CS 760: Machine Learning
Unsupervised Learning I

Fred Sala

University of Wisconsin-Madison

Nov. 16, 2021

Announcements

- **Logistics**:
  - HW6 Due Thursday. HW7 out today
- Class roadmap:

Tuesday, Nov. 16	**Unsupervised Learning I**
Thursday, Nov. 18	Unsupervised Learning II
Tuesday, Nov. 23	Learning Theory
Tuesday, Nov. 30	RL I
Thursday, Dec. 2	RL II

Outline

- **Review & Self-Supervised Learning**
  - Contrastive learning, pretext tasks, SimCLR
- **Clustering**
  - k-means, hierarchical, spectral clustering
- **Gaussian Mixture Models**
  - Mixtures, Expectation-Maximization algorithm

Outline

- **Review & Self-Supervised Learning**
  - Contrastive learning, pretext tasks, SimCLR
- **Clustering**
  - k-means, hierarchical, spectral clustering
- **Gaussian Mixture Models**
  - Mixtures, Expectation-Maximization algorithm

Self Supervision: Basic Idea

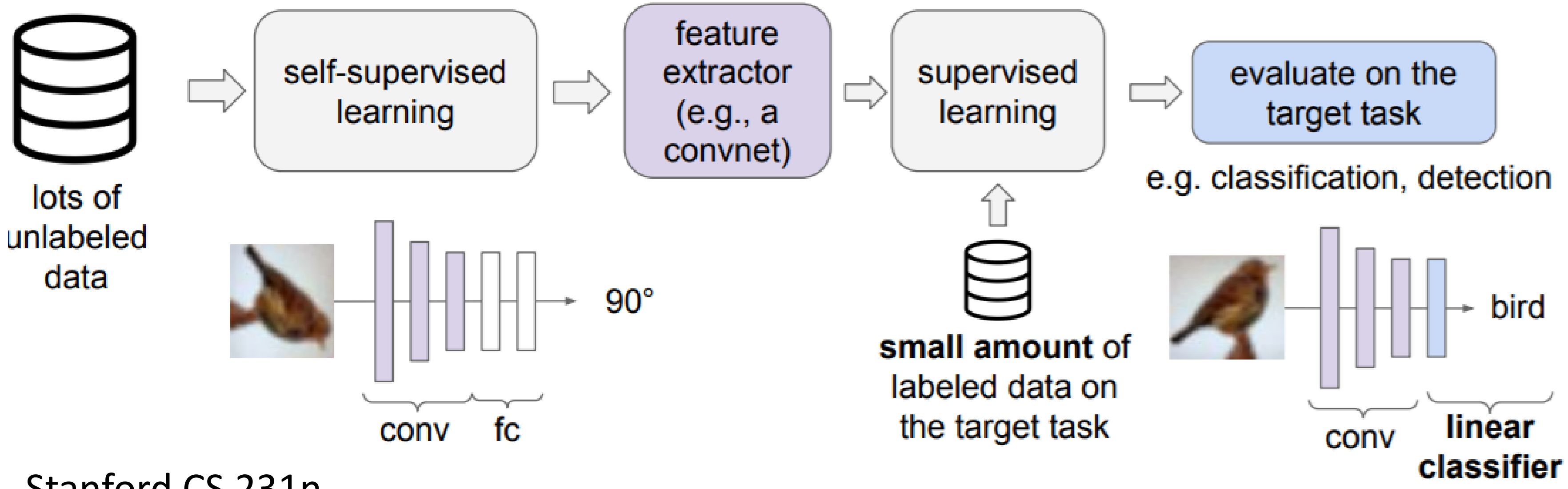
- Suppose we have no labeled data, nor weak sources
- What can we do with unlabeled data?
  - Generative modeling, etc.
  - Could also obtain **representations** (ie new features) for **downstream use.**
- Need to create tasks from unlabeled data: “Pretext tasks”
  - Ex: predict stuff you already know

image completion | rotation prediction | “jigsaw puzzle” | colorization

Stanford CS 231n

Self Supervision: Using the Representations

- Don't care specifically about our performance on pretext task
- Use the learned network as a feature extractor
- Once we have labels for a particular task, train
  - A small amount of data



Stanford CS 231n

Self Supervision: Pretext Tasks

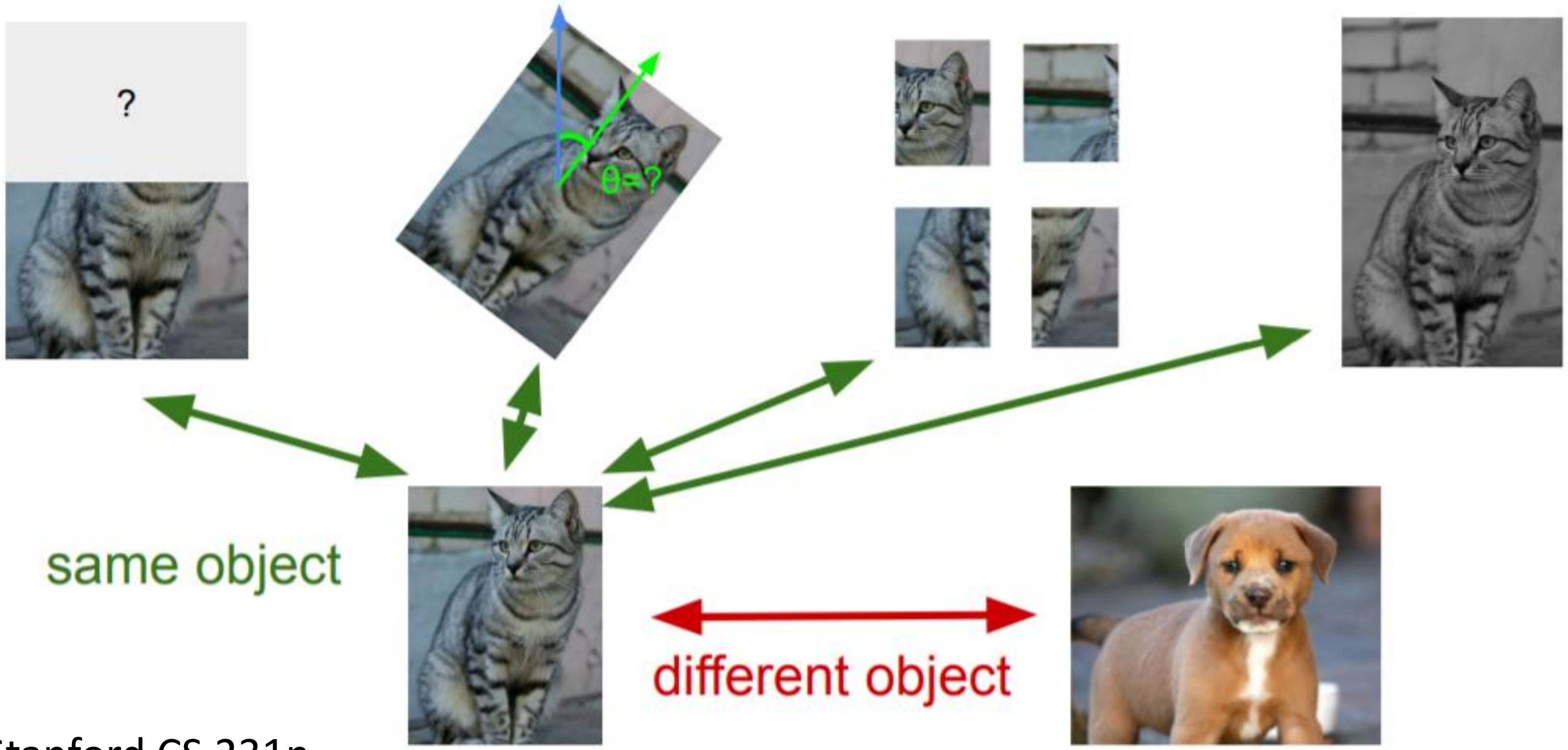
- Lots of options for pretext tasks
  - Predict rotations
  - Coloring
  - Fill in missing portions of the image
  - Solve puzzles:

