20211004 Soft Clustering

Monday, October 4, 2021 3:14 PM

- Final Project
  - Paired up with team
  - Emails from project managers to schedule kick off meeting
    - Face time with project manager, client
  - Deliverable 0: high level, reiterate project description and evaluate if scope is appropriate
    - Preliminary research on problem before getting started
    - May discover later you don't have the right information available, and then you need to be agile and change scope
    - Agile framework
      - Retrospective every 3 weeks with team to address issues as we go along
        - If needed, the PM and prof will help facilitate this
- The need to have started HW1
  - Office hours on Wednesday
- Facebook was down today
- Last time:
  - Probability review
  - Didn't finish
  - Last step: what is the distribution of sample average
  - Sample 10 times the waiting time
    - The sum of exponentials is an Errlong? distribution
    - Central limit theorem: with large enough samples, the sample mean will converge to normal distribution
      - CONVERGE to it in the LIMIT
      - Only a good enough approximation with a big enough sample
- Today: soft clustering

Soft Clustering

-

Soft Clustering

So far, clustering was done using **hard assignments** (1 point -> 1 cluster)

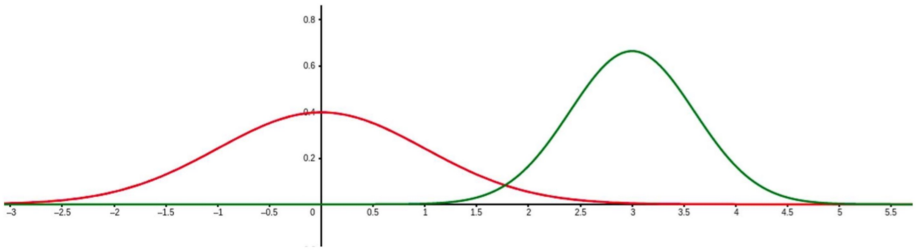
Sometimes this doesn't accurately represent the data: it seems reasonable to have overlapping clusters.

In this case, we can use **soft assignment** to assign points to every cluster with a certain probability.

- Every point either assigned to a cluster or a hierarchy of clusters
  - Cut dendrogram into a specific cluster

Soft Clustering - Example

Generate data using $\mathbf{N(\mu_1, \sigma_1)}$ and $\mathbf{N(\mu_2, \sigma_2)}$

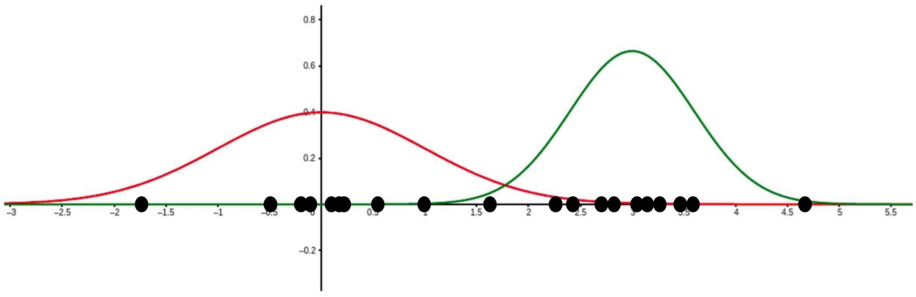


Or: we are given the weights of animals. Unknown to us these are weights from two different species. Can we determine the species (group / assignment) from the height?

- Imagine 0 is further down
  - Distribution weights of two different species
- There are points that fall into the range where they may be part of one cluster of the other

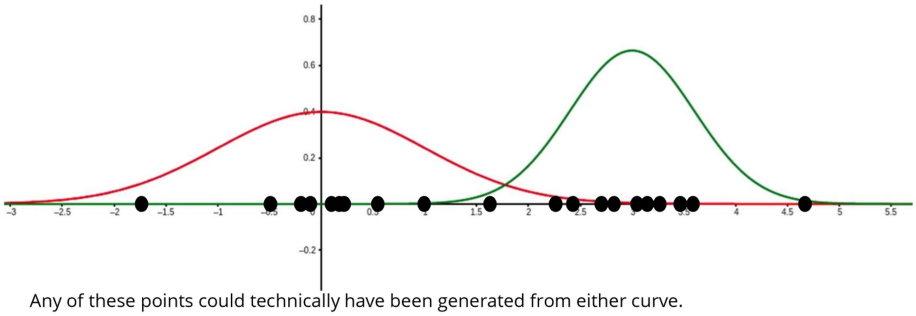
Soft Clustering - Example

Generate data using $\mathbf{N(\mu_1, \sigma_1)}$ and $\mathbf{N(\mu_2, \sigma_2)}$



