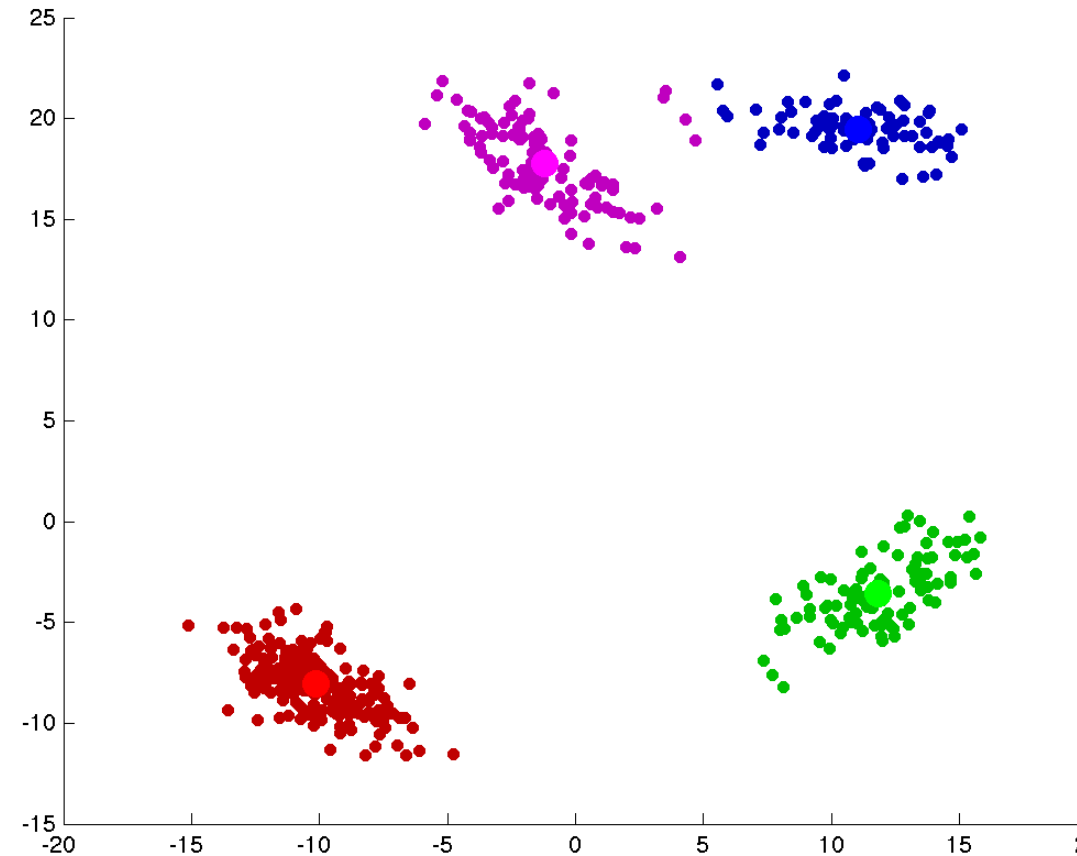


# CPSC 340: Machine Learning and Data Mining

More Clustering

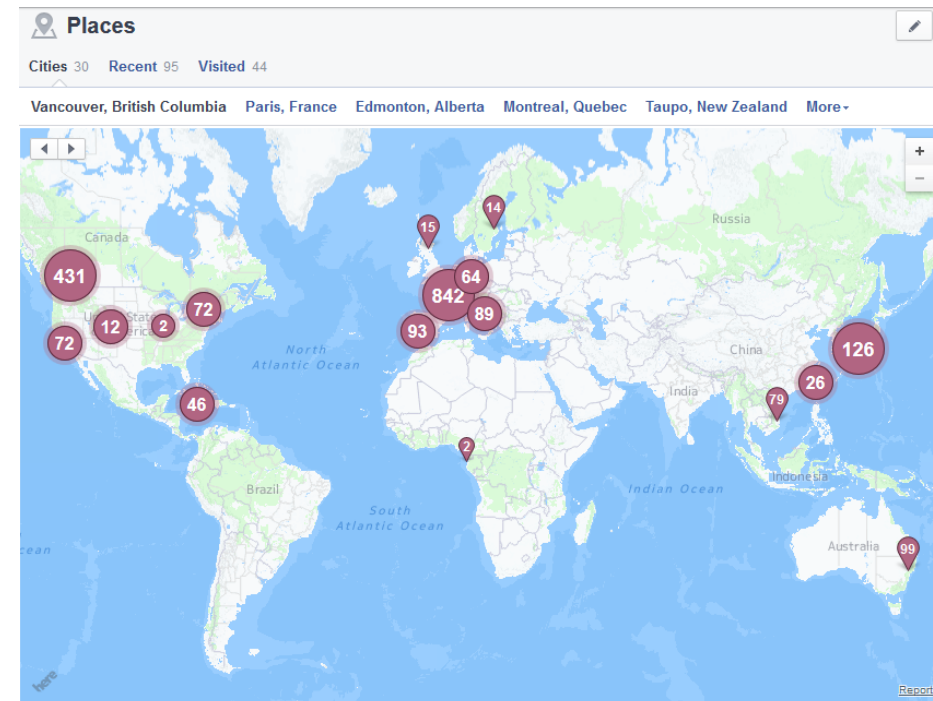
# Last Time: K-Means Clustering

- We want to **cluster** data:
  - Assign examples to groups.
- **K-means clustering**:
  - Define groups by “means”
  - Assigns examples to nearest mean.  
(And updates means during training.)
- Issues with k-means:
