**CSE/ECE-471: Statistical Methods in AI** **Spring 2020**

# Gaussian Mixture Models (continued), Hierarchical Clustering

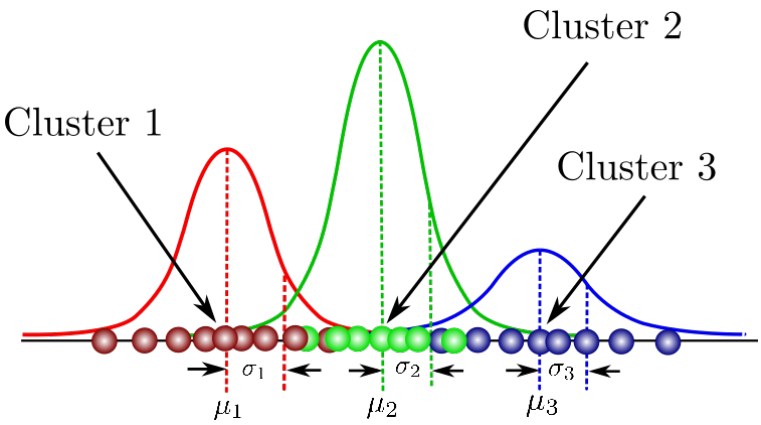
*Prepared by: 2019201015, 2019201080, 2019201097*

One important characteristic of K-means is that it is a hard clustering method, which means that it will associate each point to one and only one cluster. A limitation to this approach is that there is no uncertainty measure or probability that tells us how much a data point is associated with a specific cluster. So what about using a soft clustering instead of a hard one? This is exactly what Gaussian Mixture Models, or simply GMMs, attempt to do. Let's now discuss this method further.

## 1 GMM variables Definitions

A Gaussian Mixture is a function that is comprised of several Gaussians, each identified by $k \in \{1, ..., K\}$, where K is the number of clusters of our data-set. Each Gaussian k in the mixture is comprised of the following parameters:

1. A mean $\mu$ that defines its centre.
2. A covariance $\Sigma$ that defines its width. This would be equivalent to the dimensions of an ellipsoid in a multivariate scenario.
3. A mixing probability $\pi$ that defines how big or small the Gaussian function will be.




Here, we can see that there are three Gaussian functions, hence K = 3. Each Gaussian explains the data contained in each of the three clusters available.

The mixing coefficients are themselves probabilities and must meet this condition:

$$\sum_{k=1}^{K} \pi_k = 1 \qquad (1)$$

Now how do we determine the optimal values for these parameters? To achieve this we must ensure that each Gaussian fits the data points belonging to each cluster. This is exactly what maximum likelihood does.

The multivariate Gaussian:

$$\mathcal{N}(\mathbf{x}|\mu,\Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)\right)$$

Where x represents our data points, D is the number of dimensions of each data point. $\mu$ and $\Sigma$ are the mean and covariance, respectively. If we have a dataset comprised of N = 1000 three-dimensional points (D = 3), then x will be a 1000  3 matrix. $\mu$ will be a 1  3 vector, and $\Sigma$ will be a 3  3 matrix. For later purposes,

we will also find it useful to take the log of this equation, which is given by:

$$\ln\mathcal{N}(\mathbf{x}|\mu,\Sigma) = -\frac{D}{2}\ln 2\pi - \frac{1}{2}\ln\Sigma - \frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu) \qquad (2)$$

If we differentiate this equation with respect to the mean and covariance and then equate it to zero, then we will be able to find the optimal values for these parameters, and the solutions will correspond to the Maximum Likelihood Estimates (MLE) for this setting. However, because we are dealing with not just one, but many Gaussians, things will get a bit complicated when time comes for us to find the parameters for the whole mixture. In this regard, we will need to introduce some additional aspects that we discuss in the next section.

## 2 Initial derivations

We are now going to introduce some additional notation. We'll try to keep the notation as clean as possible for better understanding of the derivations. First, let's suppose we want to know what is probability that a data point $x_n$ comes from Gaussian k.

We can express this as:

$$p(z_{nk} = 1|\mathbf{x}_n)$$

Which reads "given a data point x, what is the probability it came from Gaussian k?" In this case, z is a latent variable that takes only two possible values. It is one when x came from Gaussian k, and zero otherwise. Actually, we don't get to see this z variable in reality, but knowing its probability of occurrence will be useful in helping us determine the Gaussian mixture parameters, as we discuss later.

Likewise, we can state the following:

$$\pi_k = p(z_k = 1)$$

Which means that the overall probability of observing a point that comes from Gaussian k is actually equivalent to the mixing coefficient for that Gaussian. This makes sense, because the bigger the Gaussian is, the higher we would expect this probability to be. Now let z be the set of all possible latent variables z, hence:

$$\mathbf{z} = \{z_1, ..., z_K\}$$

We know beforehand that each z occurs independently of others and that they can only take the value of one when k is equal to the cluster the point comes from. Therefore:

$$p(\mathbf{z}) = p(z_1 = 1)^{z_1} p(z_2 = 1)^{z_2} ... p(z_K = 1)^{z_K} = \prod_{k=1}^{K} \pi_k^{z_k}$$

Now, what about finding the probability of observing our data given that it came from Gaussian k? Turns out to be that it is actually the Gaussian function itself! Following the same logic we used to define p(z), we can state:

$$p(\mathbf{x}_n|\mathbf{z}) = \prod_{k=1}^{K} \mathcal{N}(\mathbf{x}_n|\mu_k, \mathbf{\Sigma}_k)^{z_k}$$

