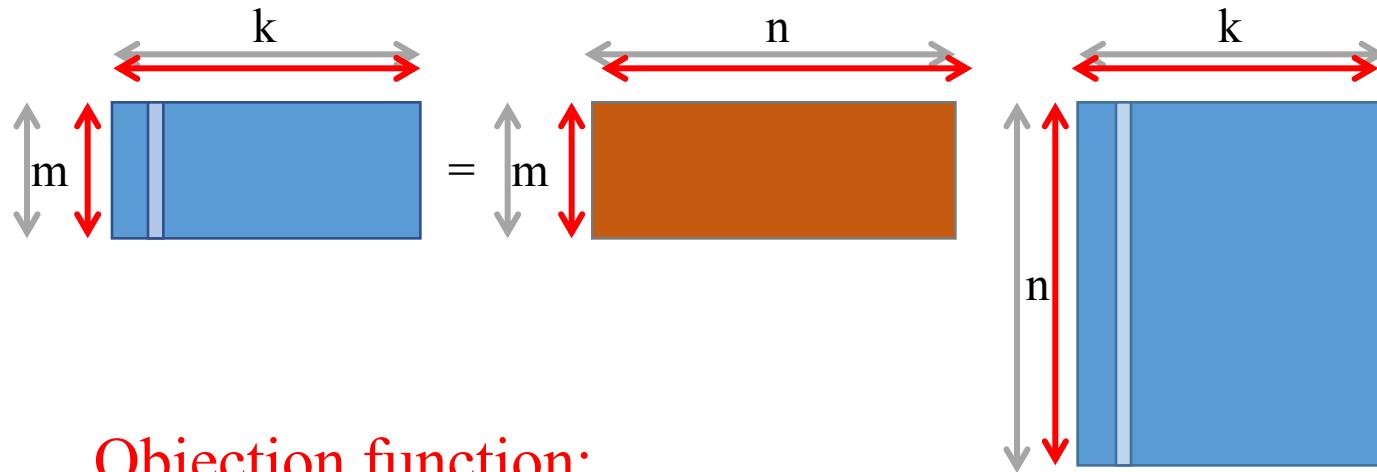
- Sparse coding is a challenging and promising theme in data analysis.
- Its main goal is to learn a sparse representation from an underdetermined dictionary.

Representation Learning

□ Low rank representation

$$A \in \mathbb{R}^{m \times n}, \quad m \ll n$$

$$Y = AX, Y = [y_1, \dots, y_k] \in \mathbb{R}^{m \times k}, X = [x_1, \dots, x_k] \in \mathbb{R}^{n \times k}$$



Objection function:

$$\min_X \text{rank}(X), \quad \text{s.t. } Y = AX$$

Low rank

Representation Learning

□ Influence

■ Academic world

- ICML: International Conference on Machine Learning is organized by the International Machine Learning Society (IMLS).

CCF class A conference.

- NIPS: Advances in Neural Information Processing Systems, The conference focuses on the progress of neural information processing systems is one of the best sessions on neural computing.

- CCF class B conference.

ICLR: International Conference on Learning Representations, since 2013

■ Industrial: great success!

Unsupervised learning

- Clustering
 - K-means method
 - Spectral clustering
- Representation learning
 - Linear coding
 - PCA
 - Others
 - Applications