  - Fast but sensitive to initialization.
  - Choosing ‘k’ is annoying.



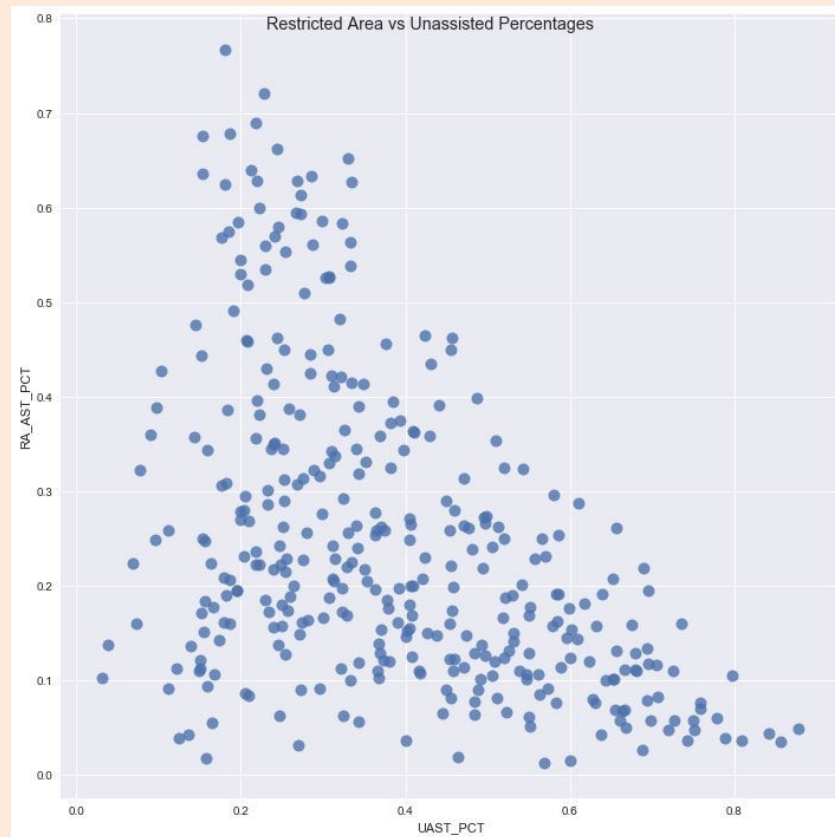
# Vector Quantization

- K-means originally comes from signal processing.
- Designed for **vector quantization**:
  - Replace examples with the mean of their cluster (“prototype”).
- Example:
  - Facebook places: 1 location summarizes many.
  - What sizes of clothing should I make?



