

CSE 446

K-means / GMM

Natasha Jaques



CSE 446

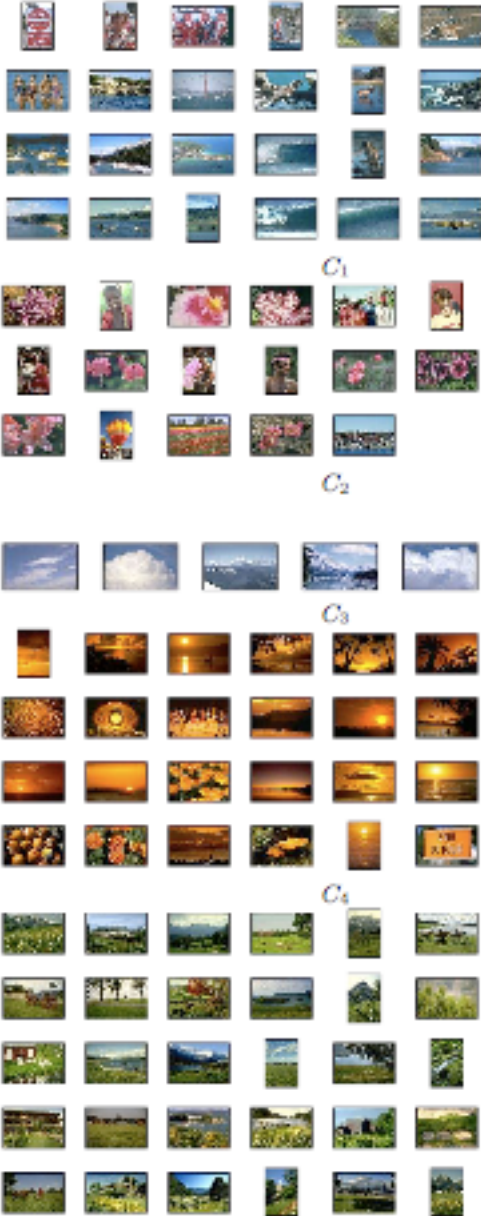
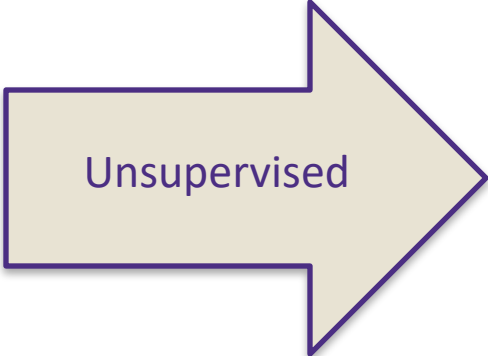
- Supervised learning
 - Linear models
 - Linear regression
 - Ridge regression
 - LASSO regression
 - Logistic regression
 - Non-linear models
 - Kernel methods
 - Neural Networks
 - Non-parametric methods
- **Unsupervised learning**
 - PCA/SVD
 - **Clustering**
 - **k-means**
 - **Gaussian Mixture Models (GMM)**
 - **Spectral Clustering**
- Modern machine learning / Advanced topics

Clustering

a fundamental unsupervised task



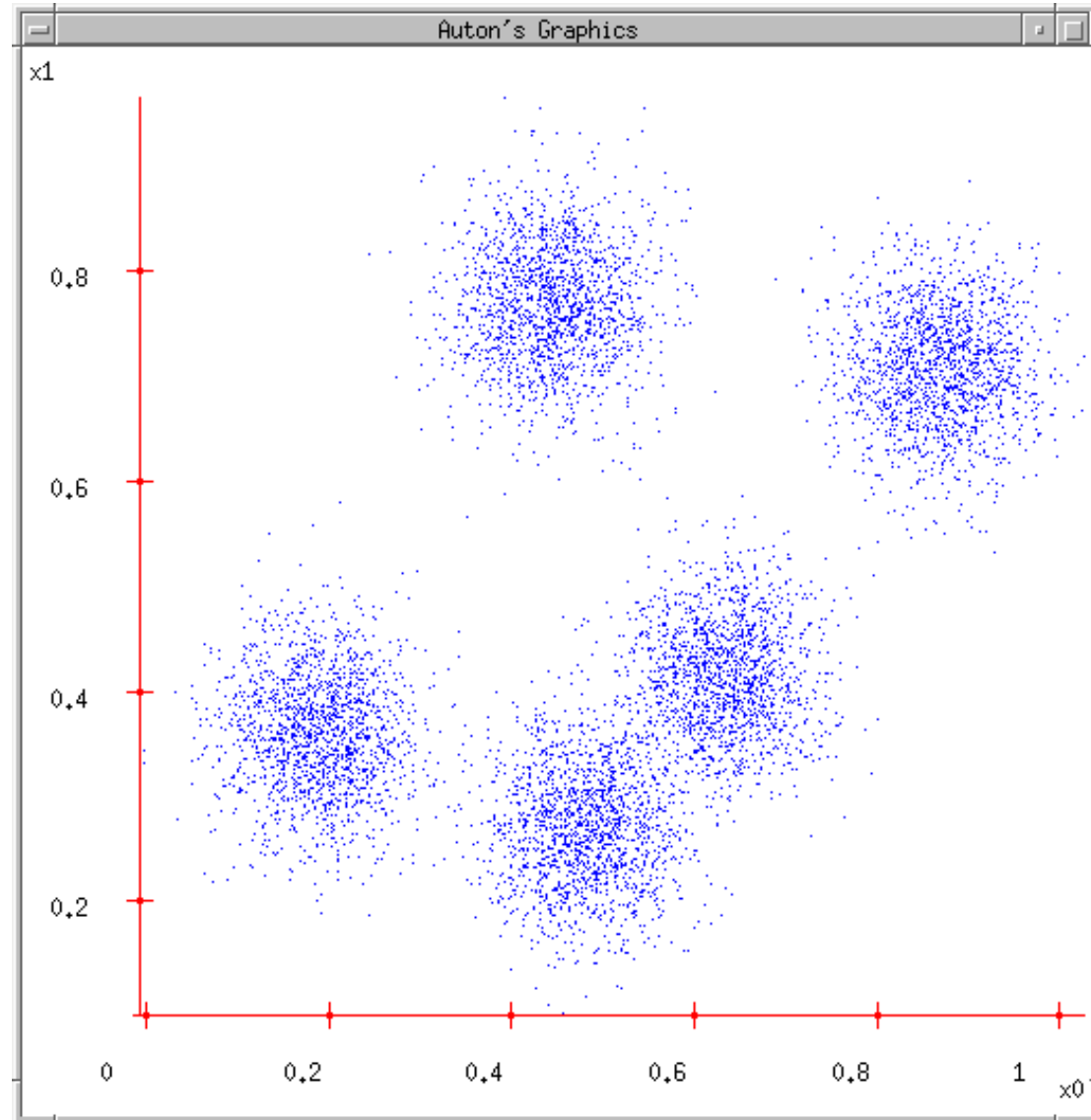
Clustering images



[Goldberger et al.]

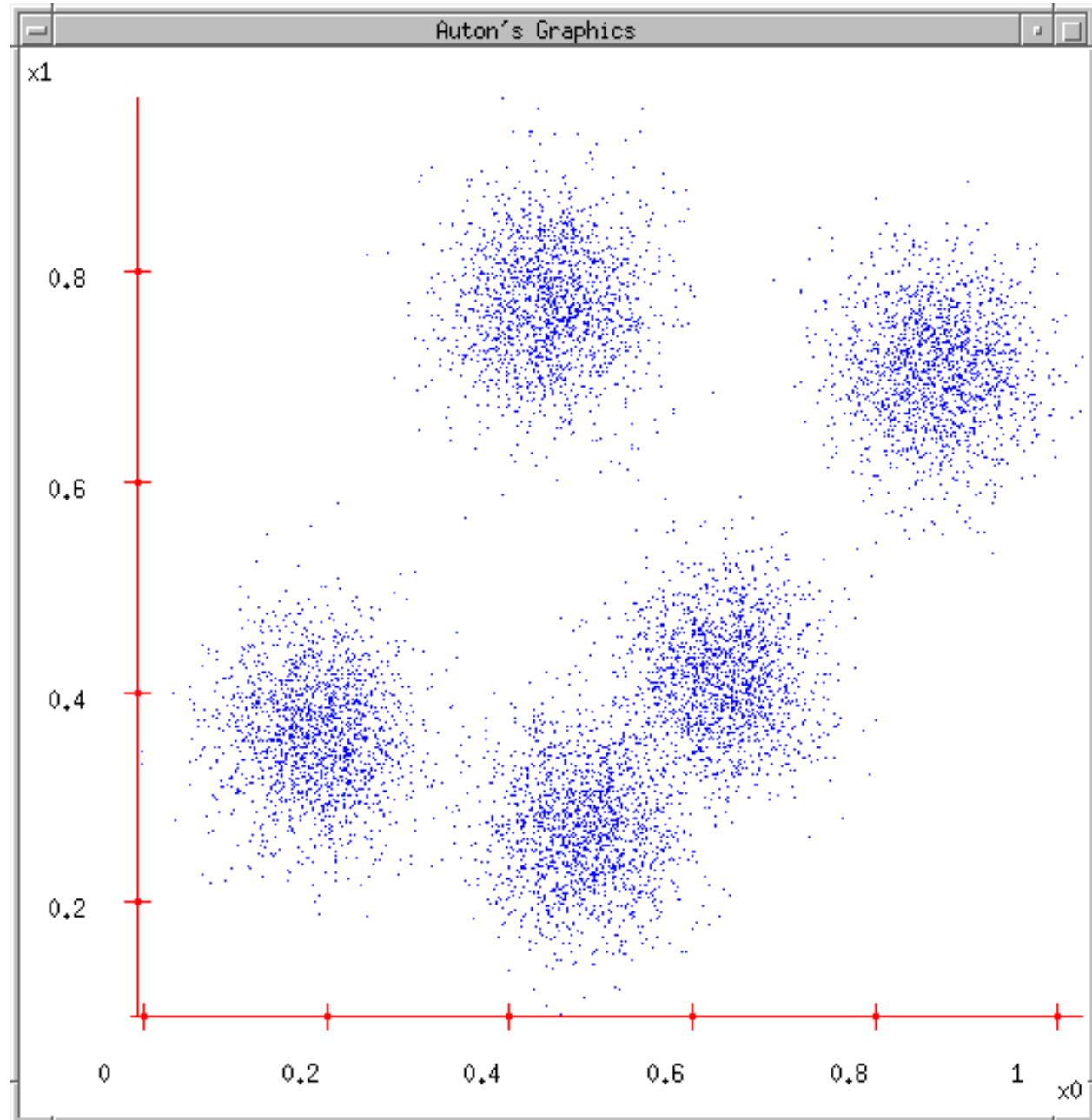
Example of 2-dimensional data points

- k -means algorithm assumes this kind of **structured** data, which will be clear once we learn what the algorithm does