Soft Clustering - Example

Generate data using $\mathbf{N(\mu_1, \sigma_1)}$ and $\mathbf{N(\mu_2, \sigma_2)}$

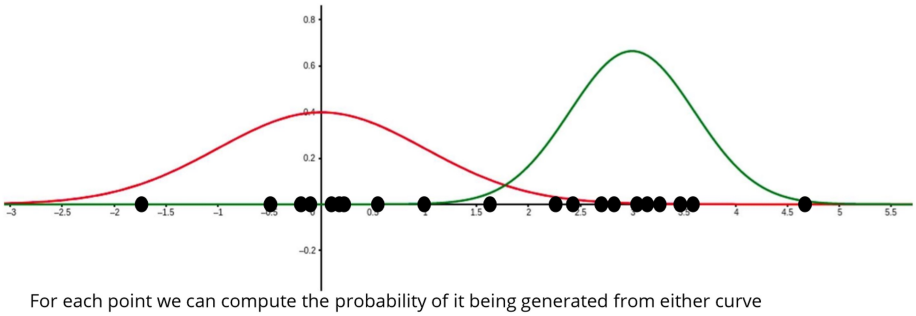


Any of these points could technically have been generated from either curve.

- Some points are more likely to be generated from one or the other
  - But even extreme points might be part of the other curve that is very low probability
- N(mu, sigma): Normal distibution: Normal distribution
- Within red curve normal distribution
- Within green curve normal distribution

Soft Clustering - Example

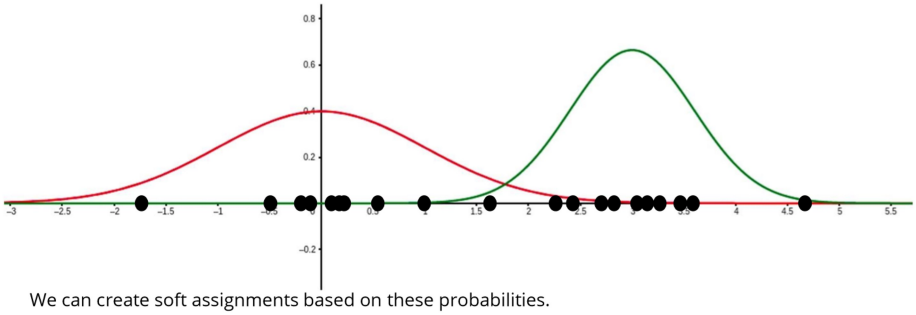
Generate data using $\mathbf{N(\mu_1, \sigma_1)}$ and $\mathbf{N(\mu_2, \sigma_2)}$



For each point we can compute the probability of it being generated from either curve

Soft Clustering - Example

Generate data using $\mathbf{N(\mu_1, \sigma_1)}$ and $\mathbf{N(\mu_2, \sigma_2)}$



We can create soft assignments based on these probabilities.

Mixture Model

X comes from a mixture model with k mixture components if the probability distribution of X is:

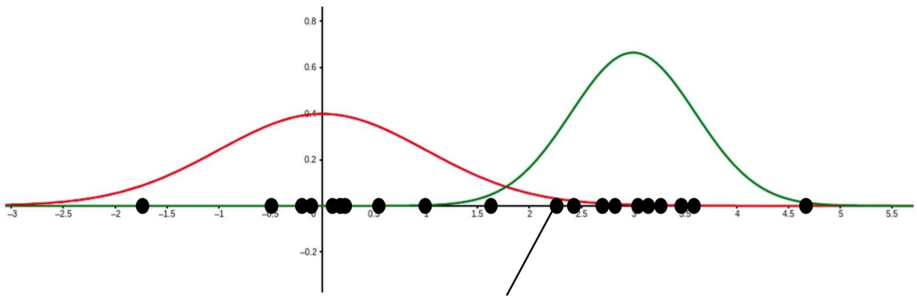
$$P(X = x) = \sum_{j=1}^{k} P(C_j)P(X = x|C_j)$$