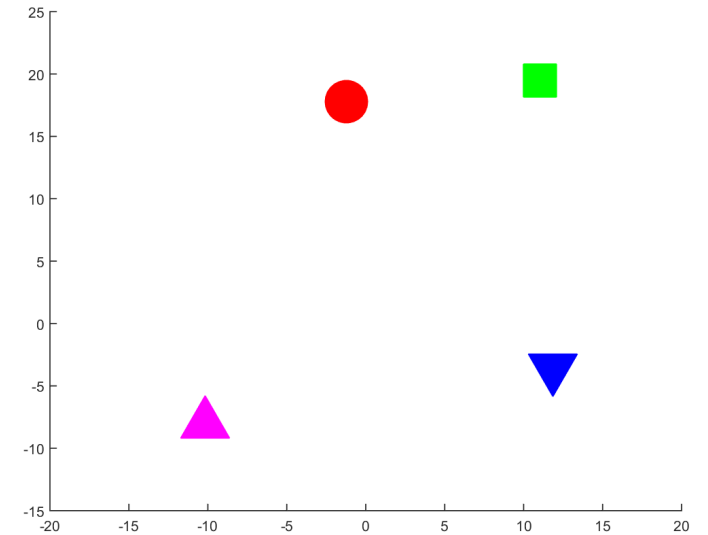
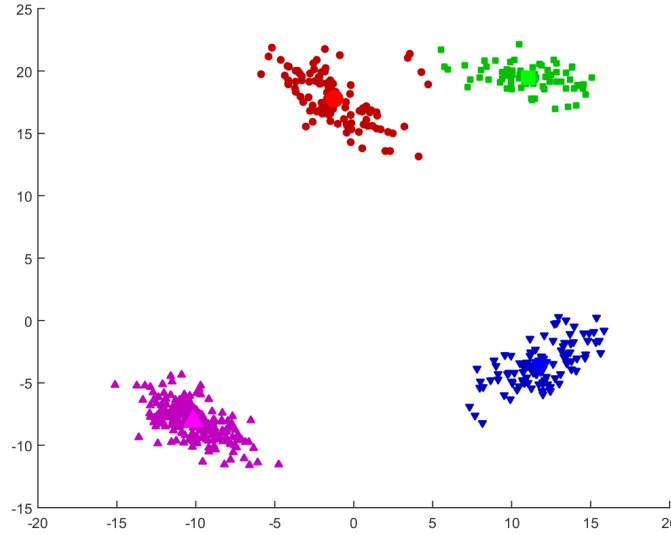
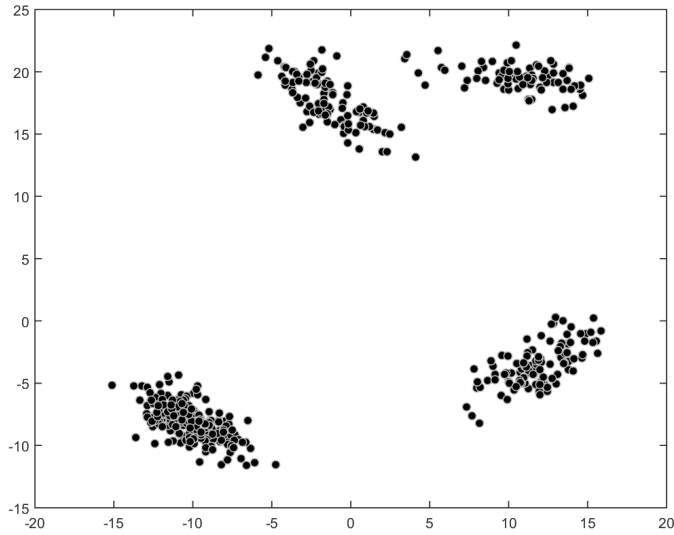
# Vector Quantization for Basketball Players

- Clustering NBA basketball players based on shot type/percentage:



- The “prototypes” (means) give offensive styles (like “catch and shoot”).

# Vector Quantization Example



$n \times d$

$$X = \begin{bmatrix} -9.0 & -7.3 \\ -10.9 & -9.0 \\ 13.7 & 19.3 \\ 13.8 & 20.4 \\ 12.8 & 20.6 \\ \vdots & \vdots \end{bmatrix}$$

Run k-means

$k \times d$

$$W = \begin{bmatrix} -1.2 & 17.8 \\ -10.2 & -8.0 \\ 11.0 & 19.5 \\ 11.8 & -3.6 \end{bmatrix}$$

$n \times 1$

$$\hat{y} = \begin{bmatrix} 2 \\ 2 \\ 3 \\ 3 \\ 3 \\ \vdots \\ 1 \end{bmatrix}$$

Approximate objects with means.

