Agglomerative Clustering

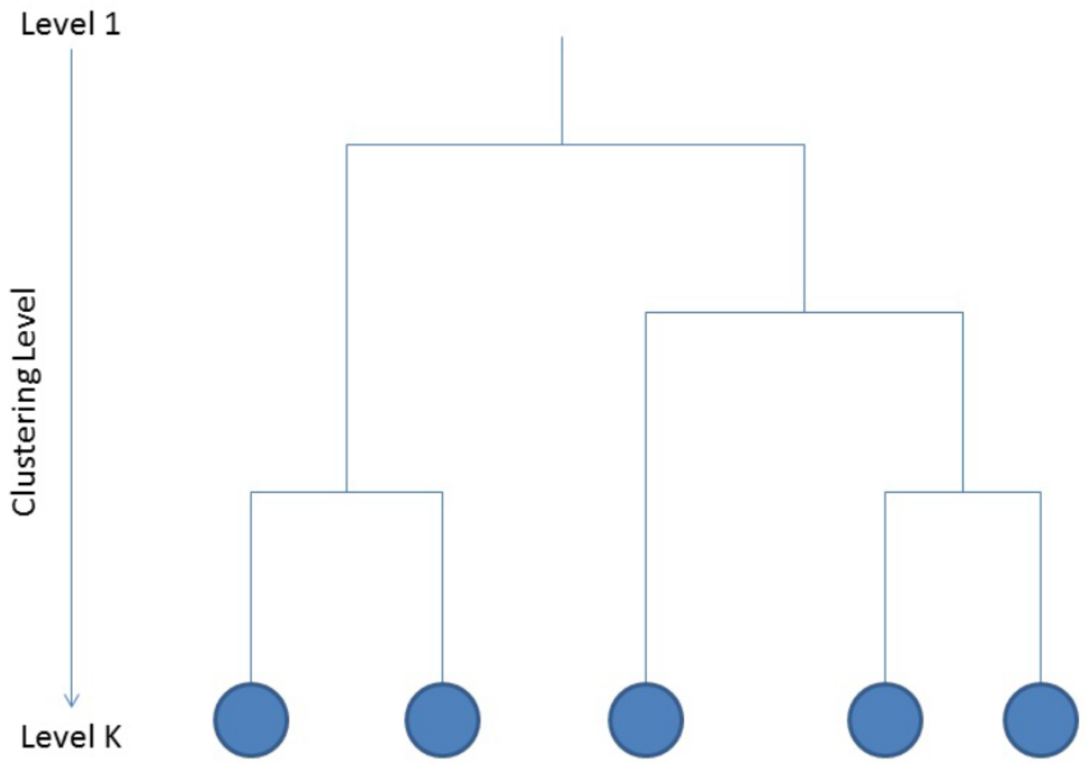
Week	THURS. Week 9
Assignment Due	
Assignment Done	☐
Due Date	
Notes Done	✓

Class Notes

Agglomerative Clustering

One weakness of k-means and mixture models is that we need to know the number of clusters *ahead of time*.

- It is more often than not that we do not know how many clusters that we want ahead of time, so we want to do data exploration.
- If we don't know this, one idea is to build a **hierarchical agglomerative clustering tree (HAC)** s.t. we have a hierarchy of clusters, from one cluster, up to N clusters.
    - This can be done either bottom up, or top down.



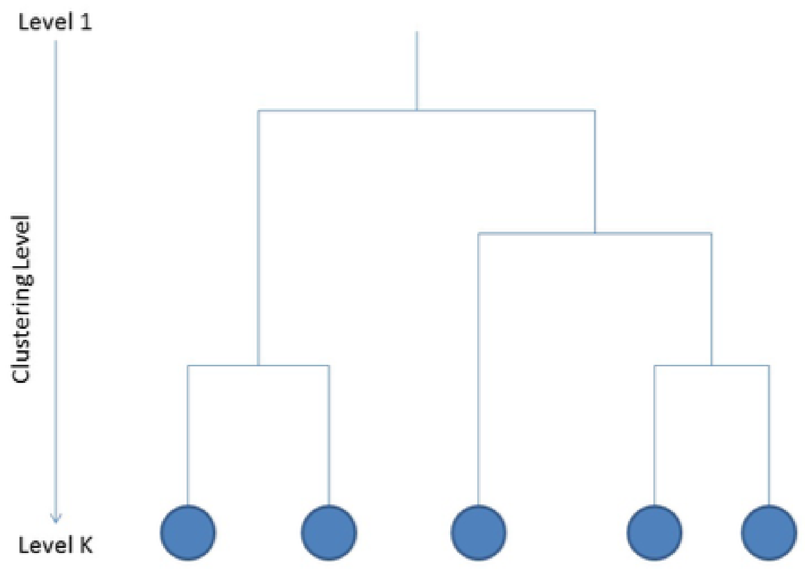
    - Depending on how we build it, if we have N observations, we'll have N different clustering levels.
        - At level N, every observation is its own cluster.
            - At the level before that, we have $N-1$ clusters, and then $N-2$ clusters, until we have only one cluster.
        - With this tree, we have some sort of clustering for values of k from 1 to N.
- We can do analytics on these individual clusterings to then decide on a value of k.

