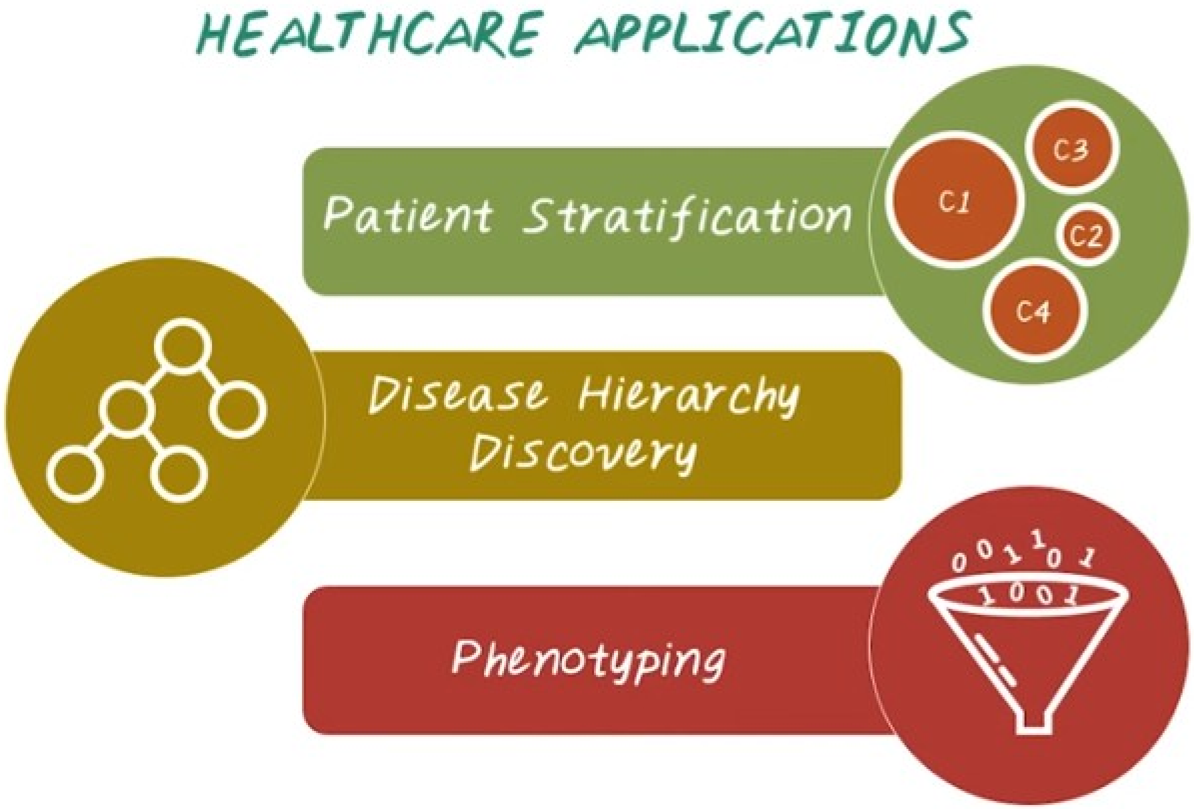
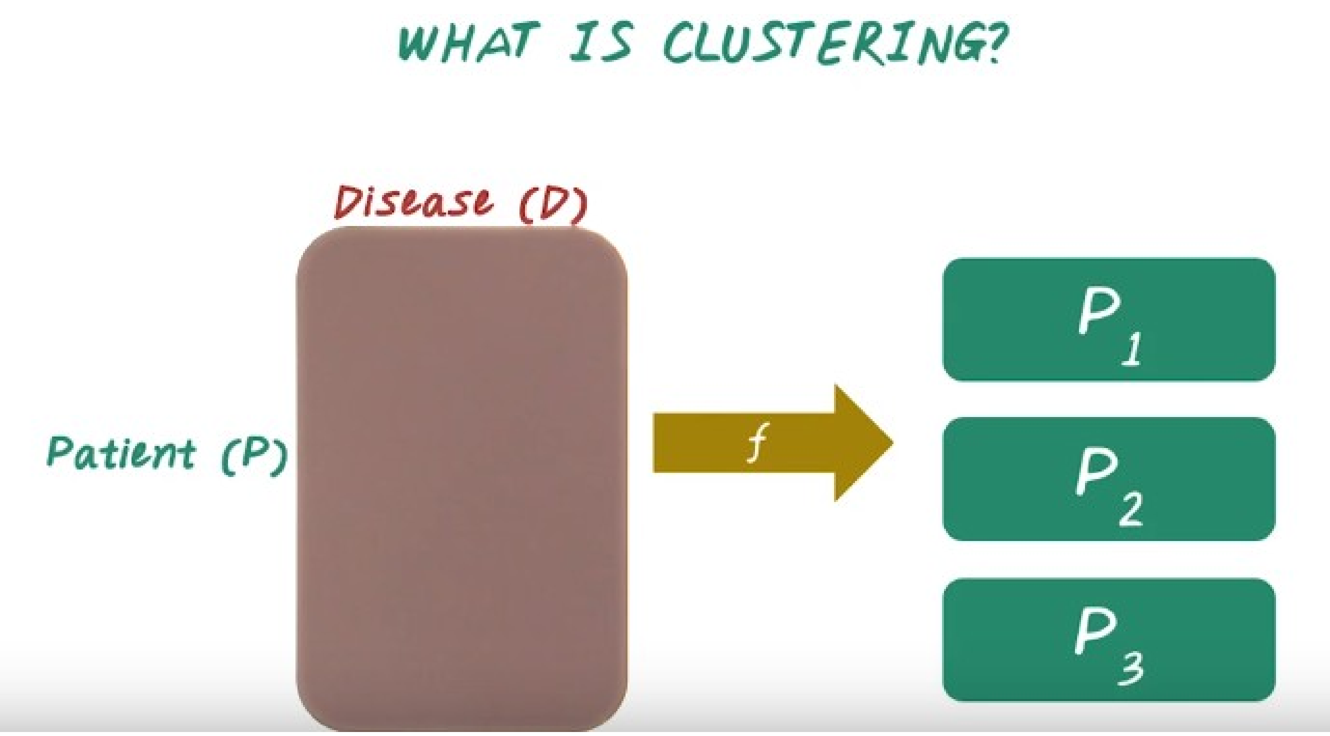
1. Now we're going to move on to a different method called clustering. In classification, we group data by label or categories that we already knew. But in clustering, we want to discover those labels or categories form raw data. We'll start by defining clustering. Then we'll describe some algorithms for clustering, such as K-means and Gaussian mixture models. Then, we'll describe scalable classroom algorithms for handling big data sets. Finally, we'll preview some of the healthcare applications of clustering.

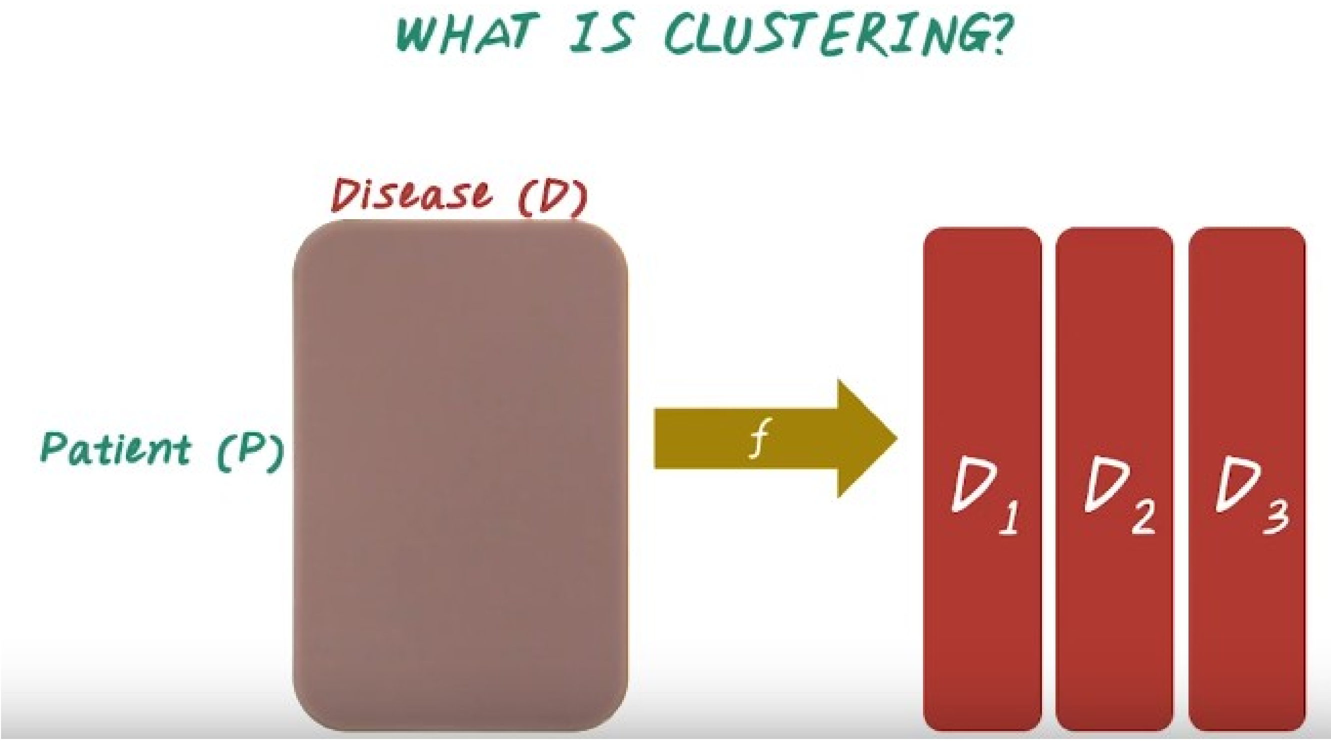


2. There are many healthcare applications for clustering. For example, patient stratification(分层) is about grouping patients into clusters such that patient within the same cluster share many common characteristics, and patients across cluster share very few common characteristics. And disease hierarchy discovery is about learning a hierarchical structure about all the diseases and how they relate to each other from data. And phenotyping is about converting raw data into meaningful clinical concepts or phenotypes(具有共同表现型的一类有机物). The phenotypes is actually a cluster of patients that share some commonalities.



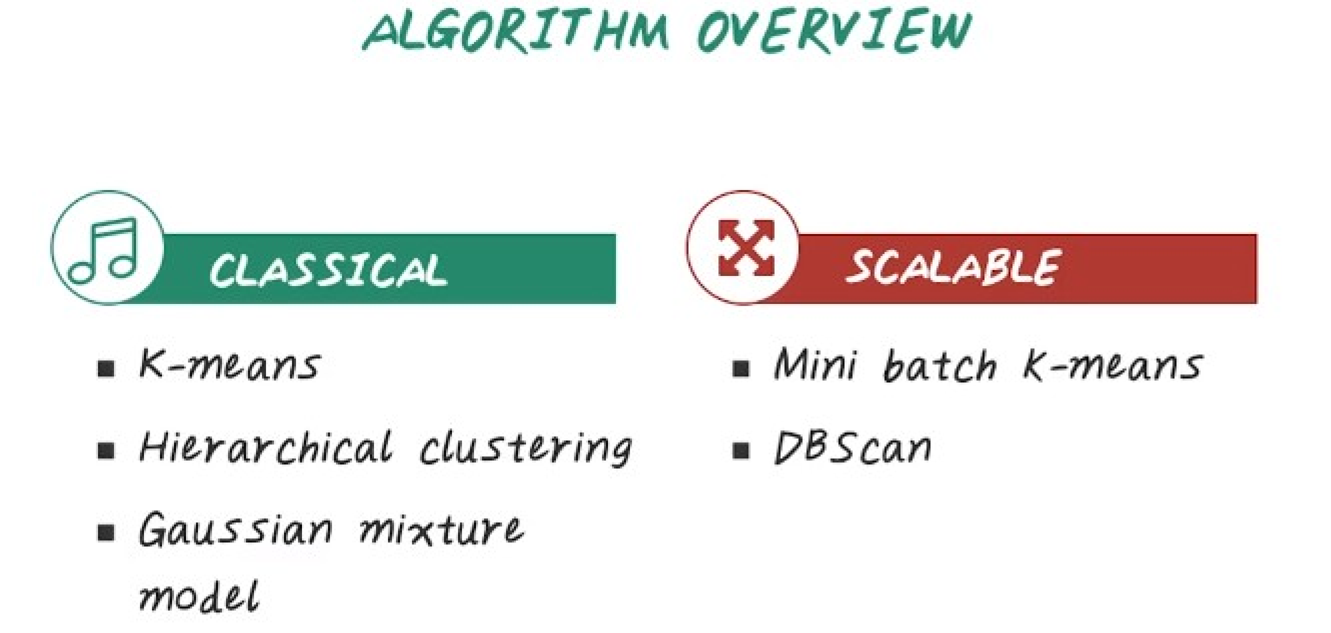
WHAT IS CLUSTERING?

3. So what is clustering? Let's illustrate clustering using the following example. Imagine we have a patient by disease matrix, where the columns are all the different diseases, and rows are different patients. And other elements indicate whether a specific patient has a specific disease. If we want to apply clustering on the rows of this matrix, which means you want to learn a function f that partitions this matrix into a set of groups. Each group corresponding to a set of patients, P1, P2 and P3. Once we have all those patient groups or patient clusters, we can utilize those to support application such as phenotyping and patient stratification.



WHAT IS CLUSTERING?

And similarly, we can apply clustering on the columns of the matrix to partition all the diseases into different disease group, such as D1,D2, and D3. And we can further group all those disease groups into a hierarchy, and this will support the disease discovery application.



ALGORITHM OVERVIEW

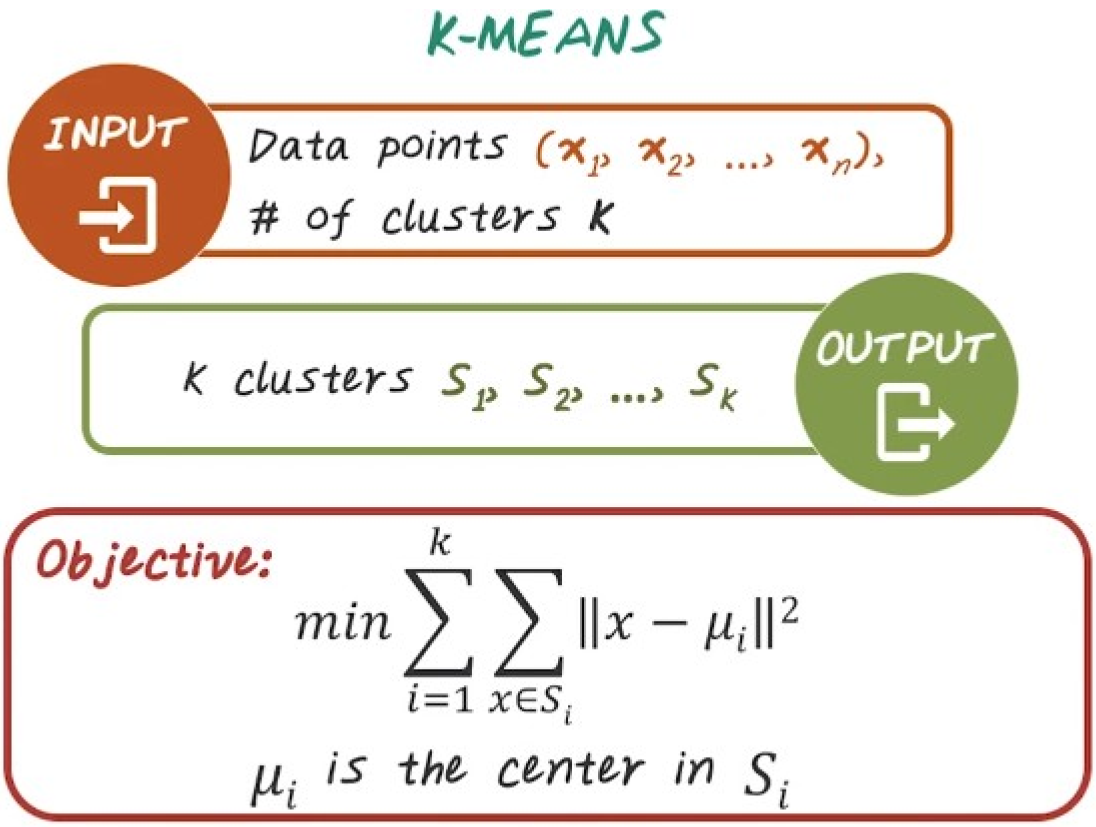
CLASSICAL

- K-means
- Hierarchical clustering
- Gaussian mixture model

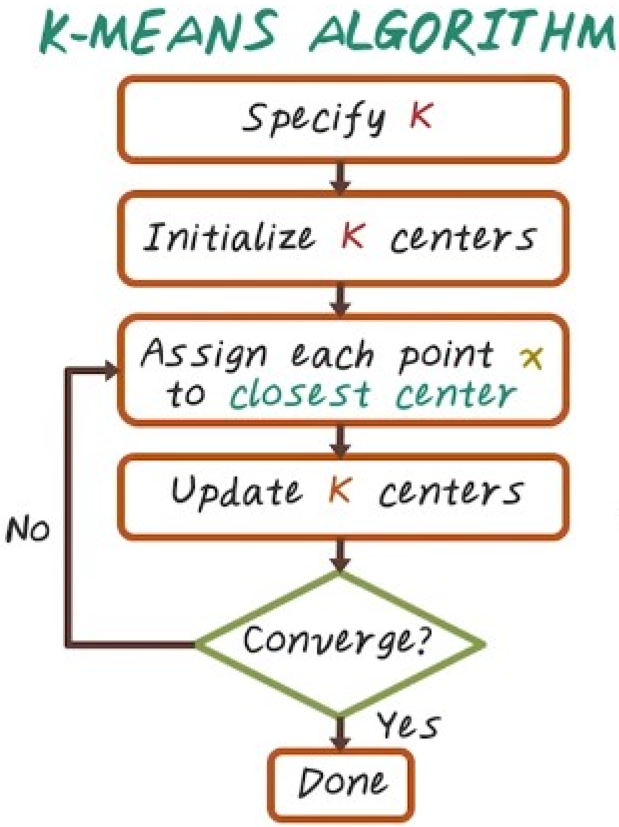
SCALABLE

- Mini batch K-means
- DBScan

4. In this class we'll introduce two sets of clustering algorithms. The classical clustering algorithms and the scalable clustering algorithms. For classical algorithms, we'll discuss K-means, hierarchical clusterings, and Gaussian mixture model. For scalable algorithms, we'll introduce mini batch K-means and DBScan.