$$\hat{X} = \begin{bmatrix} -10.2 & -8.0 \\ -10.2 & -8.0 \\ 11.0 & 19.5 \\ 11.0 & 19.5 \\ -1.2 & 17.8 \\ \vdots & \vdots \end{bmatrix}$$

these mean values are the prototypes. Assume each example can be approximated by prototype

# (Bad) Vector Quantization in Practice

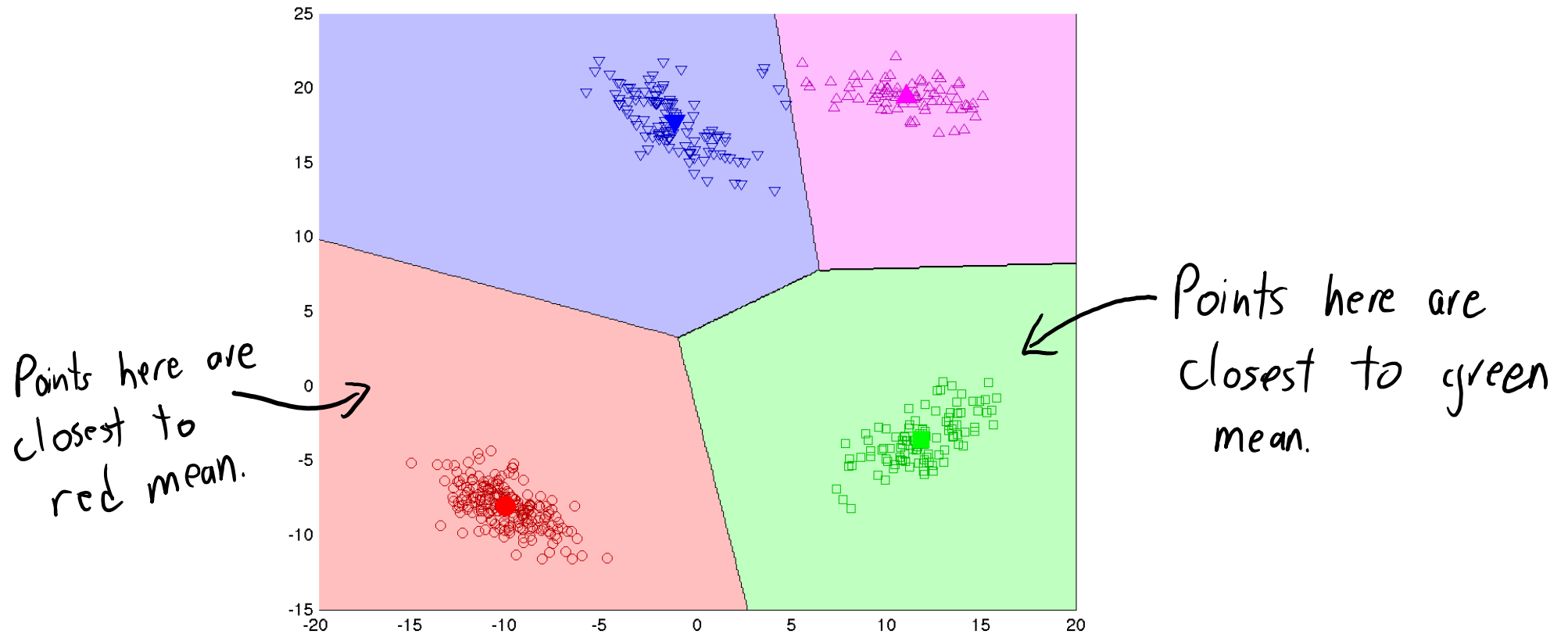
- Political parties can be thought as a form of vector quantization:



- Hope is that parties represent what a cluster of voters want.
  - With larger 'k' more voters have a party that closely reflects them.
  - With smaller 'k', parties are less accurate reflections of people's views.