Top-Down Approach

With this approach, initially everything is a part of a *single cluster*, which we may call level 1.

- To go to cluster level 2, we must break this cluster into 2 cluster.
- To go to cluster level 2, we take one of the clusters and break it down.

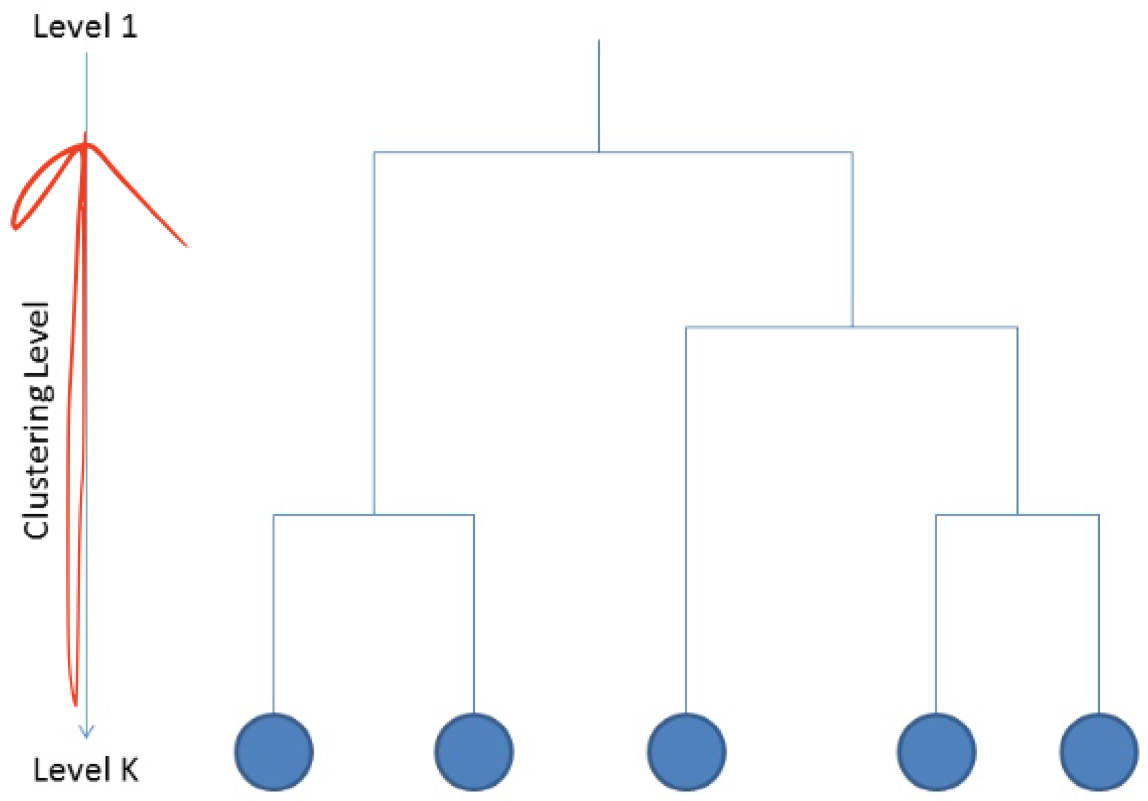
- We can consider this to be a divisive/segmentation type of approach, continuing until each cluster has only one observation in it (called a singleton).



Bottom-Up Approach

The opposite is a bottom-up approach, which is a merging/grouping type of approach.

- We begin with level N with $N$$ clusters, where each observation is its own cluster.
  - We find two clusters to merge and merge them together, giving us $N-1$ clusters.
    - Among these, we continue to identify two to merge together, until we have a single cluster.



Ignore existing above blue arrow, the red arrow is meant to be overwriting it.