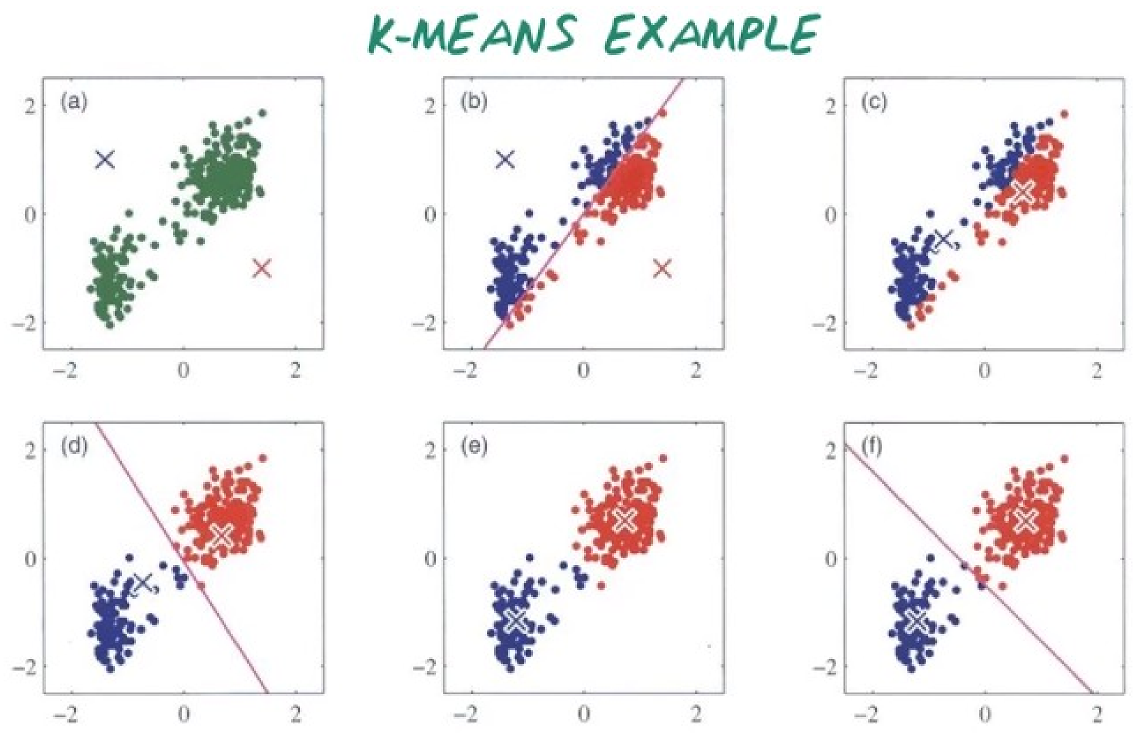

K-MEANS
INPUT
Data points $(x_1, x_2, \ldots, x_n)$,
# of clusters K
K clusters $S_1, S_2, \ldots, S_K$
OUTPUT
Objective:
$\min \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2$
μ_i is the center in S_i

5. Now let's talk about K means clustering. So the input to the K means algorithm is a set of data points X1 X2 to Xn and the perimeter K indicate the number of clusters. And the output of K means clustering are K clusters S1, S2 to SK and each cluster contains a set of data points and the union of all those clusters give us all the data points. The objective of K-means clustering is it minimize the sum of all the data points, x, to its corresponding center, mu i. Here mu i is the center for cluster Si. In order to minimize this objective, we want to find the optimal assignment of all this data points into K cluster, such that this term is minimized.



$$\mu_i = \frac{1}{|S_i|}\sum_{x \in S_i} x$$

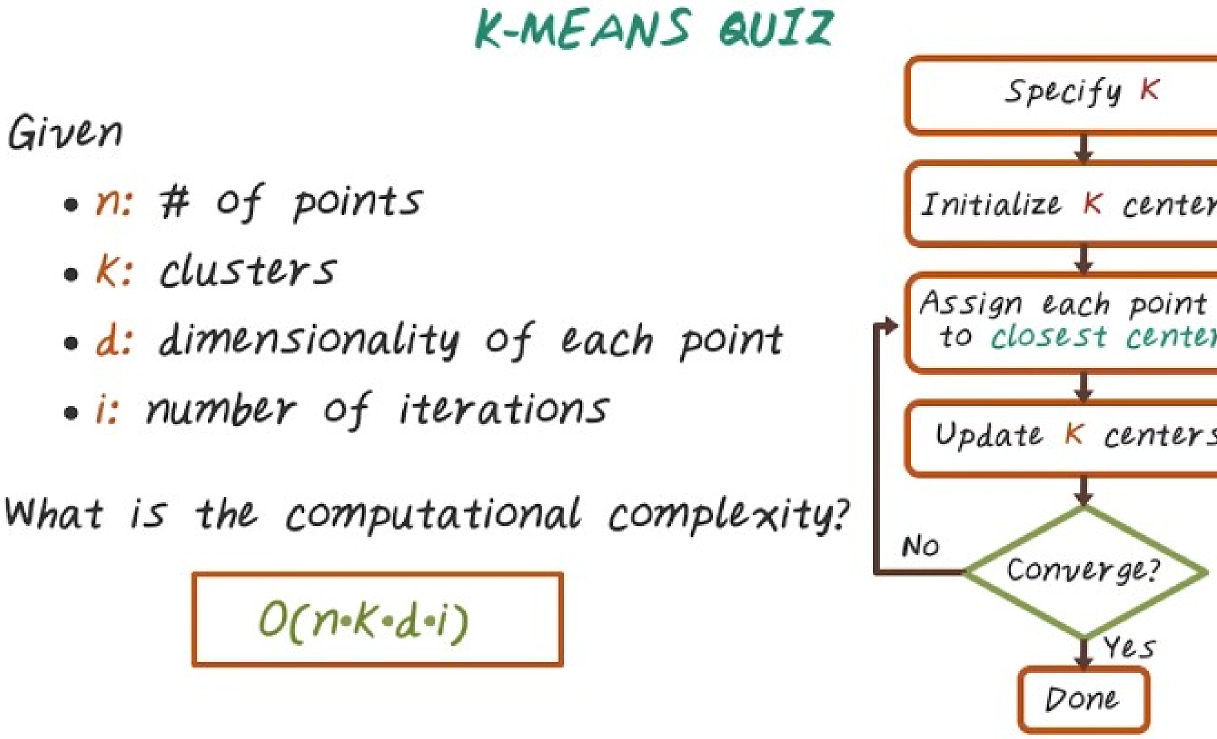
Now let's talk about the algorithm for K means. First we specified K as the number of clusters. Then we initialized K centers. Then we'll assign each data point to its closest center. Once we have all the assignments we update the K centers. For this step we update the new center mu i by averaging all the data points within the cluster Si. Then we check whether the algorithm converged or not. Which means we can check whether any cluster assignment has changed from the previous iteration. Or some pre defined maximum number of iteration has been reached. Then we iterate into the convergence criteria as math then we output the clusters. And this is the K means algorithm.



上先從左到右看完第一行, 再看第二行

Now, let's visually illustrate how K means works using this example. We want to run K means algorithm on this set of points with K = 2. To start, let's initialize the two cluster center with the blue cross and red cross. Then we'll assign all the data points to as close the center. And all those blue points are assigned to the blue cluster, and all the red points are assigned to the red cluster. Then we find the new cluster center by averaging all the blue points and all the red points. So this blue and red cross are the two new centers. Then we iterate this process over three iterations, and two had converged. This is the final result when we want K means with setup points, with K equal to two.

6. Now let's do a quiz on K-means. Given n is the number of data points, k is the number of clusters, and d is dimensionality of each data points, and i is the number of iteration of the K-means algorithm, what is the computational complexity of K-means? This is the algorithmic illustration to help you find the answer and put your answer in this box.



K-MEANS QUIZ

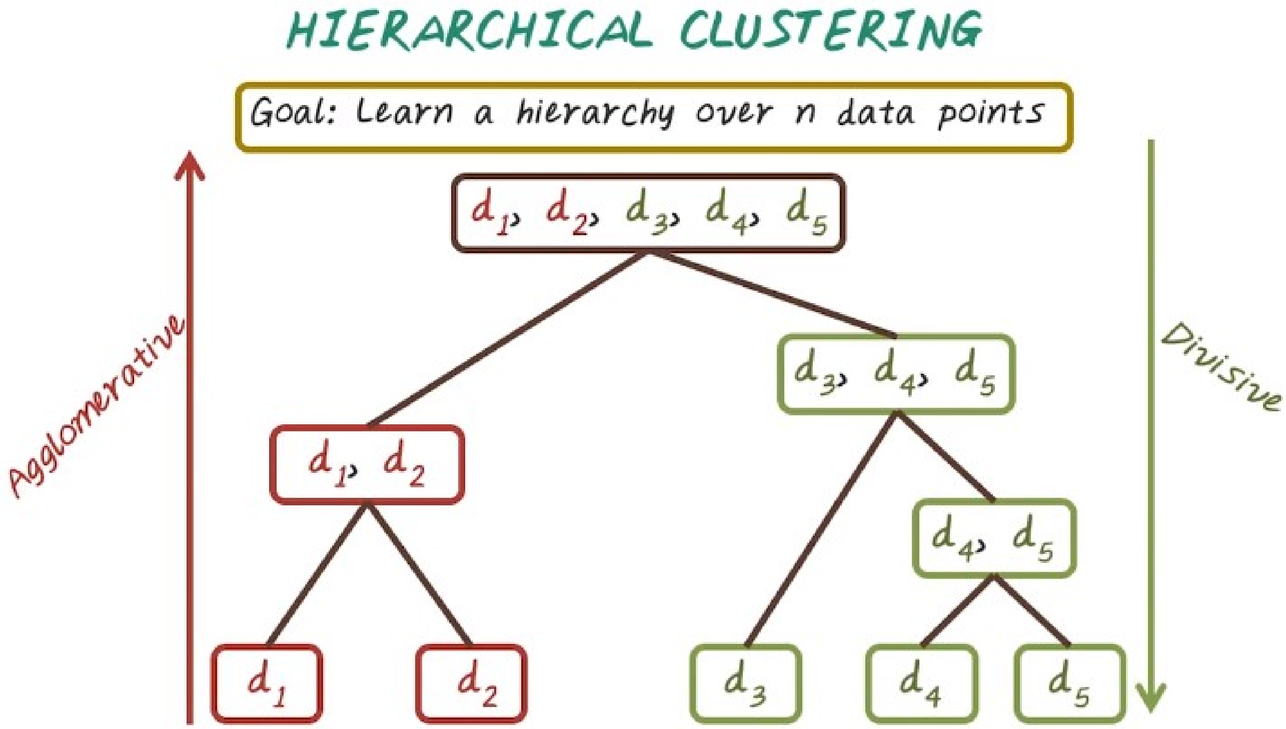
Given

- n: # of points
- k: clusters
- d: dimensionality of each point
- i: number of iterations

What is the computational complexity?

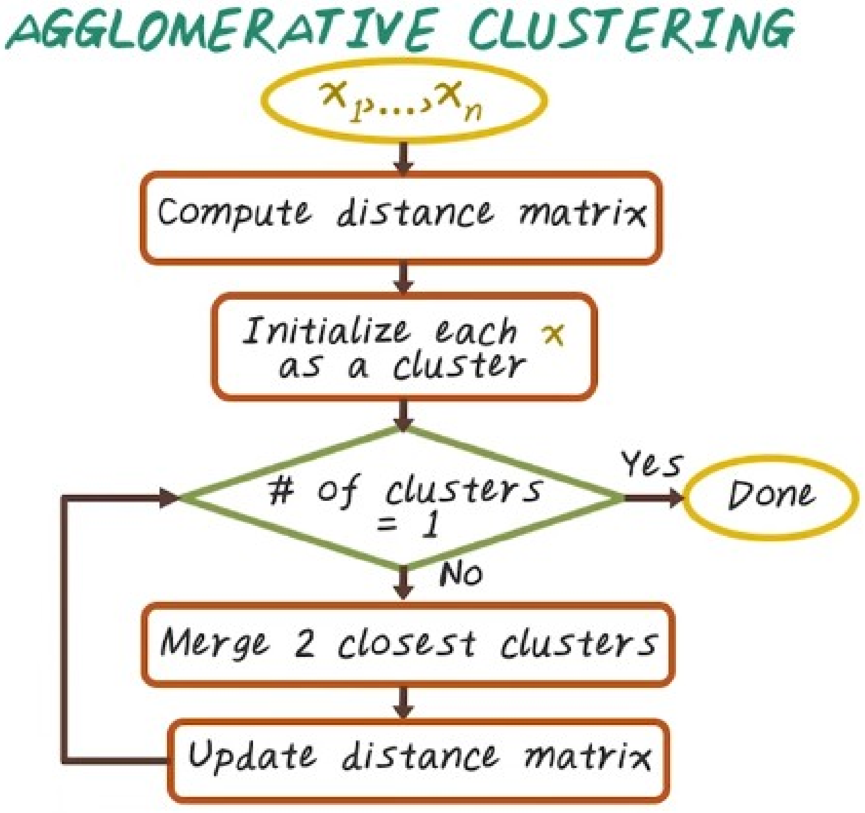
$O(n \cdot k \cdot d \cdot i)$

7. And the answer is big O of n times k times d times i. Now let's see how we got this answer. When we run k-means algorithm most of the computation goes to clustering assignment and center update. For clustering assignment, we need to compare end data points to K centers. So that takes n times k comparisons. Since each comparison is order d operation, because each datapoint has d dimensions and the total complexity for each iteration is n times k times d. Similarly, we can derive the center update of the same complexity and the total complexity of k means algorithms becomes n times k times d times i.

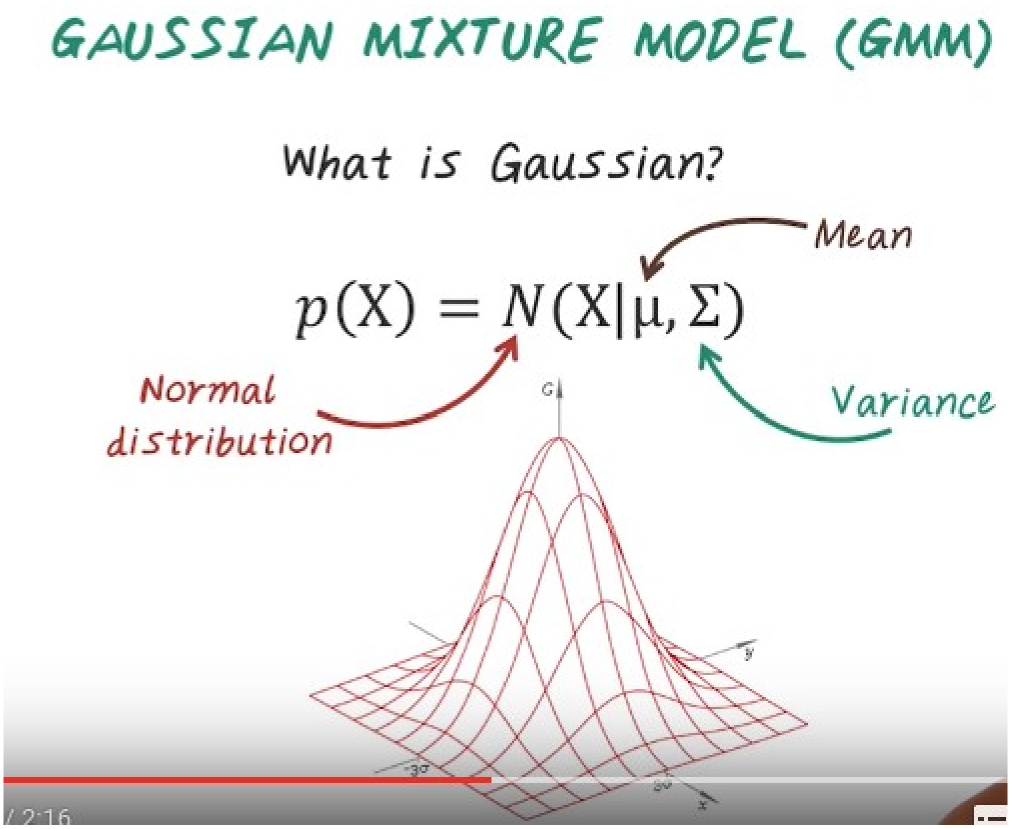


上圖的 Agglomerative method 有點像 ML clustering 課中講的 SLC (Single Linkage Clustering)算法.