Mixture proportion
Represents the probability of belonging to C_j

Probability of seeing x when sampling from C_j

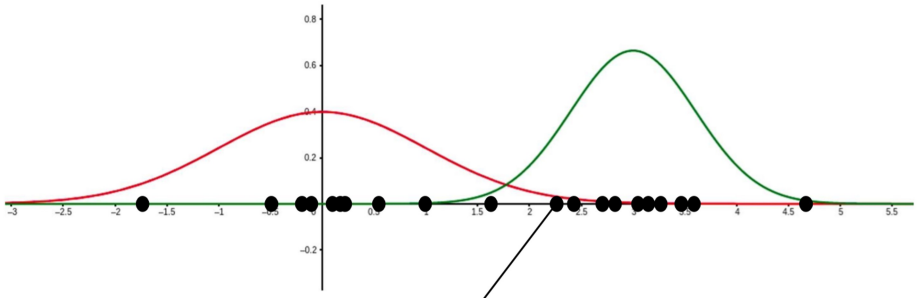
- Distribution of X is the sum of each of the k curves
  - Inherit probability of being within the species, once you are within that species, there is a probability of seeing that point given that you are coming from the green cluster
    - Add the ones from the red cluster too
- This is the mixture proportion
- Probability of seeing x when sampling from Cj
  - Right now we are discussing the discrete case, otherwise the probability that something is the EXACT weight is 0
    - Integrals are needs for the continuous case

Example



What is the probability distribution here?

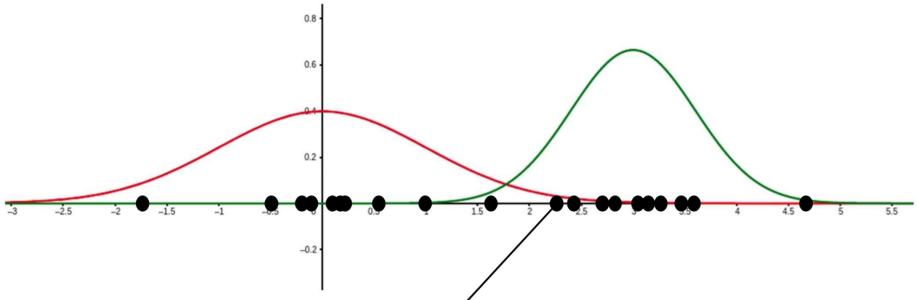
Example



$$P(X = x) = P(C_1)P(X = x|C_1) + P(C_2)P(X = x|C_2)$$

- Probability of being in red cluster, times probability of seeing that weight in the red curve, plus the probability for the same thing on the green curve side too

Example



$$P(X = x) = P(C_1)\frac{1}{\sigma_1\sqrt{2\pi}}e^{-\frac{1}{2}\left(\frac{x-\mu_1}{\sigma_1}\right)^2} + P(C_2)\frac{1}{\sigma_2\sqrt{2\pi}}e^{-\frac{1}{2}\left(\frac{x-\mu_2}{\sigma_2}\right)^2}$$

- Normal distribution!
- with mean mu2 and standard deviation sigma2 (on the right side)

Gaussian Mixture Model

A Gaussian Mixture Model (GMM) is a mixture model where

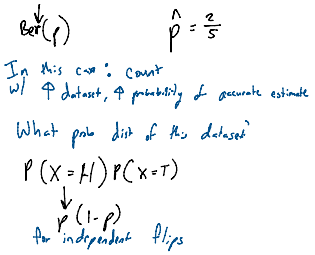
$$P(X = x|C_i) \sim N(\mu, \sigma)$$

- Normal distributions that are the INTRA-cluster distribution

H, T, T, H, T

$\text{Ber}(p)$ $\quad \hat{p} = \frac{2}{5}$

In this case: count
w/ # dataset, # probability of accurate estimate



GMM Clustering

Goal: Find the GMM that maximizes the probability of seeing the data we have.

The probability of seeing the data we saw is (assuming each data point was sampled independently) the product of the probabilities of observing each data point.
Finding the GMM means finding the parameters that uniquely characterize it.
What are these parameters?