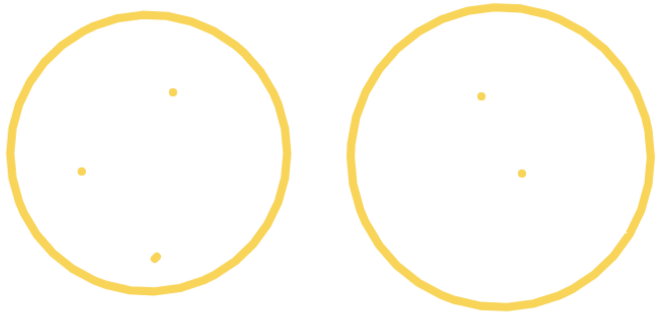
You can do it either way, but typically, for conceptual and mathematical reasons, the bottom-up approach is easier to do than the decisive/segmenting approach.

Closest Pair of Clusters

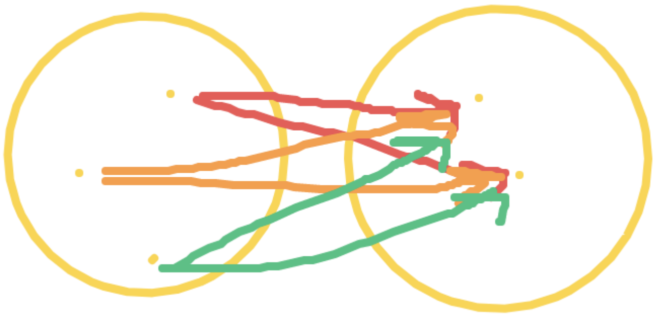
For the bottoms up approach, at any given level, we need to decide whcih two clusters to merge:

- We'll do this by determining which two clusters are most *similar* (if using similarity as our measurement) or are *closest* (if using distance as our measurement).
    - Depending on various factors, in these hierarchical agglomerative algorithms, there are typically at least three different options when deciding on a measurement on the similarity of two clusters:
        - Remember: A cluster can have a lot of observations.
        - When wanting to compute similarity between the two clusters, we can compute the similarity *between* the clusters.

- For the actual similarity *of* the cluster, we can choose/use the largest similarity.
    - Ex: Imagine having two clusters:



        - Say we want a measurement of similarity between the two clusters:
            - We can compute the similarity of each observation in one cluster to each observation in the other cluster.



                - Using these similarities between the elements of the clusters, one option called **single link** says to use the highest similarity of the elements between the clusters.
                    - As long as there's one cluster in the sample that is very similar, we will consider the clusters themselves to be very similar.
                        - This can almost generate a chaining effect, elongated clusters since as long as there are two very similar observations, we can say these clusters are similar.

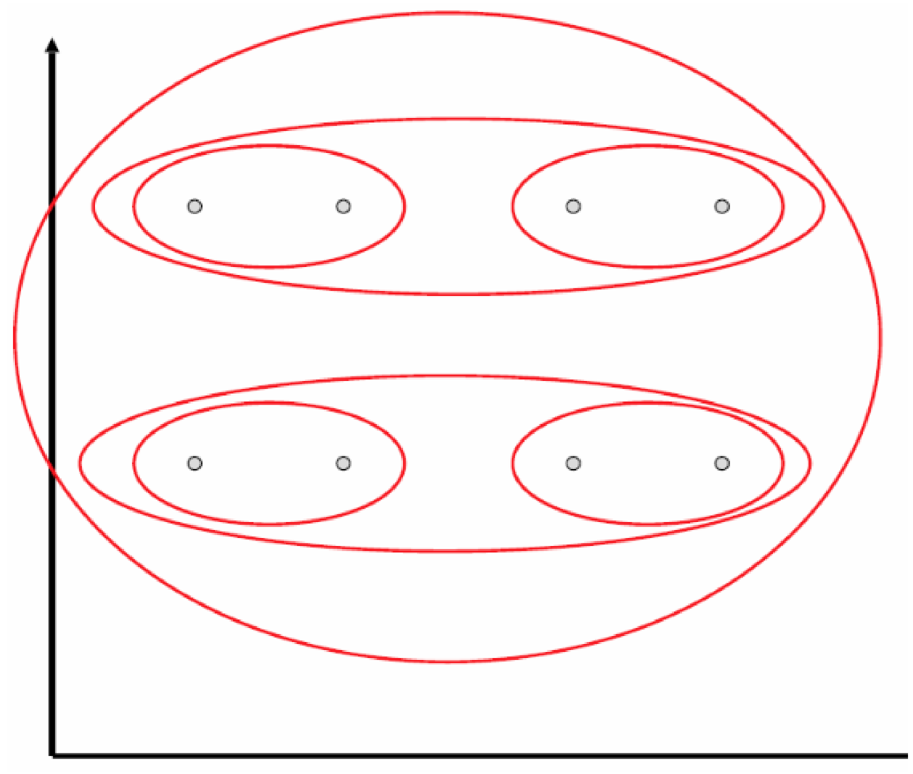
- We can conversely go the other way around from **single link** and protect against merging clusters that have something that is very dissimilar, referred to as **complete link**.
    - As a result, for measuring similarity between two clusters, we'll consider the lowest similarity between elements in the clusters we're considering.
- Probably, the logical starting point, unless the application dictates to avoid merging very dissimilar observations or a desire to merge 2 clusters if they have something very similar, the default go-to is the **average link**.
    - This computes the similarities between elements in each observation and then averaging them, using this as our measurement of similarity.

