CS 760: Machine Learning
Unsupervised Learning I

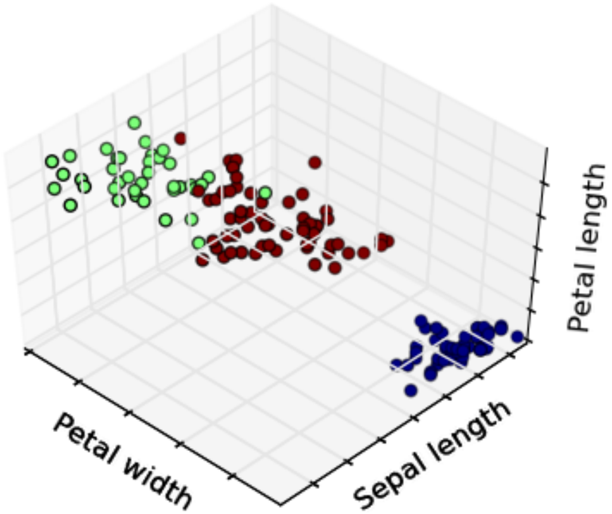
Kirthi Kandasamy

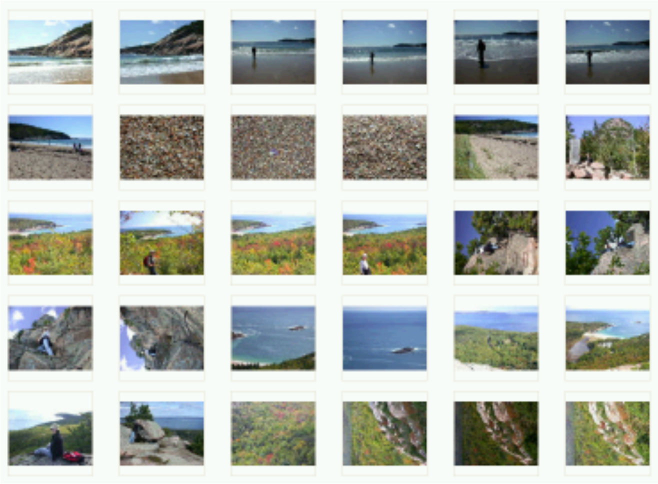
University of Wisconsin-Madison

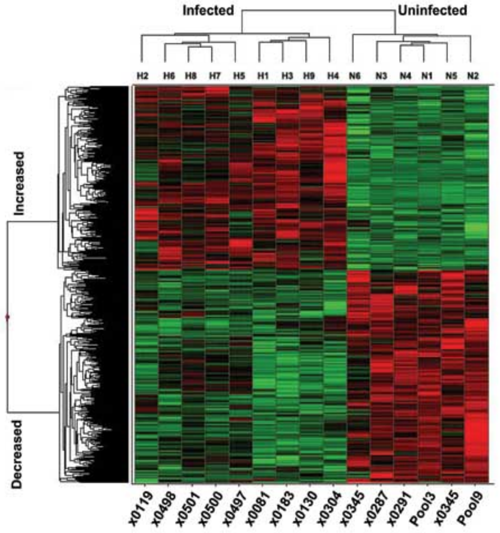
March 8, 2023

Unsupervised Learning

- Goal: find patterns & structures that help better understand data.
- No labels; generally won’t be making predictions
- Sometimes model a distribution, but not always







Mulvey and Gingold

Outline

- **K-means clustering**
- **Gaussian Mixture Models**
  - Mixtures, Expectation-Maximization algorithm
- **Advanced clustering methods**
  - hierarchical, spectral clustering

Outline

- **K-means clustering**

- **Gaussian Mixture Models**
  - Mixtures, Expectation-Maximization algorithm

- **Advanced clustering methods**
  - hierarchical, spectral clustering

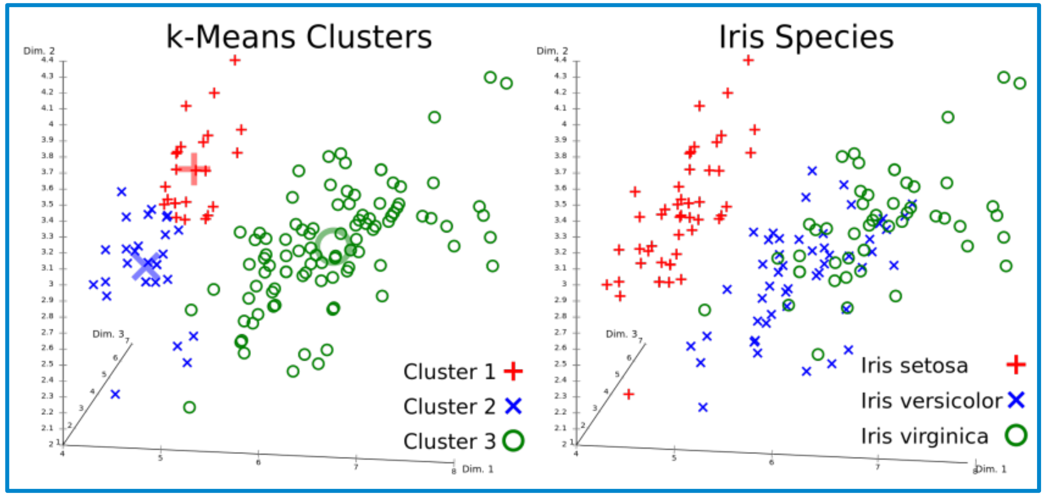
Clustering

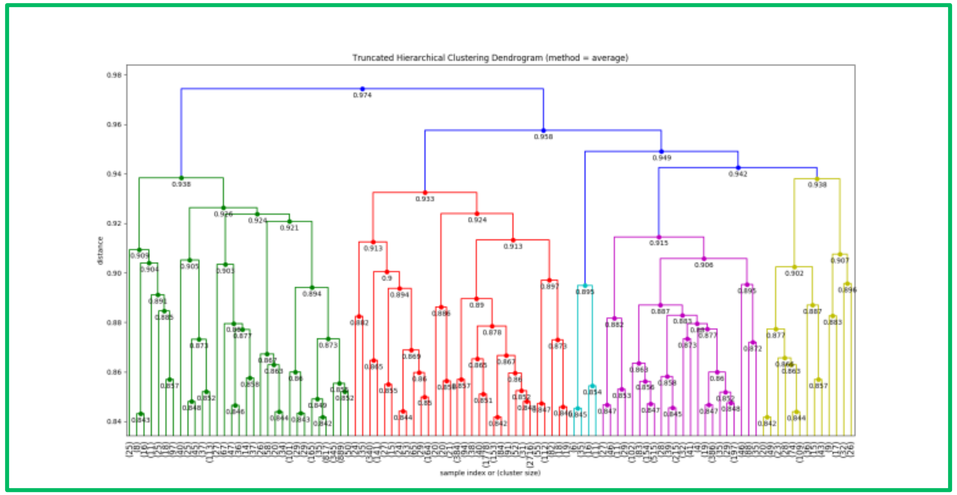
Several types:

Partitional
- Centroid
- Graph-theoretic
- Spectral

Hierarchical
- Agglomerative
- Divisive

Bayesian
- Decision-based
- Nonparametric



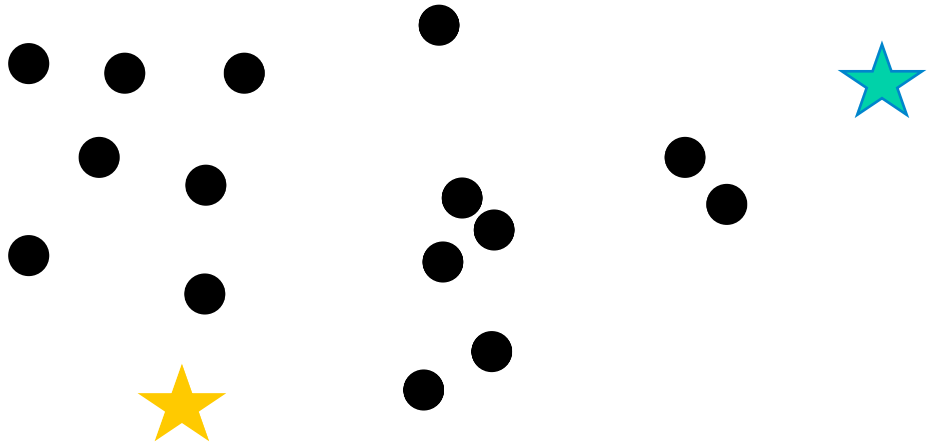


K-Means Clustering

k-means is a type of partitional **centroid-based clustering**

Algorithm:

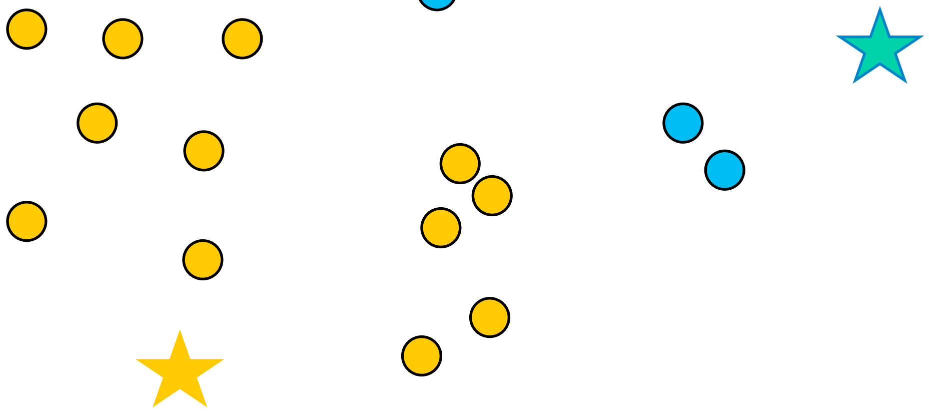
1. Randomly pick k cluster centers



K-Means Clustering: Algorithm

K-Means clustering

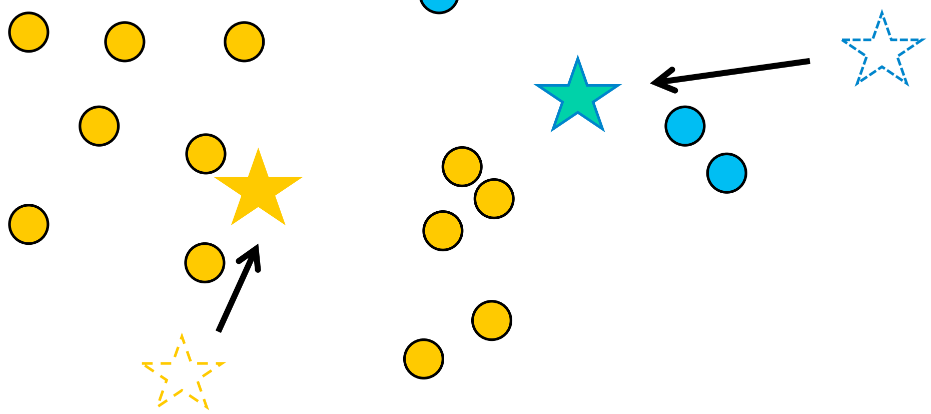
2. Find closest center for each point



K-Means Clustering: Algorithm

K-Means clustering

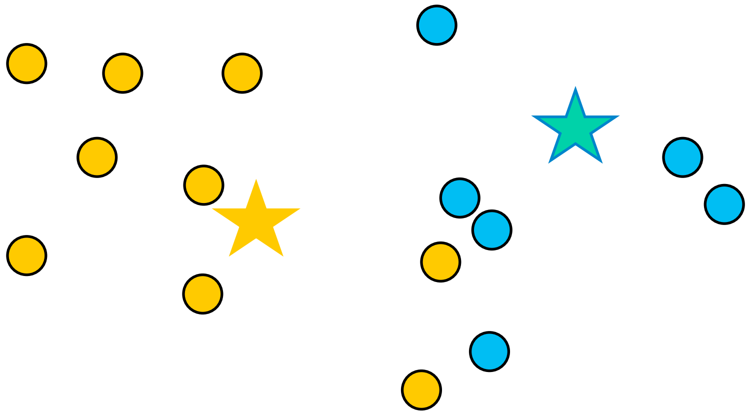
3. Update cluster centers by computing centroids



K-Means Clustering: Algorithm

K-Means clustering

Repeat Steps 2 & 3 until convergence



K-means clustering (Lloyd's) algorithm

- Input: $x_1 \ldots x_n$, k
- **Step 1**: select k cluster centers $c_1 \ldots c_k$
- **Step 2**: for each point x, determine its cluster assignment: find the closest center in Euclidean distance

$$y(x) = argmin_{i=1:k} ||x - c_i||$$

- **Step 3**: update all cluster centers as the centroids

$$c_i = \frac{\sum_{x:y(x)=i} x}{\sum_{x:y(x)=i} 1}$$

- Repeat step 2, 3 until cluster centers no longer change

Questions on k-means

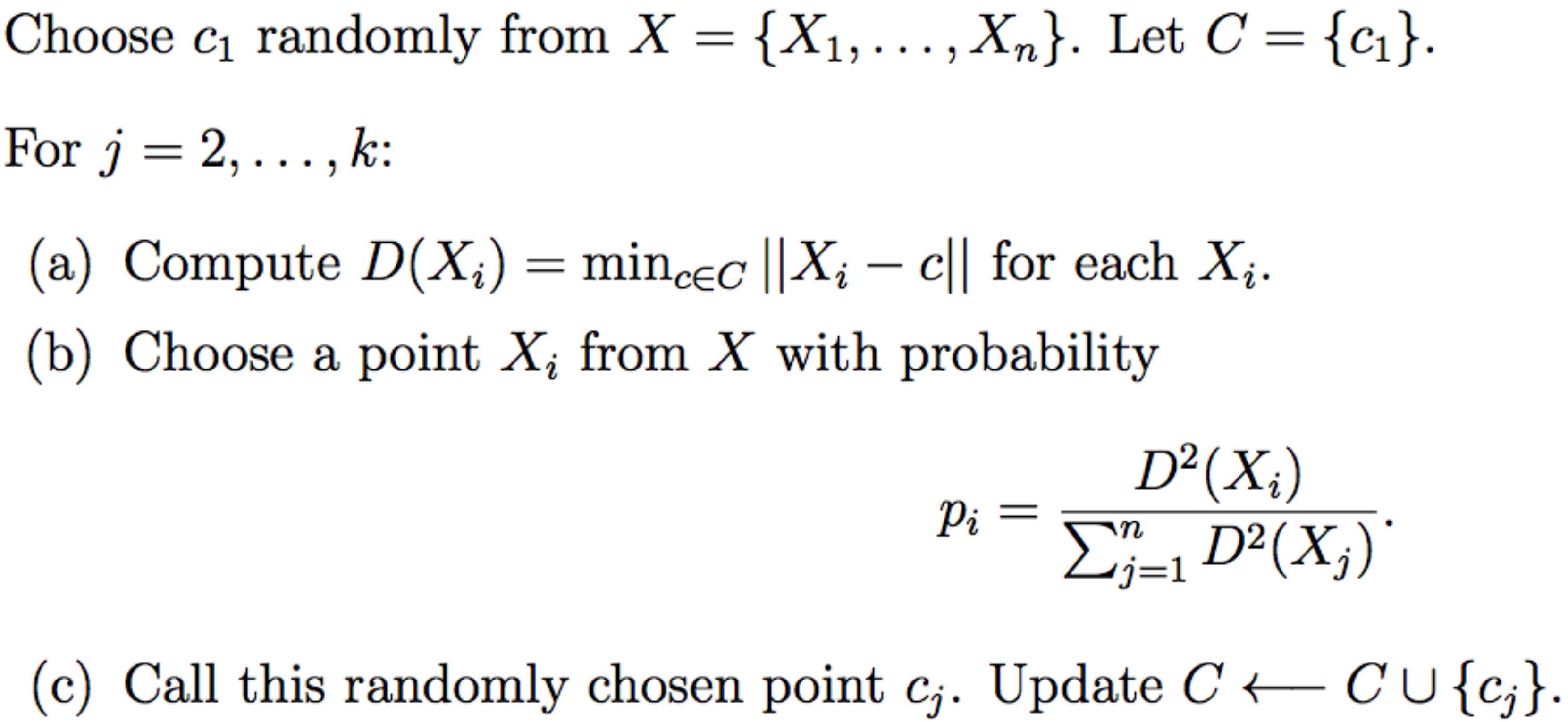
- What is k-means trying to optimize?

$$J(\{y^{(i)}\}_i, \{c_j\}_j) = \sum_{i=1}^{n} \|x^{(i)} - c_{y^{(i)}}\|^2$$

- Will k-means stop (converge)?
- Will it find a global or local optimum?
- How to pick starting cluster centers?
- How many clusters should we use?

How to pick starting cluster centers?

- Randomly choosing starting centers can lead to poor performance.
- A smarter strategy: k-means ++ (Arthur & Vassilivitski '07)



Choose c_1 randomly from $X = \{X_1, \ldots, X_n\}$. Let $C = \{c_1\}$.

For $j = 2, \ldots, k$:

(a) Compute $D(X_i) = \min_{c \in C} ||X_i - c||$ for each X_i.

(b) Choose a point X_i from X with probability

$$p_i = \frac{D^2(X_i)}{\sum_{j=1}^n D^2(X_j)}.$$

(c) Call this randomly chosen point c_j. Update $C \longleftarrow C \cup \{c_j\}$.