$P(C_i)$ & **μ_i** & **σ_i** for all **k** components.

Lets call $\Theta = \{\mu_1, ..., \mu_k, \sigma_1, ..., \sigma_k, P(C_1), ..., P(C_k)\}$

- Step back: address simpler problem
  - Flip a coin
    - Estimate distribution of this coin
      - Why are we doing this?
      - Best: assume we've seen a likely dataset, and estimate parameters that maximize the probability of seeing the distribution we saw
- We are assuming that…
  - ?
  - Each point sampled independently
  - Bernouli random variable
- How many samples do you need to be representative?
  - Depends on distribution, dataset, and what is good enough
- Probability of seeing a given distribution is a Gaussian Mixture distribution
- The probability of seeing this specific dataset is…
- Assuming all of these points are from the same mixture distribution
- Want to find the parameters that uniquely characterize this distribution
  - In normal, need mean and standard deviation
- In this case
  - We have K clusters, K mixture components
  - Probability of being part of one cluster from the other
  - A lot of parameters to estimate
- Traditionally, how would you maximize these parameters?
  - From things in your prereq

    - Take a derivative, 2nd derivative
    - The function to find the maximum is very complicated
- Theta will be the set off all the parameters we need to esimate

GMM Clustering

Goal:

$$\theta^* = \arg\max_{\theta} \prod_{i=1}^{n} \sum_{j=1}^{k} P(C_j) P(X_i \mid C_j)$$

Where $\Theta = \{\mu_1, ..., \mu_k, \sigma_1, ..., \sigma_k, P(C_1), ..., P(C_k)\}$

Joint probability distribution of our data

Assuming our data are independent

- Theta star: set of parameters that maximizes the following function
- This is the joint distribution of n points
- Big pi symbol: is product symbol from i=1 to n
- Does order matter?
    - No, we are assuming order doesn't matter and every event is independent
        - Otherwise you would have to do…
- Multiplying by a constant won't change where the maximum is

GMM Clustering

How do we find the critical points of this function?

Notice: taking the log-transform does not change the critical points

Define:

$$l(\theta) = \log(L(\theta))$$
$$= \sum_{i=1}^{n} \log(\sum_{j=1}^{k} P(C_j)P(X_i \mid C_j))$$

- Normally you apply transforms to make it easier to deal with
- Transforms doesn't change critical points
- Hand wavy math
- If given function is too complicated, maybe a transform will be easier to deal with

GMM Clustering

For $\boldsymbol{\mu} = [\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_k]^T$ and $\boldsymbol{\Sigma} = [\boldsymbol{\Sigma}_1, ..., \boldsymbol{\Sigma}_k]^T$

We can solve

$$\frac{d}{d\Sigma} l(\theta) = 0 \qquad \frac{d}{d\mu} l(\theta) = 0$$

- After transform, can take the partial derivative
- Partial sigma: is the variance

- In past he had this as a multivariate distribution
- These should be lower case sigmas

GMM Clustering

To get

$$\hat{\mu}_j = \frac{\sum_{i=1}^{n} P(C_j|X_i) X_i}{\sum_{i=1}^{n} P(C_j|X_i)}$$

$$\hat{\Sigma}_j = \frac{\sum_{i=1}^{n} P(C_j|X_i)(X_i - \hat{\mu}_j)^T (X_i - \hat{\mu}_j)}{\sum_{i=1}^{n} P(C_j|X_i)}$$

$$\hat{P}(C_j) = \frac{1}{n} \sum_{i=1}^{n} P(C_j|X_i)$$

- mu-j-hat
- Sigma-j
- Probability of coming from particular cluster
- Key:
  - Observe these three equations
  - What terms do you have access to?
  - What terms do you not have access to?
    - What term isn't defined?
      - P(Cj given Xi)
        - Need this term, it's missing

GMM Clustering

Do we have everything we need to solve this?