Representation Learning

□ Applications

Facial Image Compression



Source image



JPEG image



JPEG2000 image



K-SVD image

Representation Learning

□ Applications

Image Deblurring



Source image



Blurred image



After deblurring

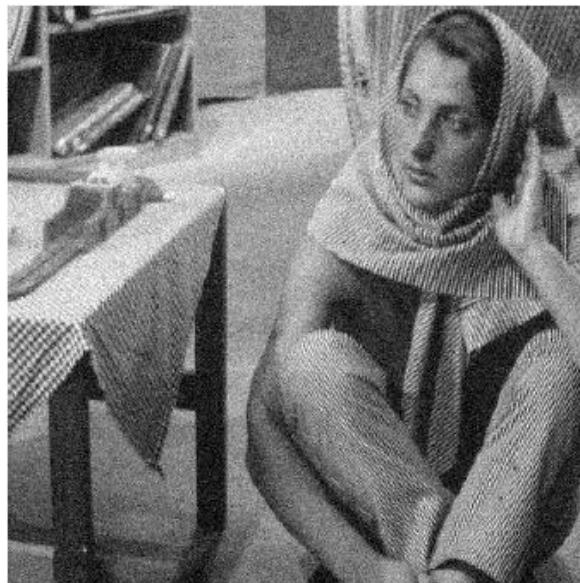
Representation Learning

□ Applications

Image Denoising



Source image



Noisy image



Denoising result

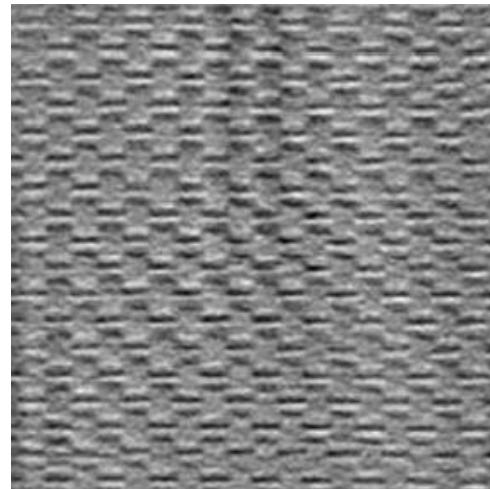
Representation Learning

□ Applications

Morphological Component Analysis



Source image



Texture component



Cartoon component

Representation Learning

□ Applications

Image Inpainting



Source image



Degraded image



Inpainting result

Representation Learning

Have a try...

- *Generate some points in two-dimensional plane.*
- *Try to design the k-means clustering algorithm or spectral clustering algorithm to cluster these points.*
- *Compare the results.*

Machine Learning

Supervised
learning

Unsupervised
learning

Reinforcement
learning

Introduction to Reinforcement learning

- Reinforcement learning
- Q-Learning



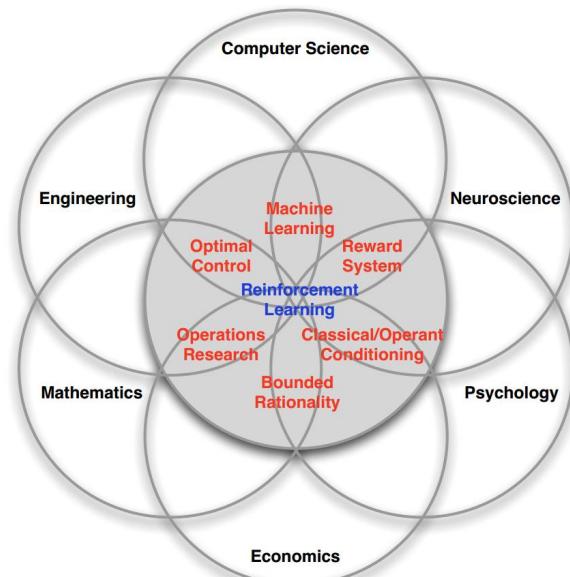
Reinforcement learning

■ Reinforcement Learning

■ “AI=RL” by David Silver

■ Agent-oriented learning—learning by interacting with an environment to achieve a goal

■ Learning by trial and error, with only delayed evaluative feedback (reward)