Ok, now you may be asking, why are we doing all this? Remember our initial aim was to determine what the probability of z given our observation x? Well, it turns out to be that the equations we have just derived, along with the Bayes rule, will help us determine this probability. From the product rule of probabilities, we know that

$$p(\mathbf{x}_n, \mathbf{z}) = p(\mathbf{x}_n|\mathbf{z})p(\mathbf{z})$$

It seems to be that now we are getting somewhere. The operands on the right are what we have just found. Perhaps some of you may be anticipating that we are going to use the Bayes rule to get the probability we eventually need. However, first we will need p($x_n$), not p($x_n$, z). So how do we get rid of z here? Marginalization! We just need to sum up the terms on z, hence

$$p(\mathbf{x}_n) = \sum_{k=1}^{K} p(\mathbf{x}_n|\mathbf{z})p(\mathbf{z}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k)$$

This is the equation that defines a Gaussian Mixture, and you can clearly see that it depends on all parameters that we mentioned previously! To determine the optimal values for these we need to determine the maximum likelihood of the model. We can find likelihood as the joint probability of all observations $x_n$, defined by:

$$p(\mathbf{X}) = \prod_{n=1}^{N} p(\mathbf{x}_n) = \prod_{n=1}^{N} \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k)$$

Like we did for the original Gaussian density function, let's apply the log to each side of the equation:

$$\ln p(\mathbf{X}) = \sum_{n=1}^{N} \ln \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k) \qquad (3)$$

Now in order to find the optimal parameters for the Gaussian mixture, all we have to do is to differentiate this equation with respect to the parameters and we are done, right? But we have an issue here. We can see that there is a logarithm that is affecting the second summation. Calculating the derivative of this expression and then solving for the parameters is going to be very hard.

What can we do? Well, we need to use an iterative method to estimate the parameters. But first, remember we were supposed to find the probability of z given x? Well, let's do that since at this point we already have everything in place to define what this probability will look like.
From Bayes rule, we know that

$$p(z_k = 1|\mathbf{x}_n) = \frac{p(\mathbf{x}_n|z_k = 1)p(z_k = 1)}{\sum_{j=1}^{K} p(\mathbf{x}_n|z_j = 1)p(z_j = 1)}$$

From our earlier derivations we learned that:

$$p(\mathbf{x}_n, \mathbf{z}) = p(\mathbf{x}_n|\mathbf{z})p(\mathbf{z})$$

So let's now replace these in the previous equation:

$$p(z_k = 1|\mathbf{x}_n) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\mu_j, \Sigma_j)} = \gamma(z_{nk}) \qquad (4)$$

And this is what we have been looking for! Moving forward we are going to see this expression a lot. Next we will continue our discussion with a method that will help us easily determine the parameters for the Gaussian mixture.

# 3 Expectation — Maximization algorithm

Well, at this point we have derived some expressions for the probabilities that we will find useful in determining the parameters of our model. However, in the past section we could see that simply evaluating (3) to find such parameters would prove to be very hard. Fortunately, there is an iterative method we can use to achieve this purpose. It is called the Expectation — Maximization, or simply EM algorithm. It is widely used for optimization problems where the objective function has complexities such as the one we have just encountered for the GMM case.
Let the parameters of our model be

$$\theta = \{\pi, \mu, \Sigma\}$$

Let us now define the steps that the general EM algorithm will follow.

**Step 1**: Initialise $\theta$ accordingly. For instance, we can use the results obtained by a previous K-Means run as a good starting point for our algorithm.

**Step 2 (Expectation step)**: Evaluate

$$p(\mathbf{Z}|\mathbf{X}, \theta)$$

Actually, we have already found an expression for this. If the take the expectation of $z_{nk}$, then we get:

$$p(z_{nk}|\mathbf{X}, \theta) = \mathbb{E}[z_{nk}] = \sum_{j=1}^{K} z_{nj}\gamma(z_{nj}) = \gamma(z_{nk}) \qquad (5)$$

Which is exactly what we ended up with in the previous section. Do you now see why it was important for us to find this probability? Yeah, because it is evaluated as part of the EM algorithm.

**Step 3 (Maximization step):** Find the revised parameters $\theta^*$ using:

$$\theta^* = \arg\max_\theta \mathcal{Q}(\theta^*, \theta)$$

where

$$\mathcal{Q}(\theta^*, \theta) = \mathbb{E}[\ln p(\mathbf{X}, \mathbf{Z}|\theta^*)] = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \theta) \ln p(\mathbf{X}, \mathbf{Z}|\theta^*) \qquad (6)$$

Okay, we already found p(Z—X, $\theta$) in step 2, but we are missing p(X, Z—$\theta$*). How can we find it? Well, actually it's not that difficult. It is just the complete likelihood of the model, including both X and Z, and we can find it by using the following expression:

$$p(\mathbf{X}, \mathbf{Z}|\theta^*) = \prod_{n=1}^{N} \prod_{k=1}^{K} \pi^{z_{nk}} \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k)^{z_{nk}}$$

Which is the result of calculating the joint probability of all observations and latent variables and is an extension of our initial derivations for p(x). The log of this expression is given by

$$\ln p(\mathbf{X}, \mathbf{Z}|\theta^*) = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} [\ln \pi_k + \ln \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k)] \qquad (7)$$

Nice! And we have finally gotten rid of this troublesome logarithm that affected the summation in (3). With all of this in place, it will be much easier for us to estimate the parameters by just maximizing Q with respect to the parameters. Besides, remember that the latent variable z will only be 1 once everytime the summation is evaluated. With that knowledge, we can easily get rid of it as needed for our derivations.