Still need **$P(C_j \mid X_i)$** (i.e. the probability that X_i was drawn from C_j)

- Can use Bayes rule to solve

GMM Clustering

$$P(C_j|X_i) = \frac{P(X_i|C_j)}{P(X_i)}P(C_j)$$
$$= \frac{P(X_i|C_j)P(C_j)}{\sum_{j=1}^{k} P(C_j)P(X_i|C_j)}$$

Looks like a loop! Seems we need $P(C_j)$ to get $P(C_j \mid X_i)$ and $P(C_j \mid X_i)$ to get $P(C_j)$

- Can we compute this now?
- We need the probability of P(Xi given Cj) and the probability of Cj
- Now we are stuck in this loop of needing estimators in order to estimate those parameters

Expectation Maximization Algorithm

1. Start with random θ
2. Compute $P(C_j \mid X_i)$ for all X_i by using θ
3. Compute / Update θ from $P(C_j \mid X_i)$
4. Repeat 2 & 3 until convergence

- Very much like kmeans at a high level
- Start with something random, and slowly get to a better and better estimate
- Use value from step 2, can update the other parameters
- Computationally, you may want to stop the algorithm at some point
  - Ideally when the parameters aren't changing anymore
- What do you check against?
  - You are assuming that your sample is representative, all you can do is best guess estimates
    - Training and test sets

Demo

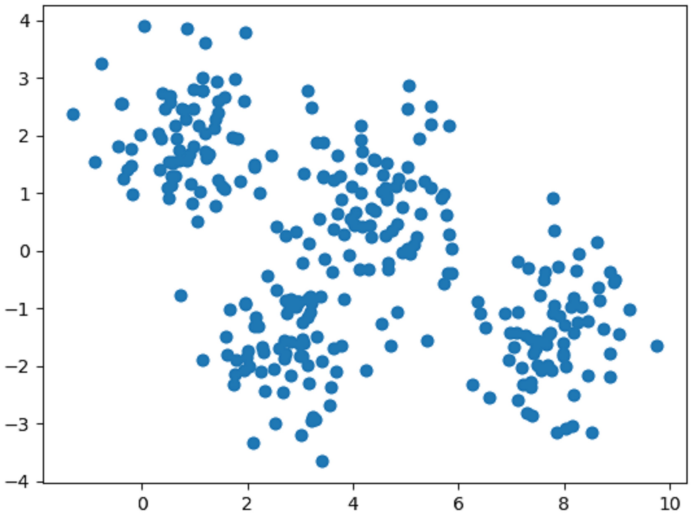
- See gmm.py in CS506 git
  - Feel free to reuse it for HW1
- GMM, at the end: does it put a point into one of the other?
  - Just use GMM predict function, it will give you the cluster with the highest probability

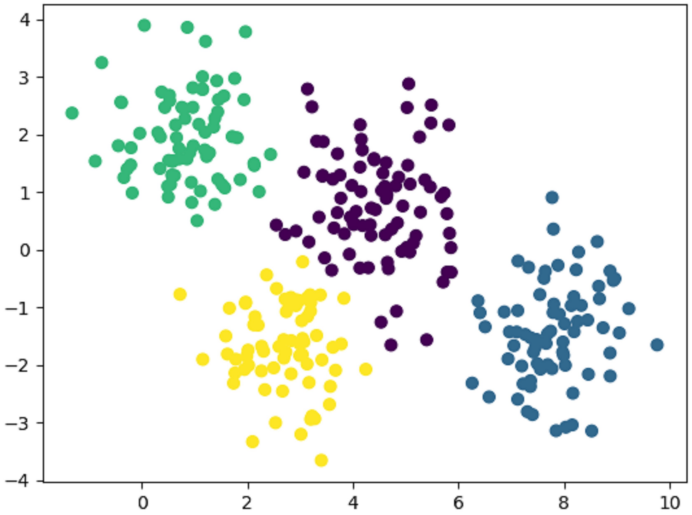
File - C:\Users\Wolfs\PycharmProjects\lec20211004\gmm.py