1. Different ML methods

■ Reinforcement Learning

■ Game Pong

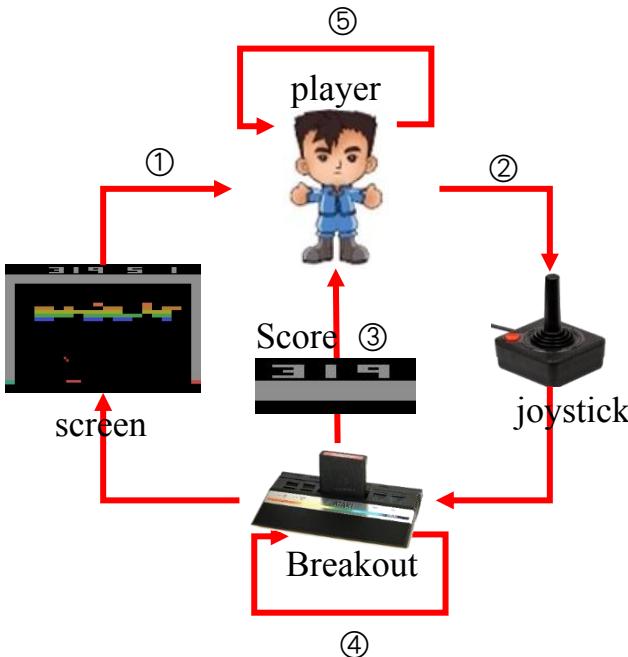


■ Game Breakout

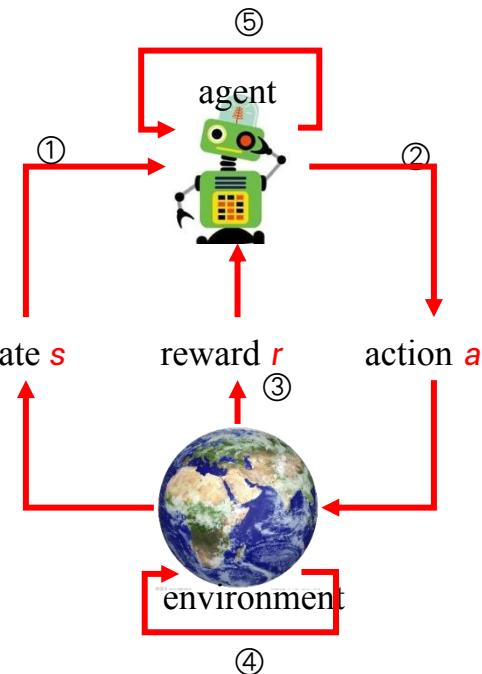


1. Different ML methods

□ Reinforcement Learning



Game Breakout



Reinforcement Learning

- Rules are unknown
 - Learn directly from the interaction
- At each time step t :
- ① Agent receives state $s(t)$
 - ② Agent executes an action $a(t)$ by his action policy $\pi(s(t))$
 - ③ Environment emits a immediate reward $r(t+1)$ to agent
 - ④ Environment changes its state to $s(t+1)$
 - ⑤ Agent improves his policy $\pi(s)$ according to the reward.

$$\{ \langle s, a, r, s' \rangle \\ s \leftarrow s' \}$$

Reinforcement learning

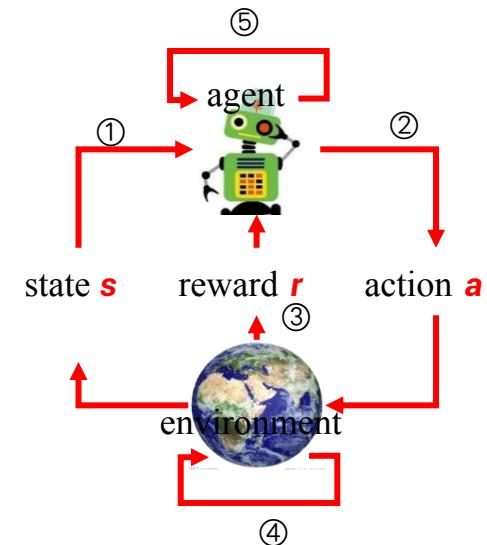
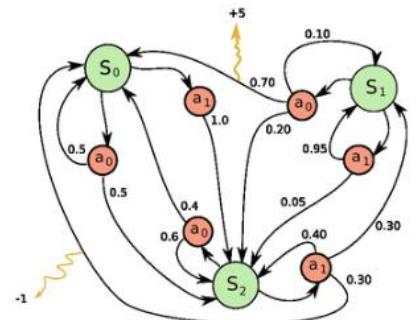
- RL problem can be described as a Markov decision process
 - The future is independent of the past given the present
- One episode of this process forms a finite sequence :

$$s(0), a(0), r(1), s(1), a(1), r(2), \dots, s(n-1), \\ a(n-1), r(n), s(n)$$

$$\left\{ \begin{array}{l} < s, a, r, s' > \\ s \leftarrow s' \end{array} \right.$$

- The agent are always trying to get the maximum rewards through policy $\pi(s)$

Question: How to define the maximum reward ?