8. Next, let's introduce hierarchical clustering. The goal of hierarchical clustering is to learn hierarchy over a set of data points. For example, given five different diseases, we want to construct such a hierarchy so that similar diseases are grouped together into this tree structure. There are two main ways for doing this. One is the bottom up approach called agglomerative method. We'll start with individual disease, d1 to d5, as their own clusters, then interactively group those smaller clusters into bigger clusters, until only one cluster remains. For example, d1 and d2 will first be grouped together, then d4, d5 will grouped together. Then subsequently, d3 and d4, d5 will be grouped together and finally, all five diseases will be merged into one cluster. The other approach is a top down approach called a divisive method. We'll start with a single cluster with all the diseases in it, then iteratively divide the bigger cluster into smaller clusters, and thus, the divisive method. In general, this agglomerative method bottom up approach is more efficient, therefore, more popular in practice.



9. Now let's talk about agglomerative clustering algorithm (就是上段中的 agglomerative method, 下段已驗證). We'll start with a set of data points x1 to xn. First we compute all paired distance matrix. This will be a n by n matrix, and every element in this matrix corresponding to the distance between the pair of points. Then we initialize each data points as their own cluster. So we have n cluster here. Then we check how many clusters are left. If we only have one cluster left, that's it, that's the final result. We output the hierarchy. If we have more than one cluster, we merge the closest clusters. Then we update the distance matrix. We will run this algorithm for the first times, we'll start with a n by n distance matrix, then we merge two closest clusters, and update the distance matrix, here the distance matrix will be of size n- 1 by n- 1. Then we iterate, continue to merge to a closest cluster together and also update the distance matrix until we have only one cluster left, and that's the final result.

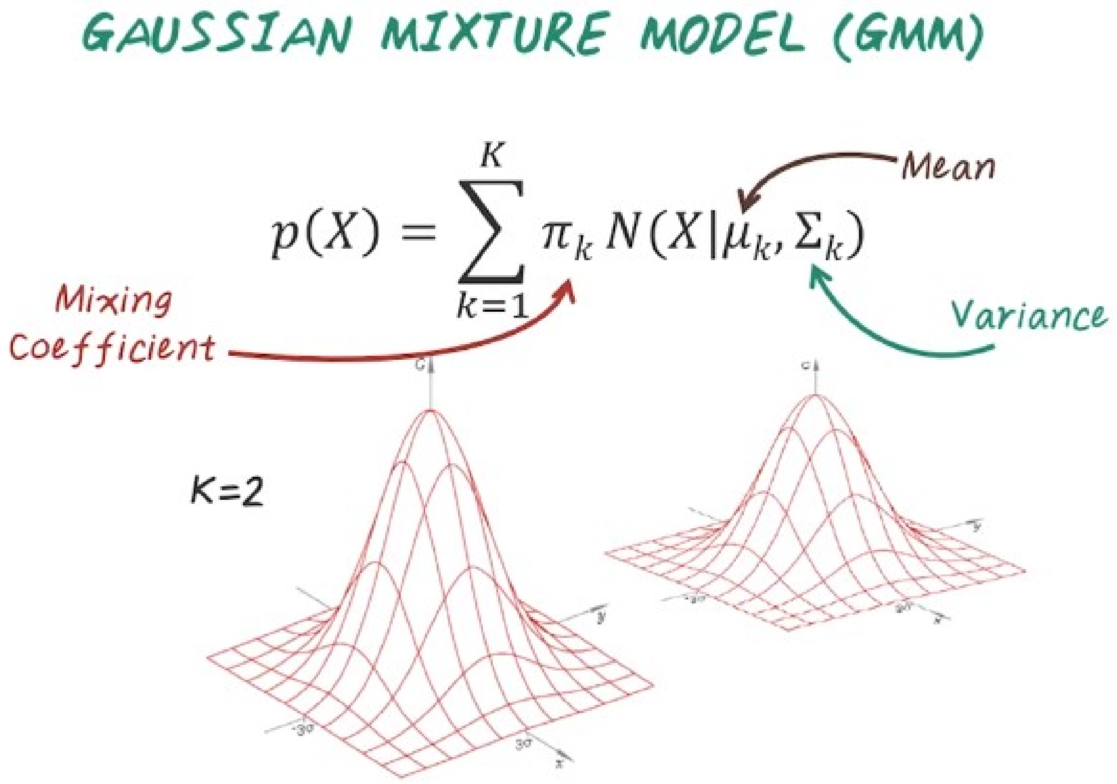


GAUSSIAN MIXTURE MODEL (GMM)

What is Gaussian?

$$p(X) = N(X|\mu, \Sigma)$$

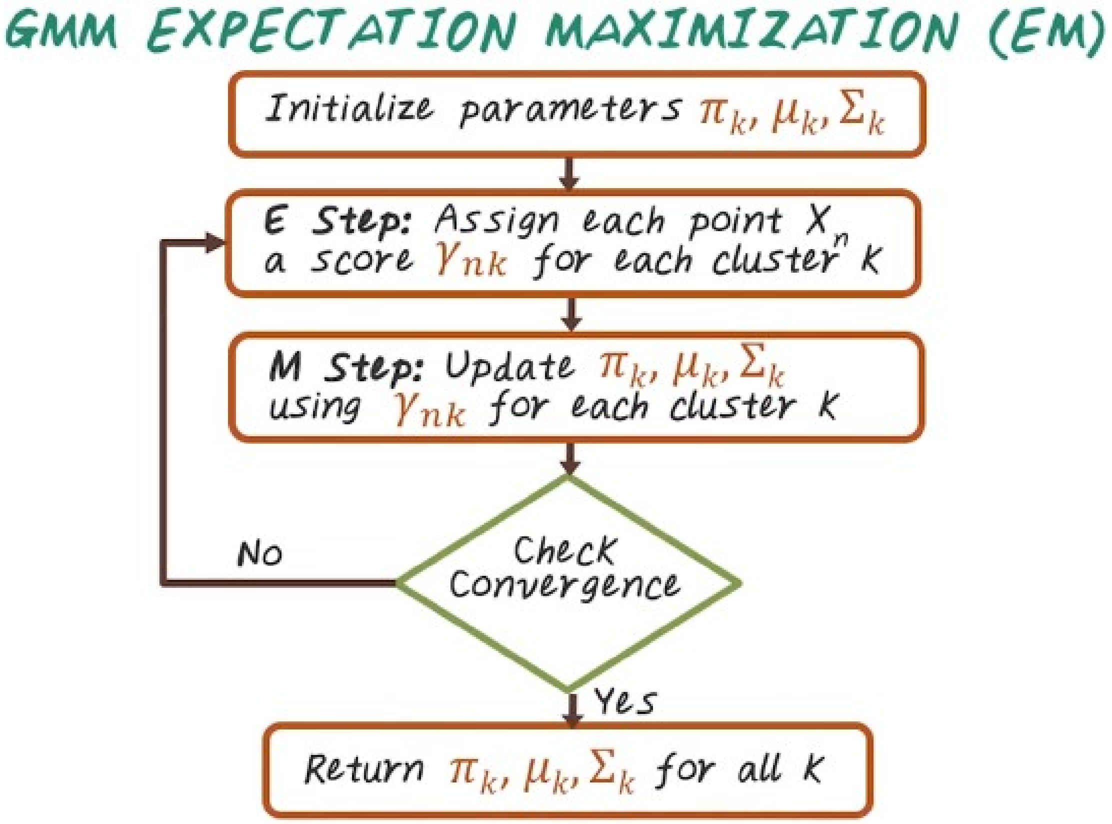
10. Next we'll talk about Gaussian mixture model. So far we have talked about two clustering algorithm, K-means and hierarchical clustering. They are both hard clustering method, which means every data points only belong to a single cluster, and Gaussian mixture model is a soft clustering method. >> Soft? >> Yes, soft clustering. So every data points can belong to multiple clusters with a different degree. So, first, what is a Gaussian? Gaussian distribution is the most popular continuous probability distribution, and it's also known as the normal distribution. The shape of Gaussian is a bell curve, like this, and Gaussian distribution has two parameters, mu and sigma. And the mu variable corresponding to the center of the bell curve, and the variance sigma capture the spread of this bell curve. When the variance is small, the spread will be small, which means all the probability will be concentrated around the mean. When the variance is large, the probability will be scattered, which means the events that are far away from the mean can still happen with large probability.



GAUSSIAN MIXTURE MODEL (GMM)

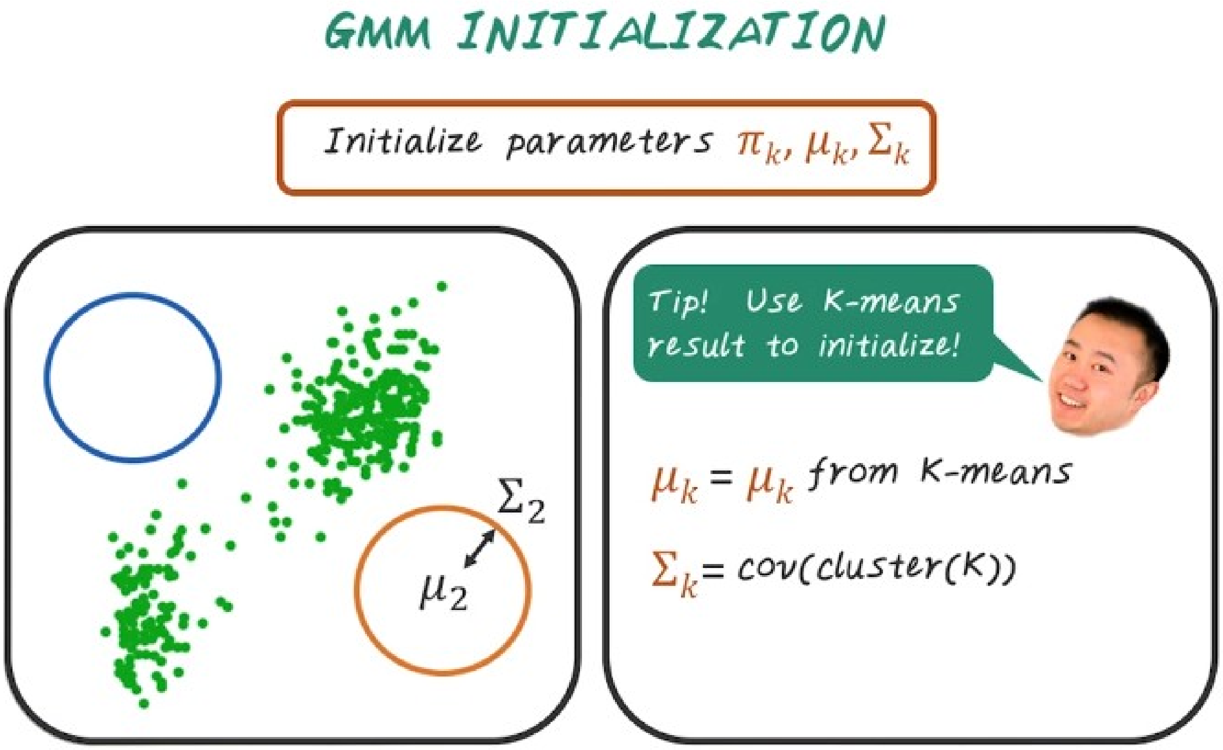
$$p(X) = \sum_{k=1}^{K} \pi_k N(X|\mu_k, \Sigma_k)$$

Now let's introduce the mathematical definition of Gaussian Mixture Model. The probability of a data point X is the weighted sum over k Gaussian distribution. Here at the pi k is the mixing coefficient for cluster k. Intuitively, it tells us how big the cluster k is, and mu k is the mean of cluster k or the cluster center for cluster k, and sigma k is the co-variance for cluster k. So here's an example when we have a two Gaussian distribution and data points x will be generated from this two underlying Gaussians. For a Gaussian mixture model, the goal is to figure out the parameters of the two underlying Gaussians, namely finding out pi k, mu k, and sigma k, given all the observed data points.


GMM EXPECTATION MAXIMIZATION (EM)
Initialize parameters π_k, μ_k, Σ_k
E Step: Assign each point X_n a score γ_{nk} for each cluster k
M Step: Update π_k, μ_k, Σ_k using γ_{nk} for each cluster k
Check Convergence
No
Yes
Return π_k, μ_k, Σ_k for all k

本課中的 EM 講得太好了, 簡直就是小清新, 不要去看 ML 的 note 中對 EM 的講解, 看了有害.

11. To compute a Gaussian mixture model, we utilize a popular optimization strategy called expectation maximization, or Algorithm. Next, let's see how Algorithm works for Gaussian mixture model. We first initialize all the parameters for Gaussian mixture model and could mix in coefficient pi k. The center mu k and variance sigma k. Then in the E Step, we assign each data point Xn with a score gamma nk for each cluster K. So in this case, data points Xn will have K scores, one for each cluster. In particular, gamma nk tells us how likely Xn is generated from cluster K. Once we have all the assignments, then in the M Steps we update all those model parameter, pi k, mu k, sigma k, using the scores we have learned from those assignments. Then we check if convergence criteria are met. For example, likelihood is no longer changing or parameters are stabilized. If no, we continue this Iterations into convergence. If yes, we return all the model parameters for Gaussian mixture model. Next, let's look at those steps in more details.



GMM INITIALIZATION

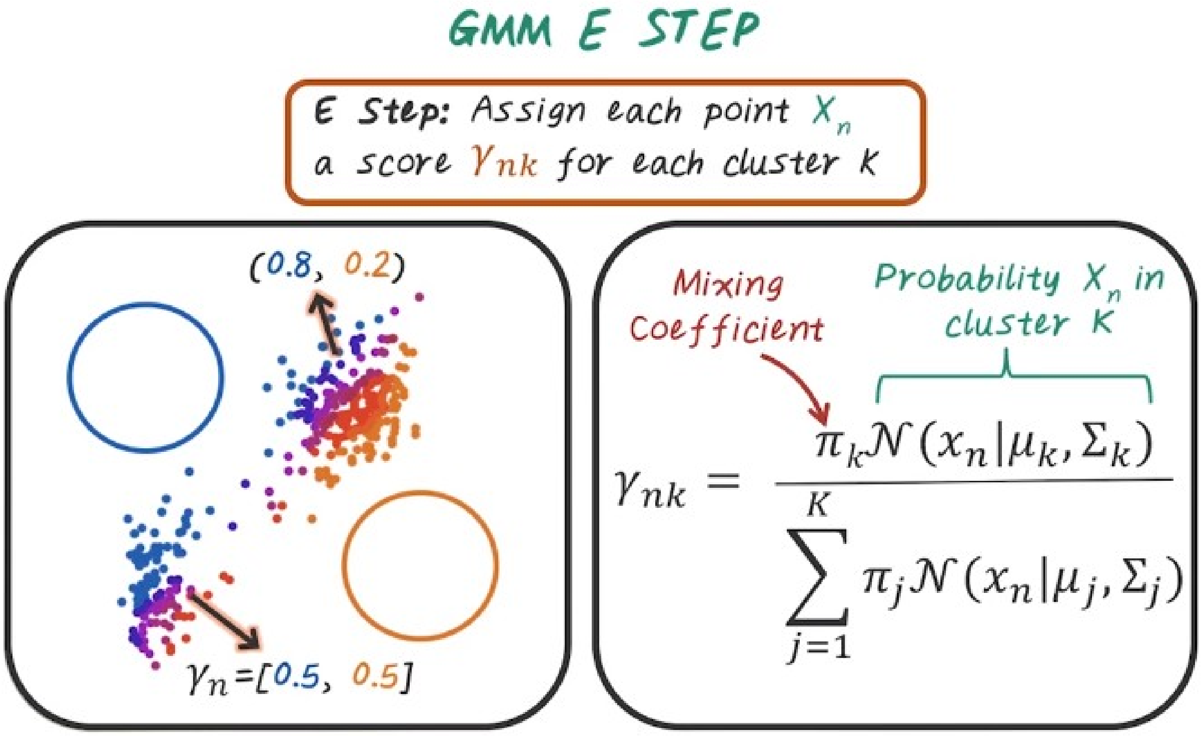
Initialize parameters π_k, μ_k, Σ_k

Tip! Use K-means result to initialize!

$\mu_k = \mu_k$ from K-means

Σ_k= cov(cluster(K))

12. In order to run Algorithms for Gaussian mixture model, we have to initialize all those parameters. In particular, we have to find the mixing coefficients, the centers, and variance for each clusters. And this initialization step is very important. A bad initialization can cause many subsequent iterations into convergence. A good initialization can save a lot of iteration so that the convergence can happen very quickly. For example, in this picture, if we initialized these two clusters here and here as the center and also give equal weights for the mixing coefficients, this will be a pretty bad initialization, because the starting point of these two cluster center do not coincide with any data points. Which means subsequently the Algorithm has to iterate many times to correct this bad initialization. So what will be a better initialization? We can actually use K-means result to initialize for Gaussian mixture model. For example, the center in the Gaussian mixture model will be the center from K-means result. The covariance matrix sigma K can be computed by using all the data points from the corresponding clusters. And finally, the mixing coefficient pi k can be simply computed as the size of the cluster K divided by the total number of data points. And usually this initialization with K-means result will be much better than a random initialization like this.