Let us now determine the optimal parameters. Replacing equations (5) and (7) in (6) and applying a suitable Lagrange multiplier for the mixing coefficients $\pi$, we get:

$$\mathcal{Q}(\theta^*, \theta) = \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma(z_{nk}) [\ln \pi_k + \ln \mathcal{N}(x_n|\mu_k, \Sigma_k)] - \lambda \left( \sum_{k=1}^{K} \pi_k - 1 \right) \qquad (8)$$

And now we can easily determine the parameters by using maximum likelihood. Let's now take the derivative of Q with respect to $\pi$ and set it equal to zero:

$$\frac{\partial \mathcal{Q}(\theta^*, \theta)}{\partial \pi_k} = \sum_{n=1}^{N} \frac{\gamma(z_{nk})}{\pi_k} - \lambda = 0$$

Then, by rearranging the terms and applying a summation over k to both sides of the equation, we obtain:

$$\sum_{n=1}^{N} \gamma(z_{nk}) = \pi_k \lambda \implies \sum_{k=1}^{K} \sum_{n=1}^{N} \gamma(z_{nk}) = \sum_{k=1}^{K} \pi_k \lambda$$

d

From (1), we know that the summation of all mixing coefficients $\pi$ equals one. In addition, we know that summing up the probabilities $\gamma$ over k will also give us 1. Thus we get $\lambda$ = N. Using this result, we can solve for $\pi$:

$$\pi_k = \frac{\sum_{n=1}^{N} \gamma(z_{nk})}{N}$$

Similarly, if we differentiate Q with respect to $\mu$ and $\Sigma$, equate the derivative to zero and then solve for the parameters by making use of the log-likelihood equation (2) we defined, we obtain:

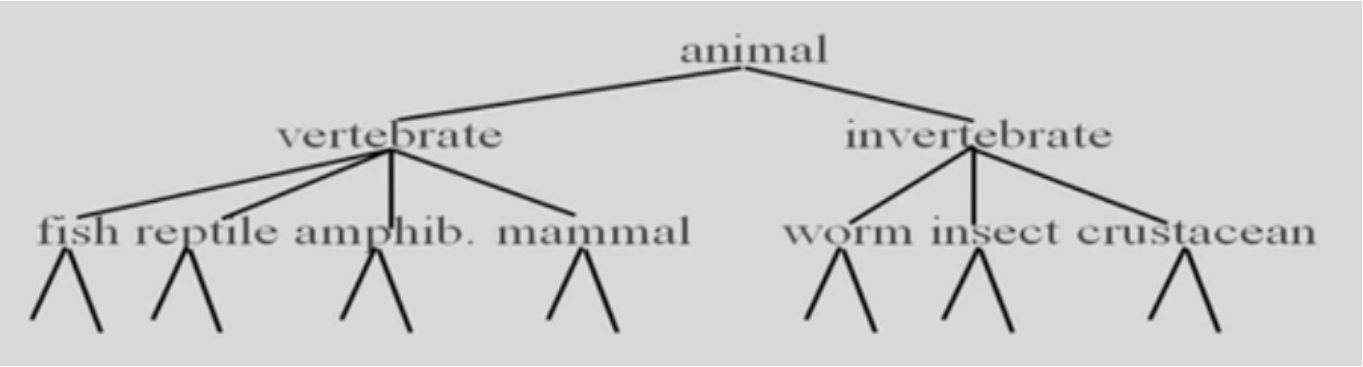
$$\mu_k^* = \frac{\sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n}{\sum_{n=1}^{N} \gamma(z_{nk})}, \qquad \Sigma_k^* = \frac{\sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \mu_k)(\mathbf{x}_n - \mu_k)^T}{\sum_{n=1}^{N} \gamma(z_{nk})}$$

And that's it! Then we will use these revised values to determine $\gamma$ in the next EM iteration and so on and so forth until we see some convergence in the likelihood value. We can use equation (3) to monitor the log-likelihood in each step and we are always guaranteed to reach a local maximum.

# 4 Hierarchical Clustering

Hierarchical Clustering, also called **Hierarchical cluster analysis** or **HCA**, is an unsupervised clustering algorithm which involves creating clusters that have predominant ordering from top to bottom.
For e.g: All files and folders on our hard disk are organized in a hierarchy.

The algorithm groups similar objects into groups called **clusters**. The endpoint is a set of clusters or groups, where each cluster is distinct from each other cluster, and the objects within each cluster are broadly similar to each other.