Reinforcement learning

One episode of this process forms a finite sequence of states, actions, and rewards:

$$s(0), a(0), r(1), s(1), a(1), r(2), \dots, s(n-1), a(n-1), r(n), s(n)$$

■ Total reward of one episode:

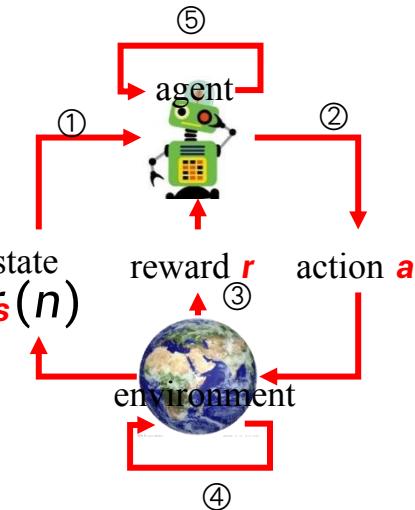
$$R = r(1) + r(2) + r(3) + \dots + r(n-1) + r(n)$$

■ Total future reward from time step t :

$$R(t) = r(t) + r(t+1) + r(t+2) + \dots + r(n-1) + r(n)$$

■ Discounted future reward from time step t :

$$R(t) = r(t) + \gamma r(t+1) + \gamma^2 r(t+2) + \dots + \gamma^{n-t} r_n$$



Question: How can agent get the maximum reward ?

Reinforcement learning

Question: How can agent get the maximum reward ?

$$\begin{aligned} R &= r(1) + r(2) + r(3) + \dots + r(n-1) + r(n) \\ &= \underbrace{r(1) + r(2) + r(3) + \dots + r(t-1)}_{\text{past reward}} + R(t) \end{aligned}$$

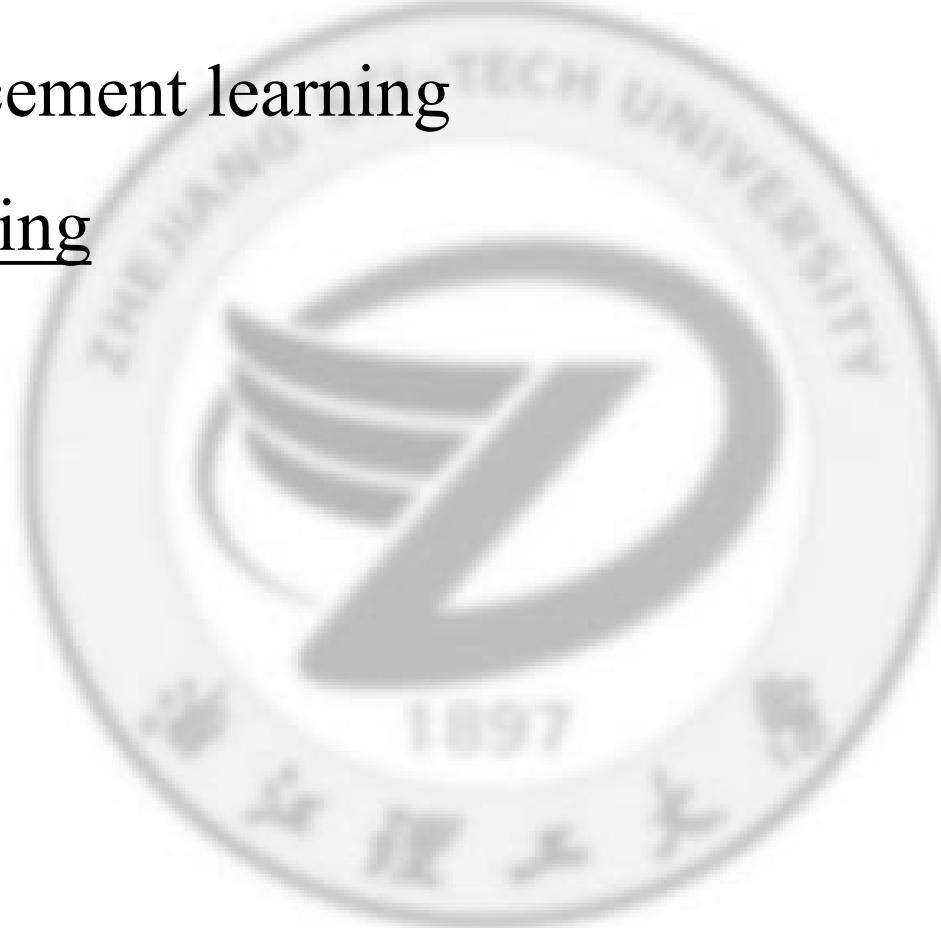
future reward

At each time step, a good strategy for an agent would be to
always choose an action that maximizes the (discounted) future reward.