Single Linkage Similarity

Single linkage similarity - use the similarity of the **most similar** observations between the clusters:

$$sim(C_i, C_j) = \max_{x \in C_i, y \in C_j} sim(\boldsymbol{x}, \boldsymbol{y})$$

- If we have several elements in cluster a and b, compute the similarity between the elements in each cluster, then we say the similarity of the clusters is the maximum of these intercluster similarities.
    - We may choose this if our application wants to merge things that ARE very similar regardless of what else may be going on between the clusters.
- This can result in *elongated/chained* clusters.



Complete Linkage Similarity

Complete linkage similarity - use the similarity of the **least similar** observations between the two clusters:

$$\mathrm{sim}(C_i, C_j) = \min_{x \in C_i, y \in C_j} \mathrm{sim}\,(\boldsymbol{x}, \boldsymbol{y})$$

- The minimum of intercluster similarities is chosen as the measure of similarity between clusters.
- Again, this is useful if we want to avoid merging clusters that have something very dissimilar.

Average Linkage Similarity

Average linkage similarity - a compromise between single and complete link

- Average over all pairs *between* the two original clusters

$$sim(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{x \in C_i} \sum_{y \in C_j} sim(\boldsymbol{x}, \boldsymbol{y})$$

- We iterate through each obseravtion from each cluster, computing the intercluster similarities and averaging them.
    - $|C_i|$ - number of elements in cluster i (same for j)

For outliers, they are likely the last elements to be merged in a cluster, since they're dissimilar from elements in other clusters.

HAC Pseudocode

1. Start with each observation as its own cluster.
2. Compute similarity between each cluster, using one of the previous linkage measurements, and create a **sorted** list (sorted by similarity, descending) of (similarity, cluster, cluster) pairs of the members that we computed the similarities for:

$$L = \{(sim(C_1, C_2), C_1, C_2), (sim(C_1, C_3), C_1, C_3), \ldots, \}$$

3. While the number of elements in L is greater than one (more to merge):
    a. Grab the head of L. This contains the two most similar clusters C_a and C_b.
    b. Merge the clusters (C_a, C_b), creating a new cluster $C' = C_a \cup C_b$
    c. Iterate through list and remove all pairs containing C_a or C_b from the sorted list, L.
    d. Compute the similarity between C' and all remaining clusters, add these new pairs to the list, and re-sort the list.

- We continue this process until everything is a *single* cluster.

Choosing the Clustering Level

If we know how many clusters we want, we could have done a bottoms-up build at that level k and see the clustering that we arrived at with merging process, possibly using reference vectors to bootstrap the k-means process.

- Or we could continue merging and seeing what clustering looks like at *other levels* of k.
- If we *DO NOT* know k beforehand, it'd be useful to see if we can use HAC to find a good value of k.
  - We'll use some analytics in our clustering process to determine when in merging at level k did something substantial happen and utilize this fact to decide on a good value of k.
- We can either look at our hierarchy tree and look at the cluster at level k and use that or use k to run a k-means or Gaussian mixture model.

Clustering Quality

Intra-Cluster Distance

At the start of our coverage of clusters, we noted that good clustering should result in clusters that have *large* ***intra-cluster similarity*** and *low* ***inter-cluster similarity***.

