Clustering

CS 480
Intro to Artificial Intelligence

Unsupervised vs Supervised

Supervised Learning
Given data with labels (inputs and outputs), find mapping from inputs to outputs

$$(\mathbf{x}^{(i)}, y^{(i)}) \in \mathcal{X} \times \mathcal{Y}, \qquad h(\mathbf{x}) = y$$

Unsupervised Learning
Given data **without** labels (inputs), find "structure" in the data

$$\mathbf{x}^{(i)} \in \mathcal{X}, \quad h(\mathbf{x}) = ?$$

What's **structure**?

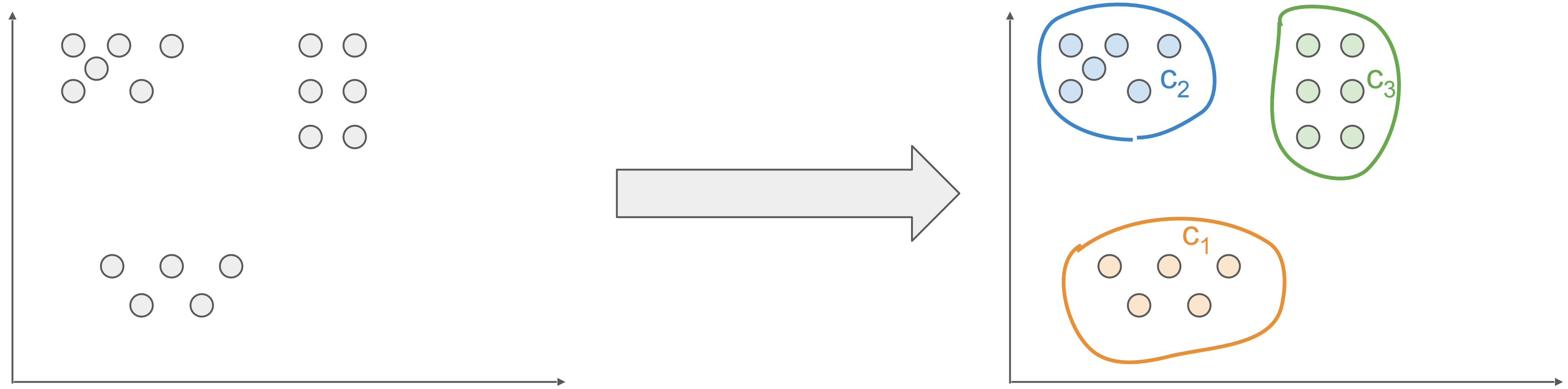
Types of structure

Distribution
Find a **probability distribution** that describes the likelihood of any **x**: p(**x**|θ) (compare with MLE/MAP: p(y|**x**,θ))

Similarity
Find a **function** that describes the "distance" between pairs of points **x**, **x**' (SVM, kNN, …)

Summarization
Find a **compact** representation of the data (compression?): partition the data into groups or **clusters**

Clustering graphically



Clustering

Given:

1. Data
2. Similarity/Distance function $d(\mathbf{x}^{(i)},\mathbf{x}^{(j)})$

Output: a partition function P_d

$$P_d(\mathbf{x}^{(i)}) = P_d(\mathbf{x}^{(j)})$$

$$\iff$$

$$\mathbf{x}^{(i)} \text{ and } \mathbf{x}^{(j)} \text{ are in the same cluster}$$

Single cluster,
$P_d(\mathbf{x}) = 1 \ \forall\ \mathbf{x}$

Want to find a happy medium

Cluster for each $\mathbf{x}$,
$P_d(\mathbf{x}) = \mathbf{x}$

Ignores similarity

Doesn’t summarize

Algorithm: Single Linkage Clustering

Init: put each **x**$^{(i)}$ into its own cluster

$$C = \left\{\{\mathbf{x}^{(1)}\}, \{\mathbf{x}^{(2)}\}, \ldots, \{\mathbf{x}^{(N)}\}\right\}$$
$$= \{c_1, c_2, \ldots, c_N\}$$