This clustering technique is divided into two types:

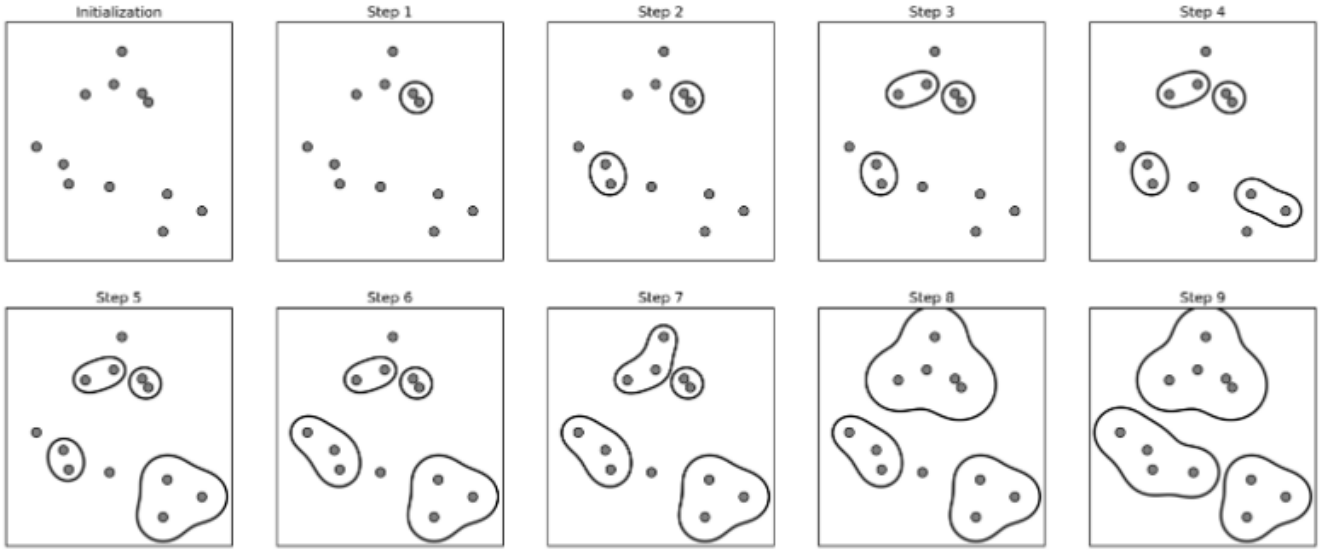
- Agglomerative Hierarchical Clustering (bottom-up)
- Divisive Hierarchical Clustering (top-down)

# 5 Agglomerative Hierarchical Clustering

The Agglomerative Hierarchical Clustering is the most common type of hierarchical clustering used to group objects in clusters based on their similarity. It's also known as **AGNES (Agglomerative Nesting)**.
It's a **bottom-up approach** : each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.

**How does it work?**

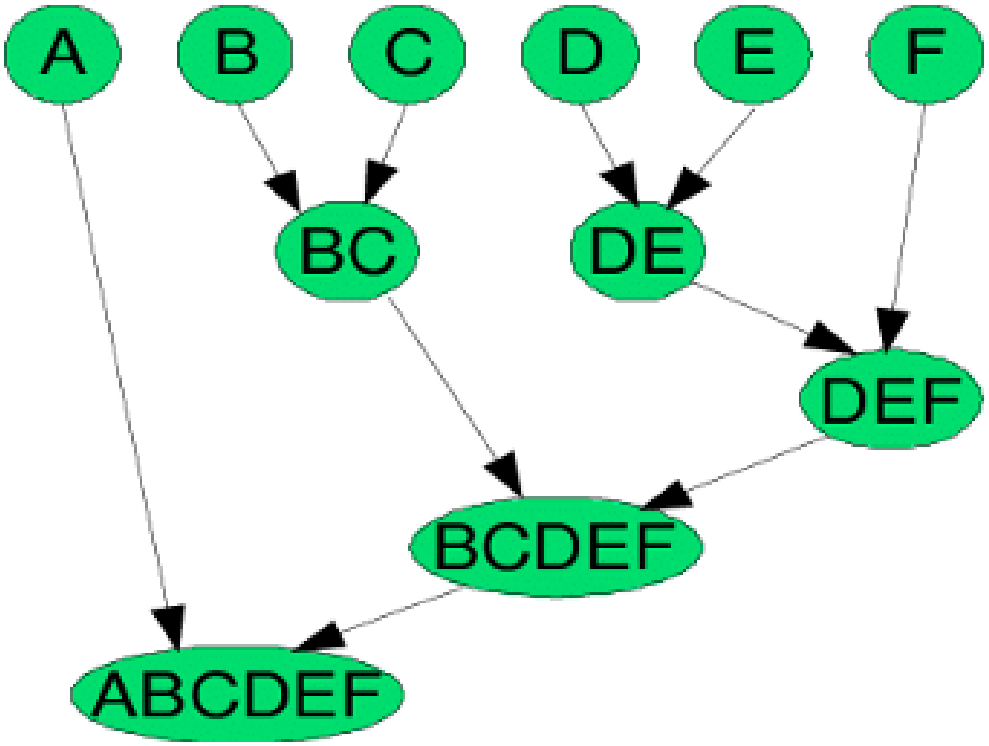
- Make each data point a single-point cluster; this forms **N clusters**
- Take the two closest data points and make them one cluster; this forms **N-1 clusters**
- Take the two closest clusters and make them one cluster; this forms **N-2 clusters**.
- Repeat step-3 until you are left with only one cluster.




Let's take a pictorial representation of the Agglomerative Hierarchical Clustering Technique. Lets say we have six data points A,B,C,D,E,F.