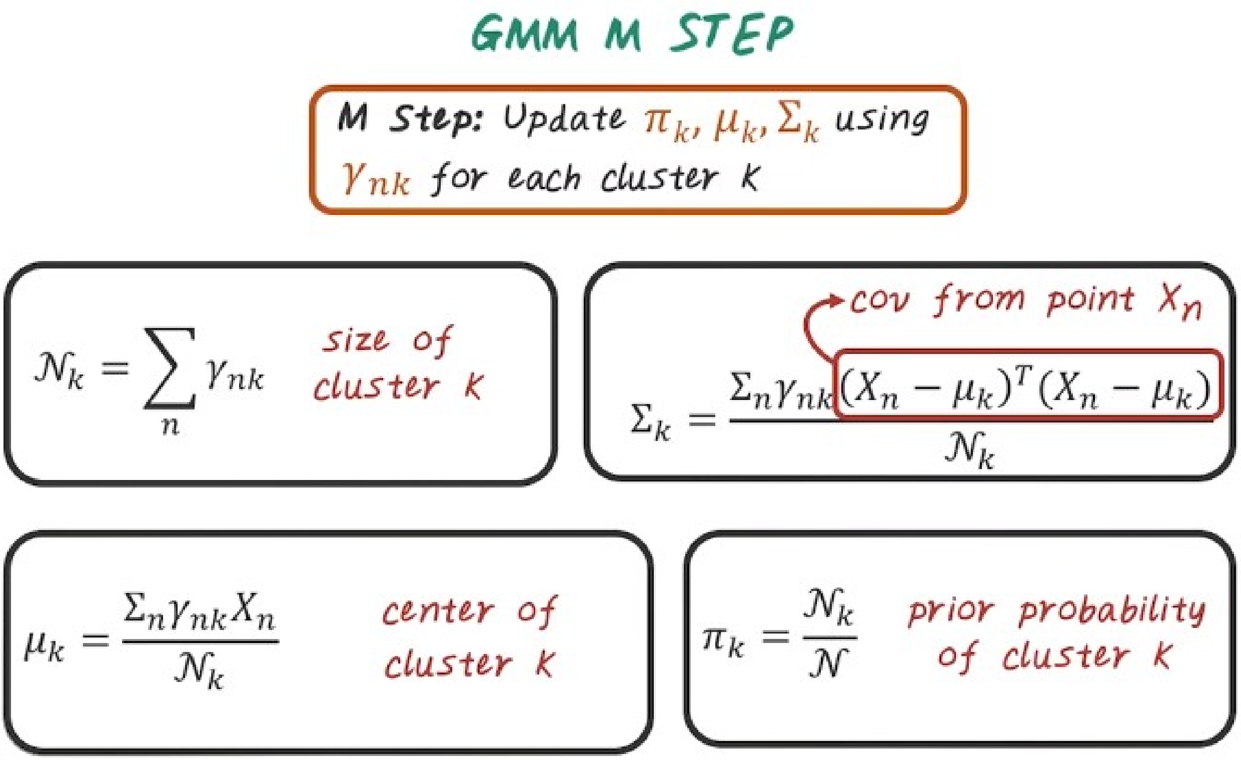
GMM E STEP

E Step: Assign each point X_n a score γ_{nk} for each cluster K

$$\gamma_{nk} = \frac{\overset{\text{Mixing Coefficient}}{\pi_k}\overbrace{\mathcal{N}(x_n|\mu_k,\Sigma_k)}^{\text{Probability } X_n \text{ in cluster } K}}{\sum_{j=1}^{K}\pi_j\mathcal{N}(x_n|\mu_j,\Sigma_j)}$$

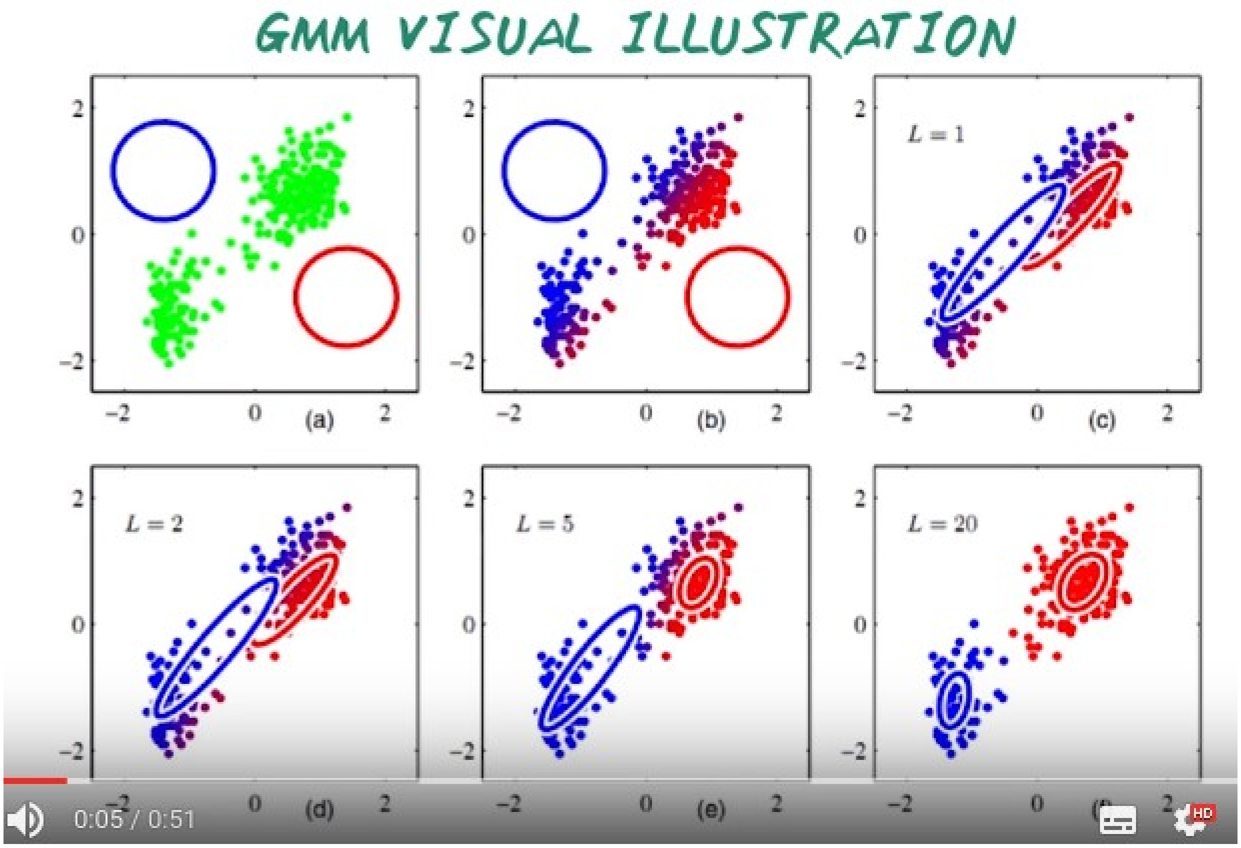
下標 n 表示第 n 個 data point.

Next, let's talk about the E step. In this step, we'll assign each data point a score gamma nk for each cluster K. And the gamma nk can be calculated with this equation. And the numerator has two terms, pi k, the mixing coefficient for cluster K, the probability of xn in cluster K. And the denominator has to normalize this assignment score to between zero and one. For example, we have one blue Gaussian over here and one orange Gaussian over here. The assignment score for this point is 0.5 for the blue Gaussian and 0.5 for the orange Gaussian. Which means this point is equally likely to belong to the blue cluster or the orange cluster. The assignment score for this point is 0.8 for blue Gaussian, and 0.2 for orange Gaussian. Which means this coin is more likely to come from this blue cluster. And the E step is to going through all these data points by assigning all these assignment scores.

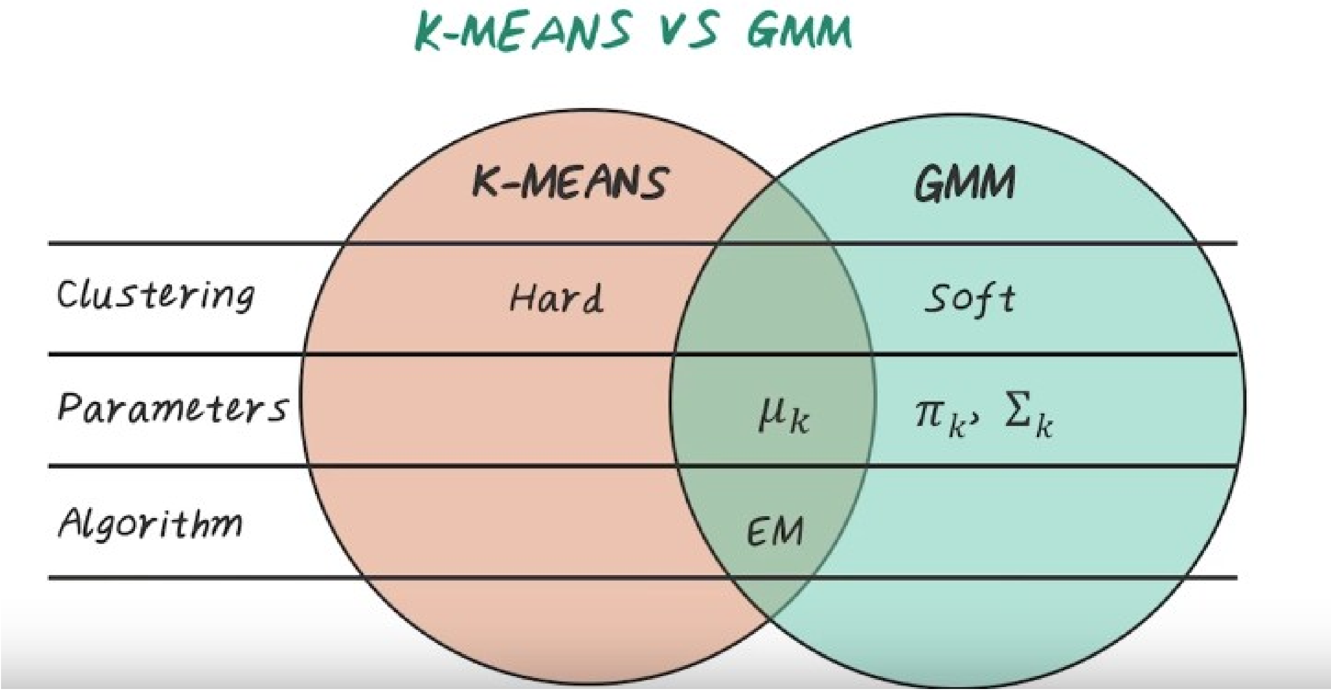

GMM M STEP
M Step: Update π_k, μ_k, Σ_k using γ_{nk} for each cluster k
$\mathcal{N}_k = \sum_n \gamma_{nk}$
size of cluster k
cov from point X_n
$\Sigma_k = \frac{\Sigma_n \gamma_{nk} (X_n - \mu_k)^T (X_n - \mu_k)}{\mathcal{N}_k}$
$\mu_k = \frac{\Sigma_n \gamma_{nk} X_n}{\mathcal{N}_k}$
center of cluster k
$\pi_k = \frac{\mathcal{N}_k}{\mathcal{N}}$
prior probability of cluster k

上圖中, 看的順序為: $\mathcal{N}_k \rightarrow \mu_k \rightarrow \Sigma_k \rightarrow \pi_k$
看 Σ_k 時, 由μ_k的表達式中, 知$\frac{\sum_n \gamma_{nk}}{\mathcal{N}_k}$就是個 求平均 的算符, 這對理解$\Sigma_k$的表達式很有幫助.

Now let's talk about the M step of Gaussian mixture model. In this step, we'll update all the model parameter pi k, mu k, sigma k, using all the assignment score we have learned from the E step for each cluster K. First let's define an auxiliary term Nk, which equals the sum of all gamma n k for a specific k. And intuitively, Nk is the size of cluster k. The we calculate mu k equals the weighted sum of the data points in cluster k divided by Nk, the size of cluster k. And intuitively, mu k is the center for cluster k. Then the covariance sigma k can be computed with this equation. These particular terms corresponding to the covariance from the data points xn. Intuitively, sigma k is the covariance of all data points in cluster k. And finally, the mixing coefficient pi k is proportional to the size of the cluster Nk. In other words, pi k is a prior probability for data points to belong to cluster k.

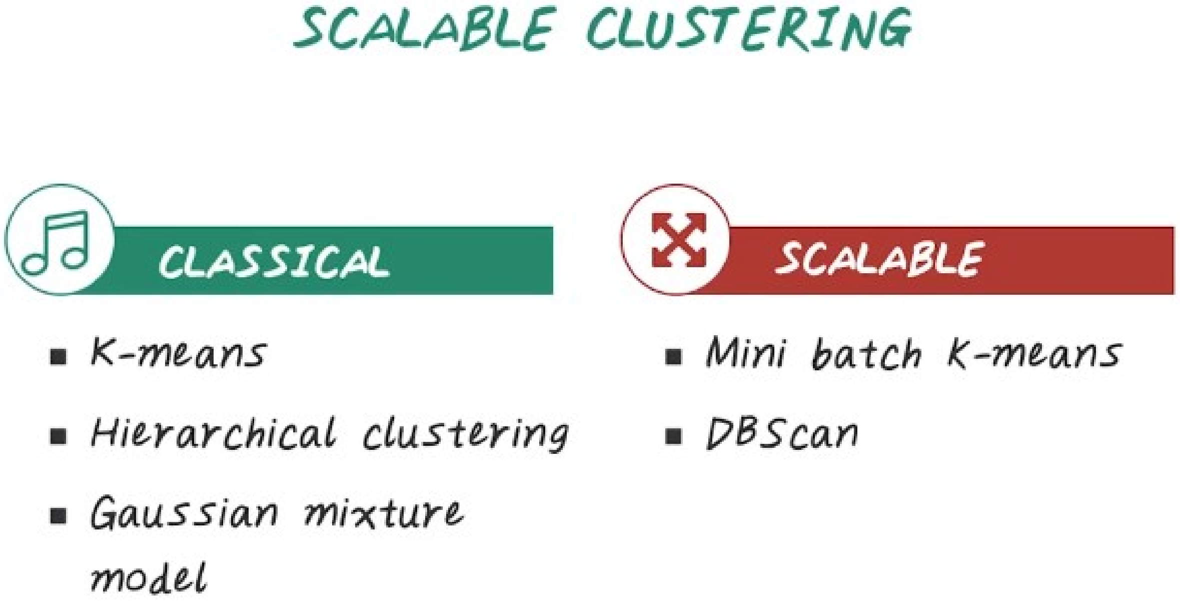


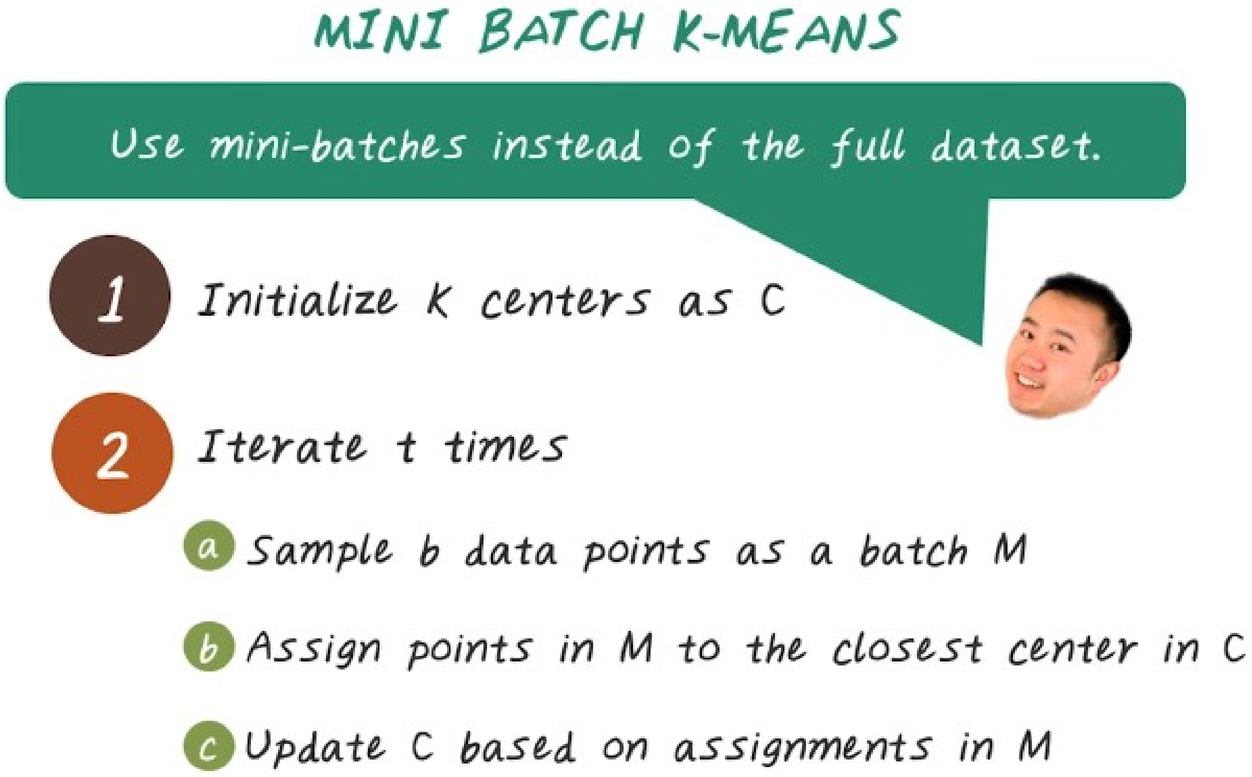
13. Now let's visually illustrate how Gaussian mixture model works. The goal is to run Gaussian mixture model over this set of points. We first initialize two Gaussians indicated by the blue and red circles. Then we compute assignment scores for each data points. The bluer points indicate the assignment score for the blue cluster are higher. Likewise, the redder points indicate the assignment score for the red cluster are higher. Then we update the Gaussian mixture model's parameters based on the new assignment scores. In particular we update mu, sigma and pi. Then we iterate the E step and N steps until converges. And here are the final result of Gaussian mixture model after 20 iterations



14. Now let's compare two clustering method. K-means and Gaussian Mixture Model. In term of clustering algorithm itself, K-mean is a hard clustering algorithm. Each data points only belong to one cluster. Gaussian Mixture is a soft clustering algorithm. Each data points can belong to multiple clusters with different weights. Or K-means the only parameter is the center, mu k, for each cluster, but for Gaussian mixture, we have three parameters. In addition to the center, mu k, we also have mixing coefficient, pi k, and covariance, sigma k. Avadöles: Both K-means and Gauss mixture model utilize EM algorithm where they iteratively update the clustering assignment and model parameters.