Loop:

1. Compute **I**nter-**C**luster **D**istance between all clusters
2. Merge the two clusters with minimum ICD
3. Stop after (N-k) iterations to generate k clusters

ICD

The minimum distance between any two points in each cluster:

$$ICD(c_i, c_j) = \min_{k,l} d(\mathbf{x}^{(k)}, \mathbf{x}^{(l)}),$$

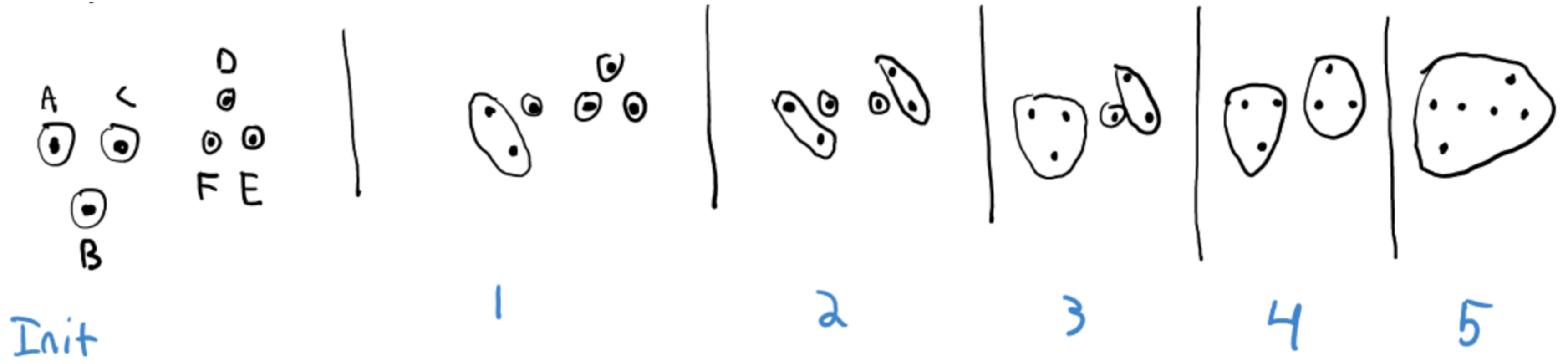
$$\mathbf{x}^{(k)} \in c_i, \quad \mathbf{x}^{(l)} \in c_j$$

Merging Clusters

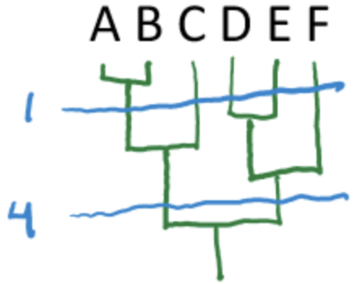
$$C \leftarrow (C/\{c_i, c_j\}) \cup \{c_i \cup c_j\}$$

Delete c_i and c_j. Add c_i union c_j.