(a) (b) (c)

Noroozi and Favaro

Contrastive Learning: Basics

- Want to learn representations so that:
  - Transformed versions of single sample are similar
  - Different samples are different



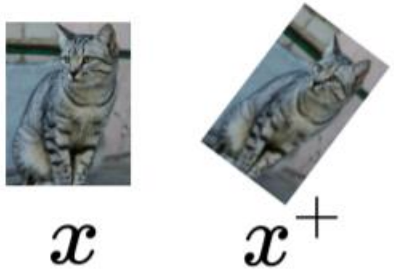
Stanford CS 231n

Contrastive Learning: Motivation

- Contrastive learning goal:
  - Keep together related representations, push unrelated apart.
  - The InfoNCE loss function:

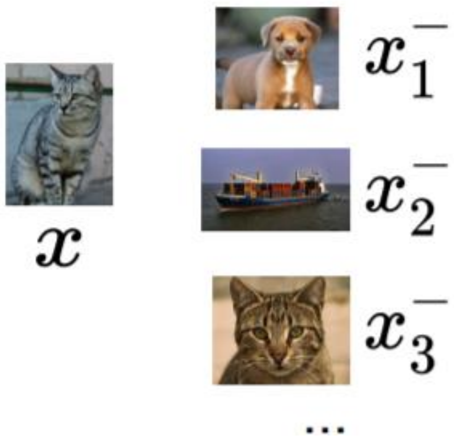
Van den Oord et al., 2018

$$L = -E_X \left[\log \frac{\exp(s(f(x), f(x^+))}{\exp(s(f(x), f(x^+)) + \sum_{j=1}^{k-1} \exp(s(f(x), f(x_j^-))} \right]$$



Positive sample: keep close

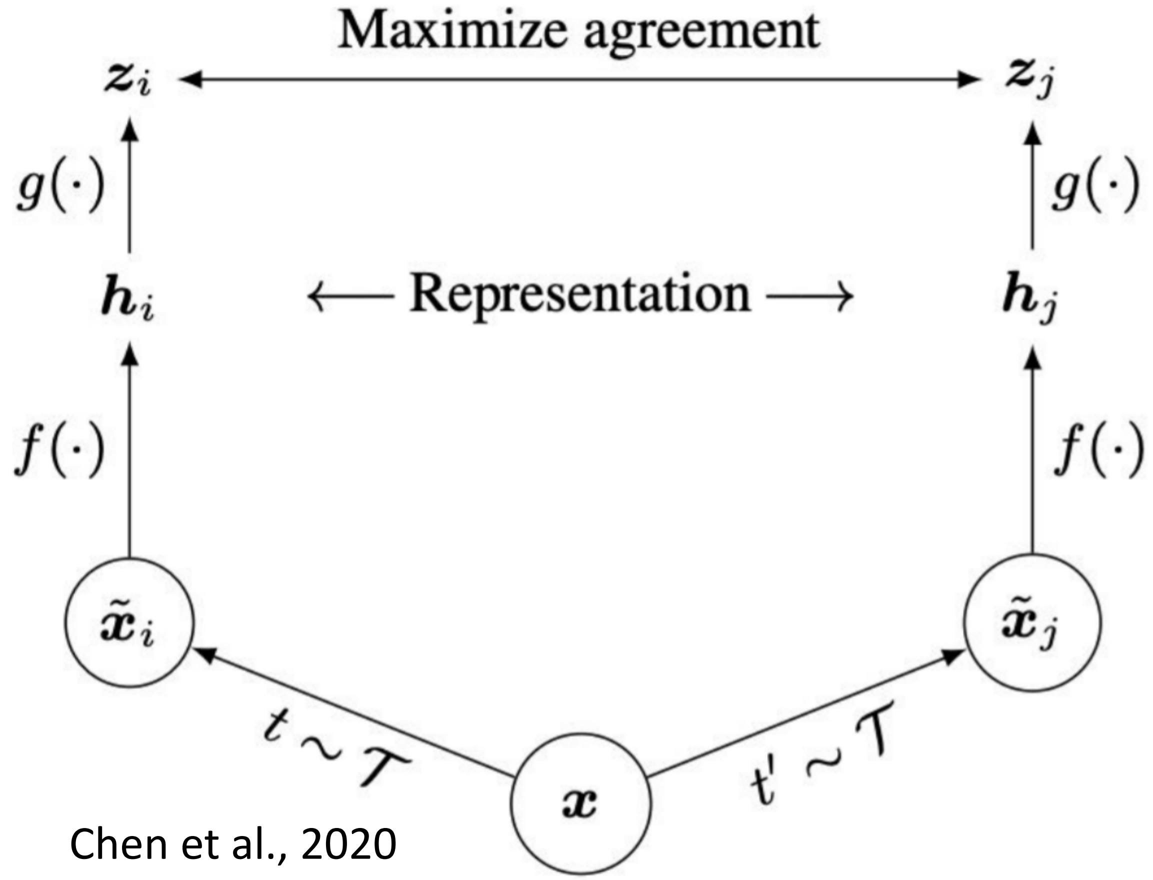
Negative sample keep far



Contrastive Learning: Frameworks

- Many approaches (very active area of research)
  - A popular approach: SimCLR. Score function is cosine similarity,