我 Conversant 電面時被到了 k-means 和 EM 的關係, 我沒答出來, 以上紅字即說得很好. 以下來自網路:
I keep reading the following: k-means is a variant of EM, with the assumptions that clusters are spherical. (後面有人解釋為何是 spherical, 但我沒認真看. 此處只要知道 k-means 是 EM 的 variant 即可, 然後看上面紅字).





MINI BATCH K-MEANS

Use mini-batches instead of the full dataset.

1 Initialize K centers as C

2 Iterate t times

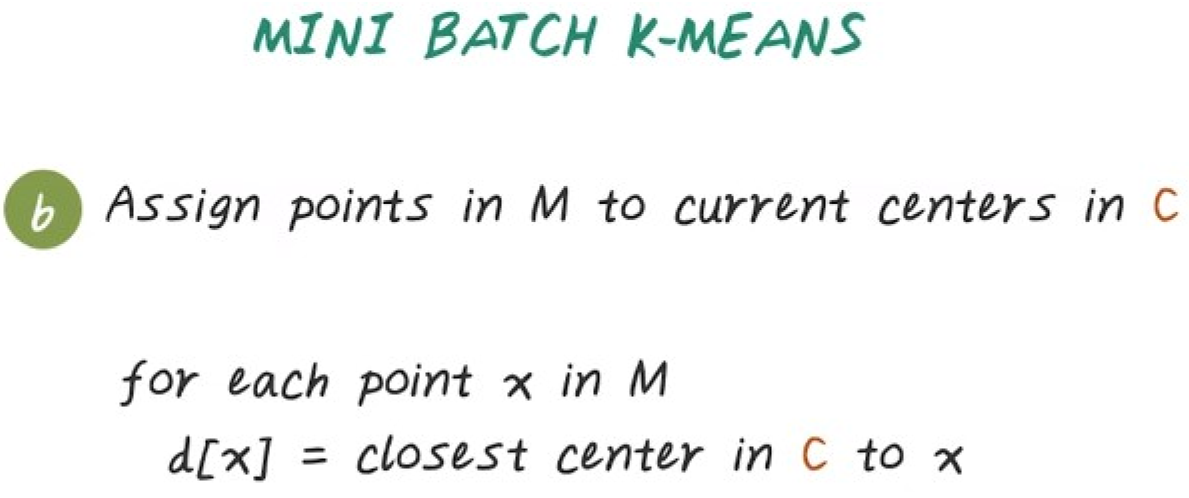
a Sample b data points as a batch M

b Assign points in M to the closest center in C

c Update C based on assignments in M

Batch: 一群
C 是那 k 個 center 的集合, M 是那 b 個 data points 的集合.

15. So far, we'll have talked about three classical clustering algorithm. K-mean, hierarchical clustering, and Gaussian mixture model. Next, we'll describe two scalable clustering algorithm. Mini batch K-means and DBScan. So one problem with K-mean's algorithm is that it needs to assign all the data points to cluster centers at each iteration. When we have a billion data points, this can be very expensive. Mini-batch K-means avoids doing these global assignments by working with smaller batches. And here's the high-level algorithm. To start, we initialize K centers as a set C. And there are many different ways for this initialization. For example, we can randomly sample K data points as the centers. Then we update those centers t times, as the following. We first sample b data points to form a batch M. Then we assign the data points in M to the closest center in C. Then we update the centers based on the assignments in M.

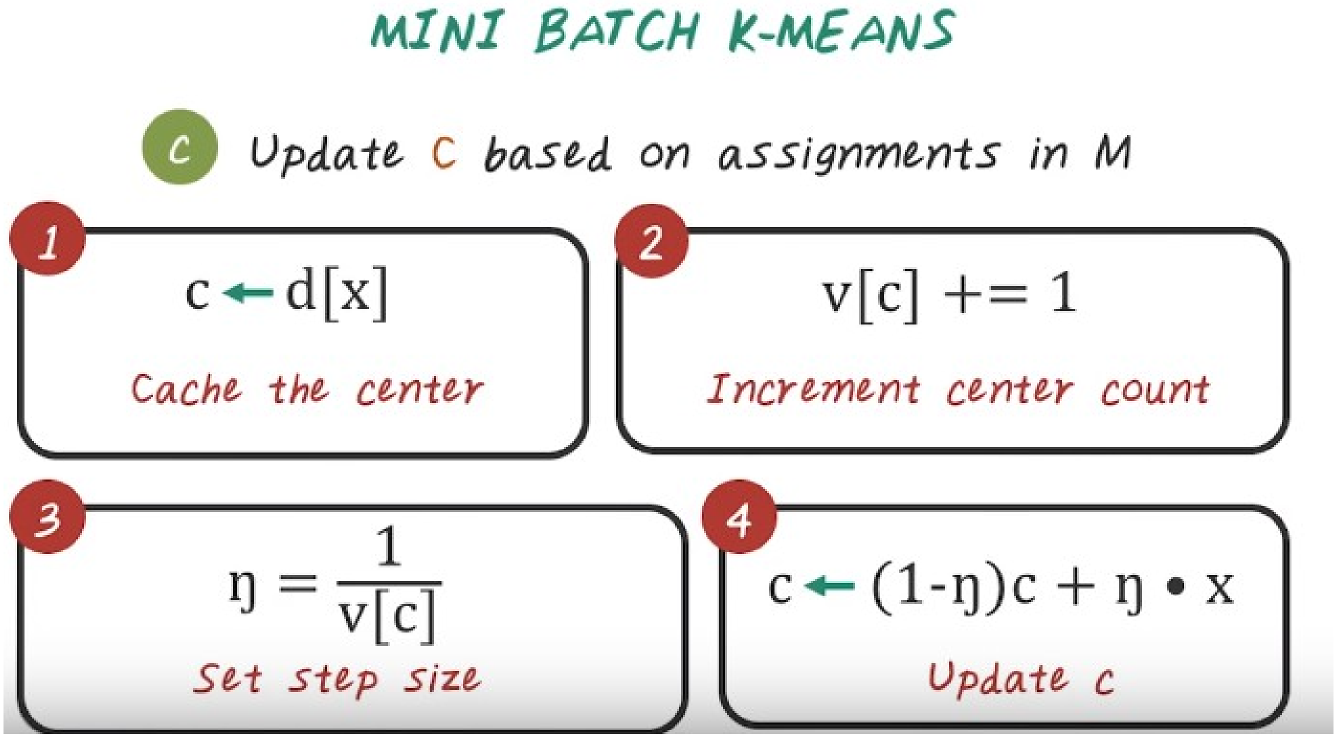


MINI BATCH K-MEANS

b Assign points in M to current centers in C

```
for each point x in M
  d[x] = closest center in C to x
```

Next let's look at step b and c and more details. In step b, for each data points x in mini batch M, we'll first find it closest center C to x, then cache this result in the hash map d[x] so that later we can retrieve this center quickly.



MINI BATCH K-MEANS

c Update C based on assignments in M

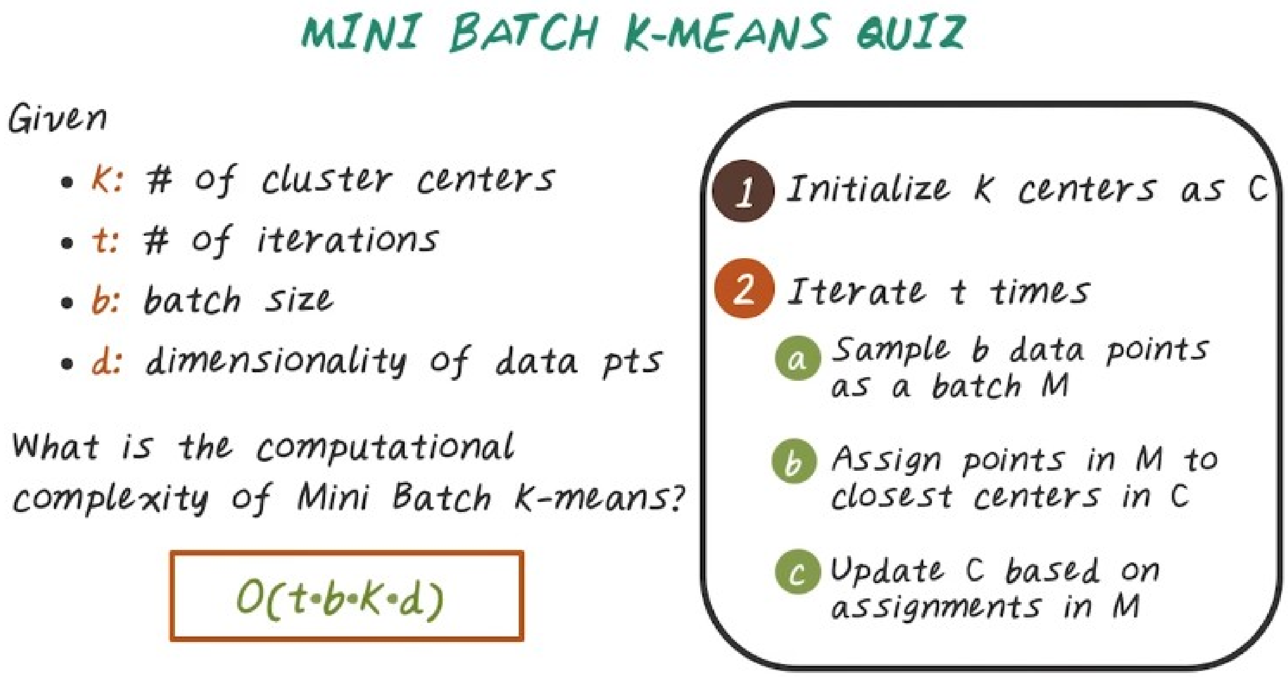
1. $c \leftarrow d[x]$ Cache the center
2. $v[c] \mathrel{+}= 1$ Increment center count
3. $\eta = \frac{1}{v[c]}$ Set step size
4. $c \leftarrow (1-\eta)c + \eta \cdot x$ Update c

別忘了上圖的順序要按 1->2->3->4 看

Next, in step C we update the center based on the assignments in this mini-batch M. In this step, again we go through all the data points x in this mini-batch. First, we retrieve the corresponding center, then we increment the corresponding center count by one. Then we set the step size as the inverse of the center count. Finally, we update the center by 1- eta times the old center c + eta times this data points

x. So intuitively, we move the old center towards this data point x by the step size eta. So note that as the center count increases, the step size becomes smaller and smaller. As a result, the center update becomes smaller and smaller over time. Therefore, the center will eventually be stabilized.

16. Now let's do a quiz on our mini batch k-means. Given k is the number of cluster centers, t is number of iterations, and b is the batch size, and d is dimensionality of the data points. What is the computational complexity of mini batch k-means? And here is the pseudo code of the algorithm to help you answer the question and put your answer in this box.



MINI BATCH K-MEANS QUIZ

Given

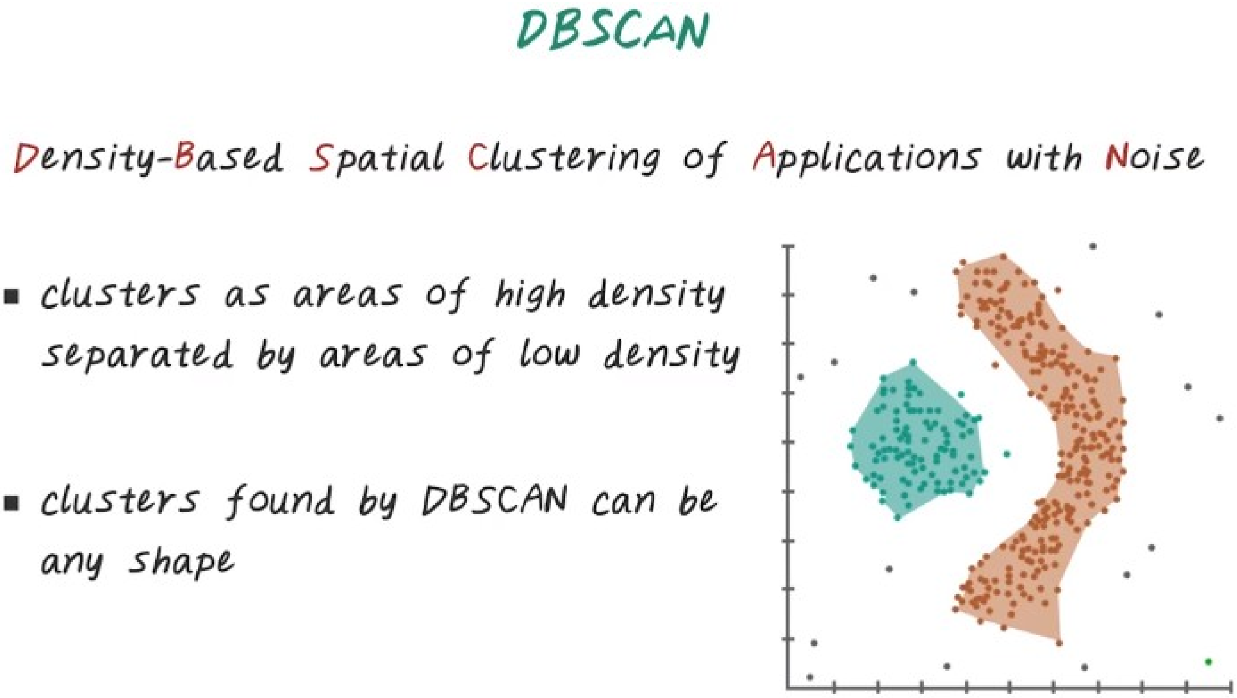
- K: # of cluster centers
- t: # of iterations
- b: batch size
- d: dimensionality of data pts

What is the computational complexity of Mini Batch K-means?

O(t•b•K•d)

```
1 Initialize K centers as C
2 Iterate t times
    a Sample b data points as a batch M
    b Assign points in M to closest centers in C
    c Update C based on assignments in M
```

17. And the answer is, big olive, t time b times k times d. The reason is, the most expensive step in this algorithm is to assign data points to its closest center. Since we have b data points in each batch and K centered. So, we have to compute k times b comparisons. And each data point is of dimensionality d, so each iteration, the total cost is b times k times d. Since we iterate this algorithm t times, the total cost becomes t times b times k times d.

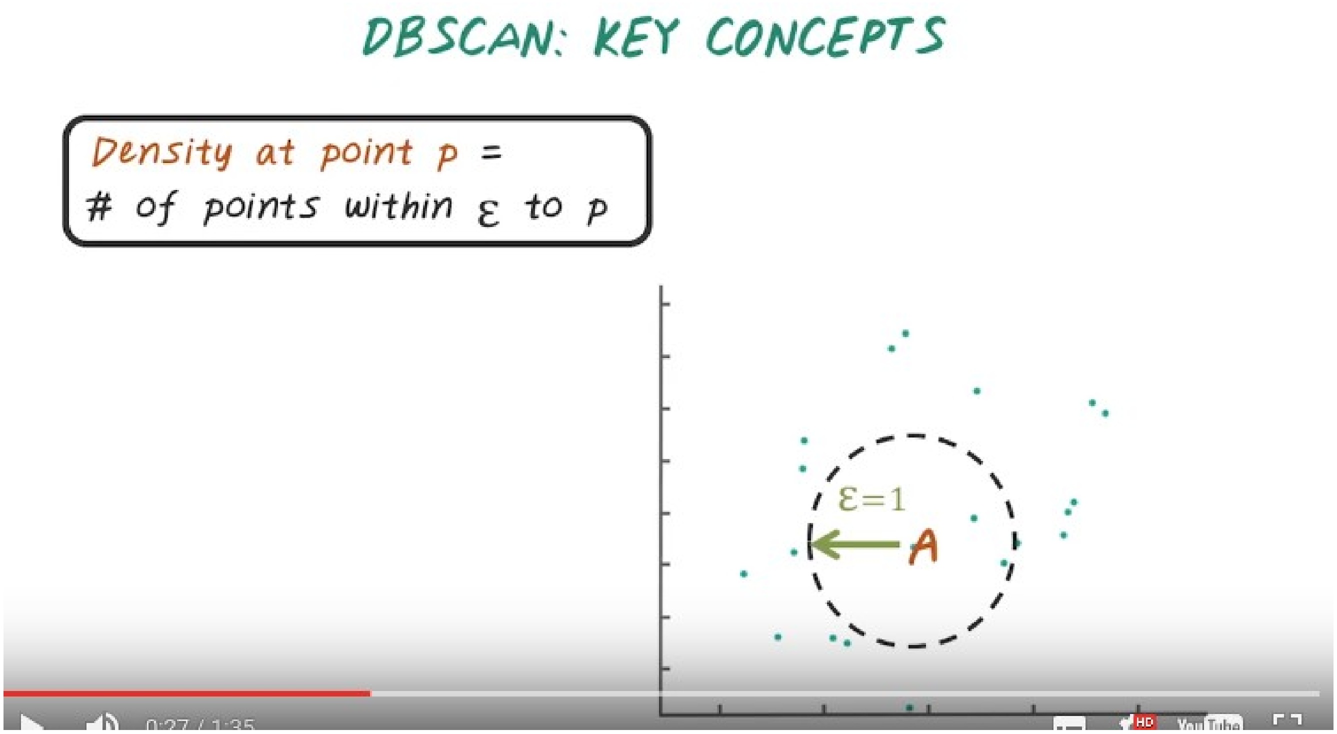


DBSCAN

Density-Based Spatial Clustering of Applications with Noise

- clusters as areas of high density separated by areas of low density
- clusters found by DBSCAN can be any shape