SLC example



Each iteration is a different “cut” through the **dendrogram**

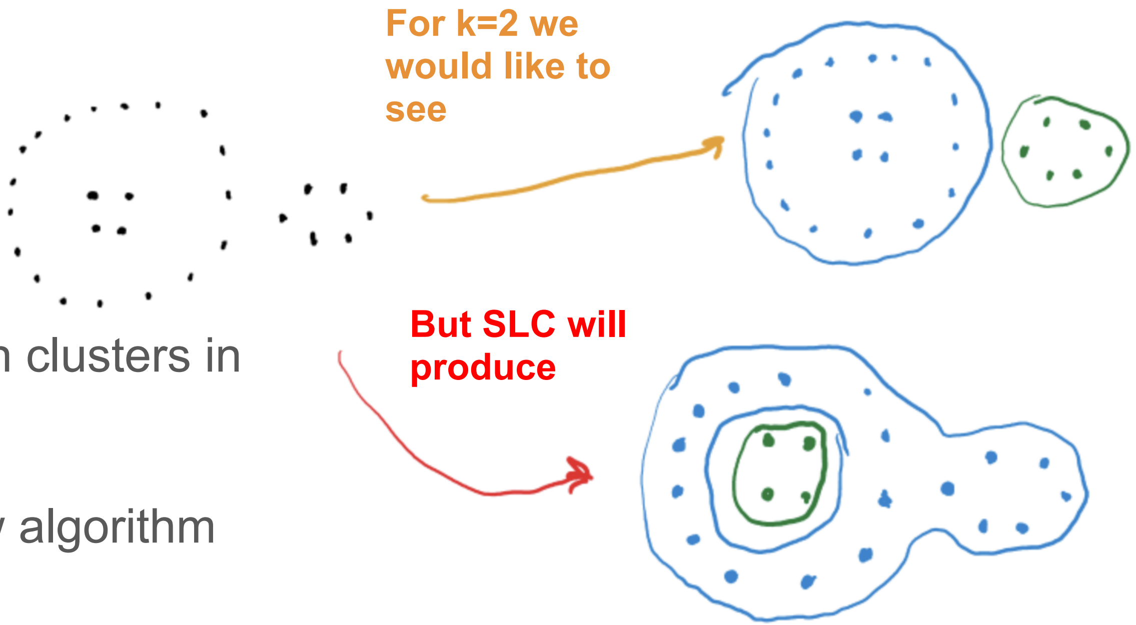


SLC notes

- SLC has a finite number of possible steps (1 to N-1)
- Start with $P_d(\mathbf{x}) = \mathbf{x}$ (not a summary)
- End with $P_d(\mathbf{x}) = 1$ (ignores similarity)
- Choosing the number of clusters == trading off the two ends of the spectrum
- Alternative choices for ICD
  - Mean over all pairwise distances
  - Median
- Running time:
  - Compare N points to N points at most N times: $O(N^3)$

SLC Pitfalls

SLC is greedy!



Can’t re-assign clusters in later iterations

Time for a new algorithm

Algorithm: *k*-means clustering

Init: pick *k* “centers” at random

$$\{\hat{c}_1^{(0)}, \hat{c}_2^{(0)}, \ldots, \hat{c}_k^{(0)}\}$$

Loop:

1. Assign each $\mathbf{x}^{(i)}$ to the nearest center
2. Recompute centers by averaging all the points assigned to it
3. Stop when the center assignments no longer change

Center i at iteration t: $\hat{c}_i^{(t)}$

Partition for **x** at time t: $P_d^{(t)}(\mathbf{x})$

Points assigned to center i at time t:

$$c_i^{(t)} = \{\mathbf{x}^{(j)} : P_d^{(t)}(\mathbf{x}^{(j)}) = i\}$$

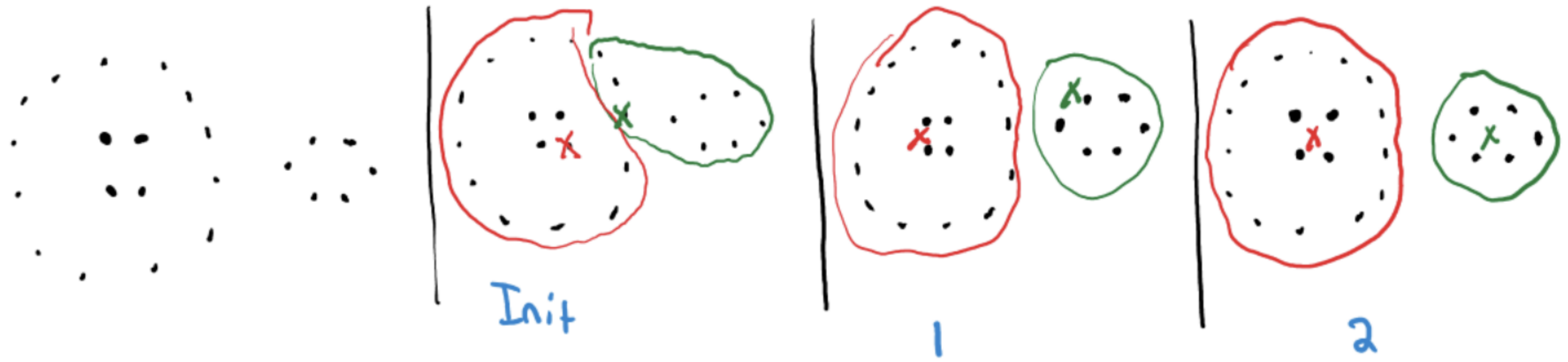
Assign $\mathbf{x}^{(i)}$ to nearest center