  - Generate positive samples:
  Choose random augmentations

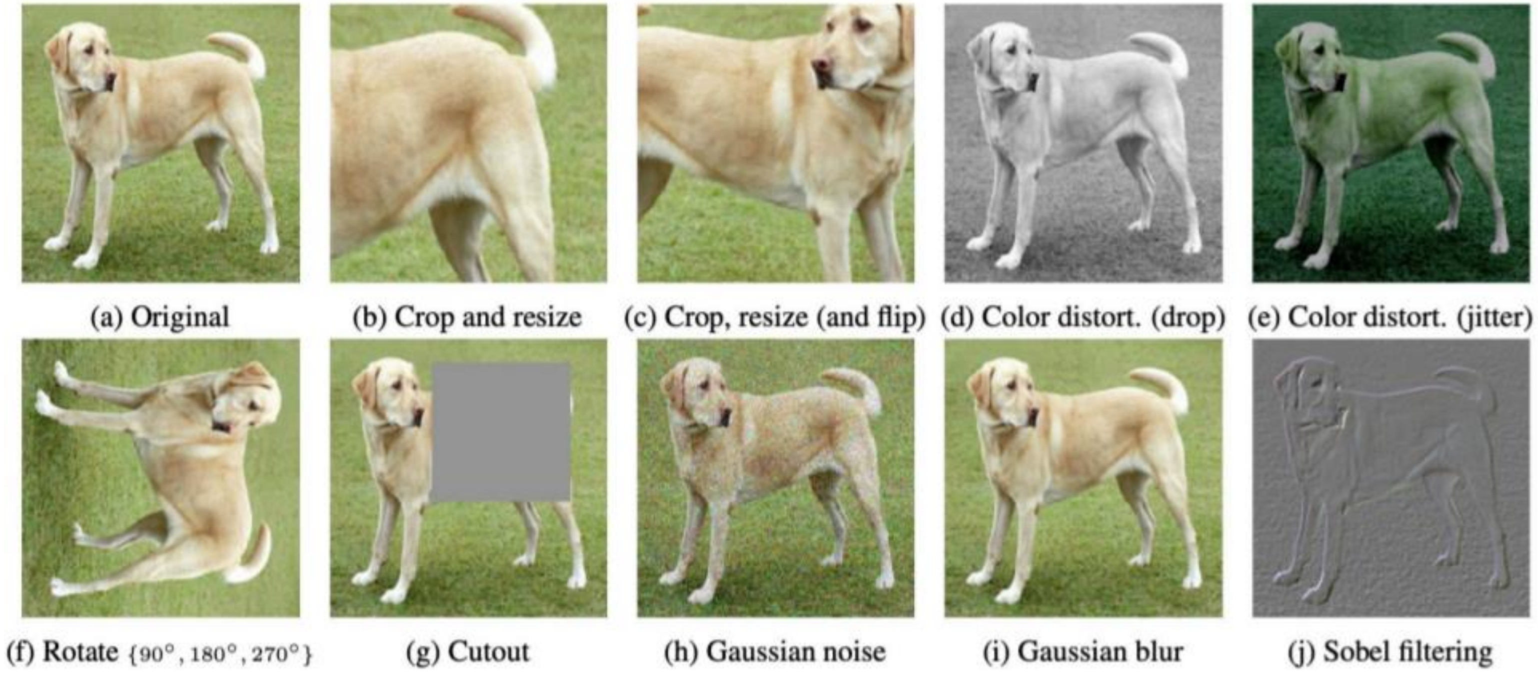


Chen et al., 2020

Contrastive Learning: Frameworks

- Many approaches (very active area of research)
  - A popular approach: SimCLR. Score function is cosine similarity,

  - Generate positive samples:
  Choose random augmentations



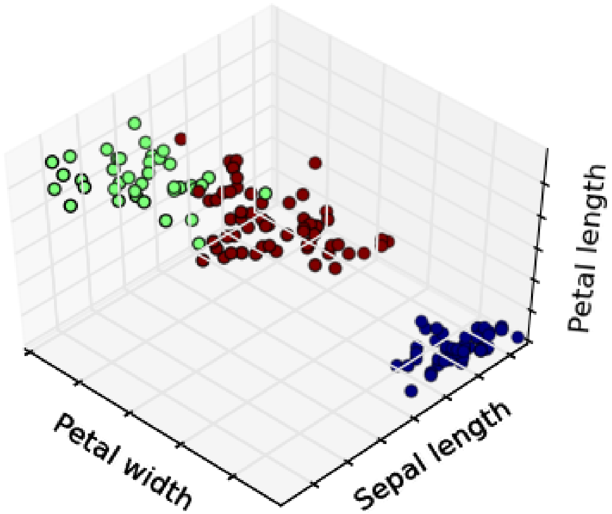
Break & Quiz

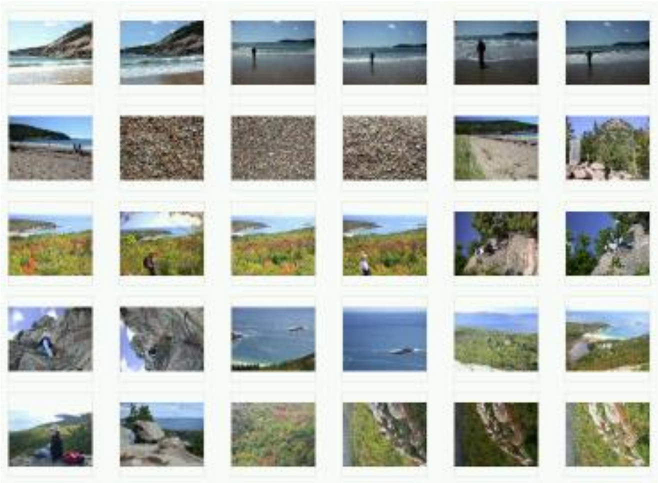
Outline

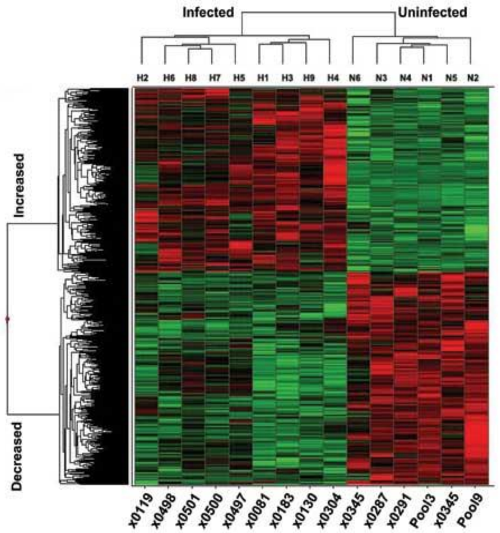
- **Review & Self-Supervised Learning**
  - Contrastive learning, pretext tasks, SimCLR
- **Clustering**
  - k-means, hierarchical, spectral clustering
- **Gaussian Mixture Models**
  - Mixtures, Expectation-Maximization algorithm

Unsupervised Learning

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







Mulvey and Gingold

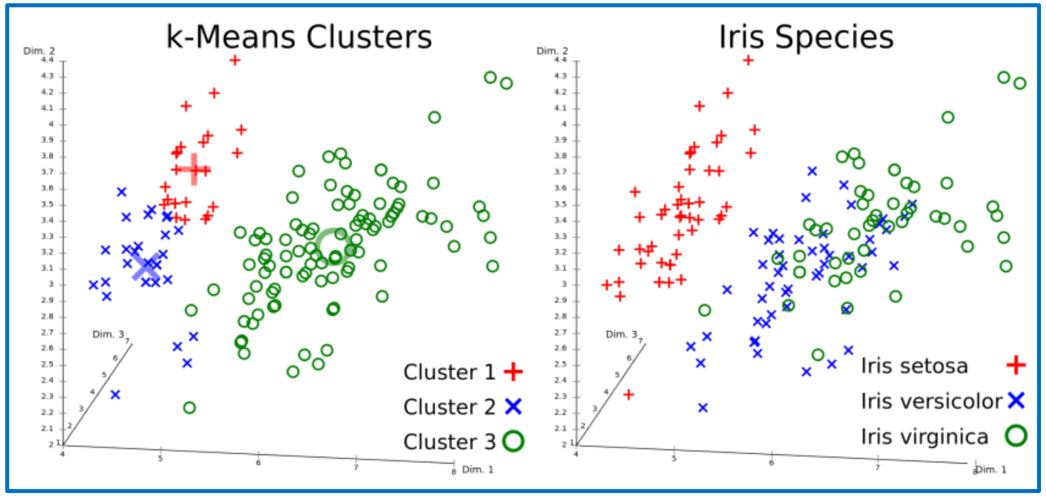
Clustering

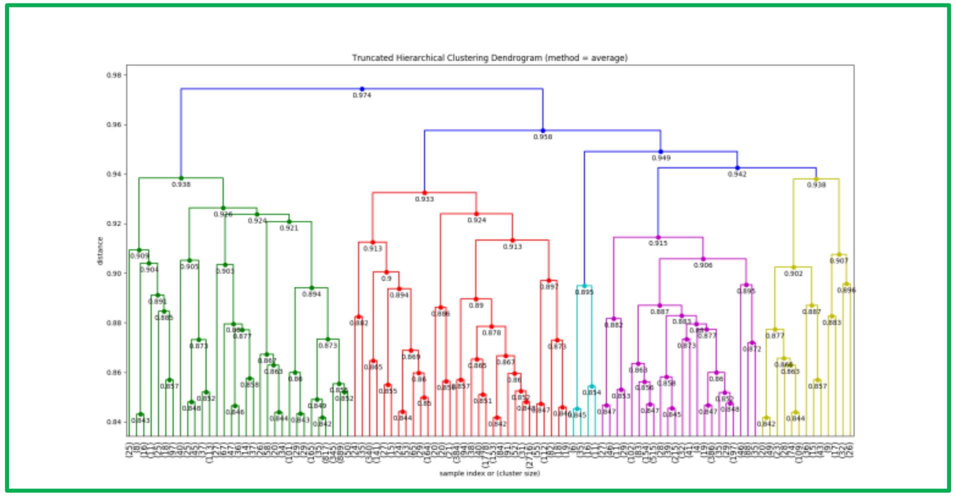
Several types:

Partitional
- Centroid
- Graph-theoretic
- Spectral

Hierarchical
- Agglomerative
- Divisive

Bayesian
- Decision-based
- Nonparametric



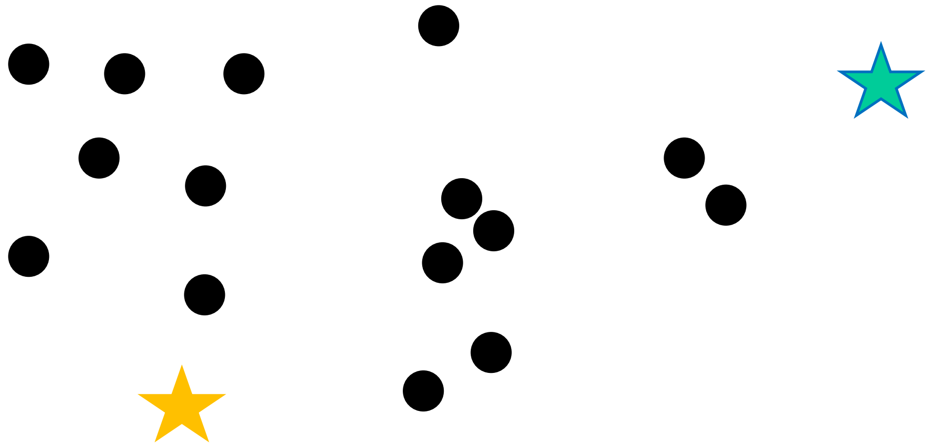


K-Means Clustering

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

Algorithm:

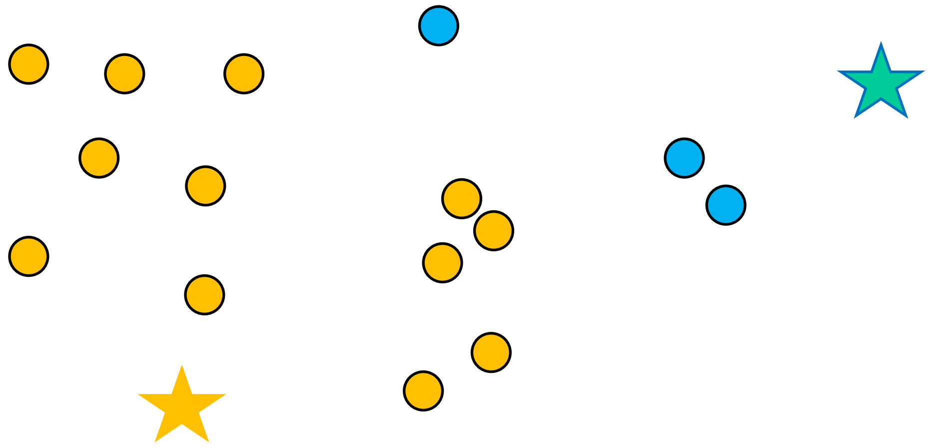
1. Randomly pick k cluster centers



K-Means Clustering: Algorithm

K-Means clustering

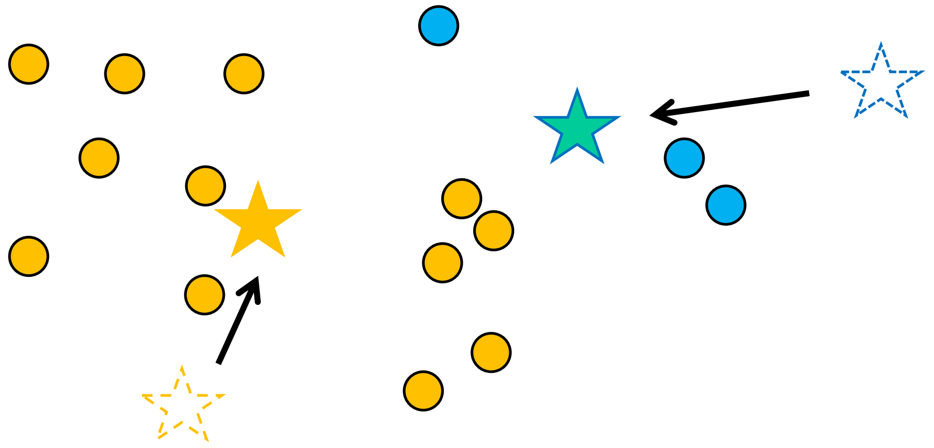
2. Find closest center for each point



K-Means Clustering: Algorithm

K-Means clustering

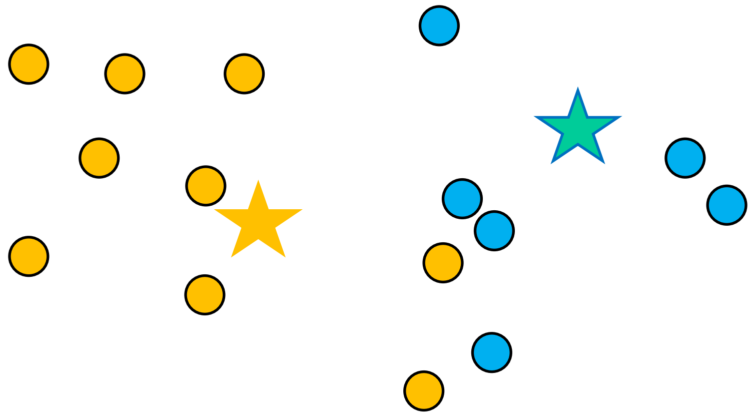
3. Update cluster centers by computing centroids



K-Means Clustering: Algorithm

K-Means clustering

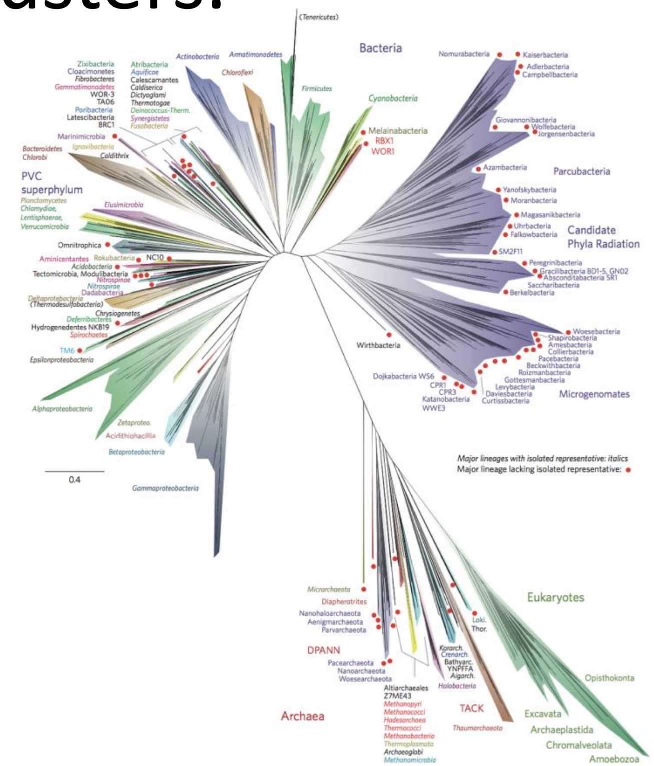
Repeat Steps 2 & 3 until convergence



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



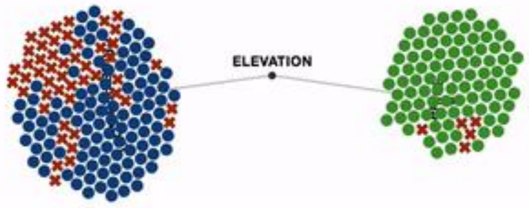
Credit: Wikipedia

HC: Agglomerative vs Divisive

Two ways to go:

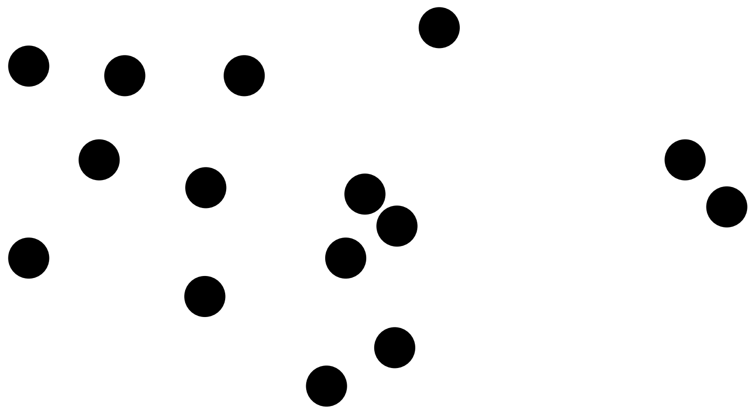
- **Agglomerative**: bottom up.
  - Start: each point a cluster.
  - Progressively merge clusters

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



HC: Agglomerative Clustering Example

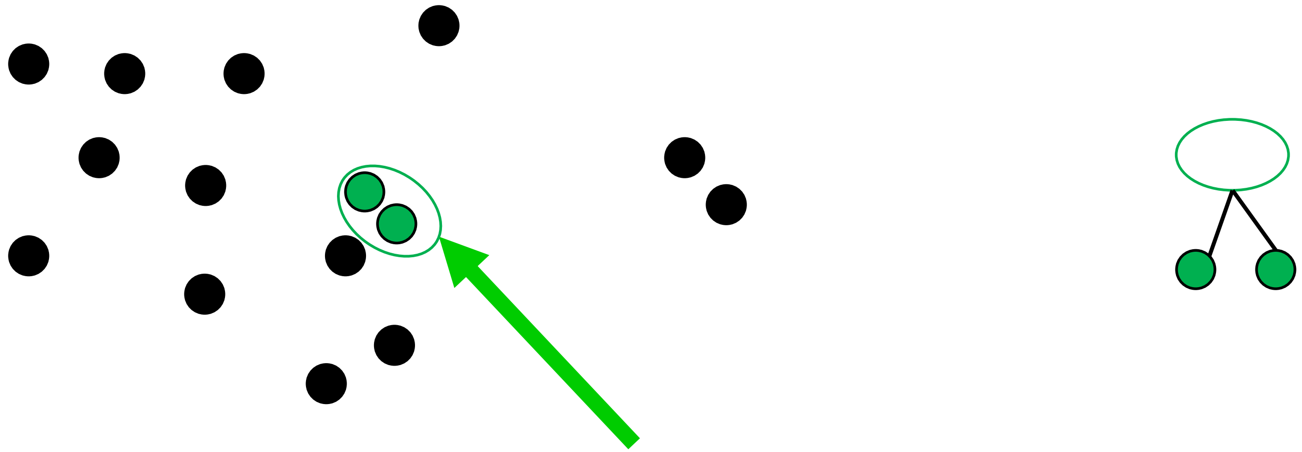
Agglomerative: Start: every point is its own cluster



HC: Agglomerative Clustering Example

Basic idea: build a “hierarchy”

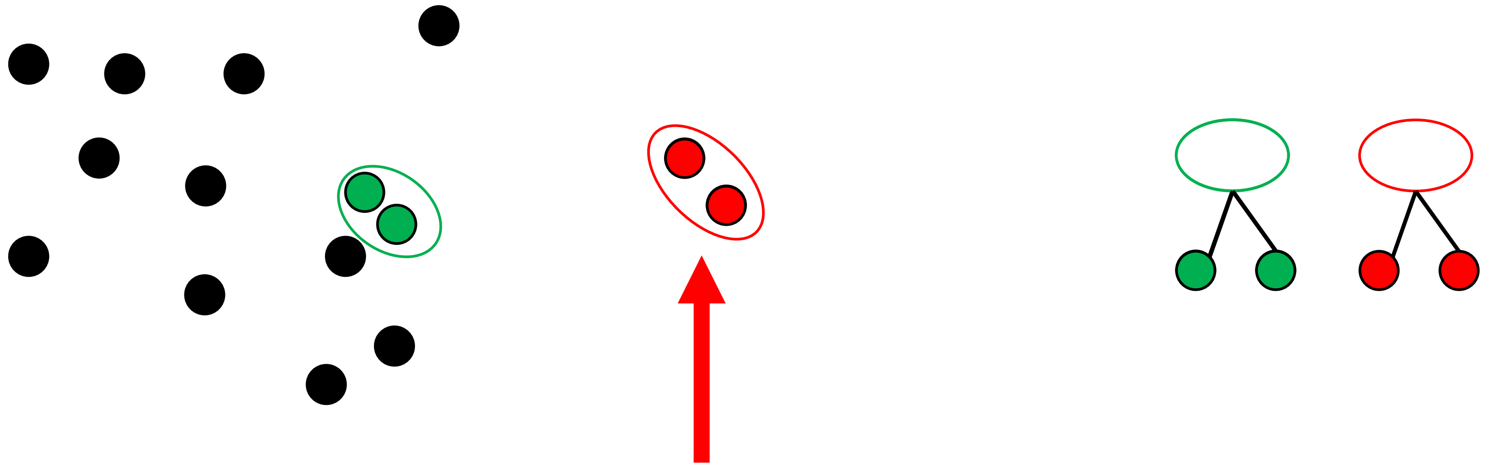
- Get pair of clusters that are closest and merge



HC: Agglomerative Clustering Example

Basic idea: build a “hierarchy”

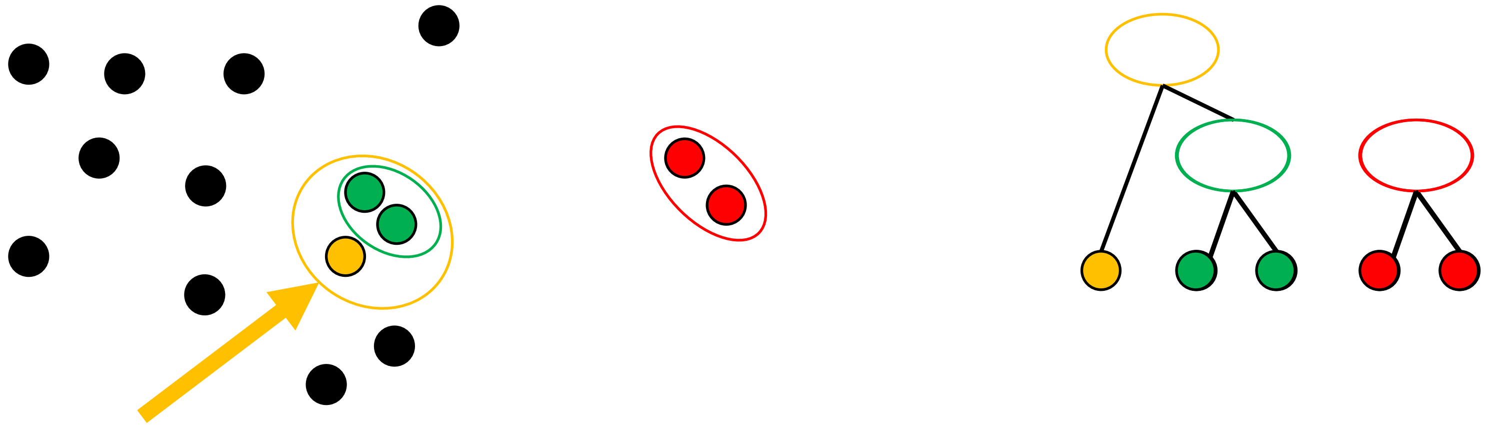
- **Repeat:** Get pair of clusters that are closest and merge



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?