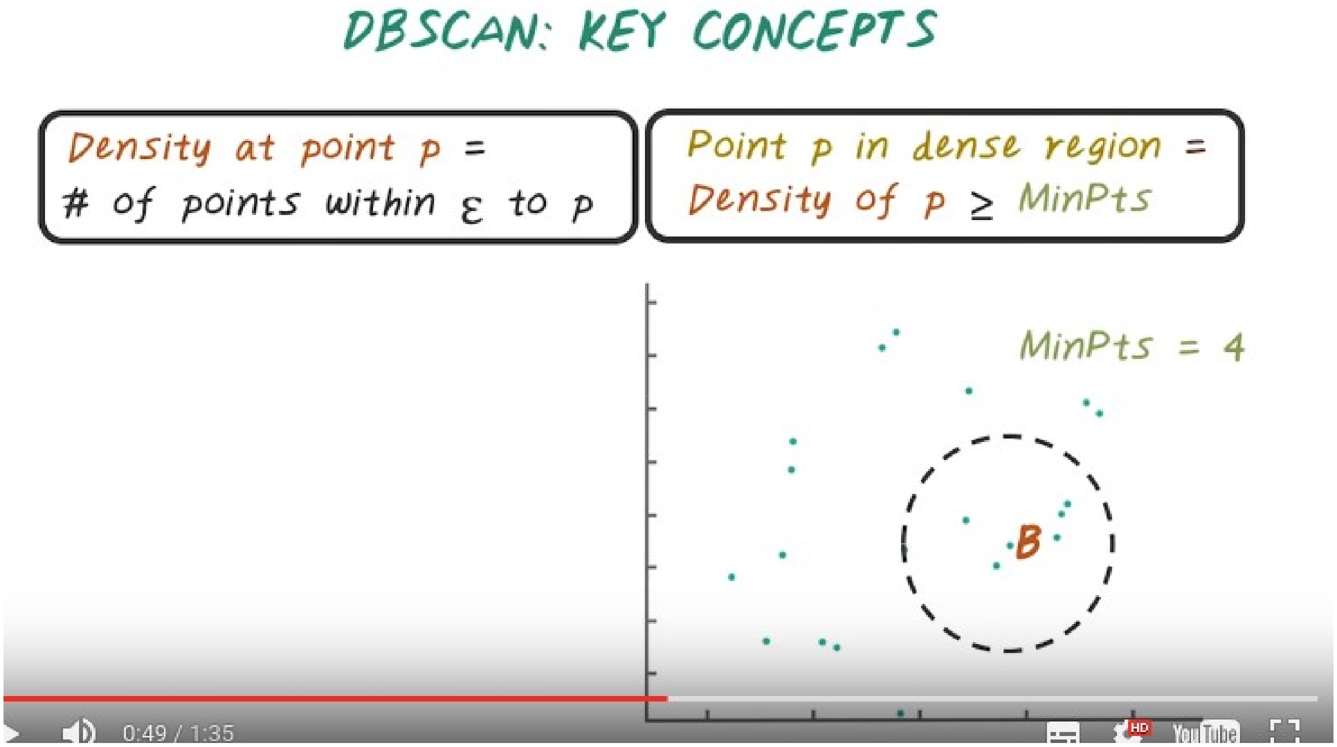
18. Now let's talk about DBSCAN algorithm. DBSCAN stands for Density-Based Spatial Clustering of Applications with Noise. The main intuition behind DBSCAN is to define clusters as areas of high densities separated by areas of low density. As a result, oftentimes the cluster found by DBSCAN can be of any shape. Here's an example. If we run DBSCAN on this data set, we'll have two clusters. The blue areas is one cluster and the red area is another cluster. Both are defined by high-density regions, and they are separated by low-density regions. More details about the algorithm can be found in the instructor notes.



DBSCAN: KEY CONCEPTS

Density at point p =
# of points within ε to p

19. In order to explain DBSCAN algorithm we have to introduce some key concepts. First, how do we measure the density? In DBSCAN, the density at a point p is defined as the number of points (包括p自己) within absolute distance to p. For example, here are a set of points in two dimensional space. This point A was equal to one, has density three, because there are three points in this circle.

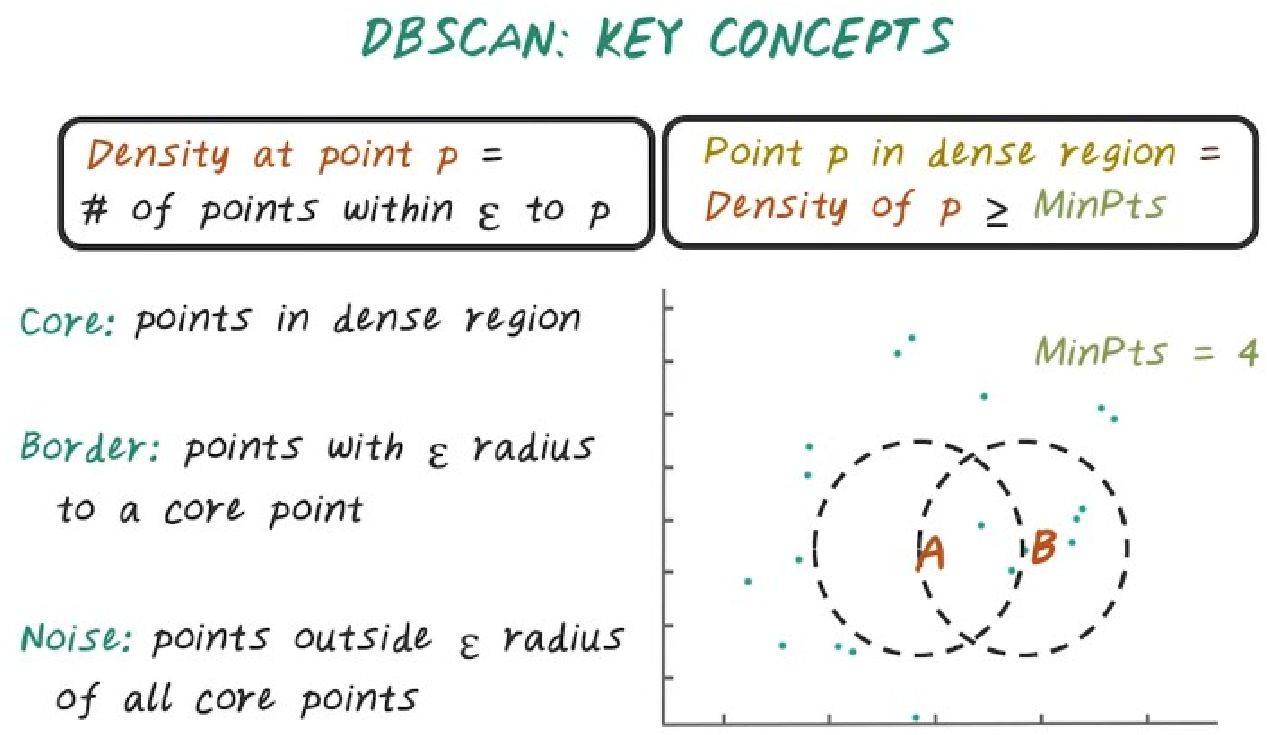


DBSCAN: KEY CONCEPTS

Density at point p =
# of points within ε to p

Point p in dense region =
Density of p ≥ MinPts

Then what is the dense region? At data point P, in a dense region, of the density of P is greater than some threshold, the mean data points. For example, this data point B is in the dense region, whereas the mean point equal to four. Because there are more than four points within radius one to data point B, But A is not in a dense region, because there are only three points within radius one to A, which is less than the mean points four.



DBSCAN: KEY CONCEPTS

Density at point p =
# of points within ε to p

Point p in dense region =
Density of p ≥ MinPts

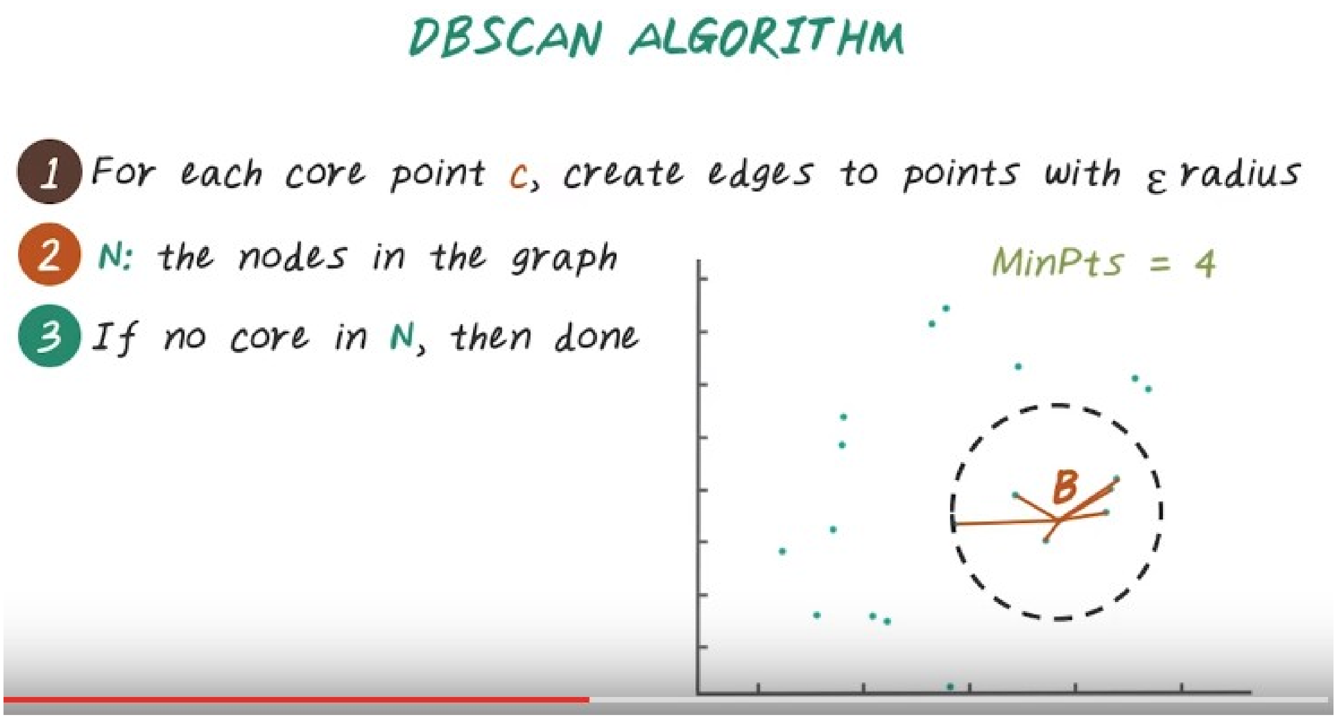
Core: points in dense region

Border: points with ε radius to a core point

Noise: points outside ε radius of all core points

注意 Noise: points outside ε radius of all core points.

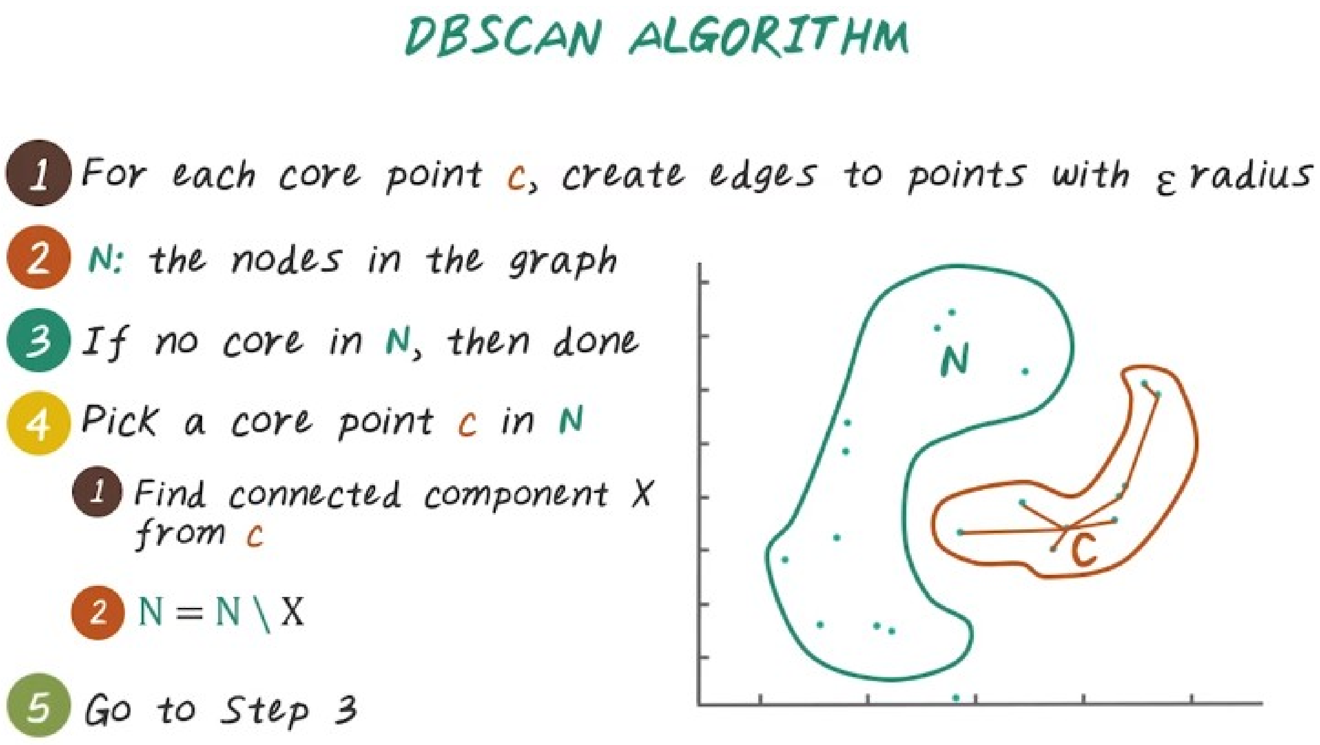
Now we understand the density and what is the dense region. Next we can define a set of key concepts. Core, border, and noise. Core points are points in the dense region. For example, B is a core point because it's in the dense region. And the border points, are points was in absolute distance to a core point. For example A is a border point because A was within absolute distance with B and B is a core point. And all the other points outside absolute distance to the core points are noise.



DBSCAN ALGORITHM

1. For each core point c, create edges to points with ε radius
2. N: the nodes in the graph
3. If no core in N, then done

20. Now we understand the key concept of core point, border points, and noise. We can use that to explain the algorithm. To start, for each core point c, we create edge to the points of absolute distance. For example, B is a core point. Then we connect B, to all the points within absolute distance to B, and those are the edges we created here. Next, we define N as a set of nodes in the graph, which are really all the data points in the data set. If no core points in the set N then we're done.