- Step- 1: In the initial step, we calculate the proximity of individual points and consider all the six data points as individual clusters as shown in the image below.
- Step- 2: In step two, similar clusters are merged together and formed as a single cluster. Let's consider B,C, and D,E are similar clusters that are merged in step two. Now, we're left with four clusters which are A, BC, DE, F.
- Step- 3: We again calculate the proximity of new clusters and merge the similar clusters to form new clusters A, BC, DEF.
- Step- 4: Calculate the proximity of the new clusters. The clusters DEF and BC are similar and merged together to form a new cluster. We're now left with two clusters A, BCDEF.

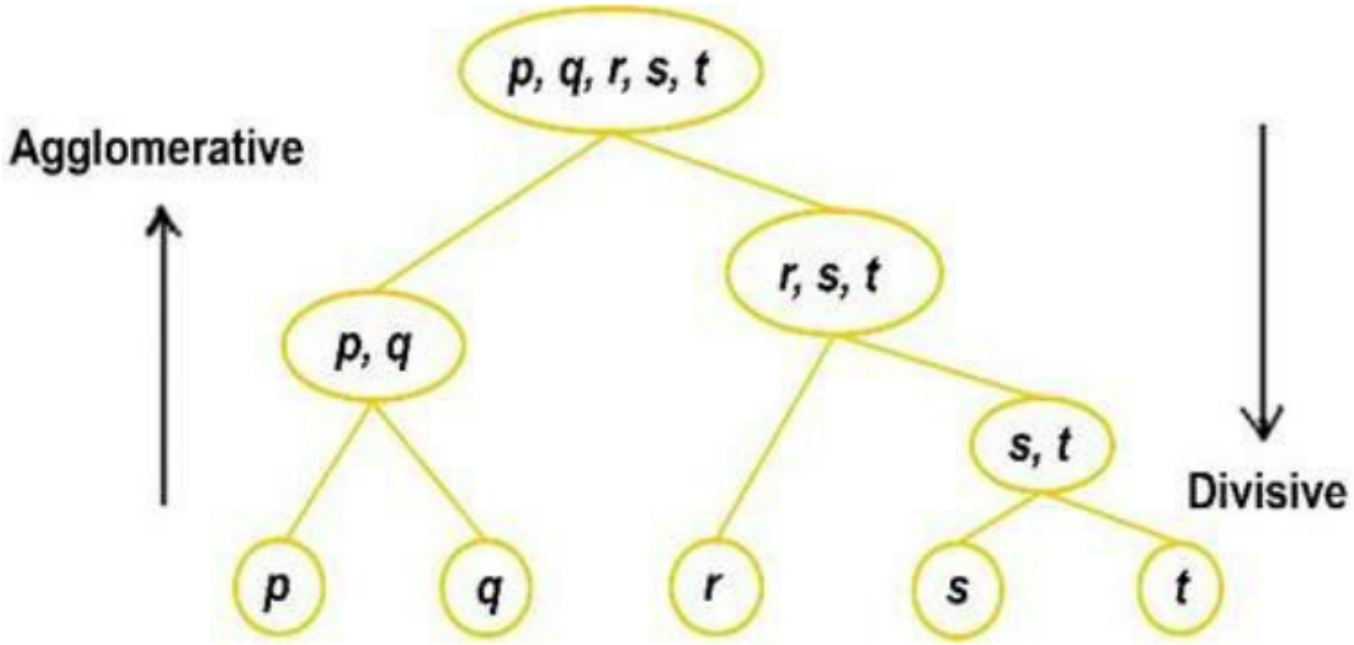
- Step- 5: Finally, all the clusters are merged together and form a single cluster.




Agglomerative Hierarchical Clustering Technique

## 6 Divisive Hierarchical Clustering

Divisive or DIANA(DIvisive ANAlysis Clustering) is a **top-down** clustering method where we assign all of the observations to a single cluster and then partition the cluster to two least similar clusters. Finally, we proceed recursively on each cluster until there is one cluster for each observation. So this clustering approach is exactly opposite to Agglomerative clustering.

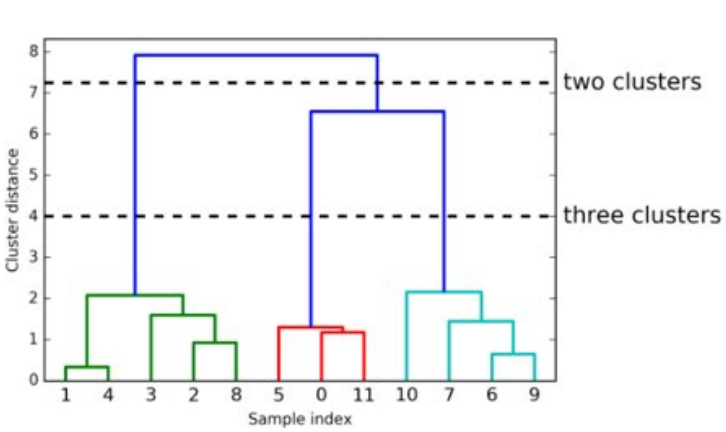


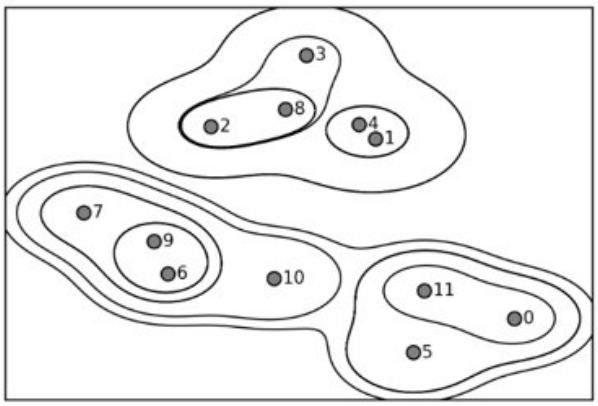

There is evidence that divisive algorithms produce more accurate hierarchies than agglomerative algorithms in some circumstances but is conceptually more complex.