k-means

1. Ask user how many clusters they'd like. (e.g. $k=5$)

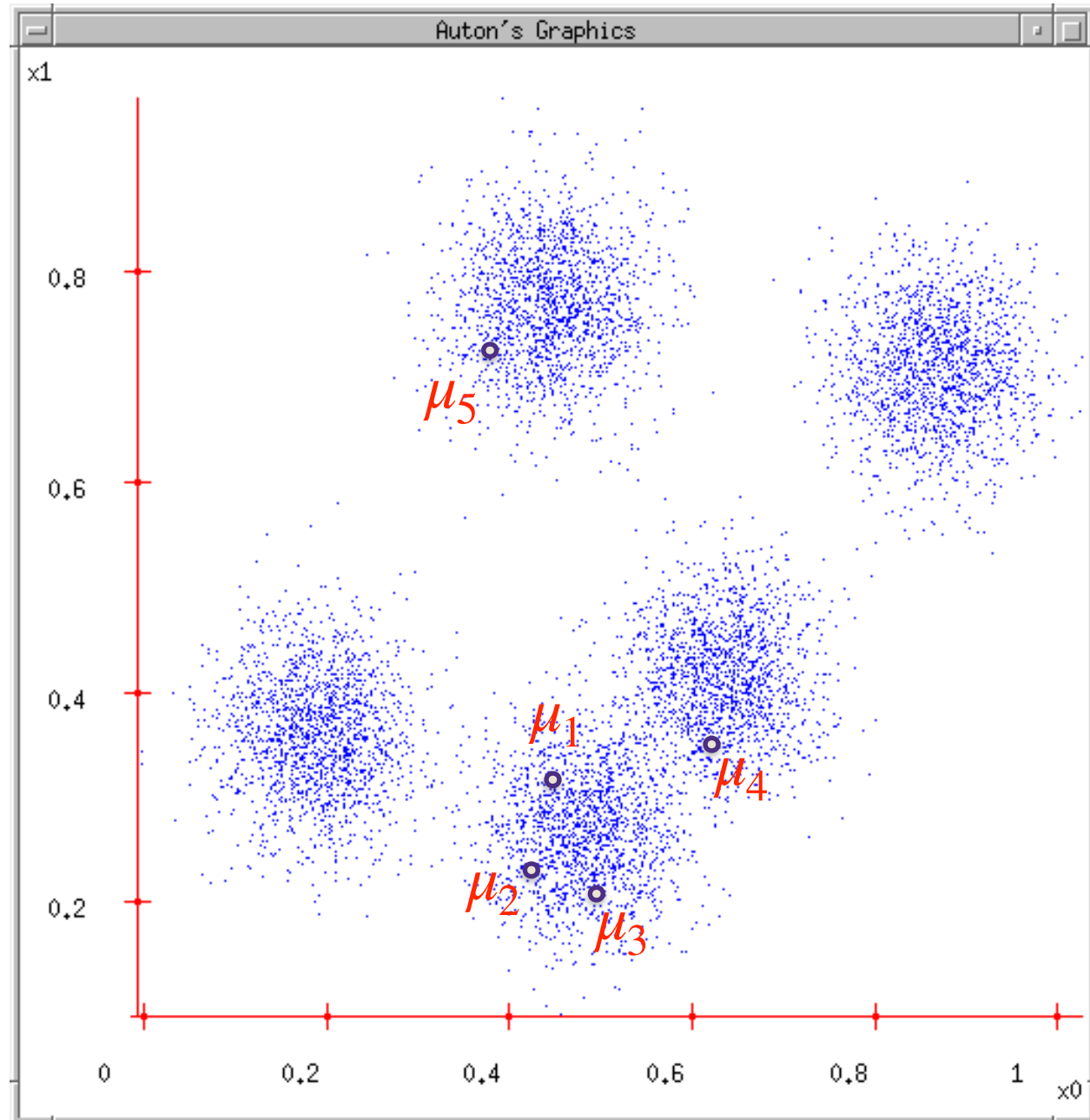


k-means

1. Ask user how many clusters they'd like. (e.g. $k=5$)

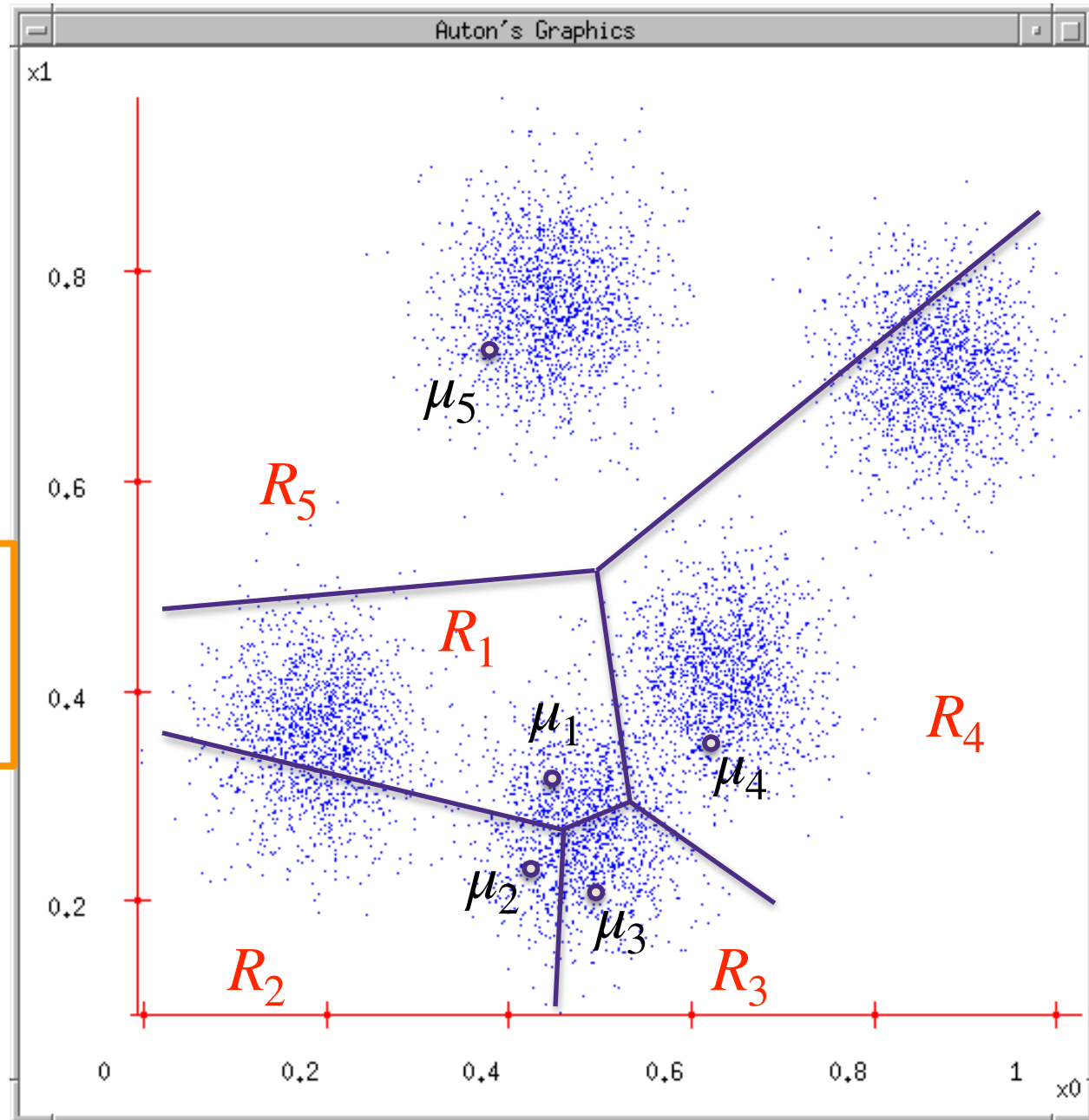
2. Randomly guess k cluster Center locations

$$\{\mu_1, \dots, \mu_5\}$$



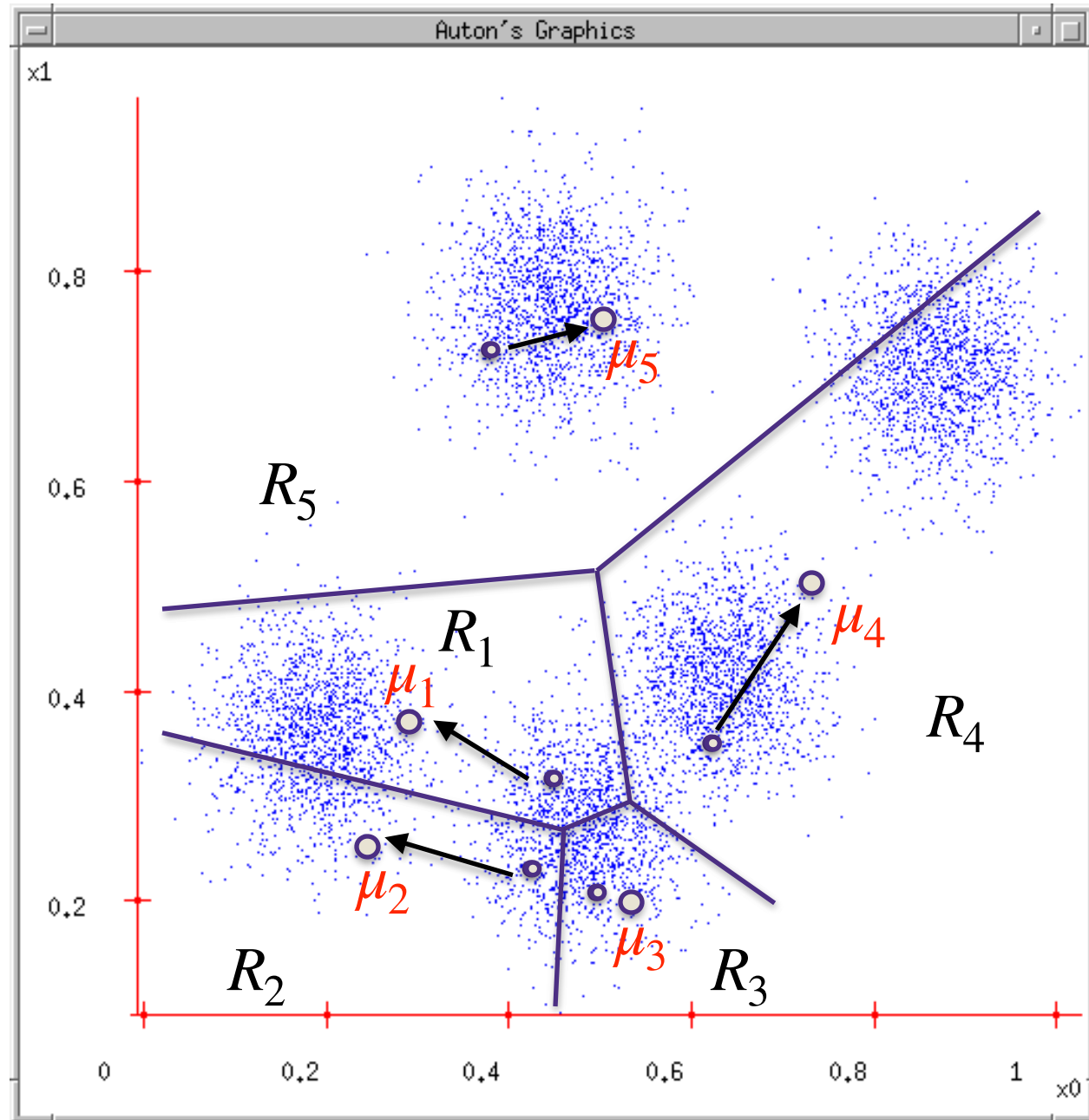
k-means

1. Ask user how many clusters they'd like. (e.g. $k=5$)
2. Randomly guess k cluster Center locations
 $\{\mu_1, \dots, \mu_5\}$
3. Each datapoint finds out which Center it's closest to.
(Thus each Center "owns" a set of datapoints)