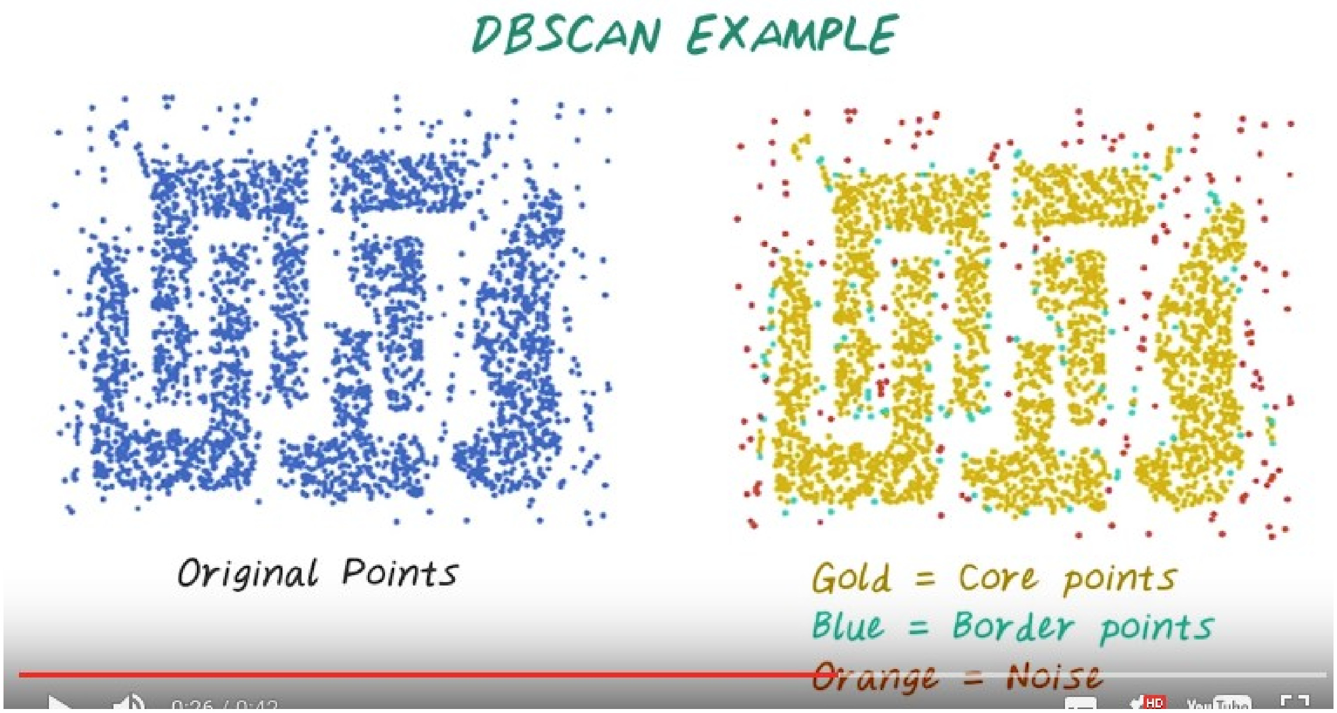


DBSCAN ALGORITHM

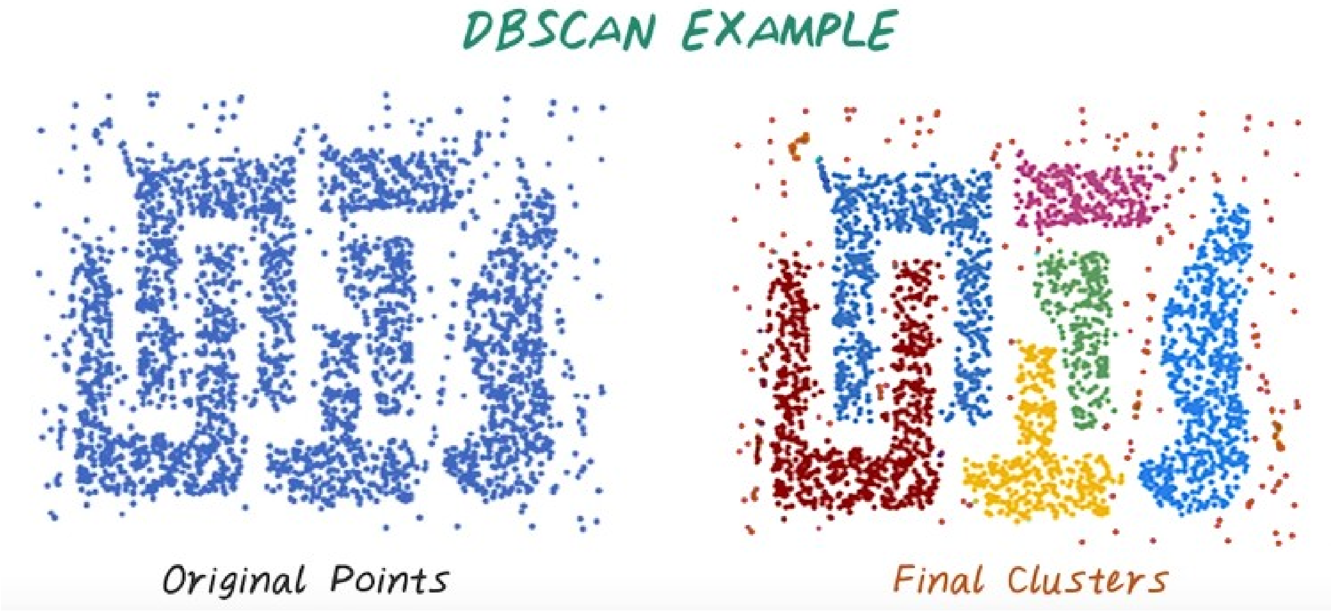
1. For each core point c, create edges to points with ε radius
2. N: the nodes in the graph
3. If no core in N, then done
4. Pick a core point c in N
   1. Find connected component X from c
   2. $N = N \setminus X$
5. Go to Step 3

第4.1步的意思應該是找出所有在C的border內的點, 這些點的集合稱為X.
第4.2步中的N = N \ X不是N除以X的意思, 而是在N中 將X中的所有點 去掉.

If there are still core points in the set N we pick a random core point C. Then we find the connected components from C. For example, if we pick this point over here as the core point C, then the connected component can look like the following. Then we update the remaining set of points N by removing all the points in x from N. So this operator indicate the set difference by removing x from N. Then we go back to step three, and continue the iteration.

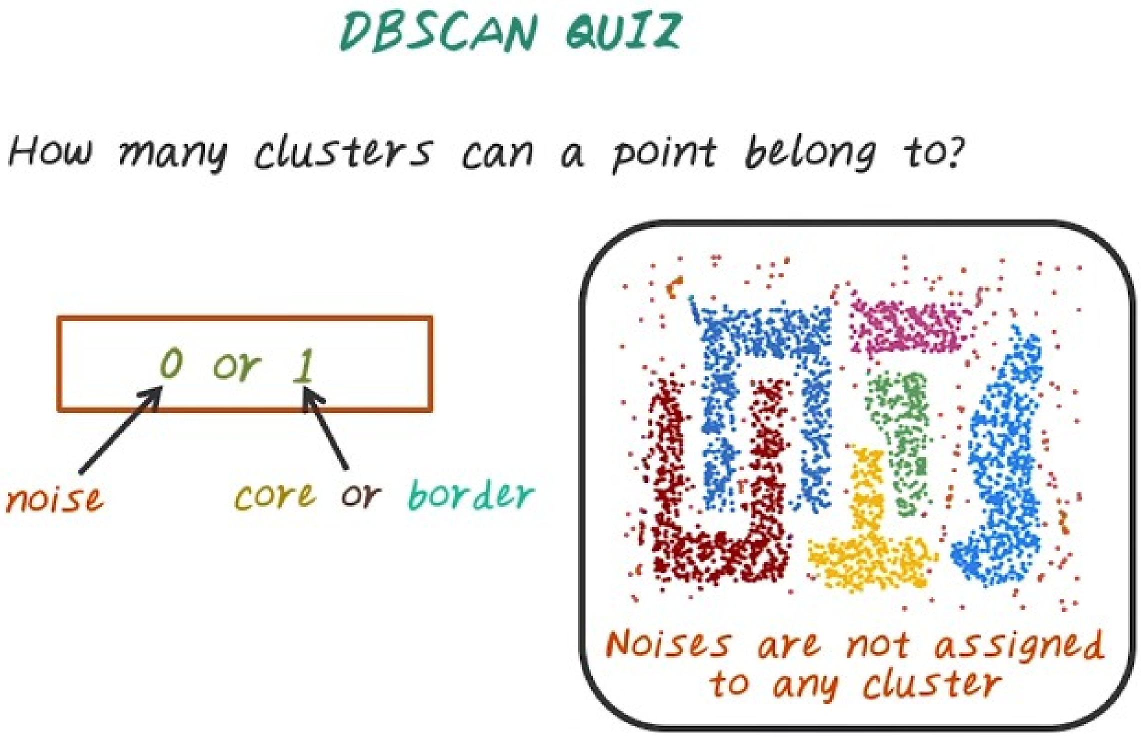


21. Now let's look at some examples of DBSCAN giving a set of two dimensional data points. Look like this. We want to find a set of clusters using DBSCAN. The first step of DBSCAN will try to classify all those data points into this three categories. The core points are those gold color points. And the border points are those blue color points. And the noises are the orange color points.



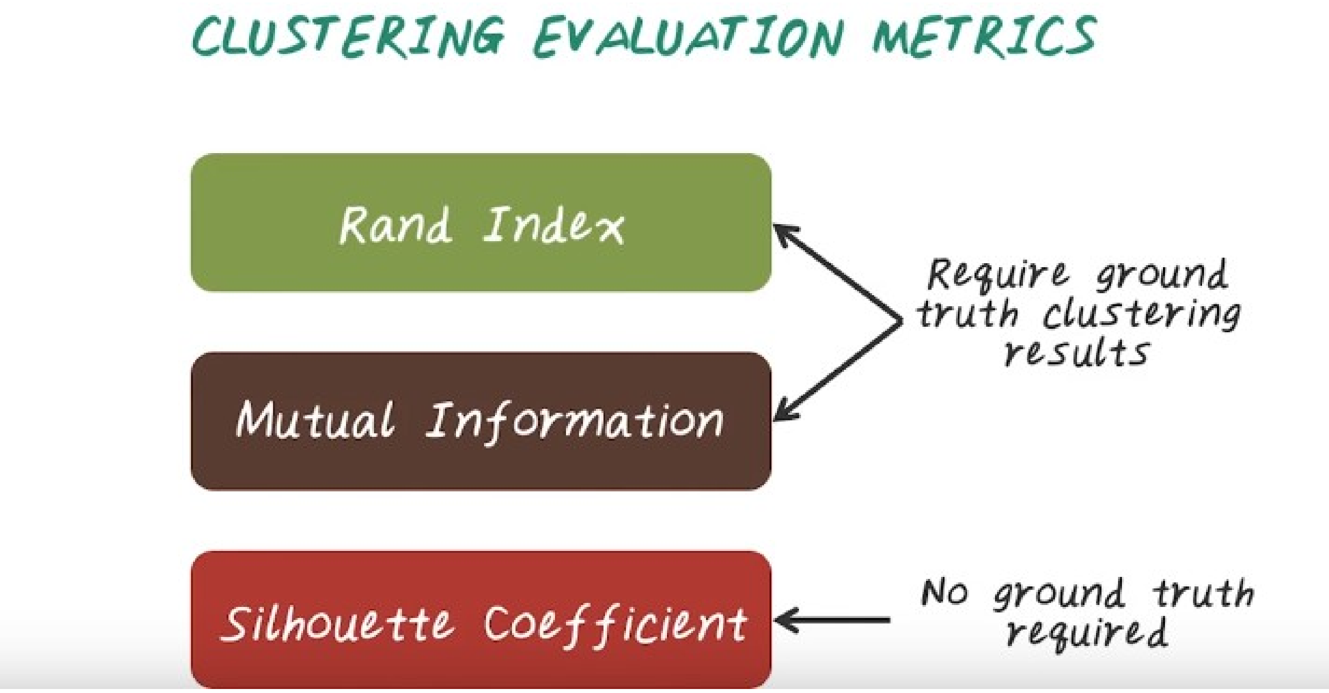
Then we iterate through DB scan algorithms, we'll find this one, two, three, four five, six clusters. And you notice that there are points not belong to any of those clusters, they are the noises.

22. Here's a quiz for DBscan. So, how many cluster can a datapoint belong to, using DBscan algorithm? Put your answer in this orange box.


DBSCAN QUIZ
How many clusters can a point belong to?
0 or 1
noise
core or border
Noises are not assigned to any cluster

23. And the answer is 0 or 1. If the data point is a noise, it won't belong to any cluster, so it will be 0. If the data point is a core points or a border point, then it will belong to 1 cluster. So the key here is all

the noises are not assigned to any cluster.



CLUSTERING EVALUATION METRICS

Rand Index

Mutual Information

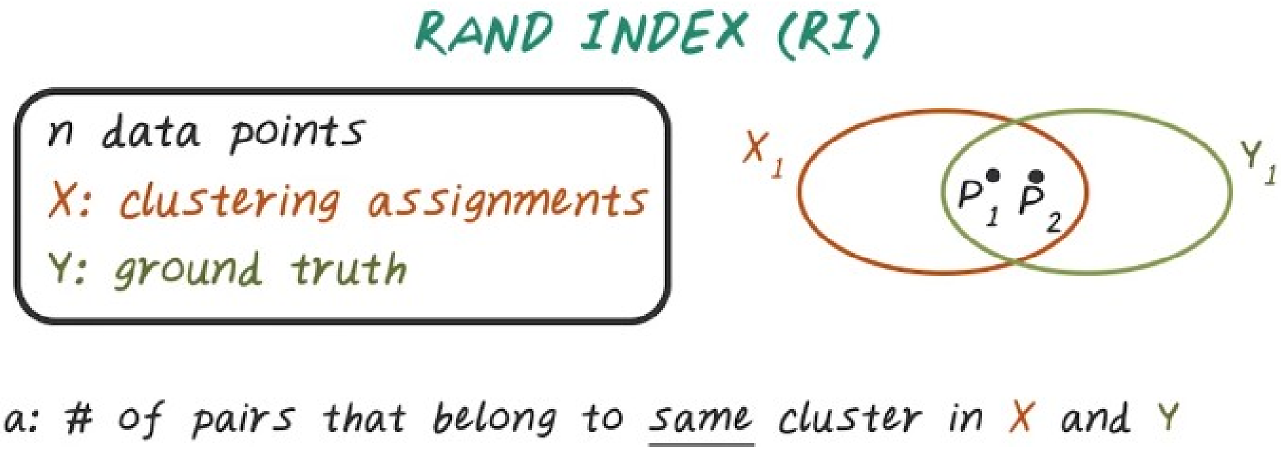
Silhouette Coefficient

Require ground truth clustering results

No ground truth required

silhouette: 轮廓

24. So far we have talked about clustering algorithms. Next we introduce a set of evaluation metrics for clustering. In particular, we'll talk about rand index, mutual information, and silhouette coefficient. And rand index and mutual information requires we know the ground truths of the clustering result. And silhouette coefficient does not require any ground truths.



RAND INDEX (RI)

n data points

X: clustering assignments

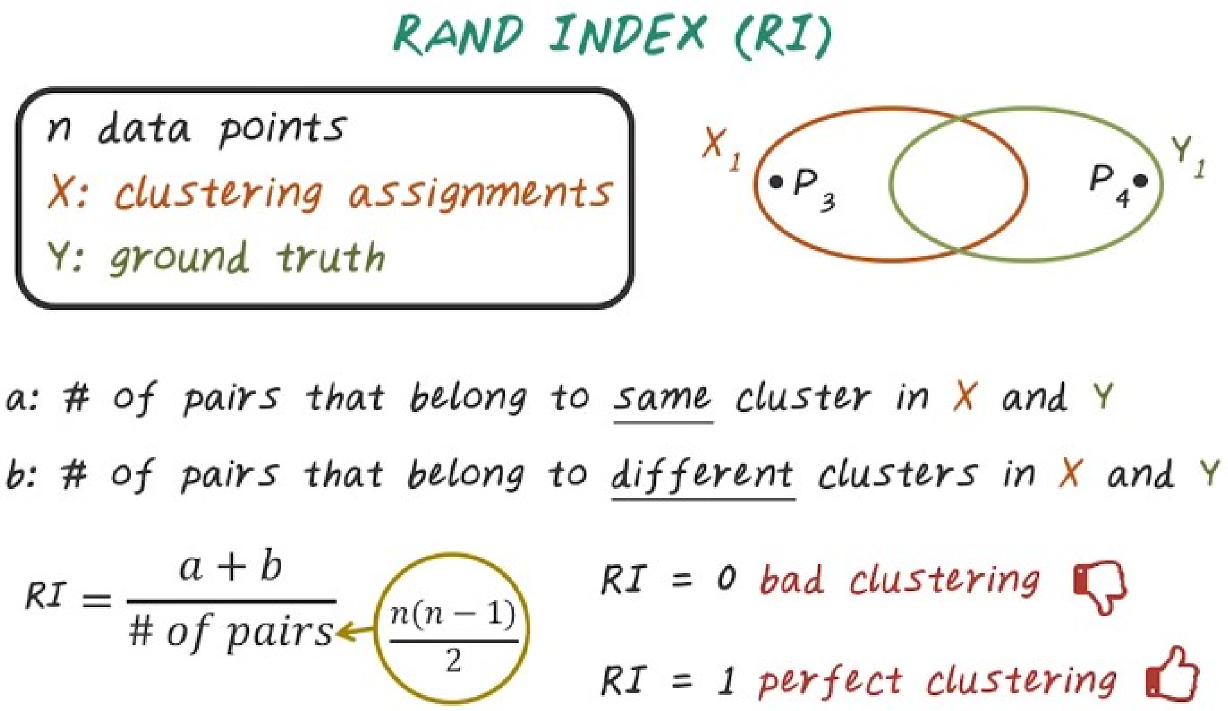
Y: ground truth

a: # of pairs that belong to same cluster in X and Y

注意是# of pairs

25. Now let's talk about Rand Index or RI. Given n data points, let's say X is the clustering assignment from the algorithm and Y is the ground truth. In here, we illustrate a cluster X1 that comes from the algorithm and cluster Y1 come from the ground truth. Then we compute the term a, which is the number of pairs that belong to the same cluster in X and in Y. For example these two data points, P1 and P2 belong to the same cluster in X because they both belong to X1. And also, they belong to the

same cluster in Y because they both belong to Y1. And this pair will be counted towards this term. We want to find all such pairs that belong to the same cluster in both X and Y.