Outline

- **K-means clustering**

- **Gaussian Mixture Models**
  - Mixtures, Expectation-Maximization algorithm

- **Advanced clustering methods**
  - hierarchical, spectral clustering

Mixture Models

- Let us get back to modeling densities in unsupervised learning.
- Have dataset: $\{(x^{(1)}, x^{(2)}, \ldots, x^{(n)}\}$
- One type of model: **mixtures**
  - A function of the **latent variable** z
  - Model:

$$p(x^{(i)} | z^{(i)}) p(z^{(i)})$$

Mixture Models: Gaussians

- Many different types of mixtures, but let us focus on Gaussians.
- What does this mean?
- Latent variable z has some multinomial distribution, $\sum_{i=1}^{k} \phi_i = 1$

$$z^{(i)} \sim \text{Multinomial}(\phi)$$

- Then, let us make x be Gaussian conditioned on z

$$x^{(i)}|(z^{(i)} = j) \sim \mathcal{N}(\underset{\text{Mean}}{\underset{\uparrow}{\mu_j}}, \underset{\text{Covariance Matrix}}{\underset{\uparrow}{\Sigma_j}})$$

Gaussian Mixture Models: Likelihood

- How should we learn the parameters? ϕ, μ_j, Σ_j
- Could try our usual way: maximum likelihood
  - Log likelihood:

$$\ell(\phi, \mu, \Sigma) = \sum_{i=1}^{n} \log \sum_{z^{(i)}=1}^{k} p(x^{(i)} | z^{(i)}; \mu, \Sigma) p(z^{(i)}; \phi)$$

  - Turns out to be **hard** to solve... inner sum leads to problems!

GMMs: Supervised Setting

- What if we knew the z's?
  - "Supervised" setting…
- First, empirically estimate the multinomial parameters:

$$\phi_j = \frac{1}{n}\sum_{i=1}^{n} 1\{z^{(i)} = j\}$$

- Next the Gaussian components:

$$\mu_j = \frac{\sum_{i=1}^{n} 1\{z^{(i)} = j\}x^{(i)}}{\sum_{i=1}^{n} 1\{z^{(i)} = j\}}$$

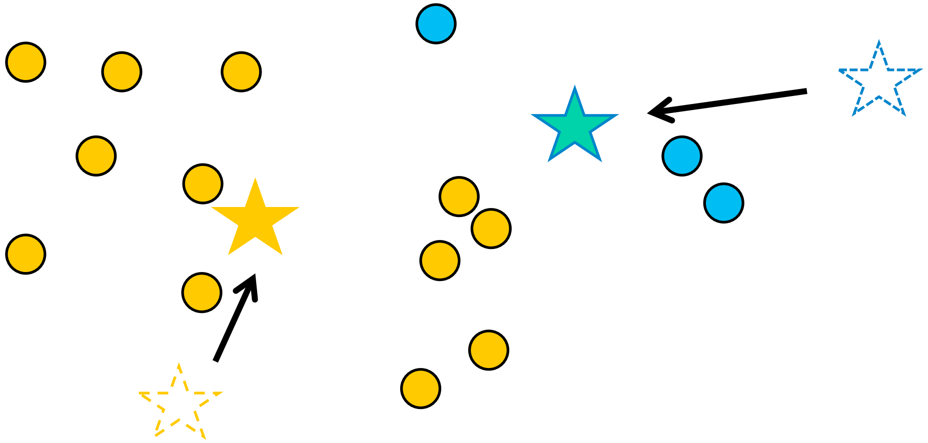
Average of x's where z = j

$$\Sigma_j = \frac{\sum_{i=1}^{n} 1\{z_j^{(i)} = j\}(x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T}{\sum_{i=1}^{n} 1\{z_j^{(i)} = j\}}$$

GMMs: Back to Latent Setting

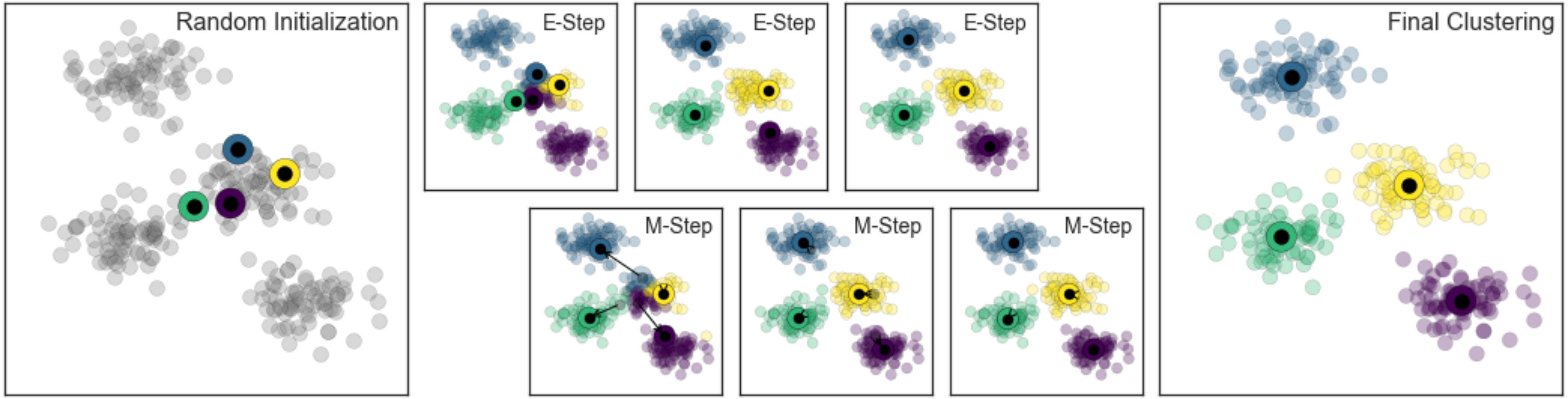
- But, we don't get to see the z's!

- What could we do instead?
- Recall our **k-means** approach: we don't know the centers, but we pretend we do, perform a clustering, re-center, iterate



GMMs: Expectation Maximization

- EM :an algorithm for dealing with latent variable problems
- Iterative, alternating between two steps:
  - **E**-step: estimate latent variable (probabilities) based on current model
  - **M**-step: update the parameters of $p(x|z)$
  - Note similarity to k-means clustering.



Jake VanderPlas

GMM EM: E-Step

- Let us write down the formulas.
- **E-step**: fix parameters, compute posterior:

$$w_j^{(i)} = p(z^{(i)} = j | x^{(i)}; \phi, \mu, \Sigma)$$

- These w's are "soft" assignments of the z terms… probabilities over the values z could take. Concretely:

$$w_j^{(i)} = p(z^{(i)} = j | x^{(i)}; \phi, \mu, \Sigma) = \frac{p(x^{(i)} | z^{(i)} = j; \mu, \Sigma) p(z^{(i)} = j; \phi)}{\sum_{\ell=1}^{k} p(x^{(i)} | z^{(i)} = \ell; \mu, \Sigma) p(z^{(i)} = \ell; \phi)}$$

GMM EM: M-Step

- Let’s write down the formulas.
- **M-step**: fix w, update parameters:

$$\phi_j = \frac{1}{n}\sum_{i=1}^{n} w_j^{(i)}$$

Soft version of our counting estimator for the supervised case.

$$\mu_j = \frac{\sum_{i=1}^{n} w_j^{(i)} x^{(i)}}{\sum_{i=1}^{n} w_j^{(i)}}$$

Soft version of our empirical mean and covariances.

$$\Sigma_j = \frac{\sum_{i=1}^{n} w_j^{(i)} (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T}{\sum_{i=1}^{n} w_j^{(i)}}$$

EM through the lens of maximum likelihood estimation

- Why is EM a sensible idea?
- Let us write out the log likelihood for our problem

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} \log p_\theta(x^{(i)}) = \sum_{i=1}^{n} \log \left(\sum_{j=1}^{k} p_\theta(x^{(i)}, z^{(i)} = j) \right)$$

- Letting $Q^{(i)} = [Q_1^{(i)}, \ldots, Q_k^{(i)}]$ be any distribution over $z^{(i)}$

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} \log \left(\sum_{j=1}^{k} Q_j^{(i)} \frac{p_\theta(x^{(i)}, z^{(i)} = j)}{Q_j^{(i)}} \right)$$

EM through the lens of maximum likelihood estimation

- Letting $Q^{(i)} = [Q_1^{(i)}, \ldots, Q_k^{(i)}]$ be any distribution over $z^{(i)}$

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} \log \left(\sum_{j=1}^{k} Q_j^{(i)} \frac{p_\theta(x^{(i)}, z^{(i)} = j)}{Q_j^{(i)}} \right)$$

- By an application of Jensen’s inequality:

$$\mathcal{L}(\theta) \geq \sum_{i=1}^{n} \sum_{j=1}^{k} Q_j^{(i)} \log \left(\frac{p_\theta(x^{(i)}, z^{(i)} = j)}{Q_j^{(i)}} \right)$$

EM through the lens of maximum likelihood estimation

- We have a lower bound on the log likelihood:

$$\mathcal{L}(\theta) \geq \sum_{i=1}^{n} \sum_{j=1}^{k} Q_j^{(i)} \log \left(\frac{p_\theta(x^{(i)}, z^{(i)} = j)}{Q_j^{(i)}} \right)$$

- If this lower bound is ***tight***, by maximizing the lower bound, we can hope to do well in maximizing the likelihood.
- A good choice is $Q_j^{(i)} = p_\theta(z^{(i)} = j | x^{(i)})$

General EM Algorithm

On round t of EM:

- E-Step (Expectation): Update $Q_j^{(i)}$ for all i and j (This effectively computes the lower bound)

$$Q_j^{(i)} \leftarrow p_{\theta_t}(z^{(i)} = j | x^{(i)})$$

- M-step: Maximize lower bound with respect to parameters θ_t

$$\theta_{t+1} \leftarrow \arg\max_{\theta} \sum_{i=1}^{n} \sum_{j=1}^{k} Q_j^{(i)} \log \left(\frac{p_\theta(x^{(i)}, z^{(i)} = j)}{Q_j^{(i)}} \right)$$

Do at home: Show that this corresponds to the GMM update equations

More on EM

- Why $Q_j^{(i)} = p_\theta(z^{(i)} = j|x^{(i)})$ in the E-step?
    - Guarantees that the log likelihood increases each iteration. (See board)

- EM works on continuous latent variables as well!
    - (HW5)

Quiz: State if the following sentences are true or false.