- Single-linkage

$$d(A,B) = \min_{x_1 \in A, x_2 \in B} d(x_1, x_2)$$

- Complete-linkage

$$d(A,B) = \max_{x_1 \in A, x_2 \in B} d(x_1, x_2)$$

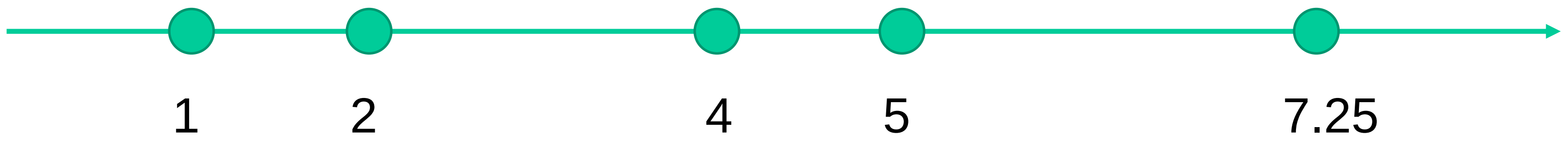
- Average-linkage

$$d(A,B) = \frac{1}{|A||B|} \sum_{x_1 \in A, x_2 \in B} d(x_1, x_2)$$

HC: Single-linkage Example

We'll merge using single-linkage

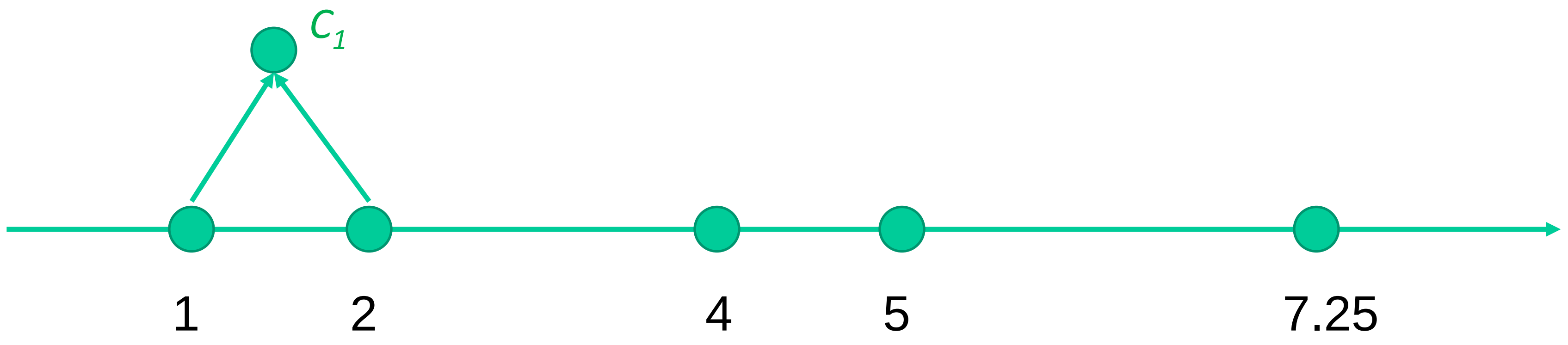
- 1-dimensional vectors.
- Initial: all points are clusters



HC: Single-linkage Example

Basic idea: build a "hierarchy"

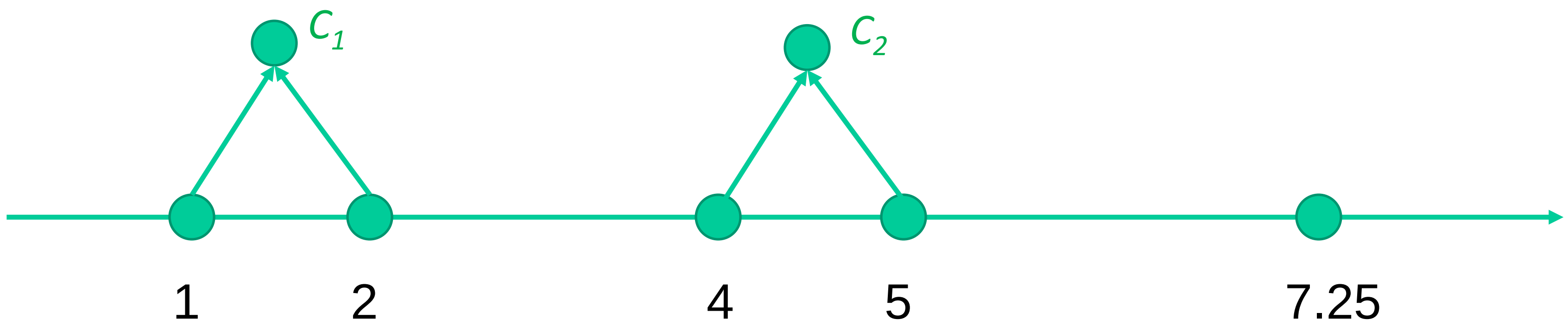
- Want: arrangements from specific to general



HC: Single-linkage Example

Basic idea: build a “hierarchy”