$$P_d^{(t+1)}(\mathbf{x}^{(j)}) \leftarrow \arg\min_i d(\hat{c}_i^{(t)}, \mathbf{x}^{(j)})$$

Recompute centers

$$\hat{c}_i^{(t)} \leftarrow \text{average}(c_i^{(t)}) = \frac{\sum_{\mathbf{x}^{(j)} \in c_i^{(t)}} \mathbf{x}^{(j)}}{|c_i^{(t)}|}$$

k-means example



Random init: pick from $\mathbf{x}^{(i)}$

Once the assignments don't change, centers stop moving and converge

k-means in Euclidean space

2-step iteration

(1) $$P_d^{(t+1)}(\mathbf{x}^{(j)}) \leftarrow \arg\min_i d(\hat{c}_i^{(t)}, \mathbf{x}^{(j)})$$

(2) $$\hat{c}_i^{(t)} \leftarrow \text{average}(c_i^{(t)}) = \frac{\sum_{\mathbf{x}^{(j)} \in c_i^{(t)}} \mathbf{x}^{(j)}}{|c_i^{(t)}|}$$

At each iteration, **(2)** moves the centers to minimize distance from each point to its assigned cluster, and **(1)** assigns points to the cluster with minimal distance.

This is a form of **local optimization**

k-means as local optimization

Fitness/cost:

$$\sum_{\mathbf{x}^{(j)}} \|\hat{c}^{(t)}_{P_d(\mathbf{x}^{(j)})} - \mathbf{x}^{(j)}\|^2$$

Parameters:

$$\hat{c}_i^{(t)}, \quad P_d(\mathbf{x}^{(j)})$$

Neighborhood:

$$P_d^{(t+1)}(\mathbf{x}^{(j)}) \leftarrow \arg\min_i d(\hat{c}_i^{(t)}, \mathbf{x}^{(j)})$$

$$\hat{c}_i^{(t)} \leftarrow \text{average}(c_i^{(t)}) = \frac{\sum_{\mathbf{x}^{(j)} \in c_i^{(t)}} \mathbf{x}^{(j)}}{|c_i^{(t)}|}$$

Step (1) can only improve fitness (argmin over possible assignments)

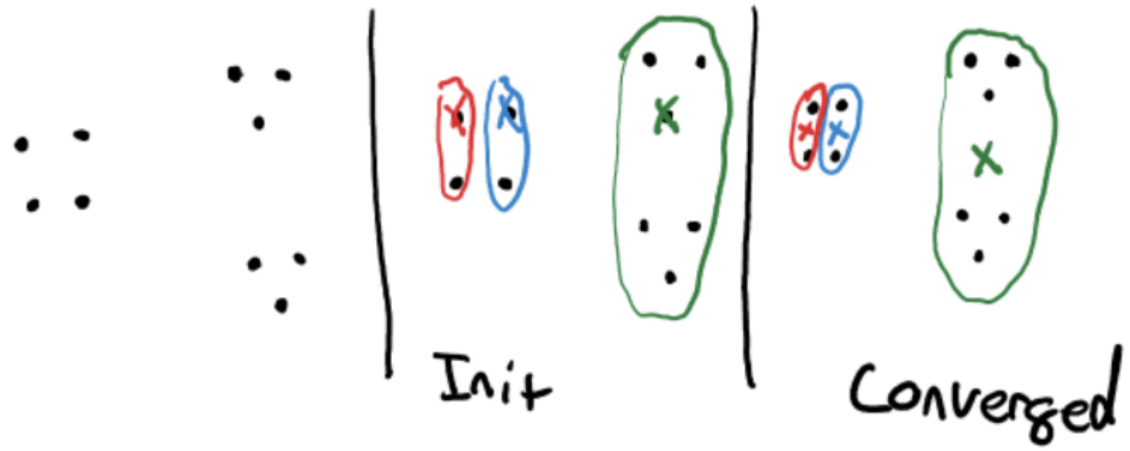
Step (2) can only improve fitness (the average has the minimum sum squared distance