A. In a Gaussian mixture model, the log likelihood is concave.

B. We can maximize the likelihood of a mixture model using gradient descent.

C. EM is always guaranteed to find a global maximum

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} \log p_\theta(x^{(i)}) = \sum_{i=1}^{n} \log \left(\sum_{j=1}^{k} p_\theta(x^{(i)}, z^{(i)} = j) \right)$$

Ans: A: false, B: true, C: false

We use EM over GD because it is more efficient than GD.

Quiz: Which of the following sentences are true.

A. GMMs are generative models

B. When you learn a GMM, you are estimating the density of the data.

C. GMMs can be used for clustering.

Ans: All are true

Outline

- **K-means clustering**

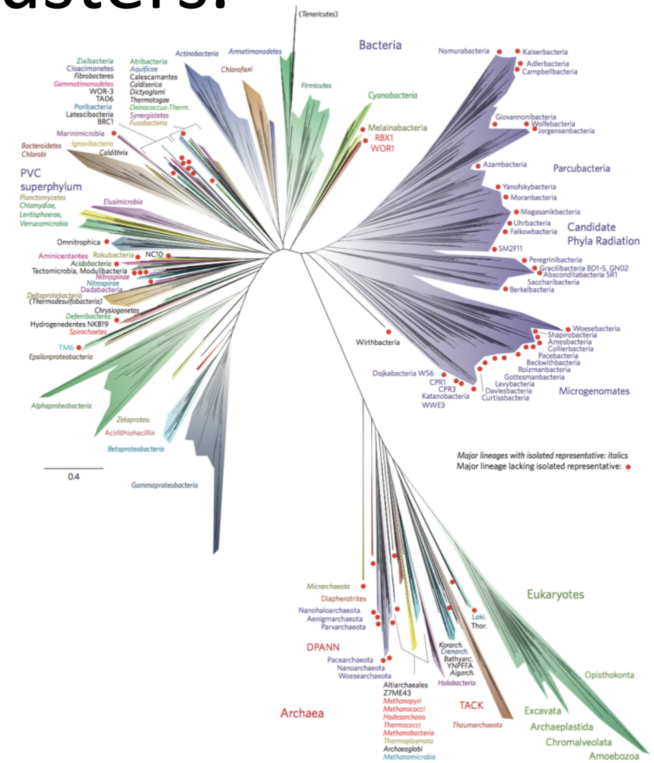
- **Gaussian Mixture Models**
  - Mixtures, Expectation-Maximization algorithm

- **Advanced clustering methods**
  - hierarchical, spectral clustering

Hierarchical Clustering

Basic idea: build a “hierarchy”

- Want: arrangements from specific to general
- One advantage: no need for k, number of clusters.
- **Input**: points.
- **Output**: a hierarchy (a binary tree)



Credit: Wikipedia

HC: Agglomerative vs Divisive

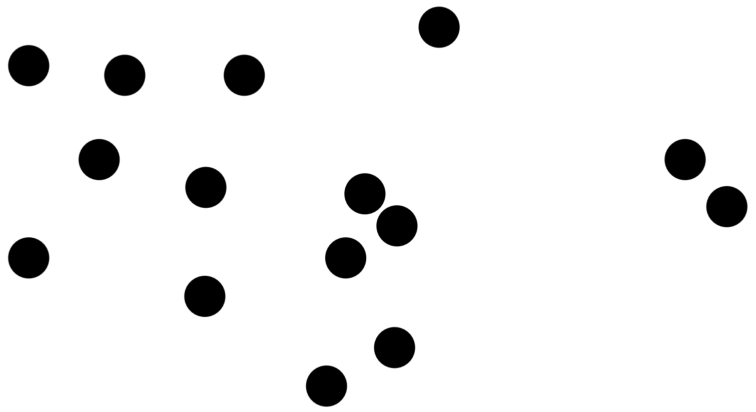
Two ways to go:

- **Agglomerative**: bottom up.
  - Start: each point a cluster.
  - Progressively merge clusters

- **Divisive**: top down
  - Start: all points in one cluster.
  - Progressively split clusters

HC: Agglomerative Clustering Example

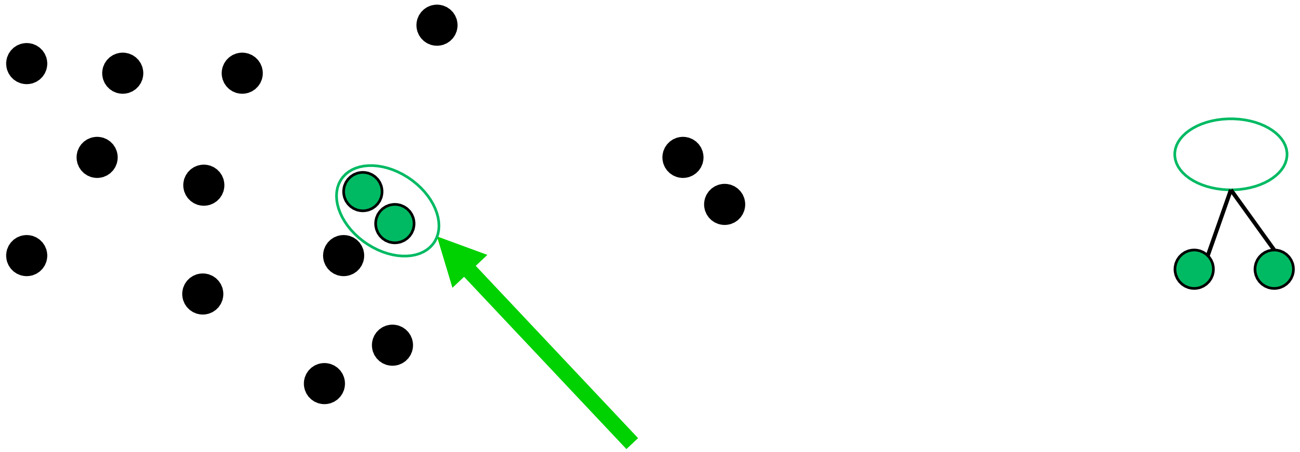
Agglomerative: Start: every point is its own cluster



HC: Agglomerative Clustering Example

Basic idea: build a “hierarchy”

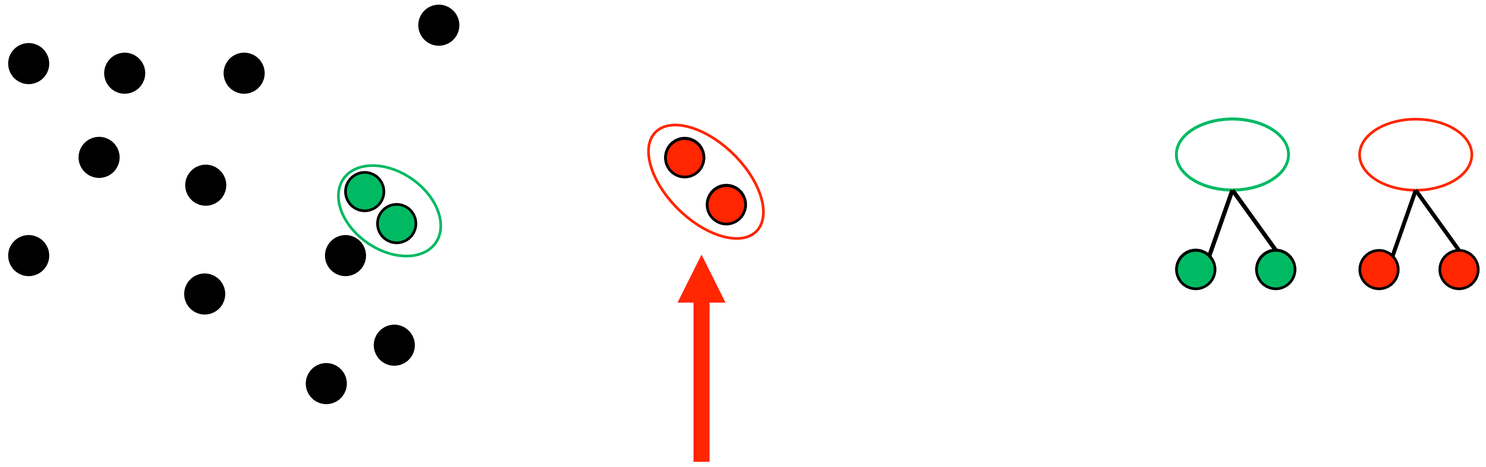
- Get pair of clusters that are closest and merge