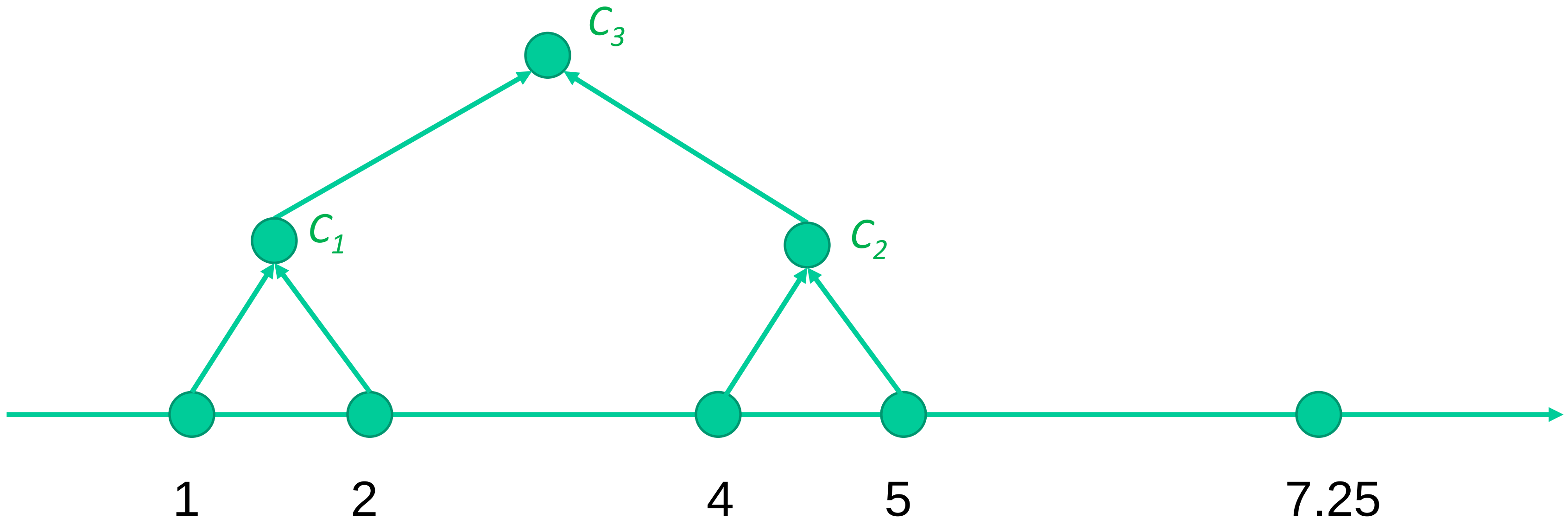
- Continue…



HC: Single-linkage Example

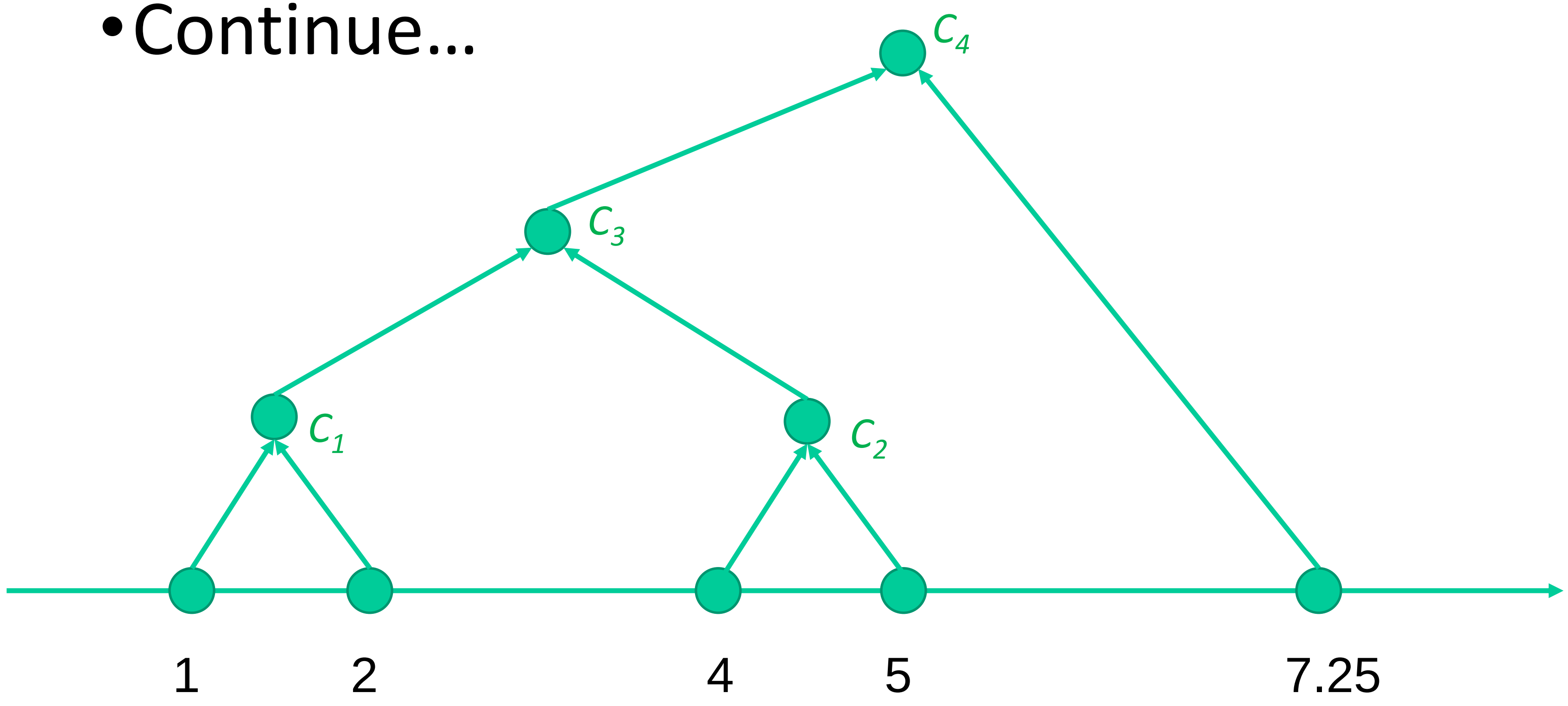
Basic idea: build a “hierarchy”

- Continue…



HC: Single-linkage Example

- Continue…

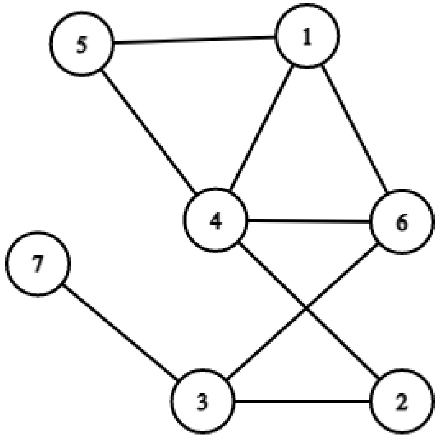


Other Types of Clustering

Graph-based/proximity-based

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

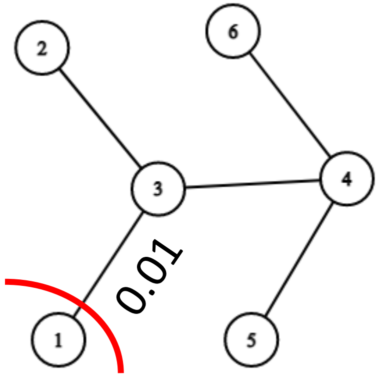
- Don't need vectors here
  - Just edges (and maybe weights)

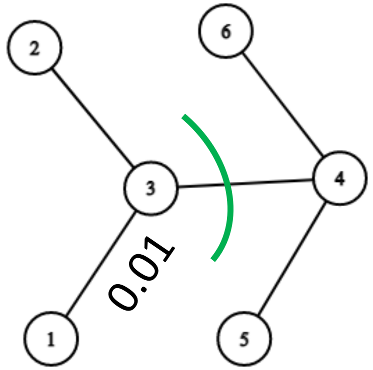


Graph-Based Clustering

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





Partition-Based Clustering

Want: partition V into V_1 and V_2

- Just minimizing weight isn't good… want **balance!**
- **Approaches:**

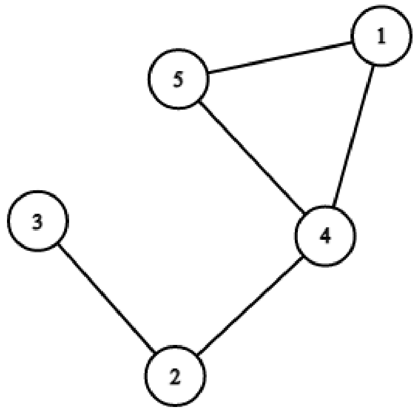
$$\mathrm{Cut}(V_1, V_2) = \frac{\mathrm{Cut}(V_1, V_2)}{|V_1|} + \frac{\mathrm{Cut}(V_1, V_2)}{|V_2|}$$

$$\mathrm{NCut}(V_1, V_2) = \frac{\mathrm{Cut}(V_1, V_2)}{\sum_{\imath \in V_1} d_i} + \frac{\mathrm{Cut}(V_1, V_2)}{\sum_{\imath \in V_2} d_i}$$

Partition-Based Clustering

How do we compute these?