In both agglomerative and divisive hierarchical clustering, users need to specify the desired number of clusters as a termination condition(when to stop merging).

## 7 Dendograms

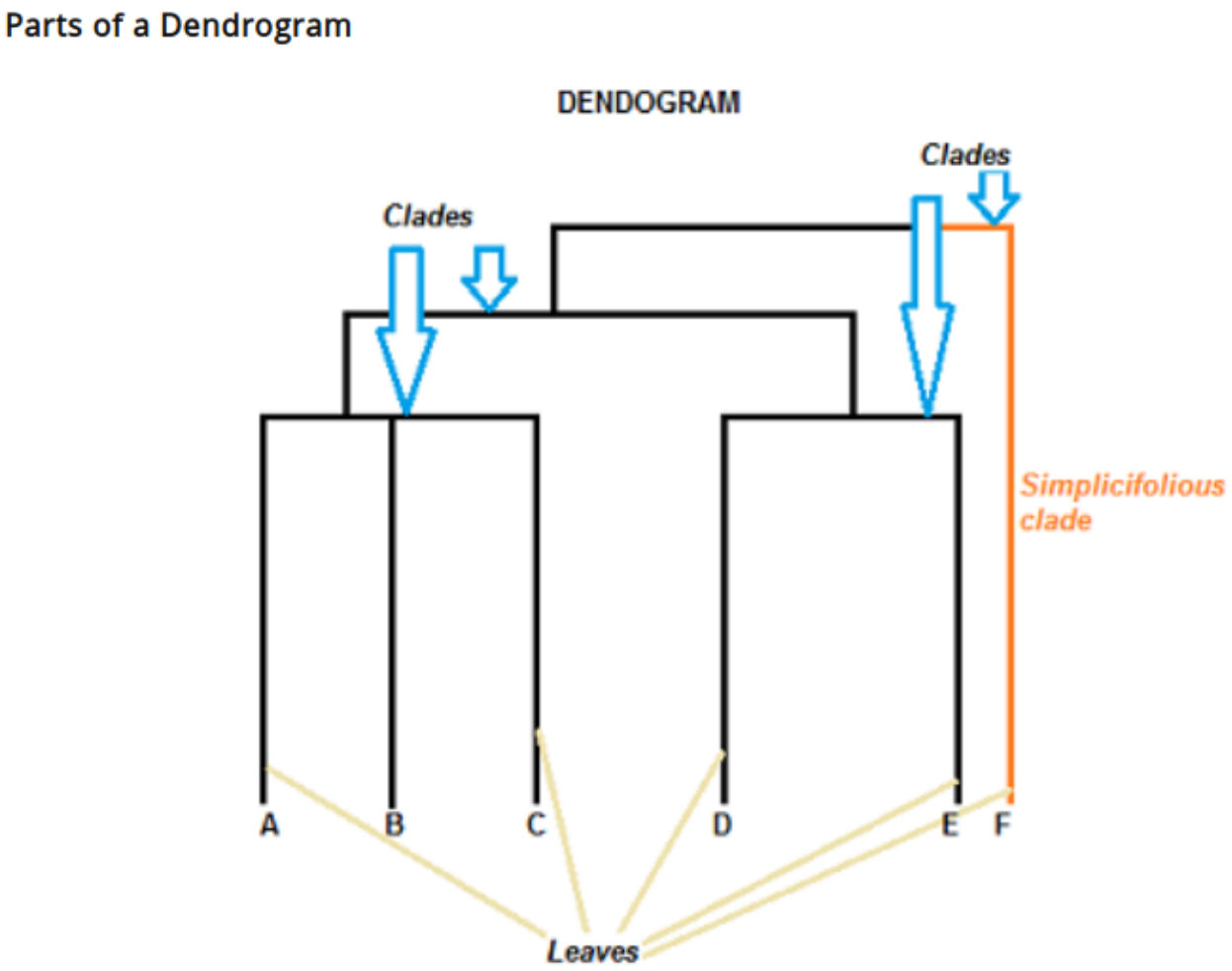
A Dendrogram is a type of tree diagram showing hierarchical relationships between different sets of data. A dendrogram contains the memory of hierarchical clustering algorithm, so just by looking at the Dendrgram you can tell how the cluster is formed.
A dendrogram can be a column graph (as in the image below) or a row graph. Some dendrograms are circular or have a fluid-shape, but the software will usually produce a row or column graph.