- We need to provide some way to measure the quality of a given clustering, possibly a weighted average intra-cluster distance.
  - For a singular cluster:
    - For a given cluster i and a chosen distance/similarity function d, the following computes the **average pairwise *intra-cluster* distance** G_i

$$G_i = \frac{\sum_{x \in C_i} \sum_{y \in C_i, x \neq y} d(x, y)}{|C_i|(|C_i| - 1)}$$

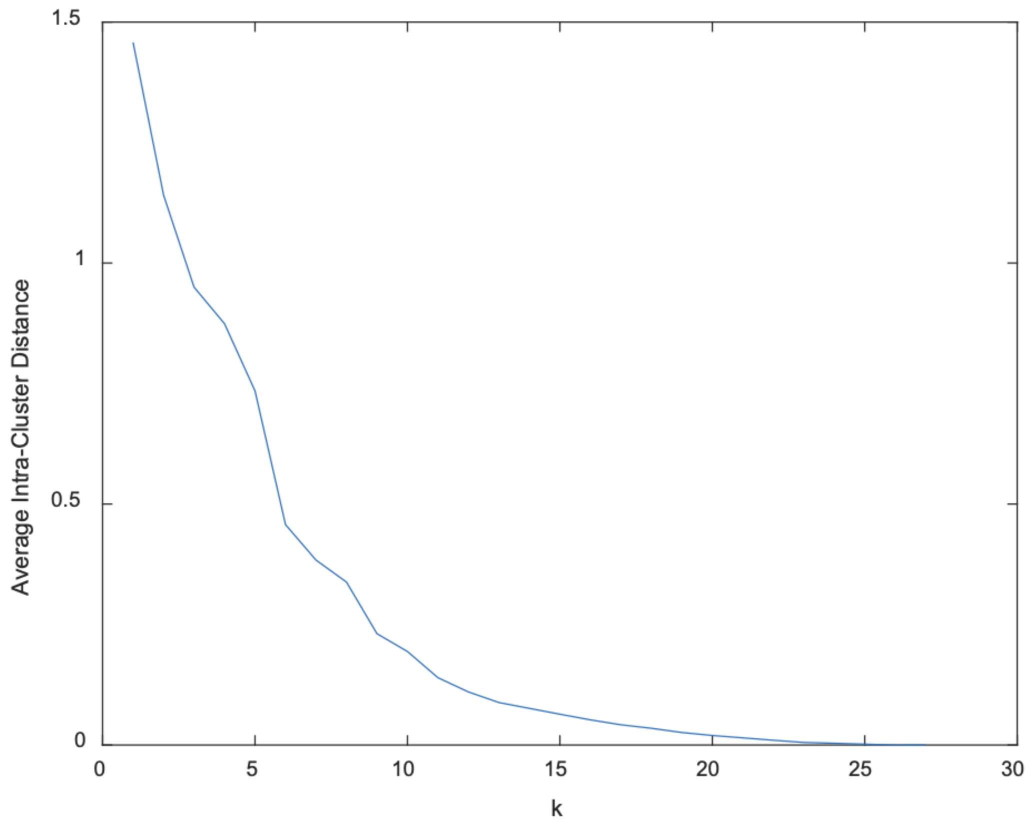
Note: The denominator is just $|C_i|$ choose 2 (permutation)

- We can go over all instances within our cluster i, compute pairwise distances/simliarites, and average them accordingly.
  - We sum over each cluster and divide by the number of members in the cluster multiplied by the number of members minus 1.

- <u>For all clusters:</u>
    - For an overall <u>**average insta-cluster distance/similarity**</u> over ALL of the clusters for clustering level j is then:
        - We iterate over all of our clusters i and for each compute the intra-cluster similarity/distance, then weigh it by the number of observations in the cluster.
- One idea to analytically look at what would be a good cluster level, k, is to look at how intra-cluster distance varied as a function of the number of clusters!