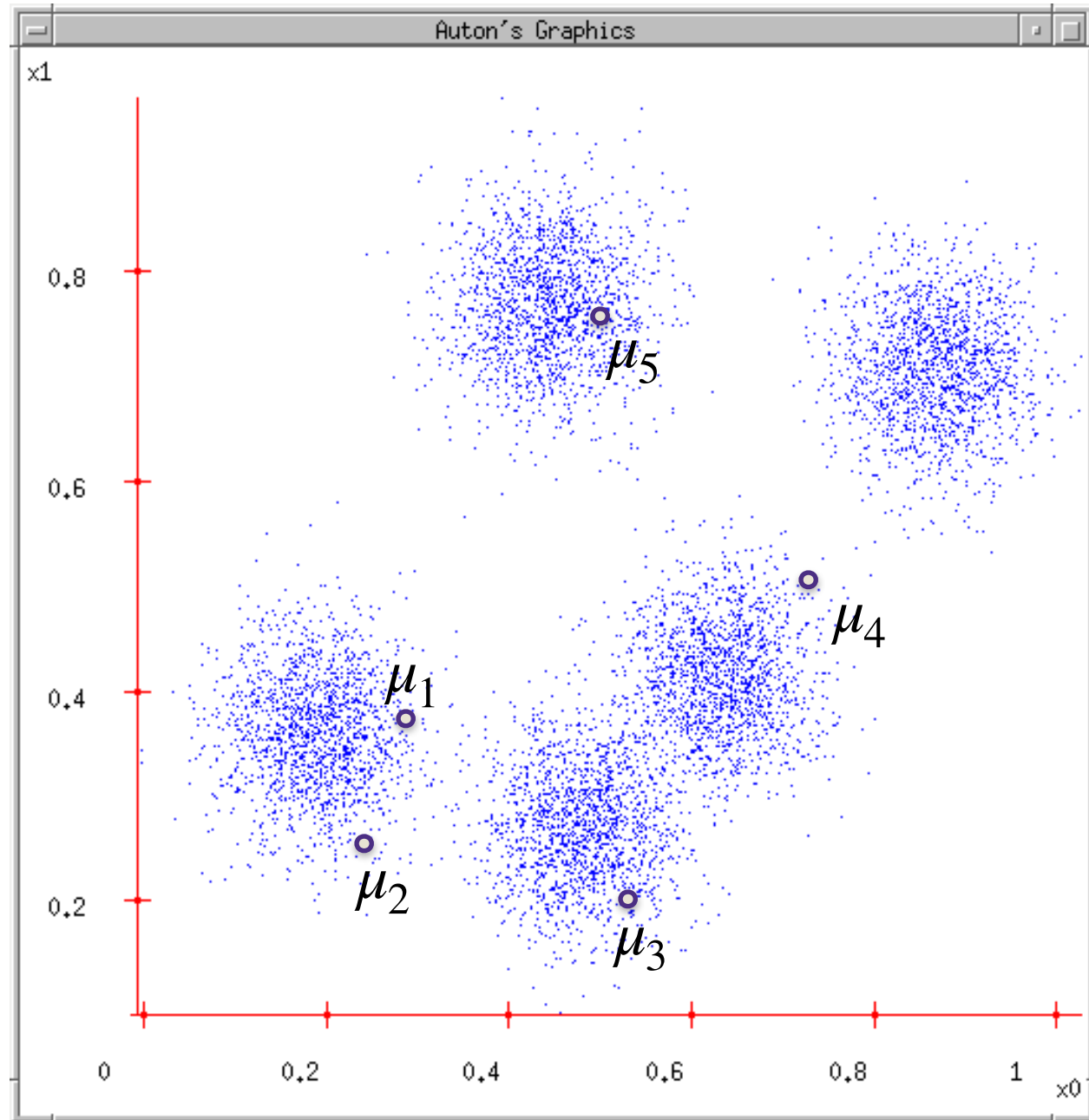
k -means

1. Ask user how many clusters they'd like. (e.g. $k=5$)
2. Randomly guess k cluster Center locations
 $\{\mu_1, \dots, \mu_5\}$
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds a new centroid of the points it owns



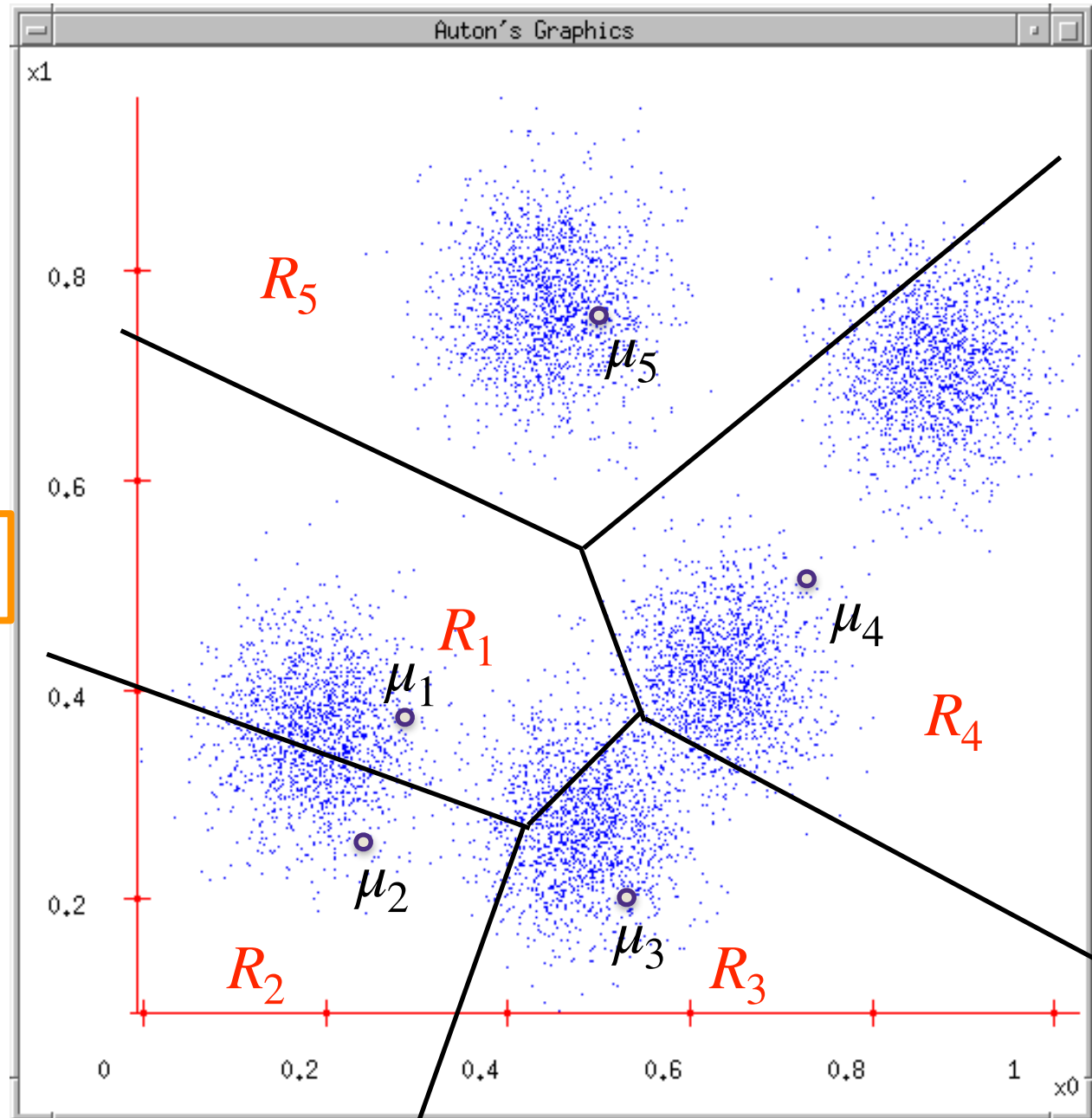
k-means

1. Ask user how many clusters they'd like. (e.g. $k=5$)
2. Randomly guess k cluster Center locations
 $\{\mu_1, \dots, \mu_5\}$
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds a new centroid of the points it owns...
5. ...and jumps there
6. ...Repeat until terminated!



k-means

1. Ask user how many clusters they'd like. (e.g. $k=5$)
2. Randomly guess k cluster Center locations
 $\{\mu_1, \dots, \mu_5\}$
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns...
5. ...and jumps there
6. ...Repeat until terminated!



k-means

<https://www.naftaliharris.com/blog/visualizing-k-means-clustering/>

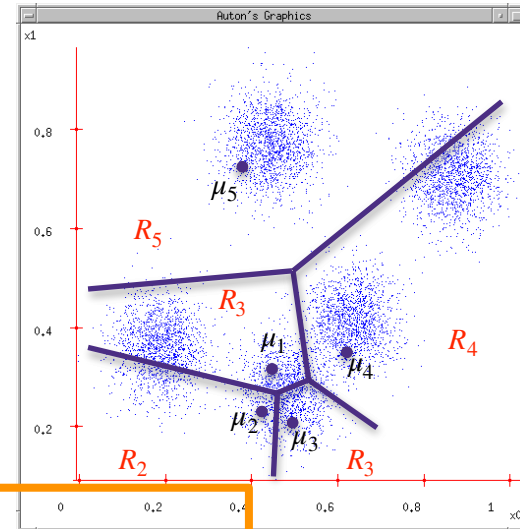
K-means algorithm

1. Choose k , how many clusters to find
2. Randomly initialize k centers
 - $\mu^{(0)} = [\mu_1^{(0)}, \dots, \mu_k^{(0)}] \in \mathbb{R}^{d \times k}$
 - Usually randomly chosen from the data points, to make sure they are in the right domain
- For $t=0,1,2,\dots$ repeat

K-means

1. Choose k , how many clusters to find
2. Randomly initialize k centers
 - $\mu^{(0)} = [\mu_1^{(0)}, \dots, \mu_k^{(0)}] \in \mathbb{R}^{d \times k}$
 - Usually randomly chosen from the data points, to make sure they are in the right domain
- For $t=0,1,2,\dots$ repeat
3. Assign each point $j \in \{1, \dots, n\}$ to its nearest center:

Assignment:



$$C_j^{(t)}$$

$$\leftarrow \arg \min_{i \in \{1, 2, \dots, k\}} \|x_j - \mu_i^{(t)}\|^2$$

Membership of the j -th data point

K-means

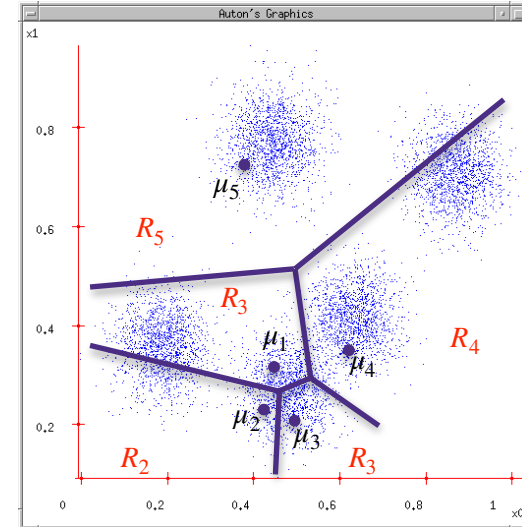
1. Choose k , how many clusters to find
2. Randomly initialize k centers
 - $\mu^{(0)} = [\mu_1^{(0)}, \dots, \mu_k^{(0)}] \in \mathbb{R}^{d \times k}$
 - Usually randomly chosen from the data points, to make sure they are in the right domain
- For $t=0,1,2,\dots$ repeat
3. Assign each point $j \in \{1, \dots, n\}$ to its nearest center:

$$C_j^{(t)} \leftarrow \arg \min_{i \in \{1, 2, \dots, k\}} \|x_j - \mu_i^{(t)}\|^2$$

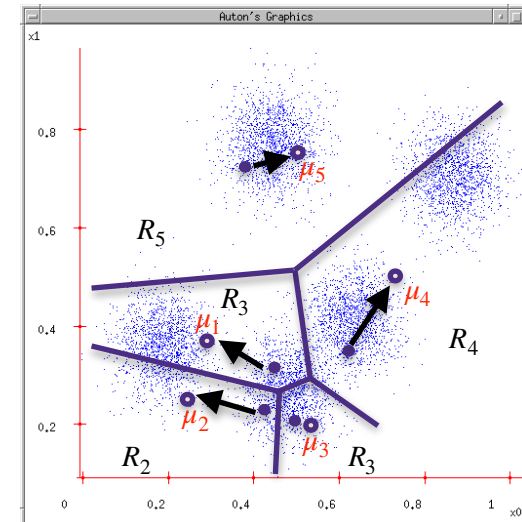
4. Recenter: μ_i becomes centroid of its point:

$$\mu_i^{(t+1)} \leftarrow \arg \min_{\mu} \sum_{j: C_j^{(t)}=i} \|\mu - x_j\|^2$$

Assignment:



Recenter:



K-means

1. Choose k , how many clusters to find
2. Randomly initialize k centers
 - $\mu^{(0)} = [\mu_1^{(0)}, \dots, \mu_k^{(0)}] \in \mathbb{R}^{d \times k}$
 - Usually randomly chosen from the data points, to make sure they are in the right domain
- For $t=0,1,2,\dots$ repeat
3. Assign each point $j \in \{1, \dots, n\}$ to its nearest center:

$$C_j^{(t)} \leftarrow \arg \min_{i \in \{1, 2, \dots, k\}} \|x_j - \mu_i^{(t)}\|^2$$

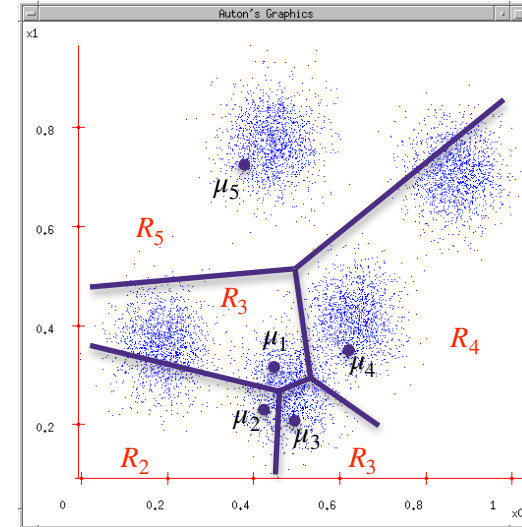
4. Recenter: μ_i becomes centroid of its point:

$$\mu_i^{(t+1)} \leftarrow \arg \min_{\mu} \sum_{j: C_j^{(t)}=i} \|\mu - x_j\|^2$$

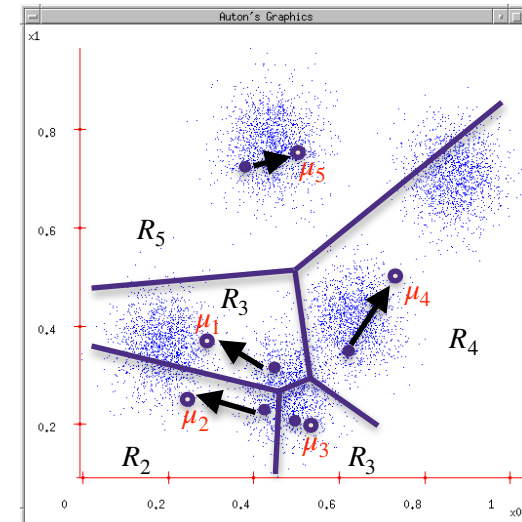
- Equivalent to

$$\mu_i^{(t+1)} \leftarrow \text{average of all the points assigned to } \mu_i^{(t)}$$

Assignment:



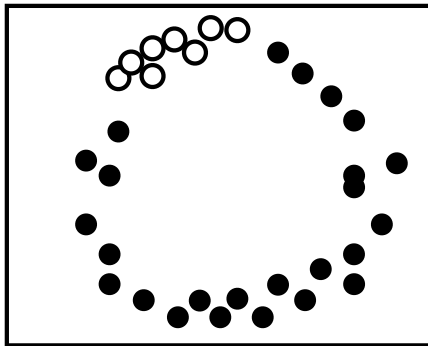
Recenter:



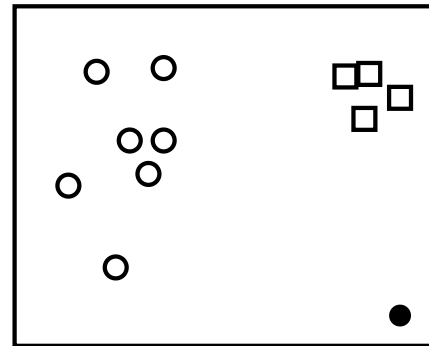
Which one is a snapshot of a converged k -means

When k -means is converged, there should be a set of centers and assignments that do not change when applying 1 step of k -means

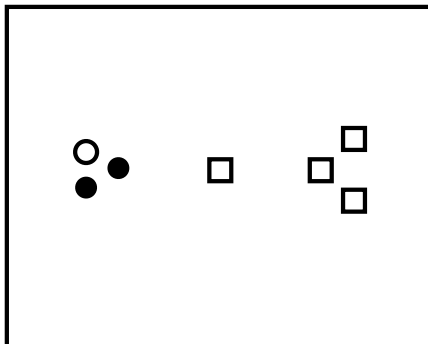
Example (a)



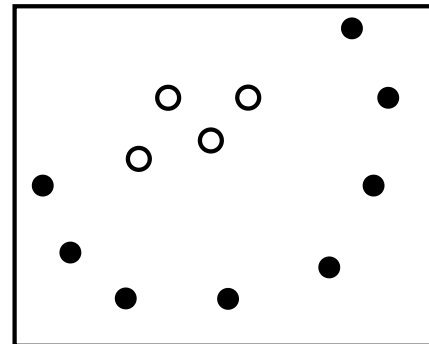
Example (b)



Example (c)



Example (d)



Does k -means converge?

- k -means is trying to minimize the following objective

$$\min_{\{\mu_i\}_{i=1}^k} \min_{\{C_\ell\}_{\ell=1}^n} \sum_{j=1}^n \|x_j - \mu_{C_j}\|^2$$

via alternating minimization
(equivalent to coordinate descent)

- Fix μ , optimize C
- Fix C , optimize μ
- Does this converge? Does this terminate in finite time?

Does k -means converge?

- there is only a finite set of values that $\{C_j\}_{j=1}^n \in \{1, \dots, k\}^n$ can take (k^n is large but finite)
- so there is only finite, k^n at most, values for cluster-centers, $\{\mu_i\}_{i=1}^k$, also
- each time we update them, we will never increase the objective function:

$$\sum_{j=1}^n \|x_j - \mu_{C_j}\|_2^2$$

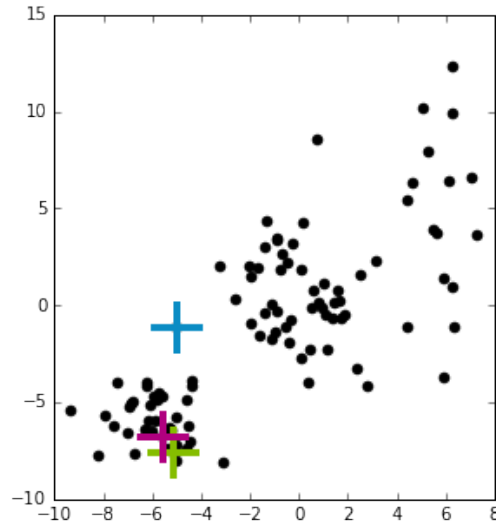
- the objective is lower bounded by zero
- after at most k^n steps, the algorithm must converge (as the assignments $\{C_j\}_{j=1}^n$ cannot return to previous assignments in the course of k -means iterations)

downsides of k -means

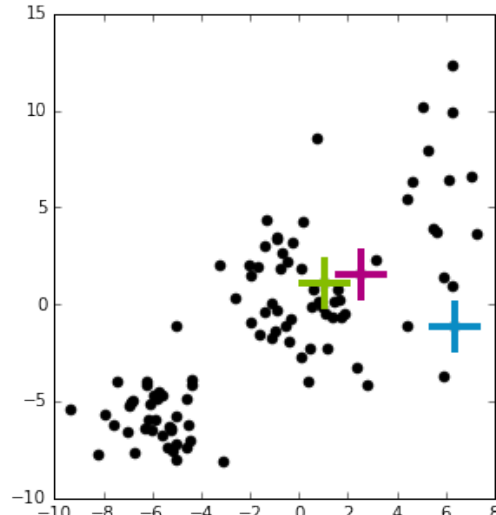
1. it requires the number of clusters k to be specified by us
2. the final solution depends on the initialization
(does not find global minimum of the objective)