HC: Agglomerative Clustering Example

Basic idea: build a "hierarchy"

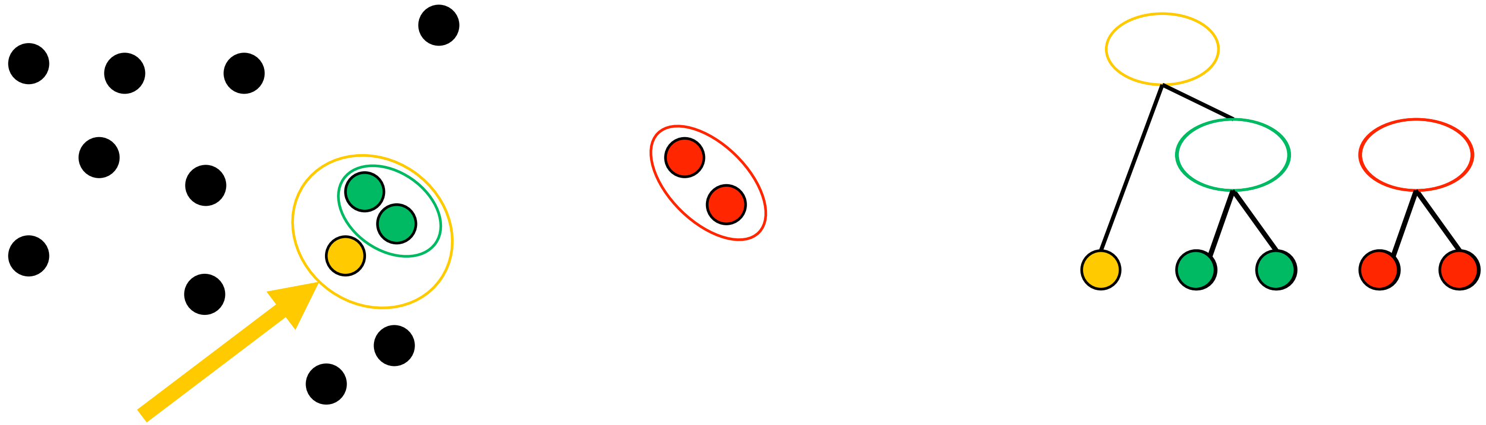
- **Repeat:** Get pair of clusters that are closest and merge



HC: Agglomerative Clustering Example

Basic idea: build a "hierarchy"

- **Repeat:** Get pair of clusters that are closest and merge



HC: Merging Criteria

Merge: use closest clusters. Define closest?

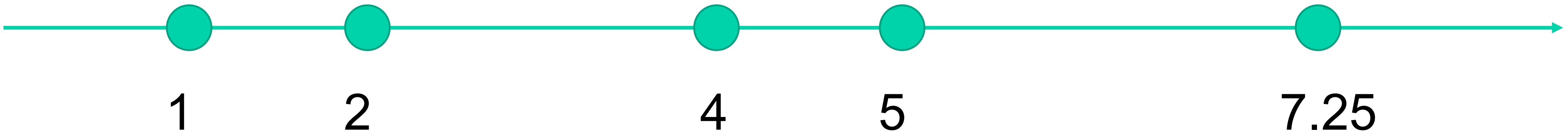
First define a distance between points $d(x_1, x_2)$. Then, define distance between clusters.

- Single-linkage $$d(A,B) = \min_{x_1 \in A, x_2 \in B} d(x_1, x_2)$$
- Complete-linkage $$d(A,B) = \max_{x_1 \in A, x_2 \in B} d(x_1, x_2)$$
- Average-linkage $$d(A,B) = \frac{1}{|A||B|} \sum_{x_1 \in A, x_2 \in B} d(x_1, x_2)$$

Single-linkage Example

We'll merge using single-linkage

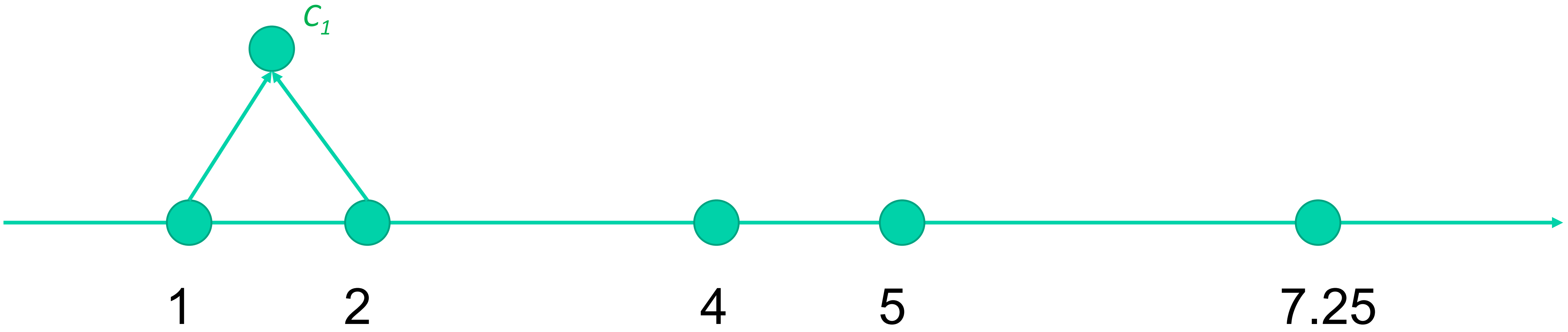
- 1-dimensional vectors.
- Initial: all points are clusters