$$\begin{aligned} R(t) &= r(t) + \gamma r(t+1) + \gamma^2 r(t+2) + \dots + \gamma^{n-t} r_n(t) \\ &= r(t) + \gamma R(t+1) \end{aligned}$$

Introduction to Reinforcement learning

- Reinforcement learning
- Q-Learning



Q-Learning

- Q function represents the “quality” of a certain action in a given state.
- It is a table of states and actions.

$$Q(s(t), a(t)) = \max R(t + 1)$$

$$\pi(s(t)) = \max_a Q(s(t), a)$$

Q-table

$Q[s, a]$	a_1	a_2	\dots	a_m
s_1				
s_2				
s_3				
\vdots				
s_n				

choose an action that maximizes the future reward.

Q-Learning

■ Bellman equation :

$< s(t), a(t), r(t + 1), s(t + 1) >$

$$Q(s(t), a(t)) = \max R(t + 1)$$



$$R(t + 1) = r(t + 1) + \gamma R(t + 2)$$

$$Q(s(t), a(t)) = r(t + 1) + \gamma \max R(t + 2)$$



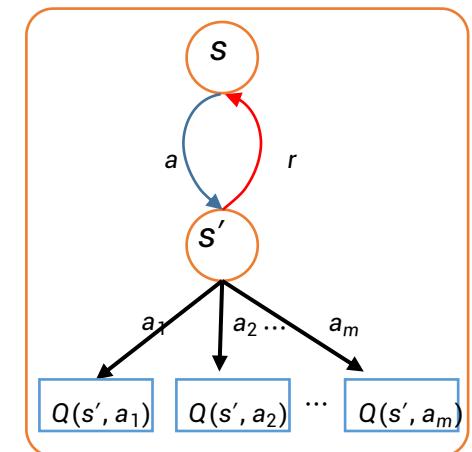
$$Q(s(t), a(t)) = r(t + 1) + \gamma \max_{a(t+1)} Q(s(t + 1), a(t + 1))$$



$$Q(s, a) = r + \gamma \max_{a'} Q(s', a')$$

current reward

maximum future
reward from
next state



Q-Learning

$\{ < s, a, r, s' >$
 $s \leftarrow s'$

Q-table

$Q[s, a]$	a_1	a_2	\dots	a_m
s_1				
s_2				
s_3				
:				
s_n				

1. Algorithm Q-Learning
2. **Input:**
 1. S is a set of states
 2. A is a set of actions
 3. γ is the discount
3. initialize $Q[S, A]$ arbitrarily
4. observe initial state s
5. **Repeat:**
 1. select and carry out an action a , randomly
 2. receive reward r
 3. observe new state s'
 4. If s' is terminal state:
 1. $Q[s, a] = r$
 5. Else:
 1. $Q[s, a] = r + \gamma \max_a Q[s', a']$
 6. $s \leftarrow s'$
6. Until terminated

Q-Learning

A tiny example:

Game description

States:

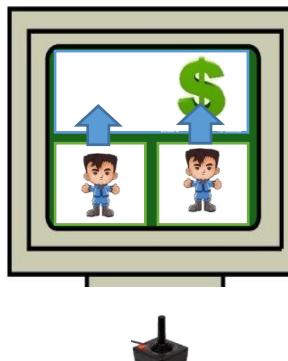
s_1, s_2, s_3 , where s_3 is terminal state

Actions:

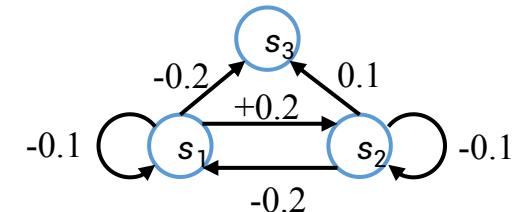
a_1 denotes *up*. The agent goes up and moves to terminal state.

a_2 denotes *left*. The agent moves to left in state s_2 with a reward -0.2 , while stay still in state s_1 with a reward -0.1 .

a_3 denotes *right*. The agent moves to right in state s_1 with a reward 0.2 , while stay still in state s_2 with a reward -0.1 .