# 8 Parts Of Dendogram:

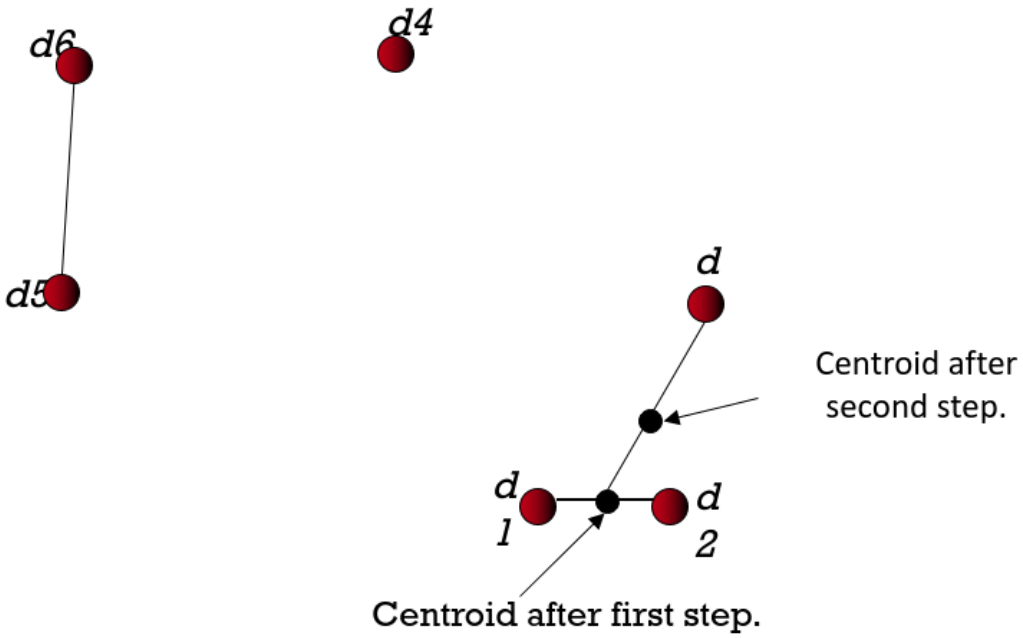
- Clades are the branch and are arranged according to how similar (or dissimilar) they are. Clades that are close to the same height are similar to each other; clades with different heights are dissimilar — the greater the difference in height, the more dissimilarity.
- Each clade has one or more leaves.
- Leaves A, B, and C are more similar to each other than they are to leaves D, E, or F.
- Leaves D and E are more similar to each other than they are to leaves A, B, C, or F.
- Leaf F is substantially different from all of the other leaves.



# 9 How to determine distance between clusters?

- Initially start from an individual cluster (each data point is considered as an individual cluster, also called leaf), then every cluster calculates their distance with each other.
- The two clusters with the shortest distance with each other would merge creating what we called node. Newly formed clusters once again calculate the member of their cluster distance with another cluster outside of their cluster.
- The process is repeated until all the data points assigned to one cluster called root.
- The result is a tree-based representation of the objects called **dendogram**.