Initial position of centers

Trial 1



Trial 2



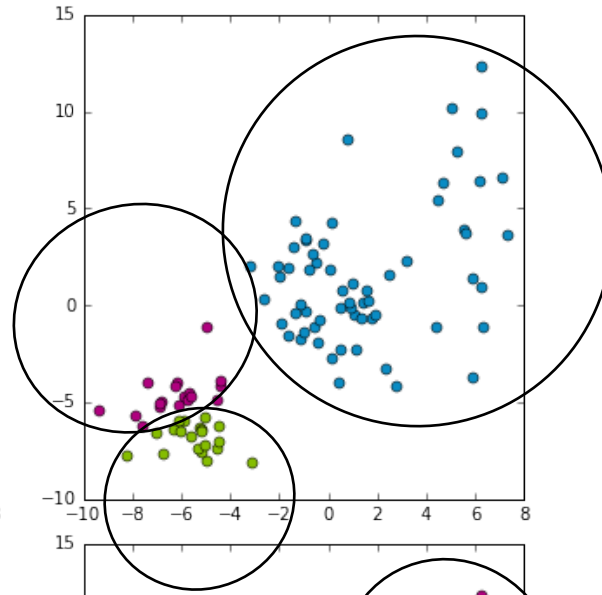
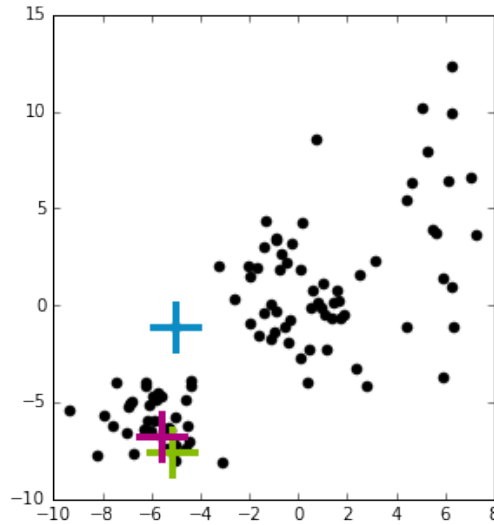
downsides of k -means

1. it requires the number of clusters k to be specified by us
2. the final solution depends on the initialization
(does not find global minimum of the objective)

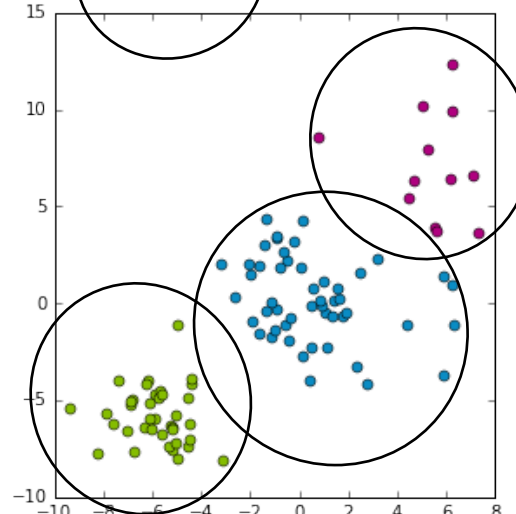
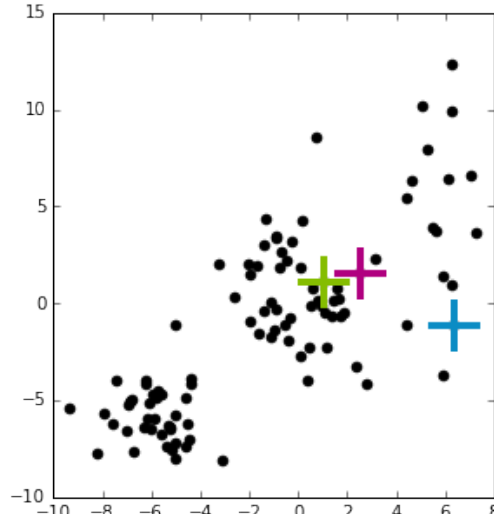
Initial position of centers

final converged assignment

Trial 1



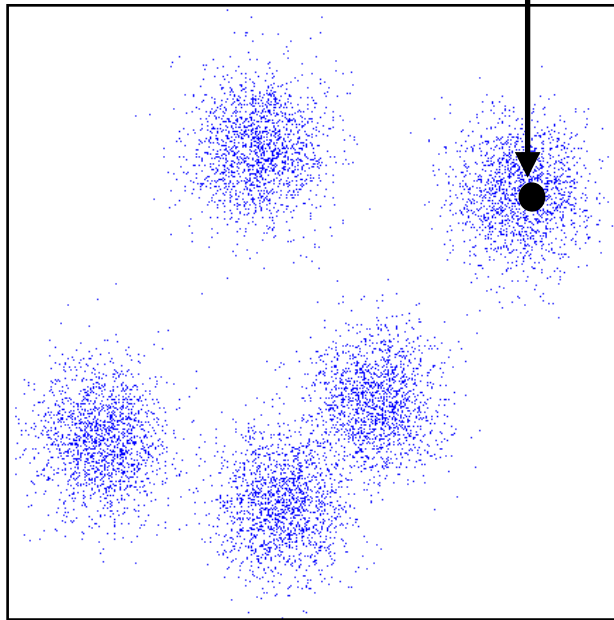
Trial 2



k -means++: a smart initialization

Smart initialization:

1. Choose **first** cluster center μ_1 uniformly at random from data points



k -means++: a smart initialization

Smart initialization:

1. Choose **first** cluster center μ_1 uniformly at random from data points

2. For $k=2, \dots, K$

3. For each data point x_i , compute distance d_i to nearest cluster center

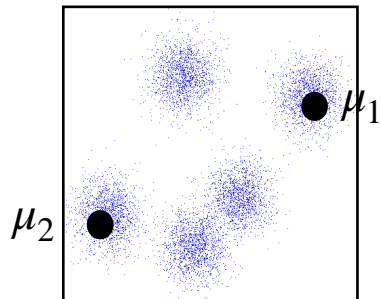
4. Choose new cluster center from amongst data points, with probability

of x_i being chosen proportional to $(d_i)^2$

precisely,

$$d_i \leftarrow \min_{j \in \{1, \dots, k-1\}} \|\mu_j - x_i\|, \text{ for all } i \text{ that is not chosen already}$$

$$\text{Prob}(x_i \text{ chosen as the next center}) = \frac{(d_i)^2}{\sum_{\ell} (d_{\ell})^2}$$



k -means++: a smart initialization

Smart initialization:

1. Choose **first** cluster center μ_1 uniformly at random from data points
2. For $k=2, \dots, K$
 3. For each data point x_i , compute distance d_i to nearest cluster center
 4. Choose new cluster center from amongst data points, with probability of x_i being chosen proportional to $(d_i)^2$

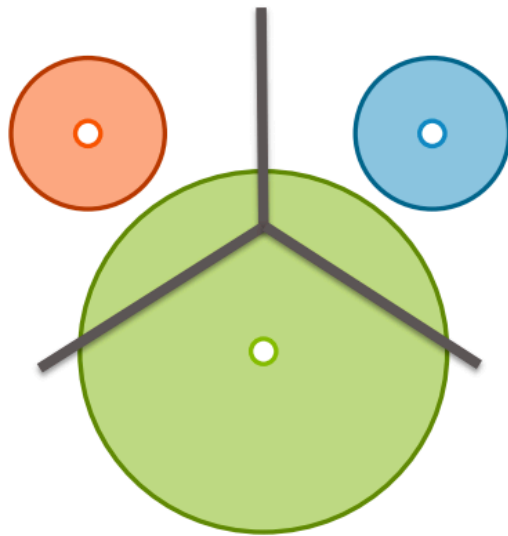
precisely,

$$d_i \leftarrow \min_{j \in \{1, \dots, k-1\}} \|\mu_j - x_i\|, \text{ for all } i \text{ that is not chosen already}$$

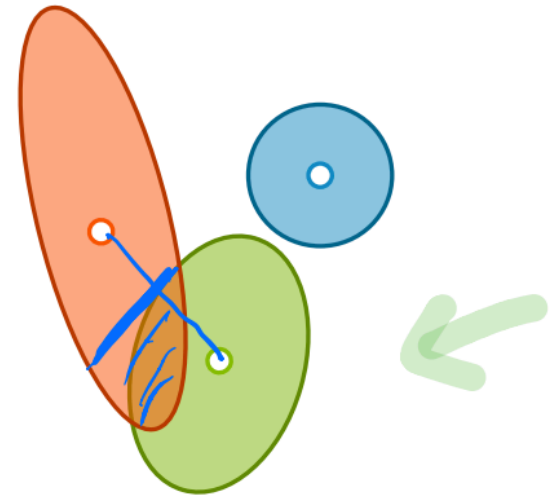
$$\text{Prob}(x_i \text{ chosen as the next center}) = \frac{(d_i)^2}{\sum_{\ell} (d_{\ell})^2}$$

- apply standard K-means after this initialization

- K-means algorithm fails, when the data has:



disparate cluster sizes



different
shaped/oriented
clusters

- What can we do?

CSE 446/546 2025 winter

- Supervised learning
 - Linear models (lectures 1-9)
 - Linear regression
 - Ridge regression
 - LASSO regression
 - Logistic regression
 - Non-linear models (lectures 10-13)
 - Kernel methods
 - Neural Networks
 - Non-parametric methods
- **Unsupervised learning (lectures 14-16)**
 - PCA/SVD
 - **Clustering**
 - **k-means**
 - **Gaussian Mixture Models (GMM)**
 - **Spectral Clustering**
- Modern machine learning (lectures 17-19)

Gaussian Mixture Models

Extension of k-means



Gaussian Mixture Model

- input: data $\{x_i\}_{i=1}^n$ in \mathbb{R}^d
- parameters of a **Gaussian Mixture Model**
 - mixing weights:
 - $\pi_j = \mathbf{P}(\text{cluster membership} = j)$ for $j \in \{1, \dots, K\}$
 - means:
 - $\mu_j \in \mathbb{R}^d$ for $j \in \{1, \dots, K\}$
 - covariance matrices:
 - $\Sigma_j \in \mathbb{R}^{d \times d}$ for $j \in \{1, \dots, K\}$
- we suppose that the given data has been generated from a GMM, and try to find the best GMM parameters (this naturally will define clustering of the training data)

Gaussian Mixture Model

- input: data $\{x_i\}_{i=1}^n$ in \mathbb{R}^d
- parameters of a **Gaussian Mixture Model**
 - mixing weights:
 - $\pi_j = \mathbf{P}(\text{cluster membership} = j)$ for $j \in \{1, \dots, K\}$
 - means:
 - $\mu_j \in \mathbb{R}^d$ for $j \in \{1, \dots, K\}$
 - covariance matrices:
 - $\Sigma_j \in \mathbb{R}^{d \times d}$ for $j \in \{1, \dots, K\}$
- we suppose that the given data has been generated from a GMM, and try to find the best GMM parameters (this naturally will define clustering of the training data)

- under the GMM, the i -th sample is drawn as follows
 - first sample a cluster $z_i \in \{1, \dots, K\}$, from $\pi = [\pi_1, \dots, \pi_K]$
 - conditioned on this cluster, x_i is sampled from

$$x_i \sim N(\mu_{z_i}, \Sigma_{z_i})$$

Maximum likelihood estimation (MLE)

- we can find the best GMM for given data, by MLE
- for simplicity, suppose $d = 1$ and $K = 2$
- Model parameters are $\pi_1, \pi_2, \mu_1, \mu_2, \sigma_1^2, \sigma_2^2 \in \mathbb{R}$
- the probability of observing a sample x_i can be written as

$$\mathbf{P}(x_i; \pi_1, \pi_2, \mu_1, \mu_2, \sigma_1^2, \sigma_2^2) = \pi_1 \underbrace{\frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{(x_i - \mu_1)^2}{2\sigma_1^2}}}_{\triangleq N(x_i; \mu_1, \sigma_1^2)} + \pi_2 \underbrace{\frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-\frac{(x_i - \mu_2)^2}{2\sigma_2^2}}}_{\triangleq N(x_i; \mu_2, \sigma_2^2)}$$