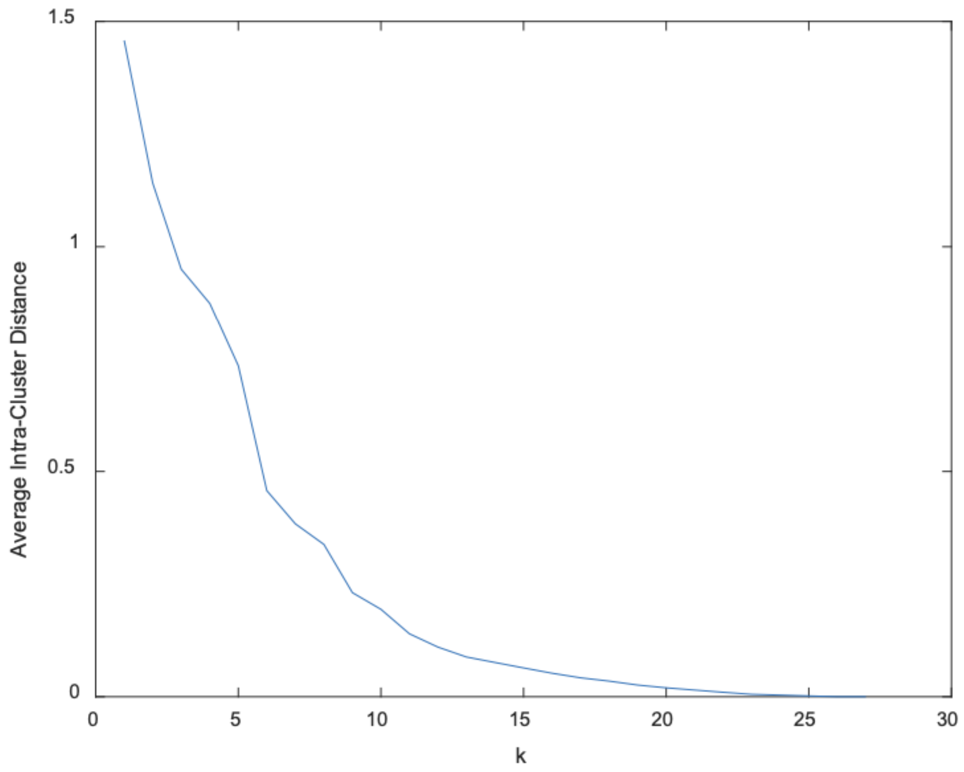
Graph-Based Approaches

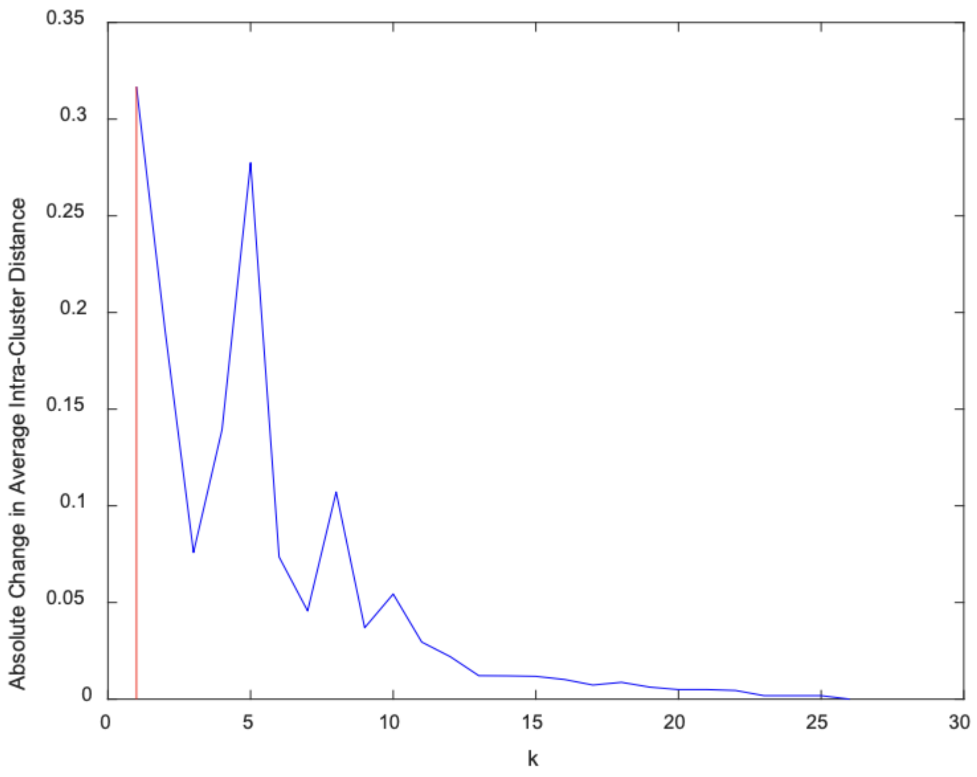
We can see on the graph in the slides that the graph displays the weighted average intra-cluster distance as a function of different k values.



There are 30 observations in this example.

- For building up our agglomerative hierarchical clustering tree, we’re look at this graph from the right to the left.
    - Initially, all 25 osbervaitons are their own clusters.
        - With one observation in a cluster, then the intra-cluster distance is naturally 0.
- As we merge clusters together, the average intracluster distance increases.
- We know we want intra-cluster distance to be small and inter-cluster distance is large.
    - Perhaps we decide to use the clustering level that has the overall minimum for intra-cluster distance.
        - However, this wouldn’t be useful as isn’t useful as this is the case when each observation is its own cluster.
    - Instead of choosing the overall minimum, we can choose where the intra-cluster distance *changes* the quickest, which is when the slope is at a maximum.
        - At this point, by doing some sort of merging, we had a large change in the intra-cluster similarity/distance occur.