Single-linkage Example

$$d(C_1, \{4\}) = d(2, 4) = 2$$

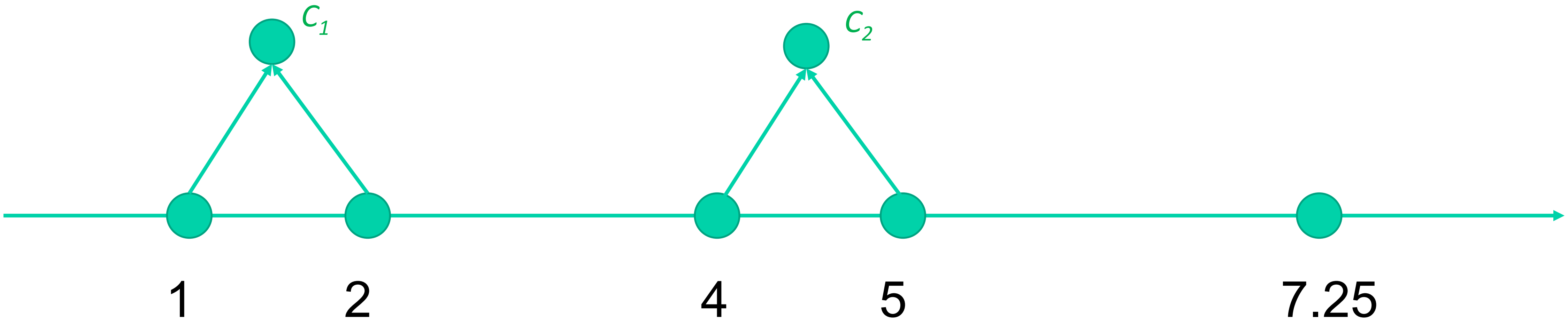
$$d(\{4\}, \{5\}) = d(4, 5) = 1$$



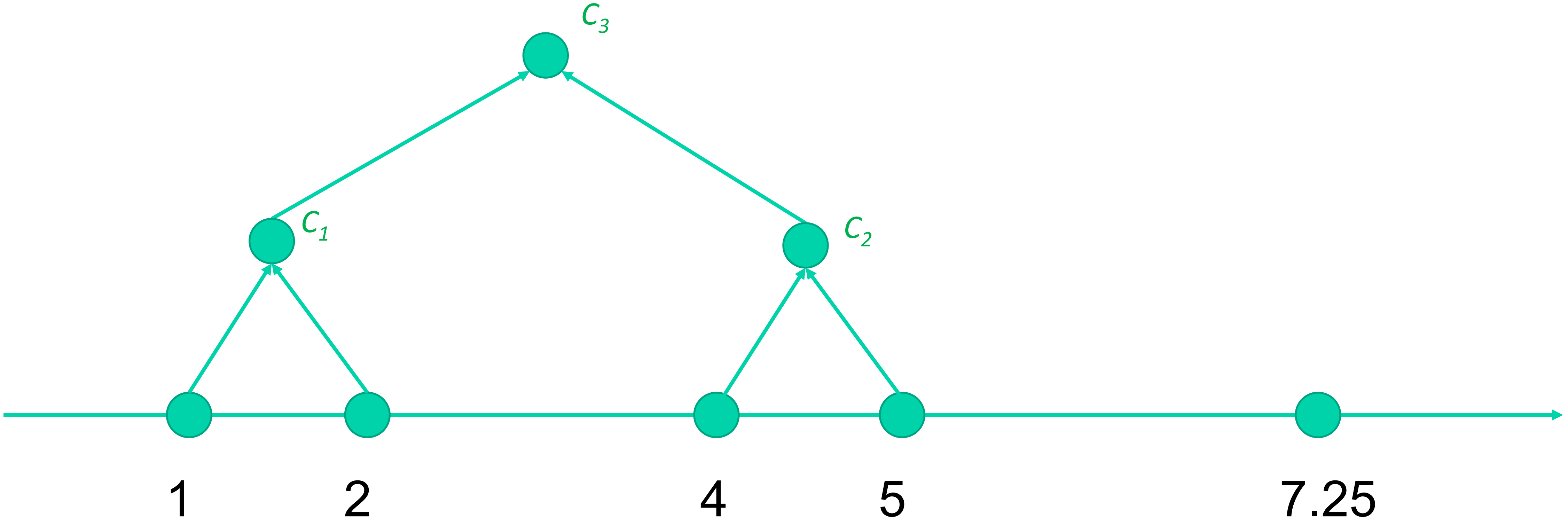
Single-linkage Example

$$d(C_1, C_2) = d(2, 4) = 2$$

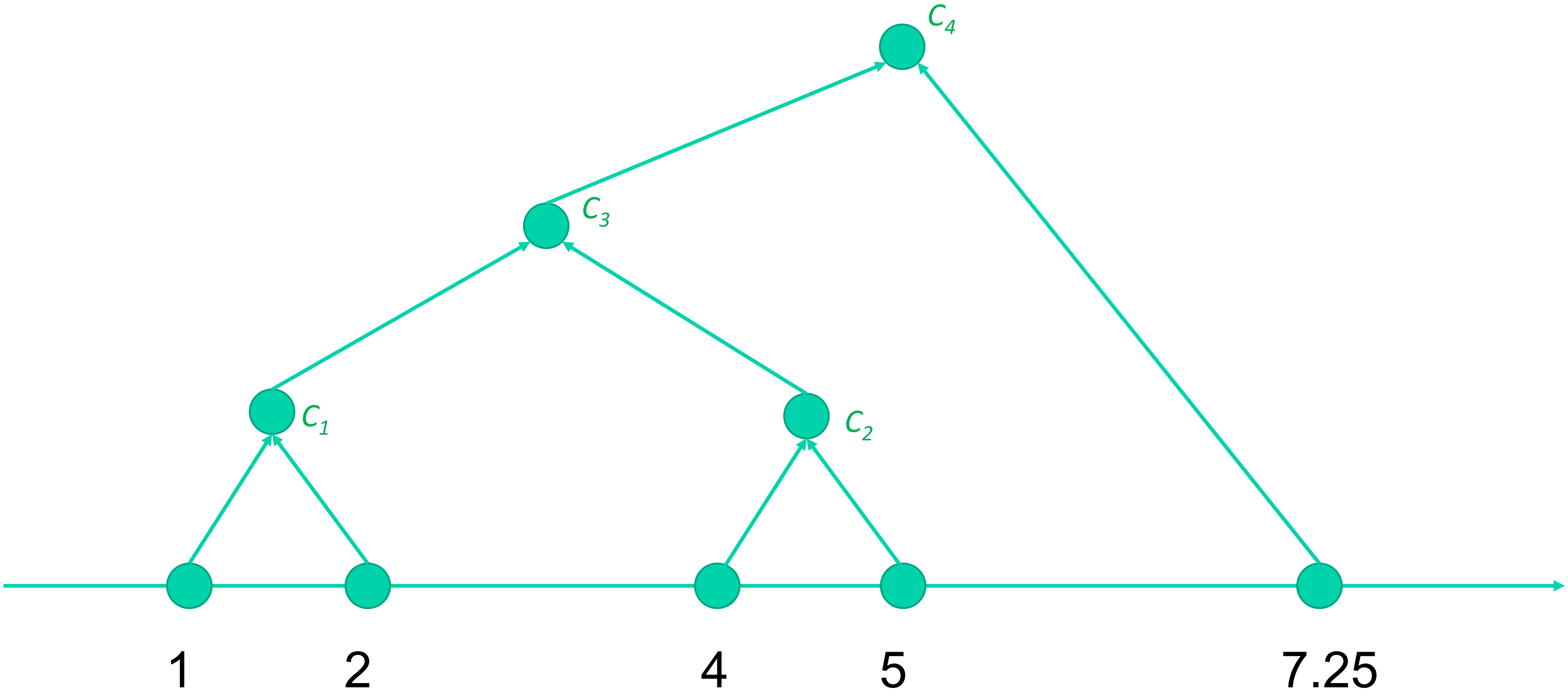
$$d(C_2, \{7.25\}) = d(5, 7.25) = 2.25$$