Maximum likelihood estimation (MLE)

- we can find the best GMM for given data, by MLE
- for simplicity, suppose $d = 1$ and $K = 2$
- Model parameters are $\pi_1, \pi_2, \mu_1, \mu_2, \sigma_1^2, \sigma_2^2 \in \mathbb{R}$
- the probability of observing a sample x_i can be written as

$$\mathbf{P}(x_i; \pi_1, \pi_2, \mu_1, \mu_2, \sigma_1^2, \sigma_2^2) = \underbrace{\pi_1 \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{(x_i - \mu_1)^2}{2\sigma_1^2}}}_{\triangleq N(x_i; \mu_1, \sigma_1^2)} + \underbrace{\pi_2 \frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-\frac{(x_i - \mu_2)^2}{2\sigma_2^2}}}_{\triangleq N(x_i; \mu_2, \sigma_2^2)}$$

- MLE tries to find

$$\arg \max_{\pi_1, \pi_2, \mu_1, \mu_2, \sigma_1^2, \sigma_2^2} \sum_{i=1}^n \log \mathbf{P}(x_i; \pi_1, \pi_2, \mu_1, \mu_2, \sigma_1^2, \sigma_2^2)$$

- however, unlike least squared or logistic regression, this is not a concave function of the parameters (thus hard to find the optimal solution)
- in general, MLE of a mixture model is not convex/concave optimization

Recall lecture 1: fitting a single Gaussian model

- given $\{x_i\}_{i=1}^n \in \mathbb{R}$, fit the best Gaussian model with mean $\mu \in \mathbb{R}$ and variance $\sigma^2 \in \mathbb{R}$
- using MLE we want to solve

$$\text{maximize}_{\mu, \sigma^2} \mathcal{L}(\mu, \sigma^2) = \sum_{i=1}^n \underbrace{\left(-\frac{(x_i - \mu)^2}{2\sigma^2} - \log(\sqrt{2\pi\sigma^2}) \right)}_{\log N(x_i|\mu, \sigma^2)}$$

- we compute gradient and set it to zero:

$$\bullet \quad \nabla_{\mu} \mathcal{L}(\mu, \sigma^2) = \frac{1}{\sigma^2} \sum_{i=1}^n (\mu - x_i)$$

which is zero for $\mu = \frac{1}{n} \sum_{i=1}^n x_i$

(which makes sense as it is the empirical mean)

Recall lecture 1: fitting a single Gaussian model

- given $\{x_i\}_{i=1}^n \in \mathbb{R}$, fit the best Gaussian model with mean $\mu \in \mathbb{R}$ and variance $\sigma^2 \in \mathbb{R}$
- using MLE we want to solve

$$\text{maximize}_{\mu, \sigma^2} \mathcal{L}(\mu, \sigma^2) = \sum_{i=1}^n \underbrace{\left(-\frac{(x_i - \mu)^2}{2\sigma^2} - \log(\sqrt{2\pi\sigma^2}) \right)}_{\log N(x_i|\mu, \sigma^2)}$$

- we compute gradient and set it to zero:

- $$\nabla_{\mu} \mathcal{L}(\mu, \sigma^2) = \frac{1}{\sigma^2} \sum_{i=1}^n (\mu - x_i)$$

which is zero for
$$\mu = \frac{1}{n} \sum_{i=1}^n x_i$$

(which makes sense as it is the empirical mean)

- $$\nabla_{\sigma^2} \mathcal{L}(\mu, \sigma^2) = \frac{\sum_{i=1}^n (x_i - \mu)^2}{2(\sigma^2)^2} - \frac{n}{2\sigma^2}$$

which is zero for
$$\sigma^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2$$

(which makes sense as it is the empirical variance)

MLE for GMM

- we want to fit a model by solving

$$\max_{\pi_1, \pi_2, \mu_1, \mu_2, \sigma_1^2, \sigma_2^2} \sum_{i=1}^n \log \left(\underbrace{\pi_1 \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{(x_i - \mu_1)^2}{2\sigma_1^2}}}_{\triangleq N(x_i; \mu_1, \sigma_1^2)} + \underbrace{\pi_2 \frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-\frac{(x_i - \mu_2)^2}{2\sigma_2^2}}}_{\triangleq N(x_i; \mu_2, \sigma_2^2)} \right)$$

MLE for GMM

- we want to fit a model by solving

$$\max_{\pi_1, \pi_2, \mu_1, \mu_2, \sigma_1^2, \sigma_2^2} \sum_{i=1}^n \log \left(\underbrace{\pi_1 \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{(x_i - \mu_1)^2}{2\sigma_1^2}}}_{\triangleq N(x_i; \mu_1, \sigma_1^2)} + \underbrace{\pi_2 \frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-\frac{(x_i - \mu_2)^2}{2\sigma_2^2}}}_{\triangleq N(x_i; \mu_2, \sigma_2^2)} \right)$$

- define $r_i = \mathbf{P}(z_i = 1 | x_i) = \frac{\mathbf{P}(z_i = 1, x_i)}{\mathbf{P}(z_i = 1, x_i) + \mathbf{P}(z_i = 2, x_i)}$
$$= \frac{\pi_1 N(x_i; \mu_1, \sigma_1^2)}{\pi_1 N(x_i; \mu_1, \sigma_1^2) + \pi_2 N(x_i; \mu_2, \sigma_2^2)}$$

Estimated soft membership

MLE for GMM

- we want to fit a model by solving

$$\max_{\pi_1, \pi_2, \mu_1, \mu_2, \sigma_1^2, \sigma_2^2} \sum_{i=1}^n \log \left(\underbrace{\pi_1 \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{(x_i - \mu_1)^2}{2\sigma_1^2}}}_{\triangleq N(x_i; \mu_1, \sigma_1^2)} + \underbrace{\pi_2 \frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-\frac{(x_i - \mu_2)^2}{2\sigma_2^2}}}_{\triangleq N(x_i; \mu_2, \sigma_2^2)} \right)$$

- define $r_i = \mathbf{P}(z_i = 1 | x_i) = \frac{\mathbf{P}(z_i = 1, x_i)}{\mathbf{P}(z_i = 1, x_i) + \mathbf{P}(z_i = 2, x_i)}$

$$= \frac{\pi_1 N(x_i; \mu_1, \sigma_1^2)}{\pi_1 N(x_i; \mu_1, \sigma_1^2) + \pi_2 N(x_i; \mu_2, \sigma_2^2)}$$

- setting the gradient to zero, we get

- $\pi_1 = \frac{N_1}{n}$ where $N_1 = \sum_{i=1}^n r_i$, and $\pi_2 = \frac{N_2}{n}$ where $N_2 = \sum_{i=1}^n (1 - r_i)$

- $\mu_1 = \frac{1}{N_1} \sum_{i=1}^n r_i x_i$ and $\mu_2 = \frac{1}{N_2} \sum_{i=1}^n (1 - r_i) x_i$

- $\sigma_1^2 = \frac{1}{N_1} \sum_{i=1}^n r_i (x_i - \mu_1)^2$ and $\sigma_2^2 = \frac{1}{N_2} \sum_{i=1}^n (1 - r_i) (x_i - \mu_2)^2$

- both LHS and RHS depend on the parameters, and no closed form solution exists

- note that if we know r_i 's it is trivial to compute parameters, and vice versa**

Expectation Maximization (EM) algorithm to approximate the solution of MLE

- EM is a popular method to solve MLE for mixture models
- input: training data $\{x_i\}_{i=1}^n$
- output: $\pi_1, \pi_2, \mu_1, \mu_2, \sigma_1^2, \sigma_2^2 \in \mathbb{R}$
- initialization: randomly initialize the parameters

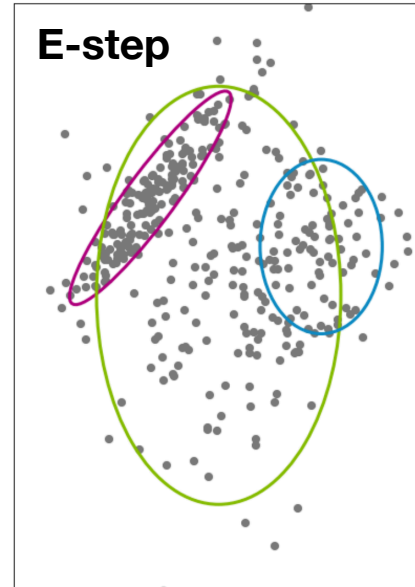
Expectation Maximization (EM) algorithm to approximate the solution of MLE

- EM is a popular method to solve MLE for mixture models
- input: training data $\{x_i\}_{i=1}^n$
- output: $\pi_1, \pi_2, \mu_1, \mu_2, \sigma_1^2, \sigma_2^2 \in \mathbb{R}$
- initialization: randomly initialize the parameters
- repeat

- **E-step** (Expectation): parameters \rightarrow soft membership

$$r_i = \frac{\pi_1 N(x_i; \mu_1, \sigma_1^2)}{\pi_1 N(x_i; \mu_1, \sigma_1^2) + \pi_2 N(x_i; \mu_2, \sigma_2^2)} \quad \text{for all } i \in \{1, 2, \dots, n\}$$

- **M-step** (Maximization): soft membership \rightarrow parameters



Expectation Maximization (EM) algorithm to approximate the solution of MLE

- EM is a popular method to solve MLE for mixture models

- input: training data $\{x_i\}_{i=1}^n$

- output: $\pi_1, \pi_2, \mu_1, \mu_2, \sigma_1^2, \sigma_2^2 \in \mathbb{R}$

- initialization: randomly initialize the parameters

- repeat

- **E-step** (Expectation): parameters \rightarrow soft membership

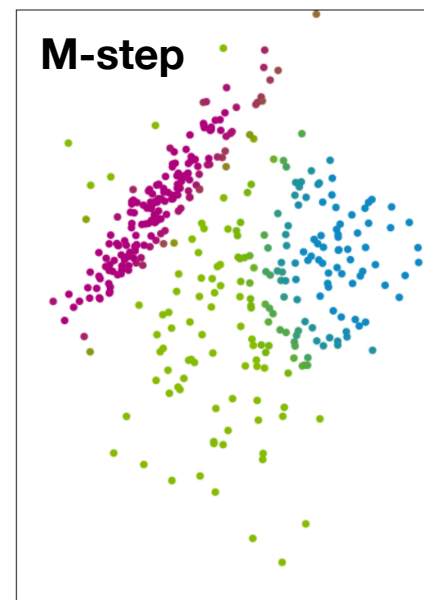
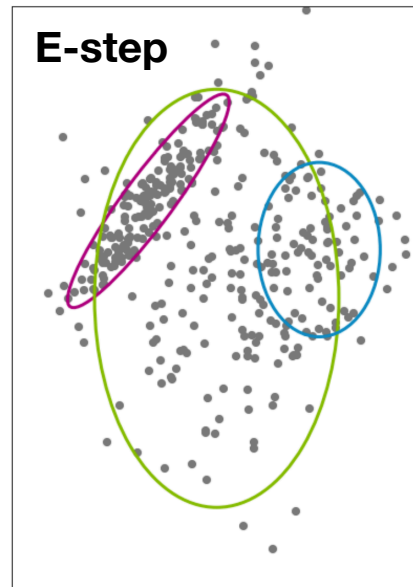
- $$r_i = \frac{\pi_1 N(x_i; \mu_1, \sigma_1^2)}{\pi_1 N(x_i; \mu_1, \sigma_1^2) + \pi_2 N(x_i; \mu_2, \sigma_2^2)} \quad \text{for all } i \in \{1, 2, \dots, n\}$$