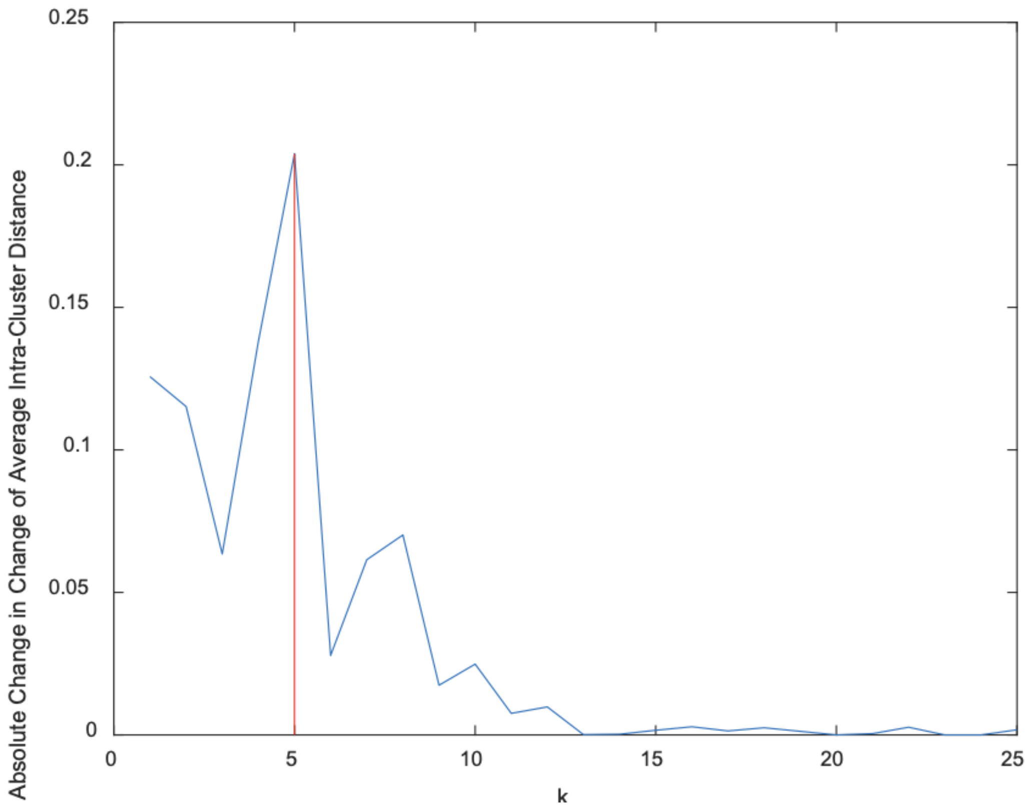


- We can see in the second bottom graph the slope (first derivative) of the average intra-cluster distance.

- The slope at location W_j is:

$$W_j' = \frac{(W_{j+1} - W_{j-1})}{2}$$

  - The intrinsic issue with this slope graph is that at the ends of the tree (going from 2 clusters to 1 or 3 to 2), we're being forced to merge clusters that we likely should not be merging.
    - As a result, we get jumps in the change in intra-cluster distance/similarity at these ends while building our tree, as we see at the last few mergings from cluster 2 to 1.
- Instead, we want to find the *knee* in the graph, where the change in the rate of change (slope) is happening the fastest.
  - We can see this below in a graph of the second derivative.



      - This would be known as the **curvature**, essentially the 2nd derivative.
        - In our graph, we see maximum curvature is at $k = 5$. The change in the rate of change was the *highest* at this point.
          - This may indicate that we're merging clusters that we possibly shouldn't.
    - We can perform this graphing by calculating the intra-cluster and inter-cluster similairity values while building our HAC.
    - Overall, as a strategy of figuring out k, we can calculate the intra-cluster similarity/distance during the construction of our HAC and look for the max of the curvature in the average intra-cluster similarity, which we can use as a guess for k.
      - Upon deciding on a k value, we can look at our HAC at that level and use that cluster set as our choice (final answer) or use this process of building the HAC to inform other clustering algorithms as k-means.

External Criteria for Clustering Quality

In addition to evaluating our intra-clustering similarity/clustering, it'd be nice to have an *external* way to know how our algorthm is doing despite the lack of supervision in our data.