Single-linkage Example



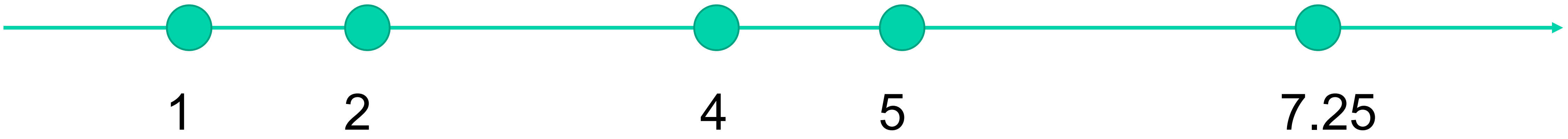
Single-linkage Example



Complete-linkage Example

We’ll merge using complete-linkage

- 1-dimensional vectors.
- Initial: all points are clusters

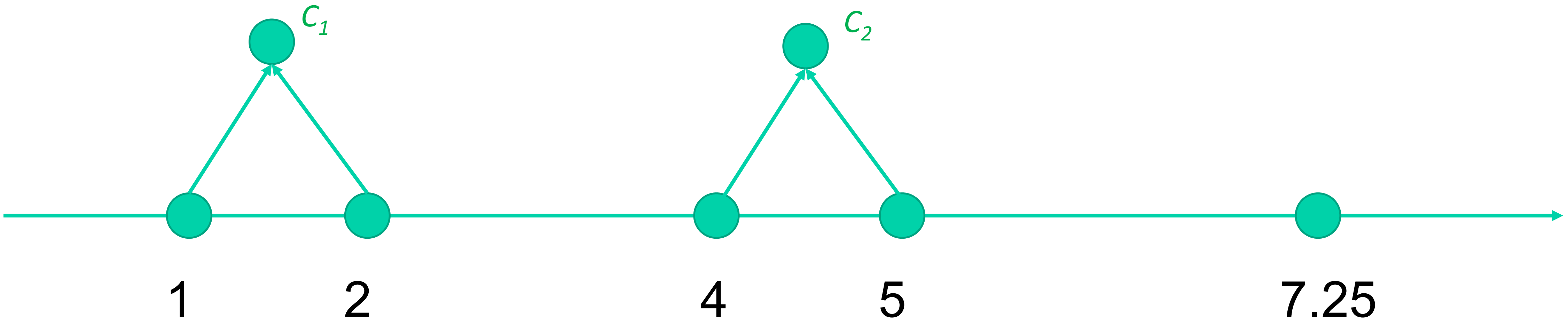


Complete-linkage Example

Beginning is the same…

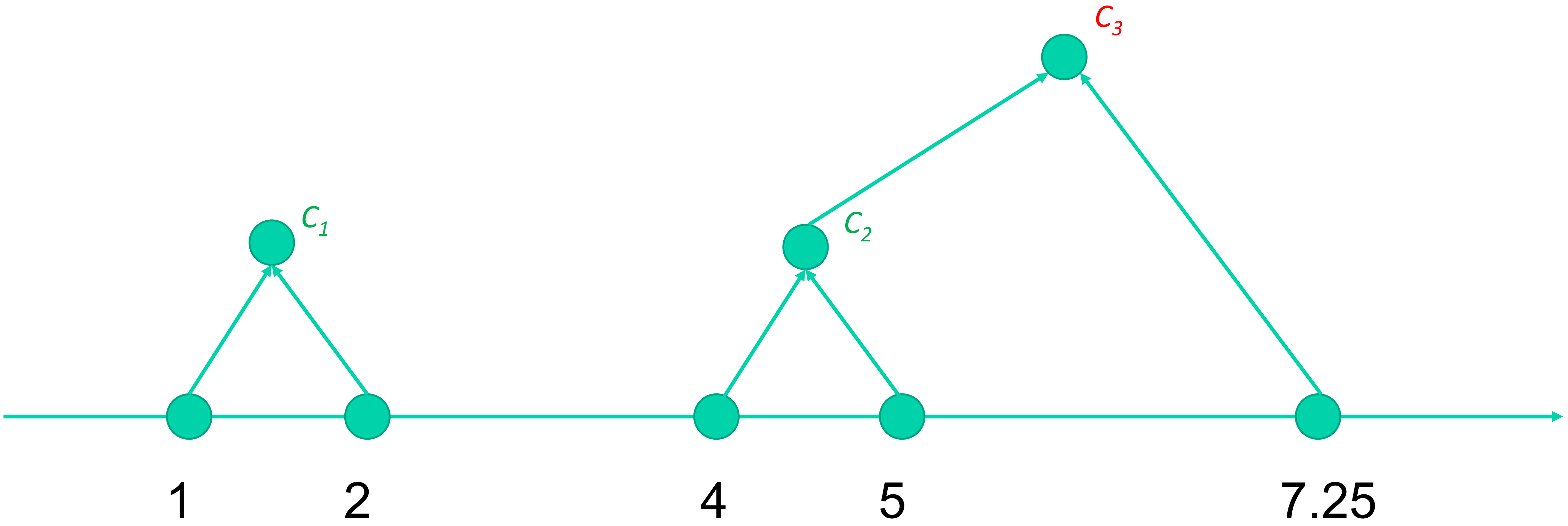
$$d(C_1, C_2) = d(1, 5) = 4$$

$$d(C_2, \{7.25\}) = d(4, 7.25) = 3.25$$

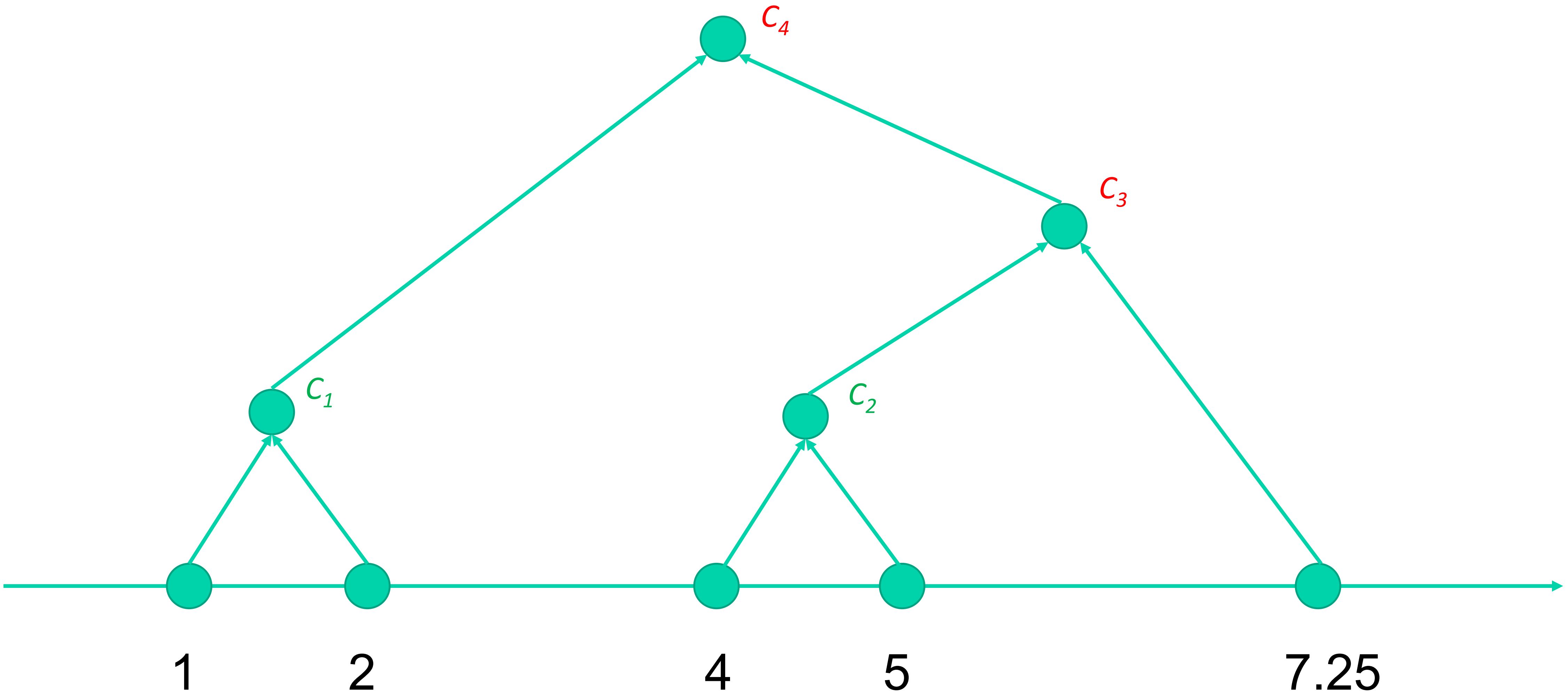


Complete-linkage Example

Now different from single linkage:



Complete-linkage Example



Break & Quiz

Break & Quiz

Q 2.2: If we do hierarchical clustering on n points, the maximum depth of the resulting tree is

- A. 2
- B. $\log_2 n$
- C. $n/2$
- D. n-1

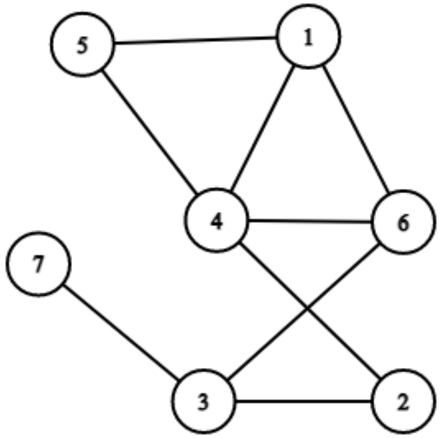
Break & Quiz

Q 2.2: If we do hierarchical clustering on n points, the maximum depth of the resulting tree is

- A. 2
- B. $\log_2 n$
- C. $n/2$
- **D. *n*-1**

Graph/proximity based clustering

- Recall: Graph G = (V,E) has vertex set V, edge set E.
  - Edges can be weighted or unweighted
  - Encode **similarity**



- Treat each data point as a node in a graph.
- Edges based on similarity of data points
- E.g. for Euclidean vectors!

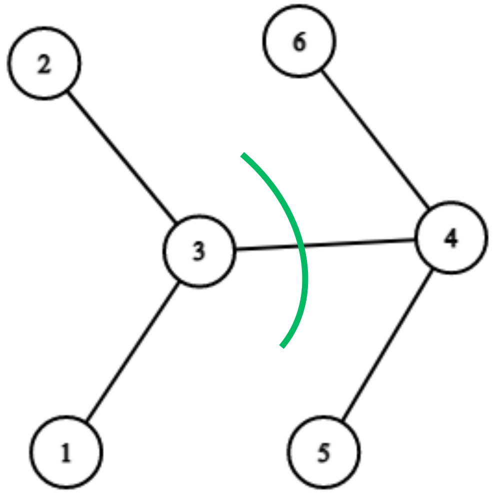
$$w_{ij} = e^{-\alpha\|x_i - x_j\|^2}$$

- But they don't need to be in Euclidean space!

Graph-Based Clustering

Want: partition V into k groups

- Implies a graph "cut"
- One idea: minimize the **weight** of the cut



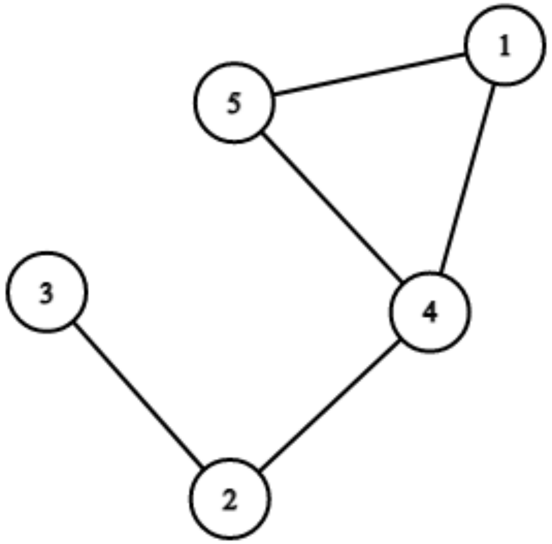
$$W(A,B) = \sum_{i \in A, j \in B} w_{ij}$$

$$\mathrm{cut}(A_1, \ldots, A_k) := \frac{1}{2} \sum_{i=1}^{k} W(A_i, \overline{A}_i).$$

Partition-Based Clustering

How do we compute these?