So at each iteration, *k*-means moves towards the neighbor with the best fitness and stops after hitting a local maximum!

k-means properties

- Will *k*-means always converge? Yes!
    - Finite number of data points → finite number of partitions
    - Each partition **exactly** defines a center
    - Steps (1) and (2) **never** decrease fitness
    - As long as ties are broken **consistently**, will never revisit previous centers/partitions, must stop eventually
- How many configurations can there be? $O(k^N)$ (permutation of points to clusters)
- In practice converges much faster than exponential
    - Never visit most configurations
    - Distance is a very strong constraint (triangle inequality, symmetry, transitivity)
- Each iteration takes polynomial time: $O(k*N)$

k-means and local optima



- Since *k*-means is a **local** optimization, it can get stuck in **local optima.**
- Solution: random restarts

Clustering algorithm properties

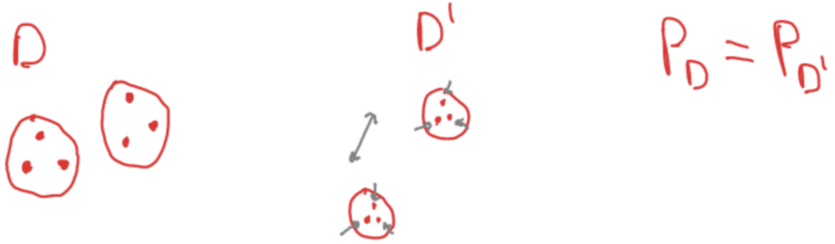
Richness
For any assignment of objects to clusters, there exists some distance function such that the algorithm returns that clustering

$$\forall P_d,\ \exists d(x,x'),\ \text{s.t. algorithm returns } P_d$$

Scale Invariance
Scaling distances by a positive value does not change the cluster assignment

$$\forall d, \forall l > 0, d_l(x,x') = l \cdot d(x,x')$$
$$P_d = P_{d_l}$$

Consistency
Decreasing within-cluster distance and increasing between-cluster distance does not change clustering



An impossibility result

No clustering algorithm can simultaneously achieve **richness**, **scale invariance**, and **consistency** (Kleinberg, 2003)

An example with SLC
The partition is determined by the stopping criteria. Here's three different versions.

1. **Stop when N/2 clusters are created**
   Scale Invariance? Yes.
   Consistency? Yes.
   Richness? **No** (number of partitions is fixed)
2. **Stop when minimum ICD is above some fixed threshold** θ
   Richness? Yes.
   Consistency? Yes.
   Scale Invariance? **No** (scale *determines* partitions)
3. **Stop when minimum ICD is greater than some fraction of the diameter of the points**
   Richness? Yes.
   Scale Invariance? Yes.
   Consistency? **No** (increasing between cluster distance changes diameter)

Summary and preview

Wrapping up

- Clustering: grouping points together based on similarity
- Single Linkage Clustering: greedily merge closest clusters
- *k*-Means: alternate between assigning and computing cluster centers
    - Can be thought of as local optimization
    - Will converge in a finite number of steps
    - May need random restarts to get out of local optima

Next time

- Soft cluster assignment with probabilities and Expectation Maximization