- **M-step** (Maximization): soft membership \rightarrow parameters

- $$\pi_1 = \frac{N_1}{n} \text{ where } N_1 = \sum_{i=1}^n r_i, \text{ and } \pi_2 = \frac{N_2}{n} \text{ where } N_2 = \sum_{i=1}^n (1 - r_i)$$

- $$\mu_1 = \frac{1}{N_1} \sum_{i=1}^n r_i x_i \quad \text{and} \quad \mu_2 = \frac{1}{N_2} \sum_{i=1}^n (1 - r_i) x_i$$

- $$\sigma_1^2 = \frac{1}{N_1} \sum_{i=1}^n r_i (x_i - \mu_1)^2 \quad \text{and} \quad \sigma_2^2 = \frac{1}{N_2} \sum_{i=1}^n (1 - r_i) (x_i - \mu_2)^2$$



For general number of clusters K and dimension d

- we can derive EM for general case, in an analogous way
- Initialize parameters: $\pi_1, \dots, \pi_K, \mu_1, \dots, \mu_K, \Sigma_1, \dots, \Sigma_K$

- **E-step:**

- For $k=1, \dots, K$

$$r_{i,k} = \frac{\pi_k N(x_i | \mu_k, \Sigma_k)}{\sum_{j=1}^K \pi_j N(x_i | \mu_j, \Sigma_j)}$$

- **M-step:**

- For $k=1, \dots, K$

$$\pi_k = \frac{N_k}{n} \quad \text{where} \quad N_k = \frac{\sum_{i=1}^n r_{i,k}}{n}$$

$$\mu_k = \frac{1}{N_k} \sum_{i=1}^n r_{i,k} x_i \quad \text{and} \quad \Sigma_k = \frac{1}{N_k} \sum_{i=1}^n r_{i,k} (x_i - \mu_k)(x_i - \mu_k)^T$$

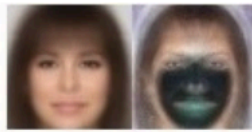
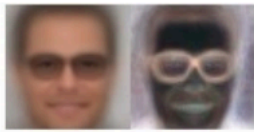
- once GMM is learned, clustering is straight forward: cluster according to the $r_{i,k}$'s

GMM for real data

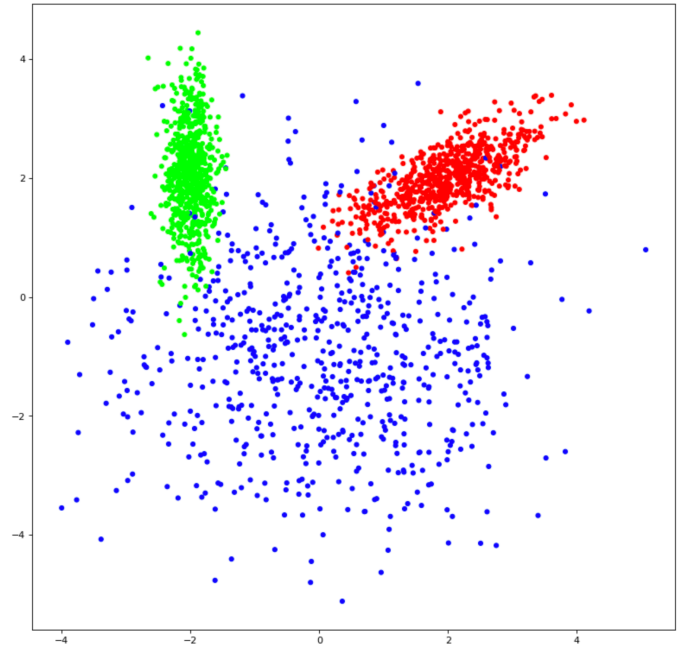
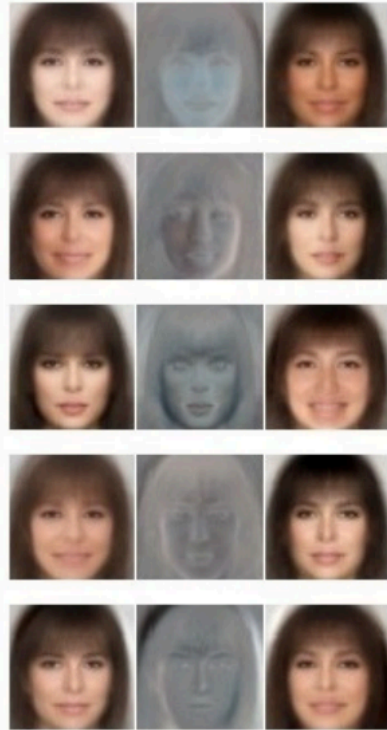
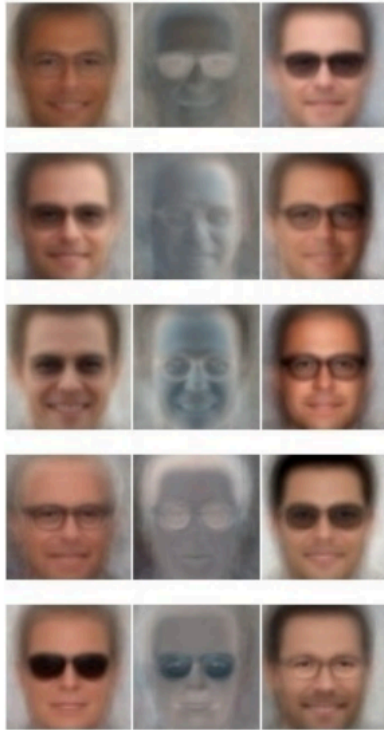


- these are generated samples, from GMM trained on CelebA dataset
- image: $64 \times 64 \times 3 = 288$ dimension
- covariance: restricted to rank-10 matrices
- mixture: $K=1,000$

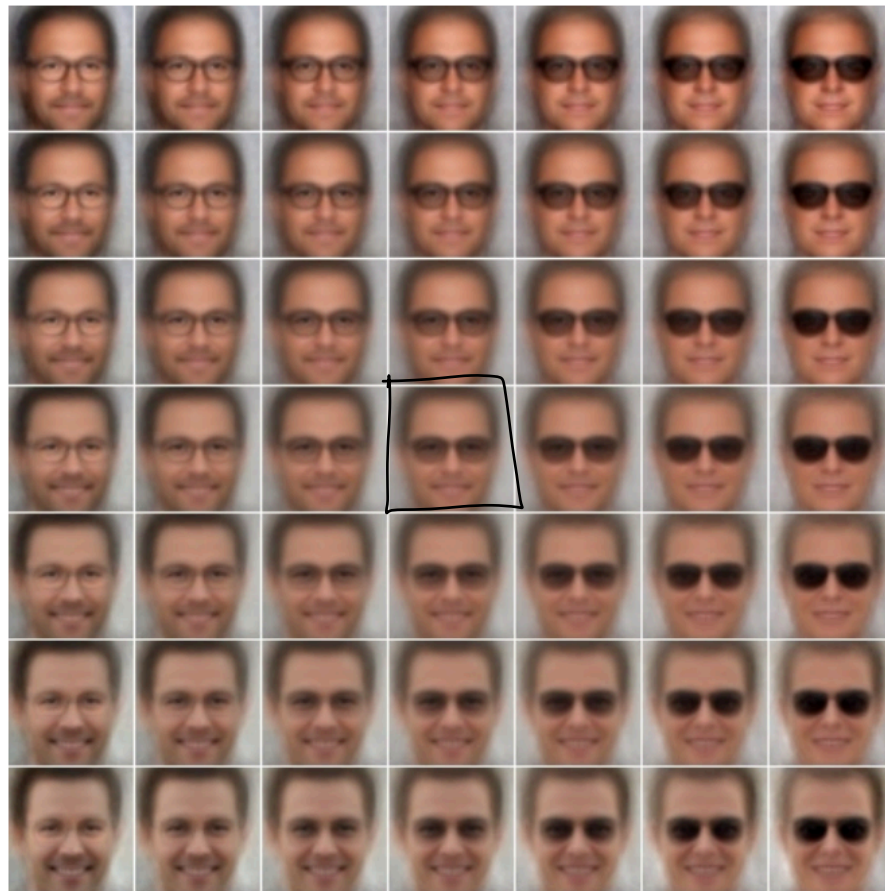
Center



Principal components

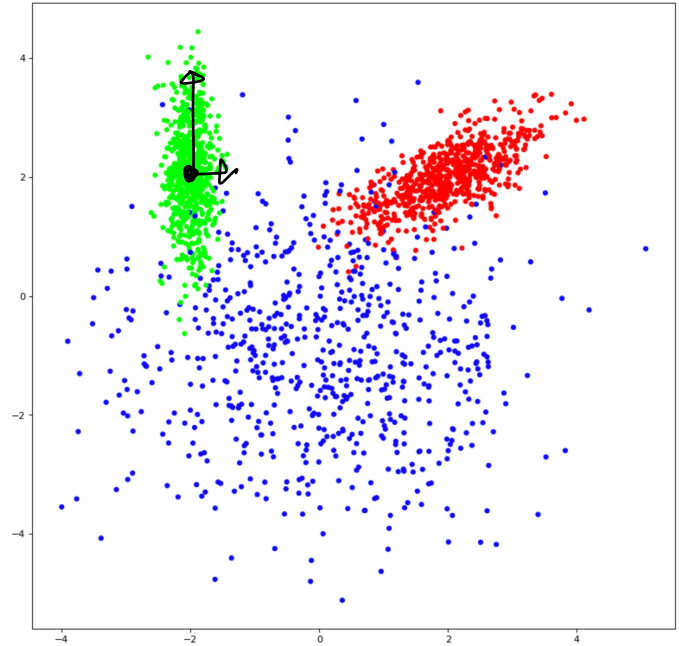


- **top:** center of a cluster μ_k and the diagonal entries of the covariance matrix Σ_k
- note that we have trained 10-dimensional covariance matrix $\Sigma_k = AA^T$, with $A \in \mathbb{R}^{288 \times 10}$, and let $A^{(j)}$ be the j -th column
- **bottom:** each row corresponds to different j , and we show $\mu_k + A^{(j)}$, $0.5 + A^{(j)}$, $\mu_k - A^{(j)}$