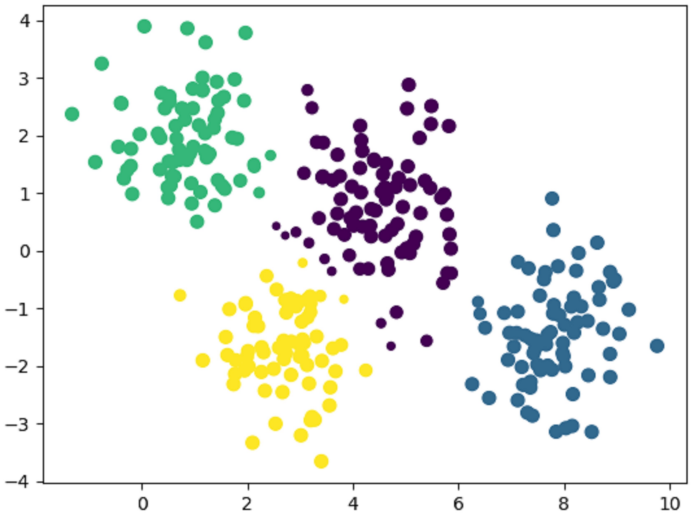
```python
import matplotlib.pyplot as plt

from sklearn.datasets import make_blobs
from sklearn.mixture import GaussianMixture

# Generate some data
X, y_true = make_blobs(n_samples=300, centers=4,
                       cluster_std=.8, random_state=0)
X = X[:, ::-1] # flip axes for better plotting

plt.scatter(X[:, 0], X[:, 1], s=40, cmap='viridis')
plt.show()

gmm = GaussianMixture(n_components=4).fit(X)
labels = gmm.predict(X)
plt.scatter(X[:, 0], X[:, 1], c=labels, s=40, cmap='viridis')
plt.show()

probs = gmm.predict_proba(X)
print(probs[:5].round(3))

size = 50 * probs.max(1) ** 2  # square emphasizes differences
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis', s=size)
plt.show()
```



```
[[0.545 0.    0.133 0.322]
 [0.    1.    0.    0.   ]
 [1.    0.    0.    0.   ]
 [0.    1.    0.    0.   ]
 [0.196 0.    0.782 0.022]]

Process finished with exit code 0
```


4
3
2
1
0
−1
−2
−3
−4
0
2
4
6
8
10


4
3
2
1
0
−1
−2
−3
−4
0
2
4
6
8
10


4
3
2
1
0
−1
−2
−3
−4
0
2
4
6
8
10

20211004 Clustering Aggregation

Monday, October 4, 2021 3:14 PM

Clustering Aggregation

- Last clustering topic before moving on

Clustering Aggregation

Some terminology:

Clustering: A group of clusters output by a clustering algorithm

Cluster: A group of points

-

Clustering Aggregation

Goals:

1. Compare clusterings
2. Combine the information from multiple clusterings to create a new clustering

---

- Want to meaningfully compare the output between different clustering algorithms
- Is there a super clustering method that will be good at everything
  - Based on which clustering algorithm is good at which aspects?

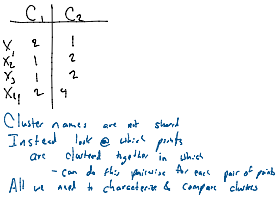
Comparing Clusterings

The many methods / cost functions make comparing clusterings difficult.

Need to compare clusterings by looking at their assignment of points to clusters.

If many points were assigned to the same clusters in both clustering C and clustering P, then C and P should have a small distance.

But identifying which clusters are the same in P and C is not easy. Why?



- Kmeans optimizes for particular cost function
  - May not be appropriate for a different clustering method
  - But maybe that is not the metric you want to use on a different clustering method
  - Need to compare clusterings by looking at their assignment of points to clusters

Comparing Clusterings

Clearly these clusterings are the same. Yet the assignments / labels are inconsistent.

Comparing Clusterings

Asking “is x in cluster “red”” in the left clustering is equivalent to asking “is x in cluster “yellow”” on the right clustering but we cannot know this conversion up front unless there is a known set of conventions.

Comparing Clusterings

Let’s not limit ourselves with such a set of convention and instead ask a different question:

Are x and y clustered together in both P and C?

Disagreement Distance

Given 2 clusterings P and C

$$D(P, C) = \sum_{x,y} \mathbb{I}_{P,C}(x, y)$$

where

$$\mathbb{I}_{P,C}(x, y) = \begin{cases} 1 & \text{if P \& C disagree on which clusters x \& y belong to} \\ 0 \end{cases}$$

- Do the clustering methods cluster the points in the same way?
  - For each pair of points, they need to agree on whether or not they are clustered together
- Disagreement distance between clustering P and clustering C
- Indicator function
- Distance function
  - If clustering are very far from each other
    - There will be a large number of disagreements
  - Trick is showing that triangle inequality holds

Disagreement Distance

	P	C
x_1	1	1
x_2	1	2
x_3	2	1
x_4	3	3
x_5	3	4

What is the disagreement distance between P and C?