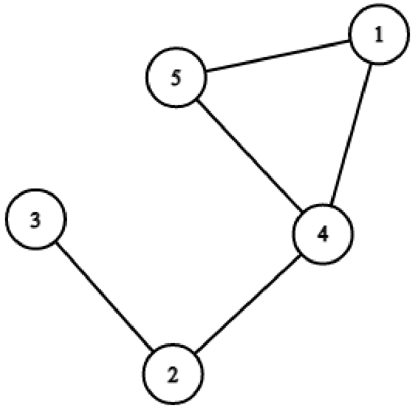
- Hard problem → heuristics
  - Greedy algorithm
  - "Spectral" approaches
- Spectral clustering approach:
  - **Adjacency** matrix



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

Partition-Based Clustering

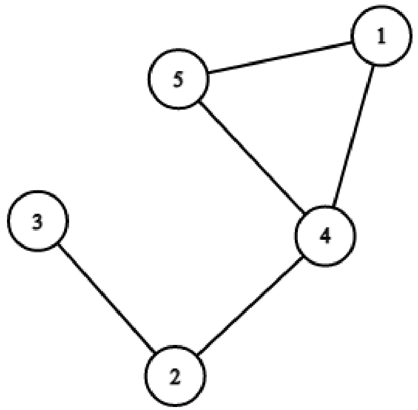
- Spectral clustering approach:
  - **Adjacency** matrix
  - **Degree** matrix



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

Spectral Clustering

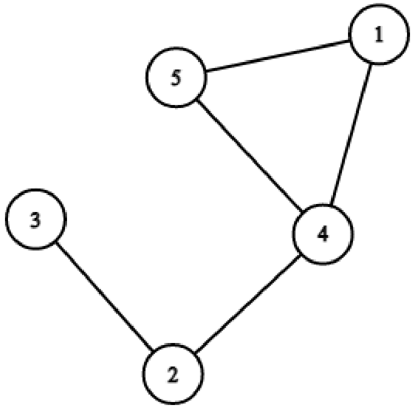
- Spectral clustering approach:
  - 1. Compute **Laplacian** **L** = **D** – **A** (Important tool in graph theory)



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

Spectral Clustering

Spectral clustering approach:

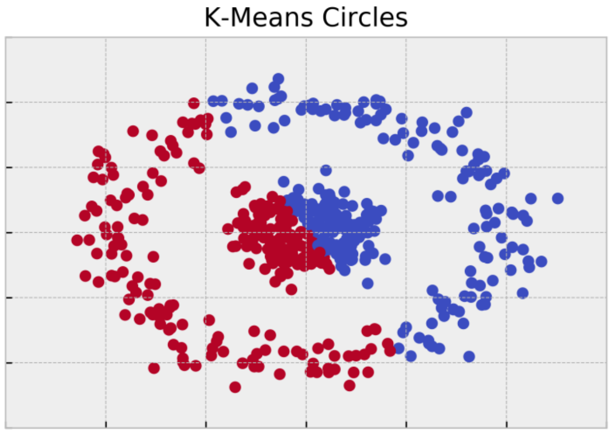


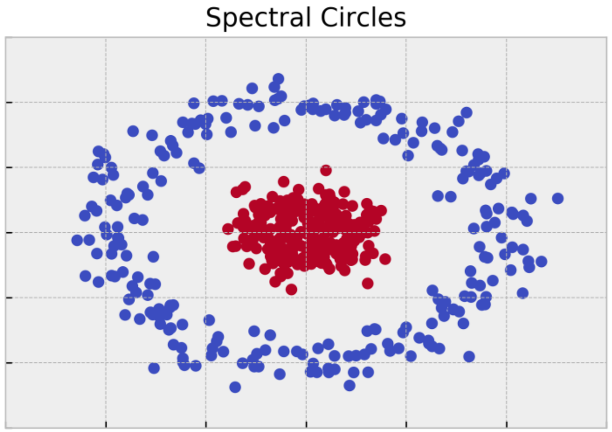
- 1. Compute **Laplacian** L = D – A
- 2. Compute *k* **smallest** eigenvectors
- 3. Set U to be the n x k matrix with u_1, u_k as columns. Take the n rows formed as points
- 4. Run k-means on the representations

Spectral Clustering

Q: Why do this?

- 1. No need for points or distances as input
- 2. Can handle intuitive separation (k-means can't!)





Credit: William Fleshman

Break & Quiz

Outline

- **Review & Self-Supervised Learning**
  - Contrastive learning, pretext tasks, SimCLR
- **Clustering**
  - k-means, hierarchical, spectral clustering
- **Gaussian Mixture Models**
  - Mixtures, Expectation-Maximization algorithm

Mixture Models

- Let's 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
  - We did something similar with flows
  - Model:

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

Mixture Models: Gaussians

- Lots of different kinds of mixtures, but let's 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's make x be conditional Gaussian

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

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… very similar to Gaussian Naïve Bayes
- First, empirically estimate the z parameters:

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

- Next the Gaussian components:

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

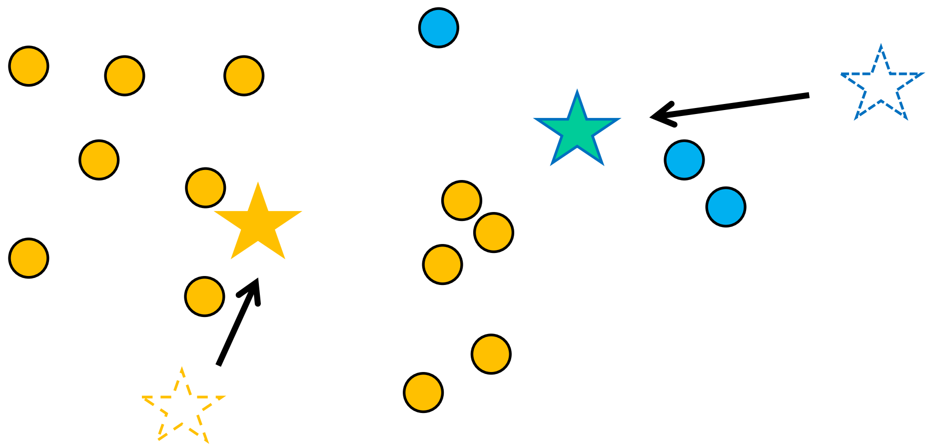
Average of x's where z = j

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

GMMs: Back to Latent Setting

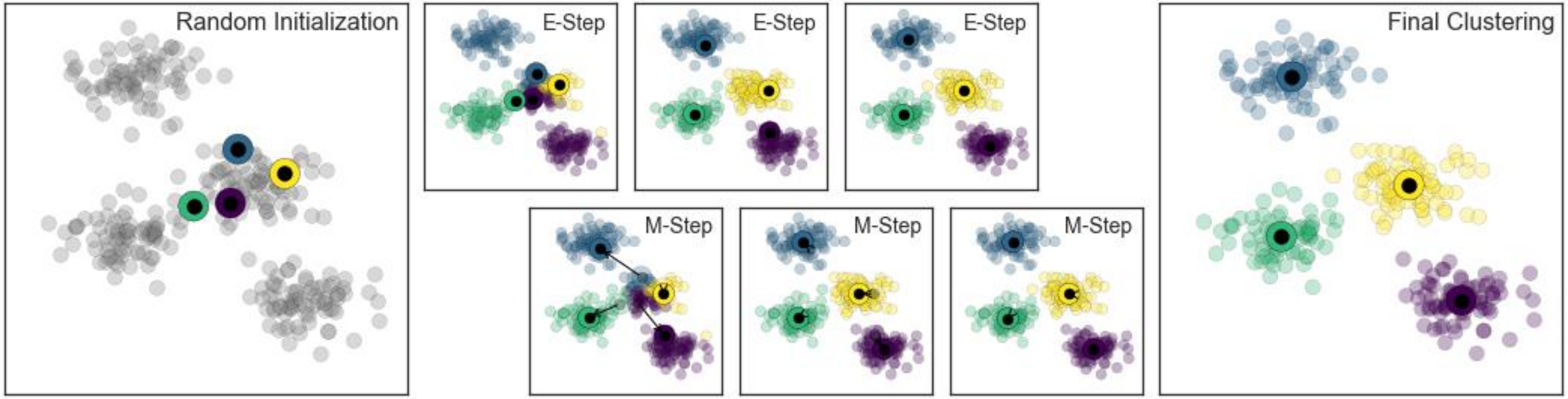
- But, we don't get to see the z's
  - Similar to the weak supervision setting from last time.

- 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 (expectation): guess the latent variables
  - **M**-step (maximization): update the parameters of the model
  - Note similarity to k-means clustering.



Jake VanderPlas

GMM EM: E-Step

- Let's 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.

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

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

Soft version of our empirical mean and covariances.

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