Move up/left/right



Q-Learning



Move up/left/right



$Q[s, a]$	a_1 up	a_2 left	a_3 right
s_1			
s_2			
s_3	-	-	-



Algorithm Q-Learning

Input:

S is a set of states

A is a set of actions

γ is the discount

initialize $Q[S, A]$ arbitrarily

observe initial state s

Repeat:

select and carry out an action a , randomly
receive reward r

observe new state s'

If s' is terminal state:

$$Q[s, a] = r$$

Else:

$$Q[s, a] = r + \gamma \max_{a'} Q[s', a']$$

$$s \leftarrow s'$$

Until terminated

Q-Learning

■ Step 1: initialize $Q[S, A]$

$$\gamma = 0.8$$

■ Step 2: training loop
1st episode:

$$s(0) = s_1, a(0) = a_3, r(1) = 0.2, s(1) = s_2, a(1) = a_3, r(2) = -0.1, s(2) = s_2, a(2) = a_1, r(3) = 0.1, s(3) = s_3$$

$$Q[s_1, a_3] = 0.2 + 0.8 * \max_{a_i} (Q[s_2, a_i]) \\ = 0.2 + 0.8 * 0.78 \\ = 0.82$$

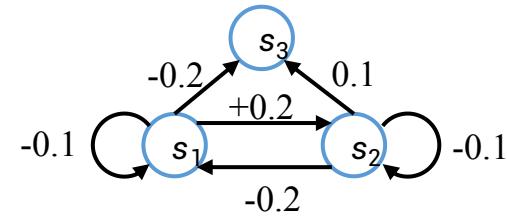
$$Q[s_2, a_3] = -0.1 + 0.8 * \max_{a_i} (Q[s_2, a_i]) \\ = -0.1 + 0.8 * 0.78 \\ = 0.52$$

$$Q[s_2, a_1] = 0.1$$

$Q[s, a]$	a_1 up	a_2 left	a_3 right
s_1	0.60	0.74	0.82
s_2	0.36	0.32	0.78
s_3	-	-	-

$Q[s, a]$	a_1 up	a_2 left	a_3 right
s_1	0.60	0.74	0.82
s_2	0.36	0.32	0.52
s_3	-	-	-

$Q[s, a]$	a_1 up	a_2 left	a_3 right
s_1	0.60	0.74	0.82
s_2	0.1	0.32	0.52
s_3	-	-	-



$$Q(s, a) = r + \gamma \max_{a'} Q(s', a')$$

Q-Learning

After
11th
episode

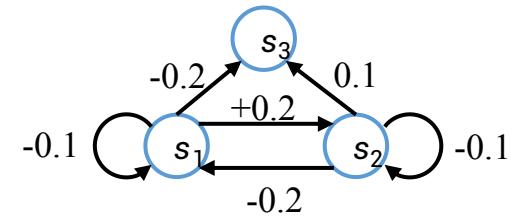
$Q[s, a]$	a_1 up	a_2 left	a_3 right
s_1	-0.2	0.5 6	0.40
s_2	0.1 0	0.2 5	0.10
s_3	-	-	-

12st episode:

$s(0) = s_1, a(0) = a_2, r(1) = -0.1, s(1) = s_1, a(1) = a_1, r(2) = -0.2, s(2) = s_3$

$$Q[s_1, a_2] = -0.1 + 0.8 * \max_{a_i} (Q[s_1, a_i]) \\ = -0.1 + 0.8 * 0.56 \\ = 0.35$$

$Q[s, a]$	a_1 up	a_2 left	a_3 right
s_1	0.20	0.35	0.40
s_2	0.10	0.25	0.10
s_3	-	-	-