**Example: n=6, k=3, closest pair of centroids**

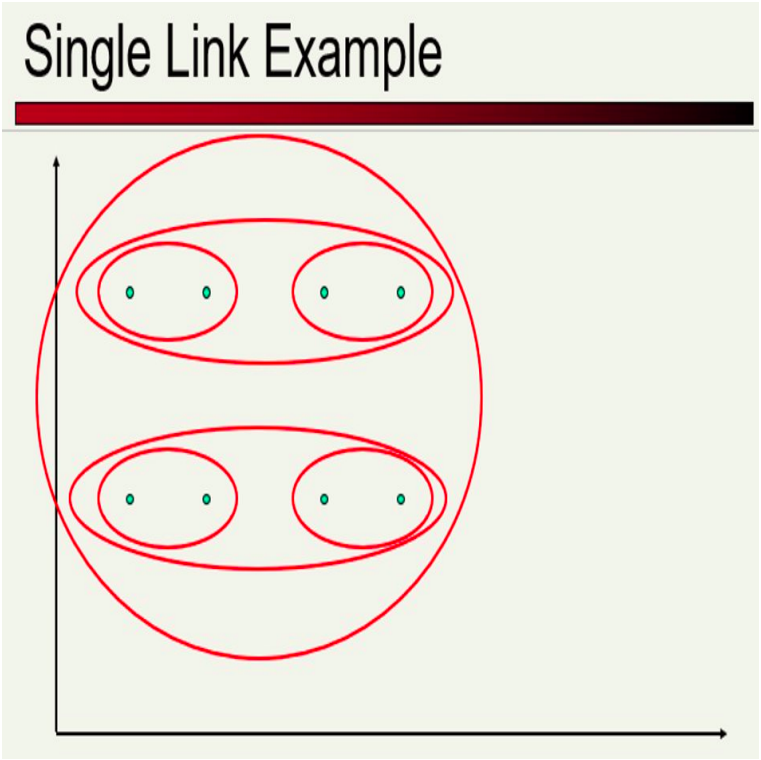



# 10 Closest Pair of Clusters

- **Single-link Clustering**

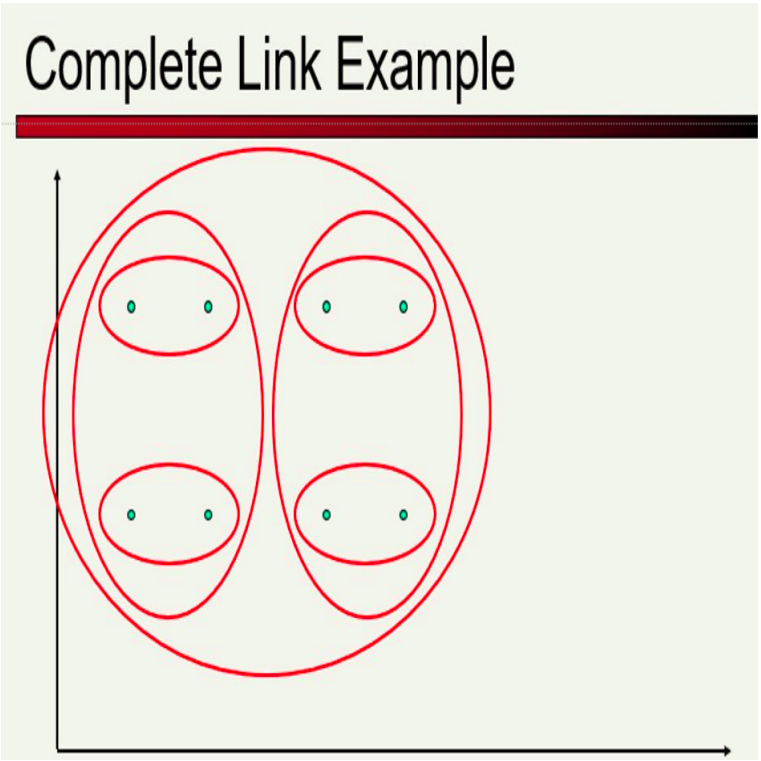
  Single-link clustering defines the distance between two clusters as the minimum distance between their members:

$$d(A, B) = \min_{\vec{x} \in A, \vec{y} \in B} \| \vec{x} - \vec{y} \|$$




- **Complete-Link Clustering**
  Most common techniques used is complete-link clustering, where the distance between clusters is the maximum distance between their members.

$$d(A, B) = \max_{\vec{x} \in A, \vec{y} \in B} \| \vec{x} - \vec{y} \|$$



- **Ward's method**

  Ward's method says that the distance between two clusters, A and B, is how much the sum of squares will increase when we merge them:

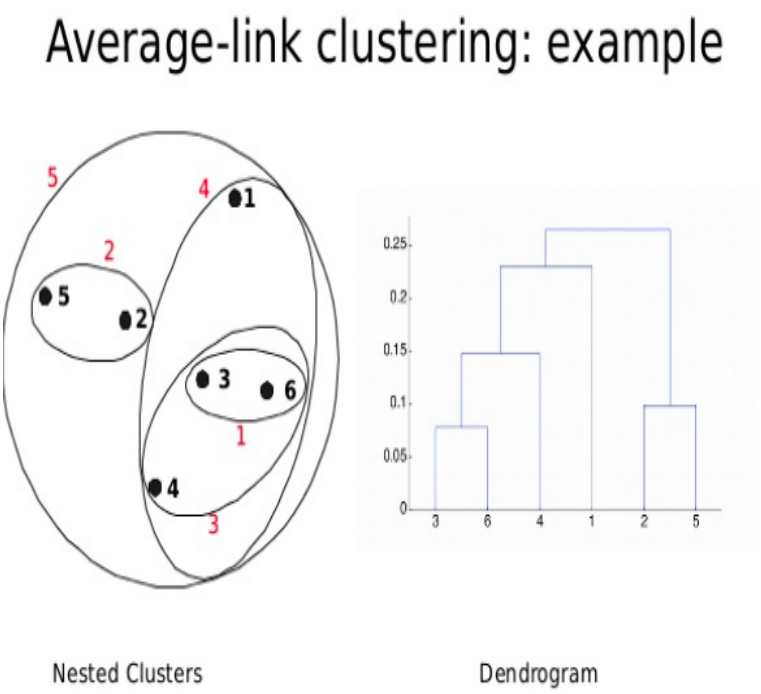
$$\Delta(A,B) = \sum_{i \in A \cup B} \| \vec{x}_i - \vec{m}_{A \cup B} \|^2 - \sum_{i \in A} \| \vec{x}_i - \vec{m}_A \|^2 - \sum_{i \in B} \| \vec{x}_i - \vec{m}_B \|^2$$

$$= \frac{n_A n_B}{n_A + n_B} \| \vec{m}_A - \vec{m}_B \|^2$$

  where $m_j$ is the center of cluster j, and $n_j$ is the number of points in it. $\Delta$ is called the merging cost of combining the clusters A and B. With hierarchical clustering, the sum of squares starts out at zero (because every point is in its own cluster) and then grows as we merge clusters. Ward's method keeps this growth as small as possible. This is nice if you believe that the sum of squares should be small. Notice that the number of points shows up in $\Delta$, as well as their geometric separation. Given two pairs of clusters whose centers are equally far apart, Ward's method will prefer to merge the smaller ones.

- **Average-Link Clustering**

  In Average linkage clustering, the distance between two clusters is defined as the average of distances between all pairs of objects, where each pair is made up of one object from each group.



# 11 Measuring the goodness of Clusters

Internal Criterion: A good clustering will produce high quality clusters in which :

- the intra-class (that is, intra-cluster) similarly is high.
- the inter-class similarity is low
- the measured quality of a clustering depends on both the data representation and the similarity measure used.
- The measured quality of a clustering depends on both the data representation and the similarity measure used.

**Dunn's index** is the ratio between the minimum inter-cluster distances to the maximum intra-cluster diameter. The diameter of a cluster is the distance between its two furthermost points. In order to have well separated and compact clusters you should aim for a **higher Dunn's index**.

# References

[1] https://towardsdatascience.com

[2] https://en.wikipedia.org/wiki/Mixture_modelGaussian_mixture_model

[3] https://www.displayr.com/what-is-hierarchical-clustering

[4] https://www.kdnuggets.com/2019/09/hierarchical-clustering.html