Principal Component direction 2

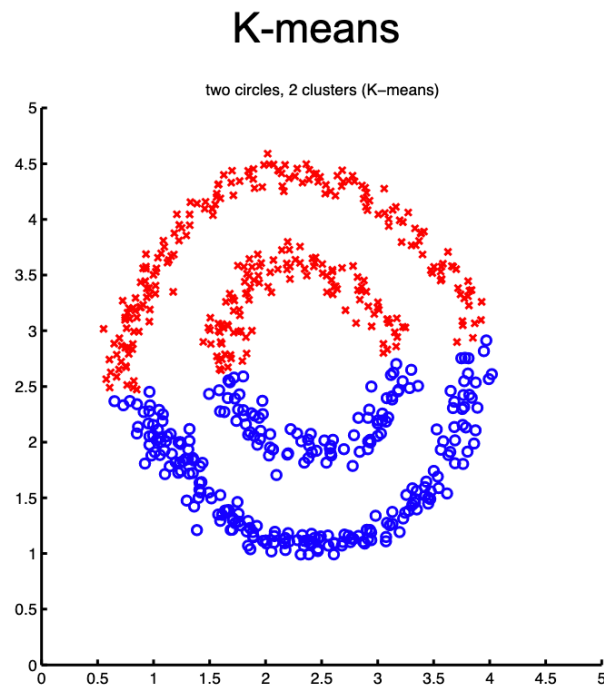
Principal Component direction 1



- middel: μ_k
- Each row: middel + $c \times A^{(1)}$
- Each column: middel + $c \times A^{(2)}$

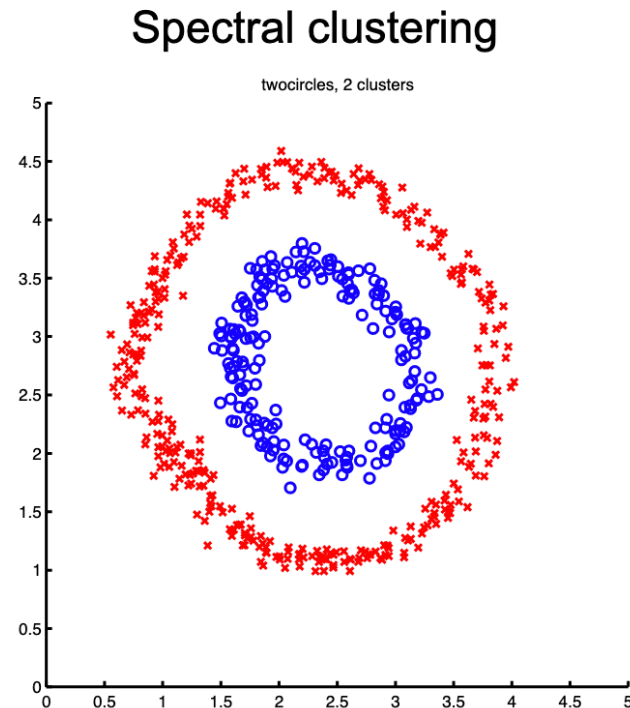
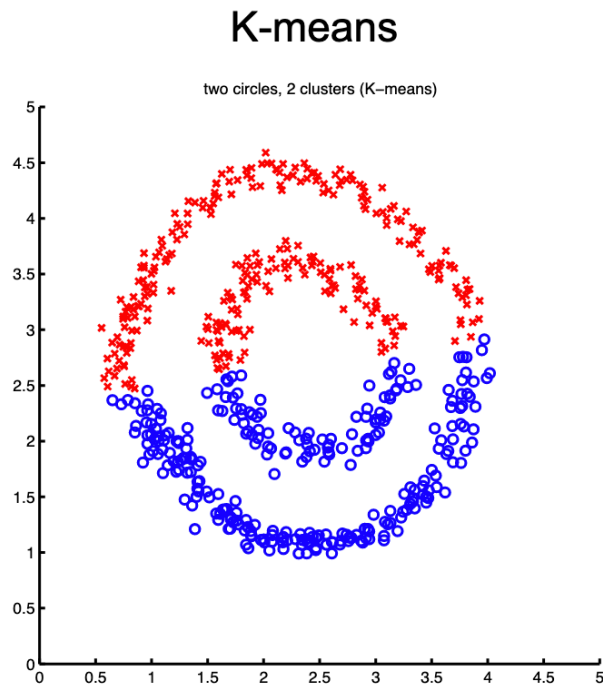
k -means and GMMs are inherently linear

- It tries to find linear boundaries between centers
- It fails completely on non-linearly clustered datasets such as



Spectral clustering

- Main idea:
 - Transform the dataset into a graph
 - Use eigenvalues (also called spectrum) and vectors of a graph to cluster



Step 1. From dataset to a graph

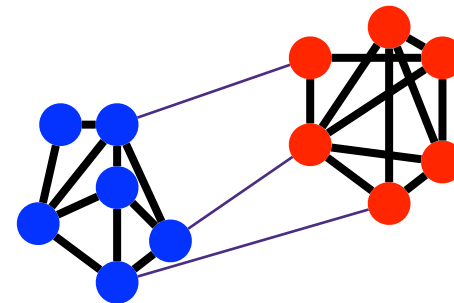
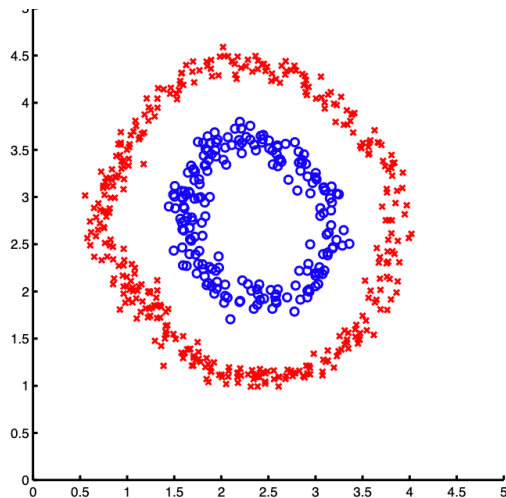
- Given $\mathcal{D} = \{x_i \in \mathbb{R}^d\}_{i=1}^n$, create a graph with n nodes and weighted edges $\{w_{ij}\}$, where each node represents each sample and each edge measures the similarity between the two nodes

- Example 1: Gaussian kernel

$$w_{ij} = e^{-\frac{\|x_i - x_j\|_2^2}{\sigma^2}}$$

- Example 2: k -nearest neighbor graph

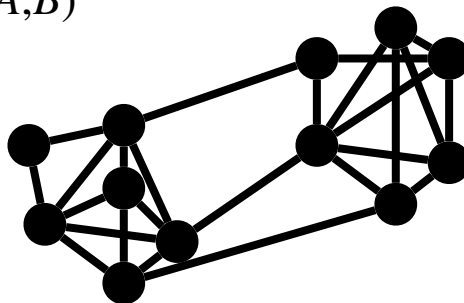
$$w_{ij} = 1 \text{ if } j \text{ is one of } k\text{-nearest neighbors of } i \text{ or } i \text{ is one of } k\text{-nearest neighbors of } j$$



Step 2. Graph partitioning

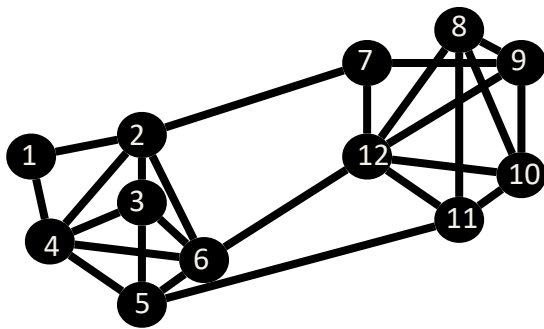
- Once we have a similarity graph, how do we partition it?
- Can we use **minimum cut** for a graph $G(V, E)$?
 - Set of nodes $V = \{1, \dots, n\}$
 - Set of edges $E = \{(i, j)\}$
 - If it is a weighted graph we have weights $\{w_{ij}\}_{(i,j) \in E}$
- **Minimum cut** of a graph is a partition $A \cup B = V$ and $A \cap B = \emptyset$ such that

$$\arg \min_{A, B} \underbrace{\sum_{i \in A} \sum_{j \in B} w_{i,j}}_{\text{cut}(A, B)}$$



Step 2. Graph partition using Graph Laplacian

- Graph Laplacian $L_G = D - A$ can capture some structure of the graph
- Consider placing each node in 1-dim line at positions $x = [x_1, x_2, \dots, x_2]$

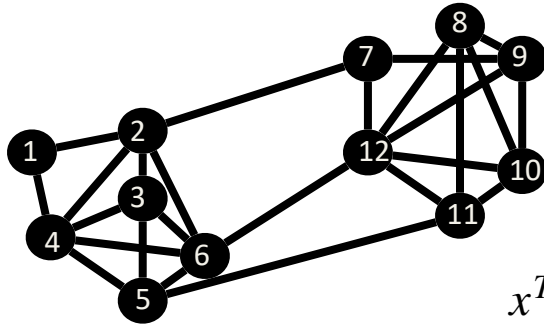


quadratic form of L_G is useful in capturing the structure of the graph:

$$\begin{aligned} \mathbf{x}^T L_G \mathbf{x} &= \sum_i d_i x_i^2 - \sum_{(i,j) \in E} 2x_i x_j \\ &= \sum_i \sum_{j:(i,j) \in E} x_i^2 - \sum_{(i,j) \in E} 2x_i x_j \\ &= \sum_{(i,j) \in E} 2x_i^2 - 2x_i x_j \\ &= \sum_{(i,j) \in E} x_i^2 + x_j^2 - 2x_i x_j \\ &= \sum_{(i,j) \in E} (x_i - x_j)^2 \end{aligned}$$

Step 2. Graph partition using Graph Laplacian

- Graph Laplacian $L_G = D - A$ can capture some structure of the graph
- Consider placing each node in 1-dim line at positions $x = [x_1, x_2, \dots, x_n]$



$$x^T L_G x = \sum_{(i,j) \in E} (x_i - x_j)^2$$

- If we want a good graph partition, we want to place nodes such that the distance between connected nodes are smaller
- This naturally leads to the following problem:

$$\arg \min_{x \in \mathbb{R}^n} x^T L_G x = \sum_{(i,j) \in E} (x_i - x_j)^2$$

- There is a trivial solution to this problem: $x_i = 1$ for all i , which achieves the minimum value of zero, so we change it to

$$\arg \min_{x \in \mathbb{R}^n} x^T L_G x = \sum_{(i,j) \in E} (x_i - x_j)^2 \quad \text{subject to } x^T \mathbf{1} = 0$$

Step 2. Graph partition using Graph Laplacian

- To solve graph partitioning, we solve

$$\arg \min_{x \in \mathbb{R}^n} x^T L_G x = \sum_{(i,j) \in E} (x_i - x_j)^2$$

$$\text{subject to } x^T \mathbf{1} = 0$$

$$\|x\|_2 = 1$$

and place nodes as per x , and find a partition using simple algorithms like k -means

- It turns out that the above optimization has a efficient solver, because The optimal x turns out to be the second smallest eigen vector of the graph Laplacian L_G
- Since, eigen values of a matrix is also called a spectrum, this is called a spectral clustering algorithm

Spectral clustering

- Step 1. Define a similarity graph $G(V, E, W)$
- Step 2. Compute the Graph Laplacian

$$L_G = D - W$$

where D is a diagonal matrix with $D_{ii} = \sum_{j=1}^n w_{ij}$

- let x be the Eigen vector corresponding to the second smallest Eigen value
 - Place samples according to x and apply k -means clustering
-
- instead of using just the second smallest Eigen pair, you can use multiple smallest Eigen pairs