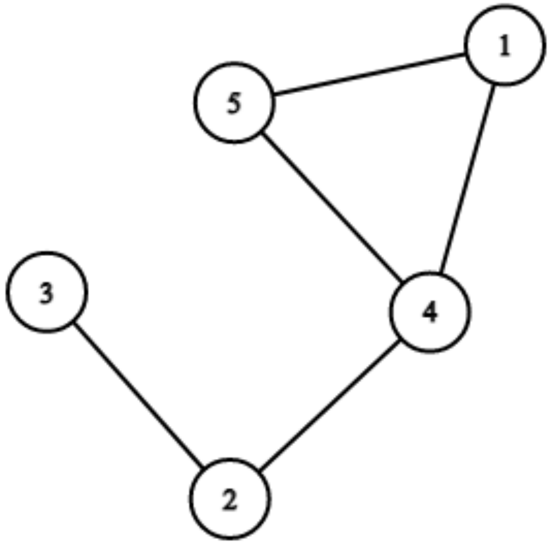
- Hard problem → heuristics
  - Greedy algorithm
  - "Spectral" approaches
- Spectral clustering approach:
  - **Adjacency** matrix



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

Partition-Based Clustering

- Spectral clustering approach:
  - **Adjacency** matrix
  - **Degree** matrix

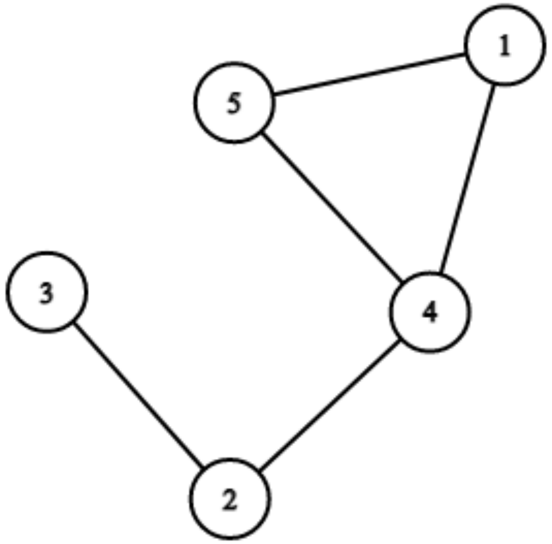


$$D = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix} \quad A = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Spectral Clustering

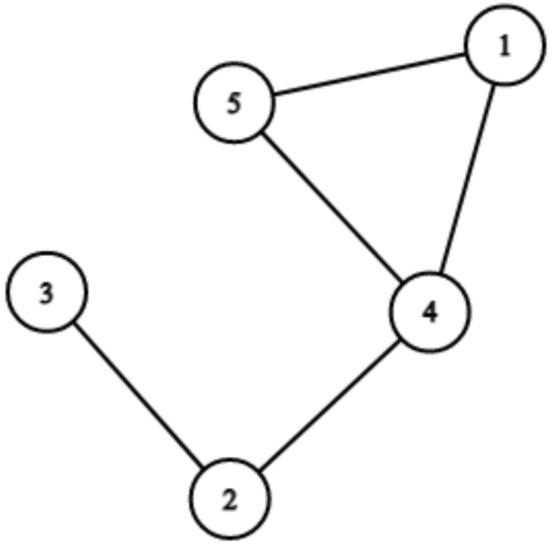
- Spectral clustering approach:
  - 1. Compute **Laplacian** L = D – A

(Important tool in graph theory)



$$L = \underbrace{\begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}}_{\text{Degree Matrix}} - \underbrace{\begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{Adjacency Matrix}} = \underbrace{\begin{bmatrix} 2 & 0 & 0 & -1 & -1 \\ 0 & 2 & -1 & -1 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ -1 & -1 & 0 & 3 & -1 \\ -1 & 0 & 0 & -1 & 2 \end{bmatrix}}_{\text{Laplacian}}$$

Spectral Clustering

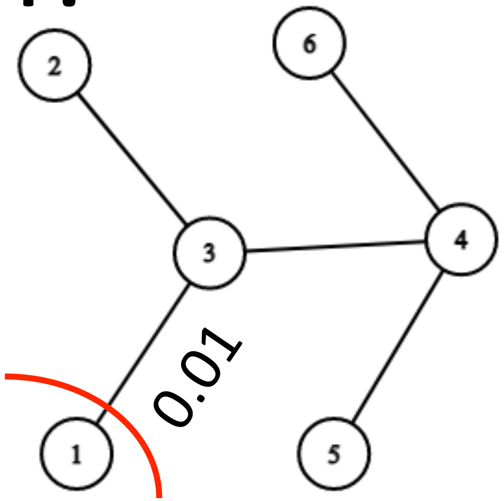


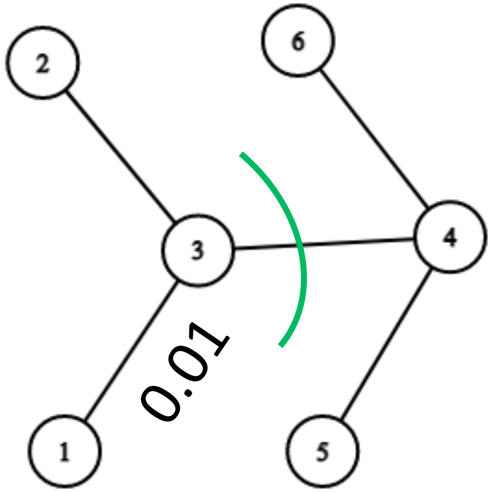
- Spectral clustering approach:
  - 1. Compute **Laplacian** L = D – A
  - 1a (optional): compute normalized Laplacian:
    $\mathbf{L = I - D^{1/2}AD^{1/2}}$, **or** $\mathbf{L = I - D^{-1}A}$
  - 2. Compute *k* **smallest** eigenvectors of **L**
  - 3. Set *U* to be the *n* x *k* matrix with u_1, …, u_k as columns. Take the *n* rows formed as points
  - 4. Run k-means on the representations

Why normalized Laplacian?

Want: partition V into V_1 and V_2

- Implies a graph "cut"
- One idea: minimize the **weight** of the cut
  - Downside: might just cut of one node
  - Need: "**balanced**" cut





Why Normalized Laplacian?

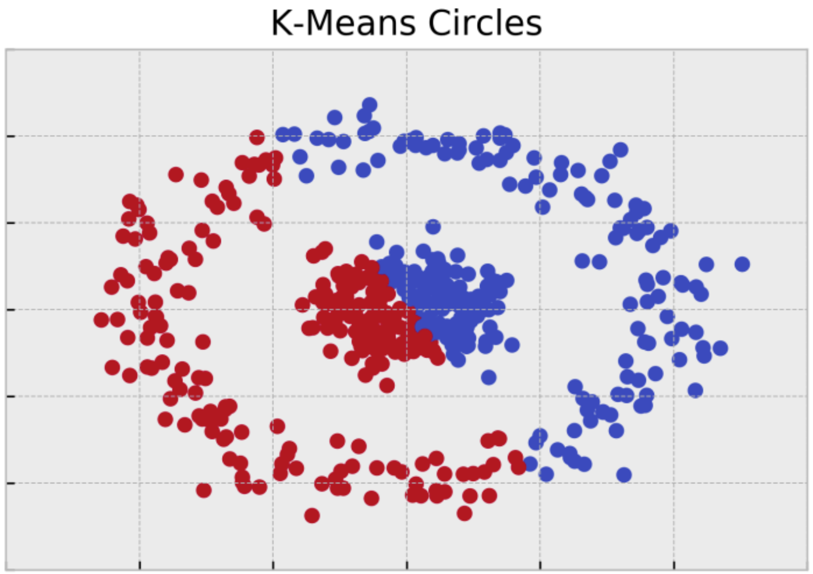
Want: partition V into V_1 and V_2

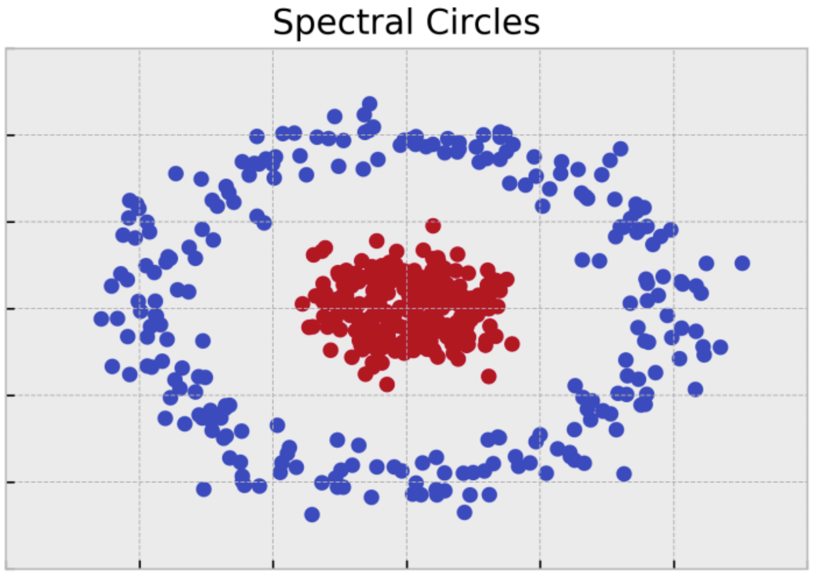
- Just minimizing weight is not always a good idea.
- We want **balance!**

$$\mathrm{Ncut}(A_1, \ldots, A_k) := \frac{1}{2} \sum_{i=1}^{k} \frac{W(A_i, \overline{A}_i)}{\mathrm{vol}(A_i)}$$

$$\mathrm{vol(A)} = \sum_{i \in A} \mathrm{degree(i)} = \sum_{i \in A} \sum_{j \in \mathrm{nbd}(i)} w_{ij}$$

Spectral Clustering





Credit: William Fleshman

Thanks Everyone!

Some of the slides in these lectures have been adapted/borrowed from materials developed by Mark Craven, David Page, Jude Shavlik, Tom Mitchell, Nina Balcan, Elad Hazan, Tom Dietterich, Pedro Domingos, Jerry Zhu, Yingyu Liang, Volodymyr Kuleshov, and Fred Sala