$Q[s, a]$	a_1 up	a_2 left	a_3 right
s_1	-0.20	0.35	0.40
s_2	0.10	0.25	0.10
s_3	-	-	-

$$Q(s, a) = r + \gamma \max_{a'} Q(s', a')$$

Q-Learning

After
15th episode

$Q[s, a]$	a_1 up	a_2 left	a_3 right
s_1	- 0.20	0.18	0.30
s_2	0.10	0.08	- 0.00
After	-	-	-

100th episode

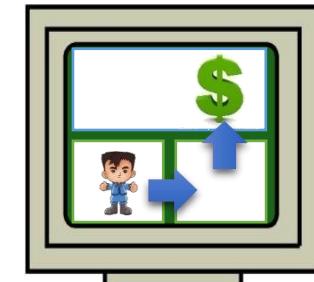
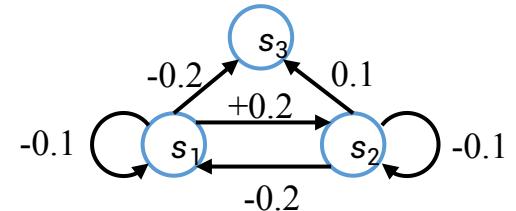
$Q[s, a]$	a_1 up	a_2 left	a_3 right
s_1	- 0.20	0.12	0.28
s_2	0.10	0.02	- 0.02
s_3	-	-	-

After
50th episode

$Q[s, a]$	a_1 up	a_2 left	a_3 right
s_1	- 0.20	0.12	0.28
s_2	0.10	0.02	- 0.02
After	-	-	-

1000th episode

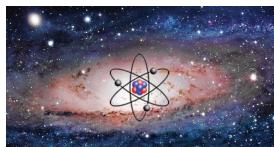
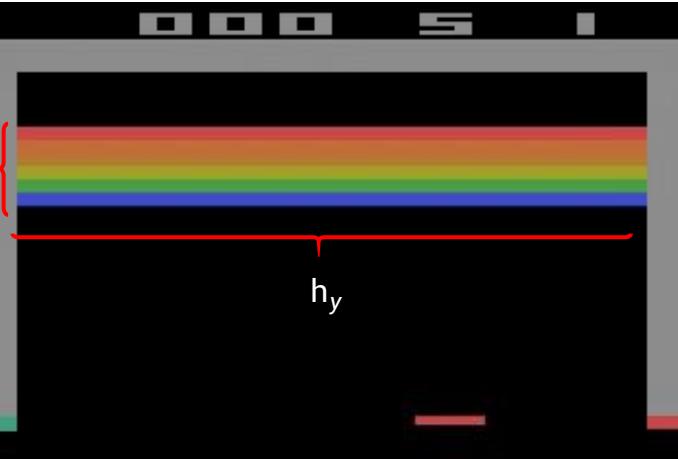
$Q[s, a]$	a_1 up	a_2 left	a_3 right
s_1	- 0.20	0.12	0.28
s_2	0.10	0.02	- 0.02
s_3	-	-	-



Move up/left/right



Q-Learning



$$< (3 * 256)^{w_x * h_y} < \text{number of states}$$

Q-table

$Q[s, a]$	a_1	a_2	\dots	a_m
s_1				
s_2				
s_3				
:				
s_n				

$\left\{ \begin{array}{l} < s, a, r, s' > \\ s \leftarrow s' \end{array} \right.$

Too huge states space to approximate Q-function iteratively by Q-table!!!

Conclusion – Machine Learning

1. Supervised Learning

- Linear Regression
- Logistic Regression
- Classification
 - Distance-based algorithms
 - Linear classifiers
 - Other classifiers

2. Unsupervised Learning

- Clustering
 - K-means method
 - Spectral clustering
- Representation learning

3. Reinforcement Learning

- Q-Learning, Q-table
- Exploration & Exploitation

Homework

1. 判断题

- PCA只能用于二维数据的降维。 ()
- 聚类分析的目标是使得簇内相似度最大，簇间相似度最小。 ()
- 无监督学习的评估通常比监督学习更容易。 ()
- 降维数据一定会导致数据信息的损失。 ()
- Q-table本质上是Agent的动作选择策略。 ()

2. 设计机器学习模型需要考虑哪些非技术因素？