RAND INDEX (RI)

n data points

X: clustering assignments

Y: ground truth

a: # of pairs that belong to same cluster in X and Y

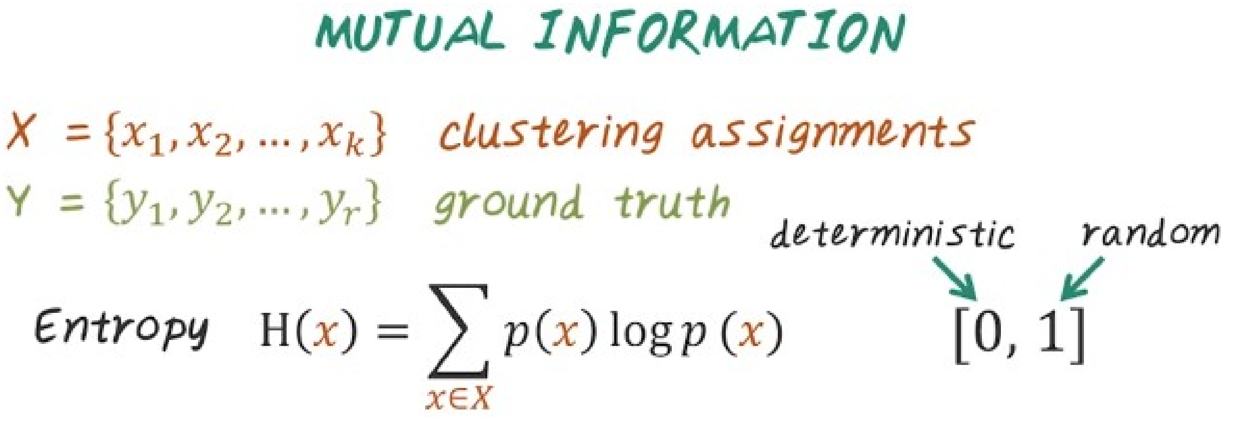
b: # of pairs that belong to different clusters in X and Y

$$RI = \frac{a+b}{\# \text{ of pairs}} \leftarrow \boxed{\frac{n(n-1)}{2}}$$

RI = 0 bad clustering

RI = 1 perfect clustering

Then we compute another term, b, corresponding to the number of pair that belong to different cluster in X and Y. For example, these two points P3 and P4. The P3 belong to X1, and P4 belong to Y1. They belong to different clusters. And we want to find all such pairs. So once we know A and B, the Rand Index is defined as a plus b divided by the total number of pairs. In this case, the total number of pairs is n times n minus 1 divided by 2. The Rand Index is between 0 and 1. 0 means bad clustering assignment, and 1 means perfect clustering assignment. So, in general, we want to have an algorithm with high Rand Index.



MUTUAL INFORMATION

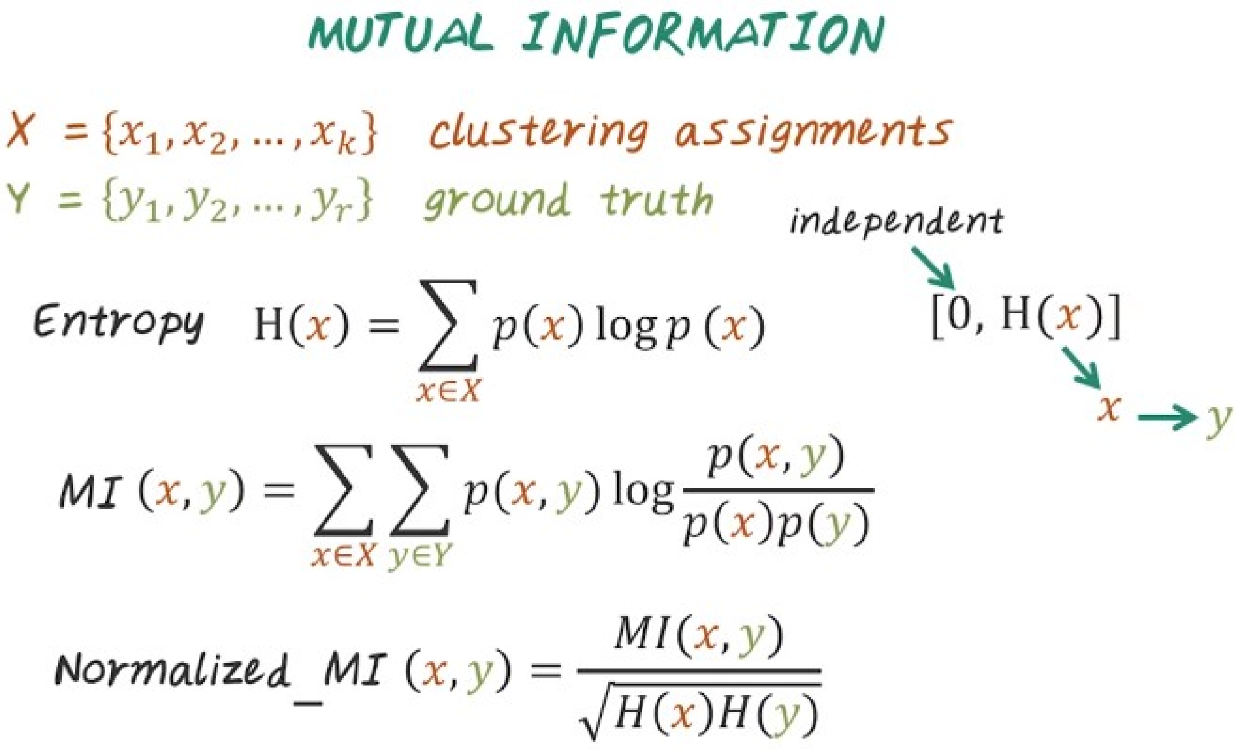
$X = \{x_1, x_2, \dots, x_k\}$ clustering assignments

$Y = \{y_1, y_2, \dots, y_r\}$ ground truth

Entropy $H(x) = \sum_{x \in X} p(x) \log p(x)$

26. Like grand index, mutual information is another way to measure clustering quality. And mutual information is a concept from information theory which measures the mutual dependency of two random variables. In this case, the two random variables are clustering assignment x and the ground truth assignment y. More specifically, X has k cluster (不是 data points), X1, X2 to Xk, and Y has r cluster, Y1, Y2 to Yr, and then the entropy of X is defined as sum of probability PX times log of PX.

And this entropy term measures uncertainty of x. And the entropy term is between 0 and 1. And 0 means the variable is deterministic. And 1 means the variable is completely random (因為我早就知道 entropy 越大就越混亂).



MUTUAL INFORMATION

$X = \{x_1, x_2, \ldots, x_k\}$ clustering assignments

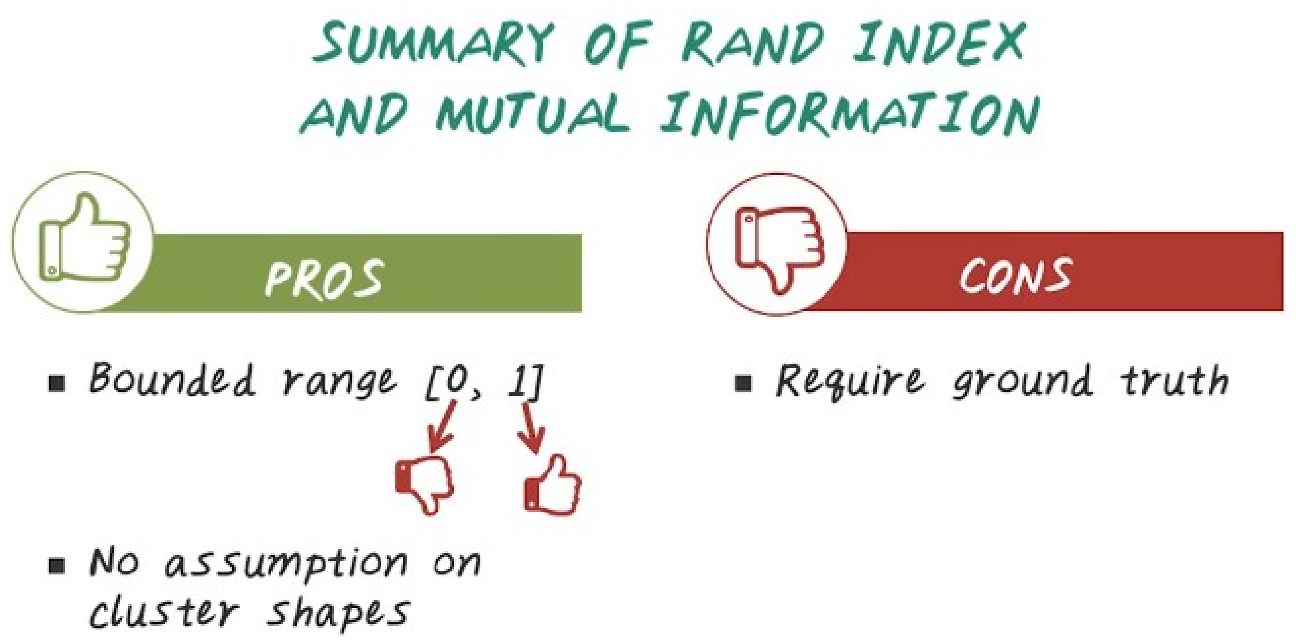
$Y = \{y_1, y_2, \ldots, y_r\}$ ground truth

Entropy $H(x) = \sum_{x \in X} p(x) \log p(x)$

$$MI(x,y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$$

$$Normalized\_MI(x,y) = \frac{MI(x,y)}{\sqrt{H(x)H(y)}}$$

Then we define the mutual information between x and y. As sum over both x and y of the joint probability p (x, y) times log of p (x, y) divided by the marginal probability p (x) times marginal probability of p (y). And the range of mutual information is between 0 and entropy of x. So when mutual information equals 0. Means x and y are independent. When mutual information equals H(x), then it means y is completely determined by x. If we apply mutual information as cluster and evaluation metric, then higher value means good clustering assignment. Sometimes, people want the metric to be normalized between 0 and 1. So in this case, we can define this normalized mutual information as the mutual information between x and y, divided by the square root of entropy of x times entropy of y.



SUMMARY OF RAND INDEX AND MUTUAL INFORMATION

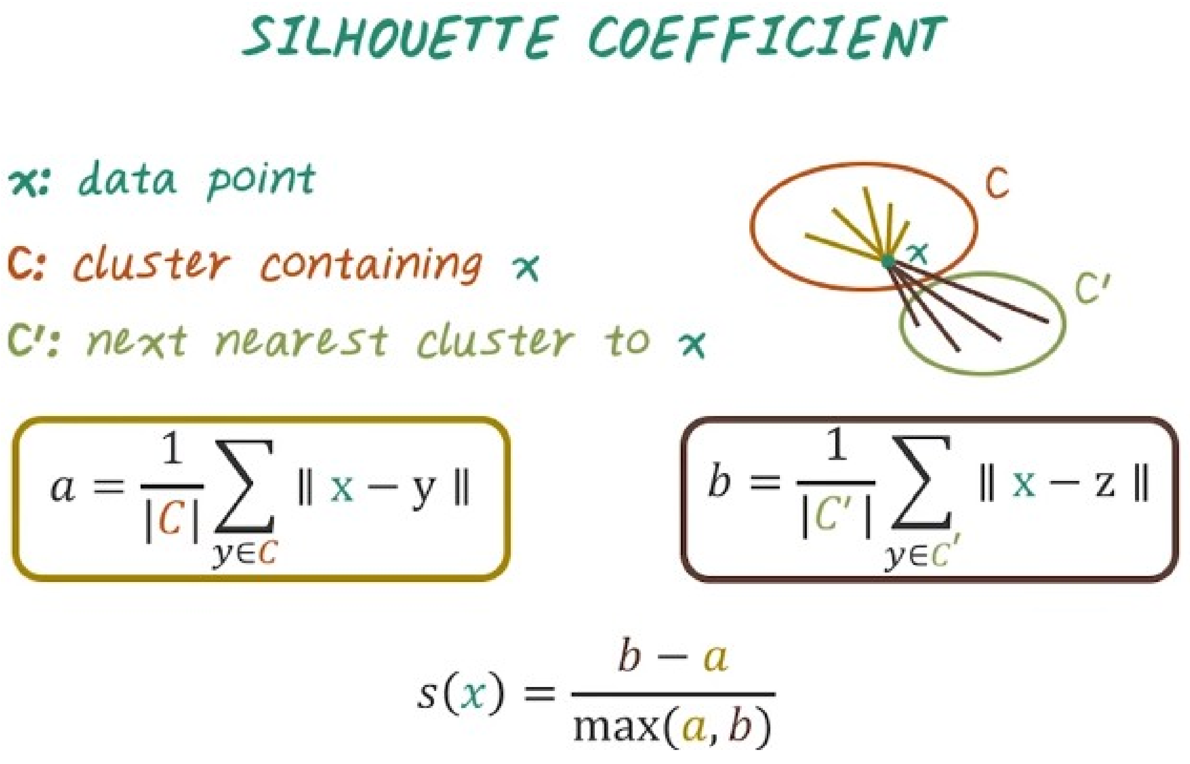
PROS

- Bounded range [0, 1]
- No assumption on cluster shapes

CONS

- Require ground truth

27. There are a lot of commonality between Rand index and mutual information. So what are the pros and cons of this Q evaluation metric. They both provided bounded ranges when the metric is close to 0, which means bad clustering assignment. When it's close to one, means good cluster assignment. And there's no assumption of clustering shape, so you work with arbitrary cluster algorithms but a disadvantage is it will require ground truth cluster assignment. In many cases we don't know the ground truth, even the data set, so this will be a big limitation for both rand index and mutual information.



SILHOUETTE COEFFICIENT

x: data point

C: cluster containing x

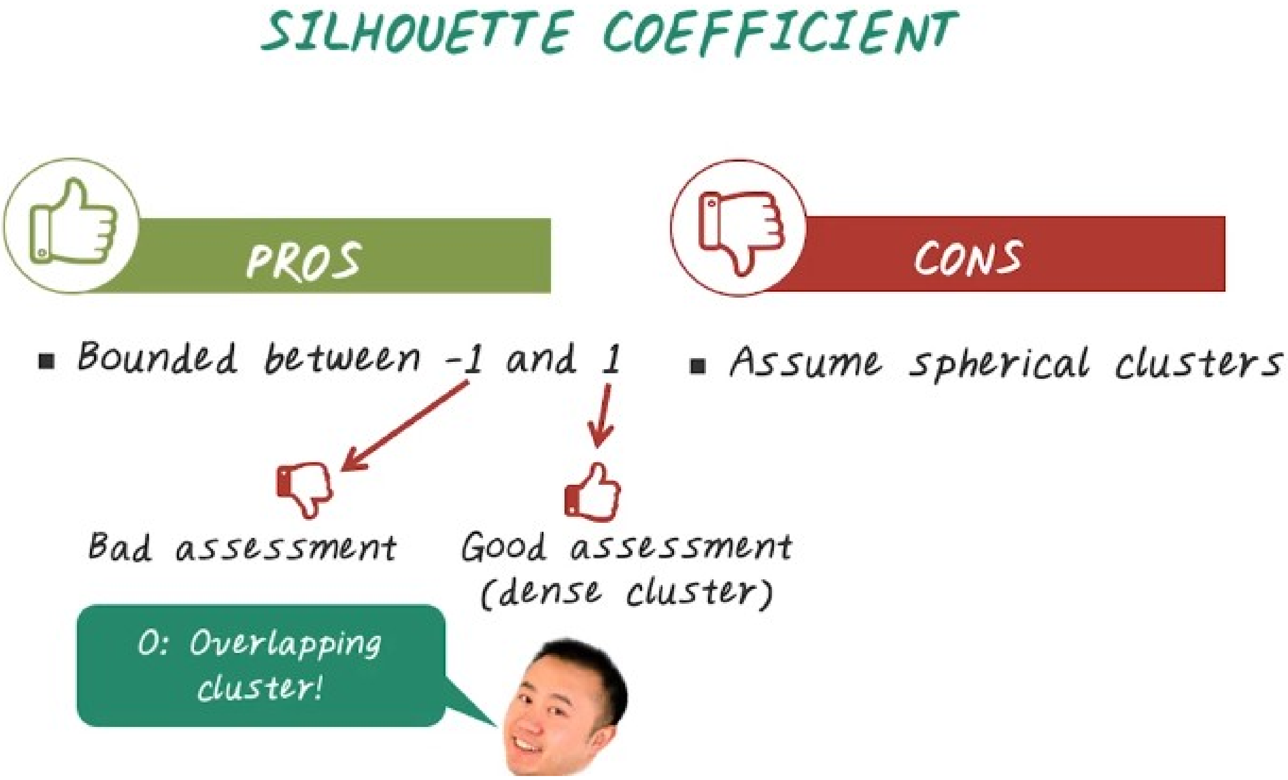
C': next nearest cluster to x

$$a = \frac{1}{|C|}\sum_{y \in C} \| x - y \|$$

$$b = \frac{1}{|C'|}\sum_{y \in C'} \| x - z \|$$

$$s(x) = \frac{b - a}{\max(a, b)}$$

28. Next, let's introduce silhouette coefficient, which is another popular clustering evaluation metric. In this case, we do not require any ground truce information. Here's the main idea behind silhouette coefficient. Given the data point x, we first want to find the cluster containing x, then we want to find the next nearest cluster to x. So visually, it's illustrated as the following. So we have these two clusters, C contains x and C prime is another cluster that's very close to x. Then we compute this measure a, which is average distance of x to all the data points in cluster C, which means we compute the distance of x to all the other data points in C and find the average. Similarly, we define the term b, which is average distance from x to all the other points in C prime. For example, in this case, we'll compute the distance from x to all the other points in C prime and then take the average. And, the silhouette coefficient on x = b- a divided by max(a,b). The intuition is if the assignment of x is good, then the difference between b and a should be large, then we'll have a large value for this coefficient, which means good clustering assignment. If x is assigned to a cluster which is so close to another cluster, then the difference will be small. In that case, the silhouette coefficient will be small. In that case, it will be a bad clustering assignment.



SILHOUETTE COEFFICIENT

PROS

- Bounded between -1 and 1

Bad assessment

Good assessment (dense cluster)

0: Overlapping cluster!

CONS

- Assume spherical clusters

29. In summary, here are the pros and cons for a silhouette coefficient. Again, it provided a bounded range between minus 1 and 1. And minus 1 means very bad clustering assignment, 1 means very good clustering assignment, and 0, in this case, means overlapping cluster. The limitation here is silhouette coefficient assumes spherical clusters. For the clustering algorithms that generated non spherical clusters, such as DBScan, silhouette coefficient would not be a good evaluation metric.