- Need to look through every pair of points and count up disagreements

Disagreement Distance

	P	C
x_1	1	1
x_2	1	2
x_3	2	1
x_4	3	3
x_5	3	4

x_2	x_1	1
x_3	x_1	1
x_4	x_1	0
x_5	x_1	0
x_3	x_2	0
x_4	x_2	0
x_5	x_2	0
x_4	x_3	0
x_5	x_3	0
x_4	x_5	1

- Generate list of all pairs of points and list whether they agree or disagree
- Are they similar or are they dissimilar
- This is really expensive
- All of these techniques are very expensive, and are not very practical
- If run time stops being an issue…

Disagreement Distance

Is D(P, C) a distance function?

1. D(C, P) = 0 iff C = P
2. D(C, P) = D(P, C)
3. Triangle Inequality:

$$\mathbb{I}_{C_1,C_3}(x,y) \leq \mathbb{I}_{C_1,C_2}(x,y) + \mathbb{I}_{C_2,C_3}(x,y)$$

Since $\mathbf{I_{C,P}}$ can only be 0 or 1, the above can only be violated if

$\mathbf{I_{x,y}(C_1,C_3) = 1}$, $\mathbf{I_{x,y}(C_1,C_2) = 0}$, $\mathbf{I_{x,y}(C_2,C_3) = 0}$ is this possible?

- The first two are easy to show
- The triangle inequality:
  - If the two clustering methods agree on whether x and y should be together…
- We like distance functions
  - You can work with them and they are intuitive
  - Triangle inequality
    - You should not have a shorter path possible by going through an intermediary
- Example of not a distance function?
- This doesn't tell us which is better, just shows the descrepancy
- If you have different parameters to tune
  - If you apply DBScan with a bunch of different parameters you want to know how different the results are

Aggregate Clustering

Goal: From a set of clusterings $C_1, ..., C_m$, generate a clustering C^* that minimizes:

$$\sum_{i=1}^{m} D(C^*, C_i)$$

The problem is equivalent to clustering categorical data

- Goal is to apply a clustering C* to minimize the disagreement distances
- Want to minimize sum of disagreement distances
- Will give us a C*
- Minimize
- If you have categorical data:
  - Your categorical data is ITSELF a clustering of your dataset

Aggregate Clustering

	City	Profession	Nationality
x_1	NY	Doctor	US
x_2	NY	Teacher	French
x_3	Boston	Lawyer	Canada
x_4	Boston	Doctor	US
x_5	LA	Lawyer	Canda
x_6	LA	Actor	French

- Categorical data:
  - Here we have three different clustering of our data
  - We want to aggregate it into one set of clusters aggregated from this data

Aggregate Clustering

Benefits:

1. Can identify the best number of clusters (optimization function does not make any assumptions on the number of clusters)
2. Can handle / detect outliers (points where there is no consensus)

2: Aggregate clustering is only better if you aren't

1. Can identify the best number of clusters (optimization function does not make any assumptions on the number of clusters)
2. Can handle / detect outliers (points where there is no consensus)
3. Improve robustness of the clustering algorithms - combining clusterings can produce a better result
4. Privacy preserving clustering (can compute aggregate clustering without sharing the data, need only share the assignments)

?: Aggregate clustering is only better if you aren't including really inappropriate clustering methods for your data?

-

Aggregate Clustering

But... The problem is NP-Hard.

Often use approximations and heuristics to solve this problem.

What about the majority rule?

This only works **if** it produces a clustering

Possible to have a majority saying:

1. x_1 & x_2 together
2. x_2 & x_3 together
3. x_1 & x_3 separate

NP-Hard def:

- Not only did all possible pairs of points, then do it multiple times?
- You have to work with NP-Hard problems
  - Sometimes there isn't a better solution
- Majority rule:
- This only works if it produces a clustering
- If one algorithm has 3 clusters, and the other assumes 4 clusters
- Possible to get a result that is worse from aggregating the tools
  - It’s a little tricky, need more time to address this question
  - We don't really know what it means for it to be good or not
  - There are no guarantees

Aggregate Clustering