- While having our algorithm create its k clustering, we can have some expert in parallel perform the clustering themselves.
  - We can use this "expert"'s grouping to compare to the grouping done by the algorithm → asks how well did our algorithm match up with the expert's grouping?
    - Although we're using someone's evaluation for this evaluation process, we did not use it in the actual classification process, so clustering is still an unsupervised learning algorithm.
- There are several ways that external criteria are computed:
  - The measure of this comparision is referred to as **purity**, which is the percentage of elements of a cluster that have the *same* external label.

        - If a cluster is completely pure, all the elements in it have the same exact label.
        - We ask what the most common label is and divide it by the number of elements in that cluster that it is a part of.
            - Since we want to do this for all clusters, we can again do a **weighted average purity**, taking each clusters' purity and weight it by the percentage of observations in that cluster.
    - Mathematical formulation of purity:

Let N_{ij} be the number of instances of (supervised) label j within cluster C_i

The purity of cluster C_i is then defined as

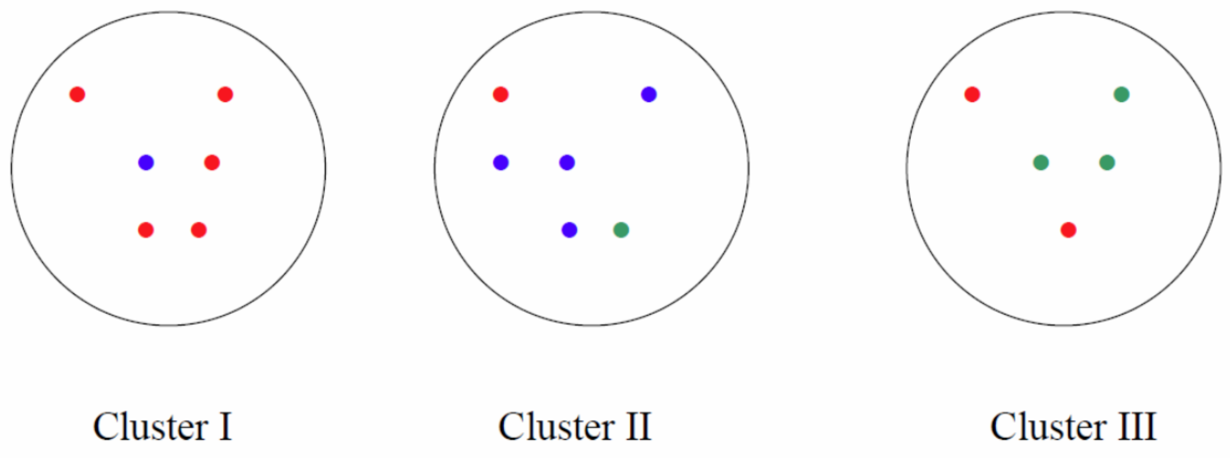
$$Purity(C_i) = \frac{1}{|C_i|} \max_j N_{ij}$$

Then we can define the average purity of this clustering as

$$Purity = \frac{1}{N} \sum_{i=1}^{k} |C_i| Purity(C_i)$$

Example

Say our algorithm made 3 clusters → cluster 1 has 6 members, cluster 2 has 6 members, cluster 3 has 5 members



Cluster I: Purity = 1/6 (max(5, 1, 0)) = 5/6

Cluster II: Purity = 1/6 (max(1, 4, 1)) = 4/6

Cluster III: Purity = 1/5 (max(2, 0, 3)) = 3/5

$$\text{Total Purity} = \frac{1}{17}\left(6 * \frac{5}{6} + 6 * \frac{4}{6} + 5 * \frac{3}{5}\right) = \frac{12}{17} \approx 70\%$$

We had an expert group these observations into 3 groups as well by color.

- To compute purity in first cluster, we see that red is the common external label, therefore the purity of the cluster if $\frac{5}{6}$.
- In calculating the purity of the other two clusters with this measurement, we can determine that the second

Purity is not perfect by any means → we would always attain maximal purity if we had n clusters

- With n clusters, each cluster has 1 and only 1 member in it, so it would of course have perfect purity.

- It is not always fair to use purity to help us decide what the value of k should be, but it could be useful to compare different algorithms that are using the same values of k .
    - In this case, whichever algorithm has the higherst purity says this system clusters stuff closest to how this expert person did.

There are various other ways that we can use external opinions to help determine cluster quality.

- Other measurements include:
    - **Silhouette**
        - https://en.wikipedia.org/wiki/Silhouette_(clustering)
    - **V-Measure**
        - https://www.aclweb.org/anthology/D07-1043