# Shape of K-Means Clusters

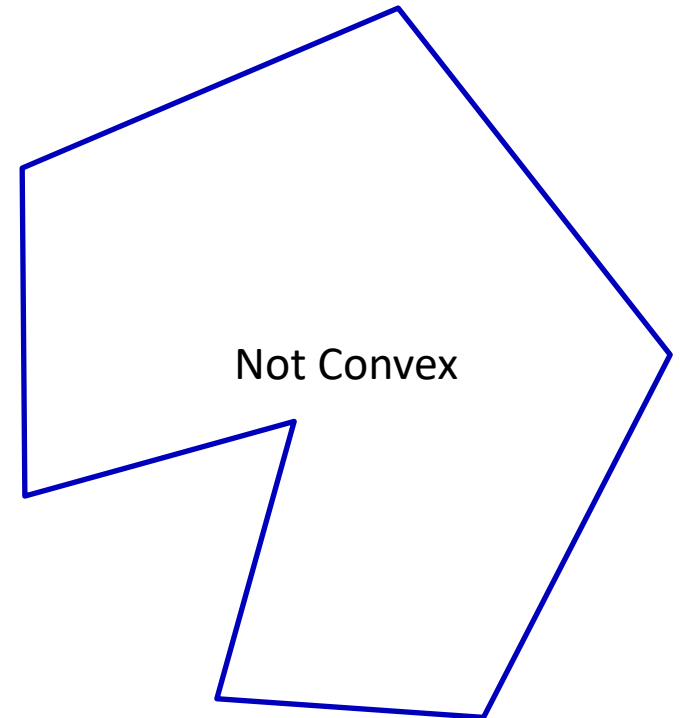
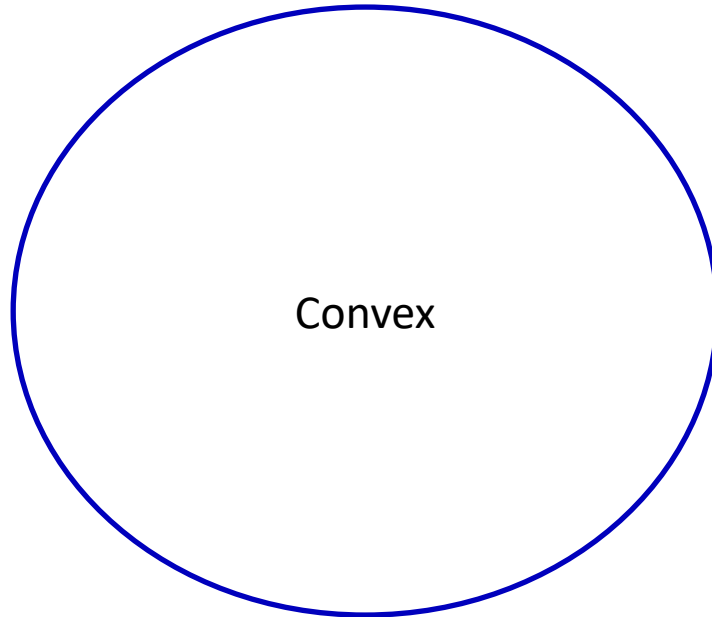
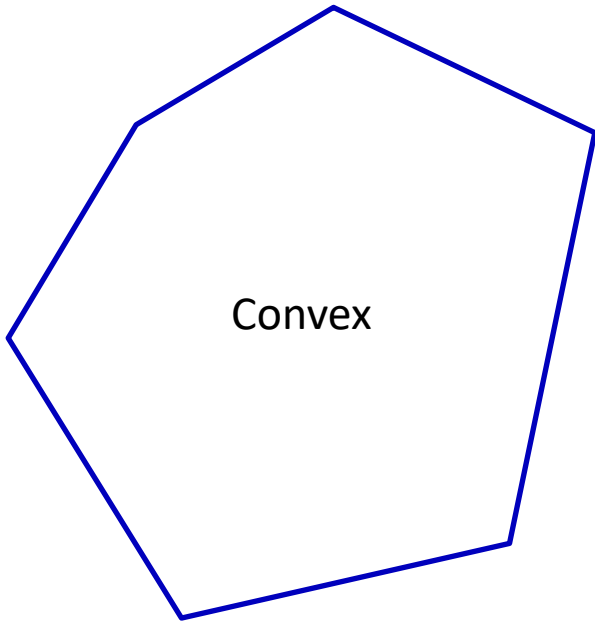
- Recall that k-means assigns cluster based on nearest mean.
- This leads to **partitions the space** :



- Observe that the **clusters are convex** regions (proof in bonus).

# Convex Sets

- A set is **convex** if **line between two points in the set stays in the set**.



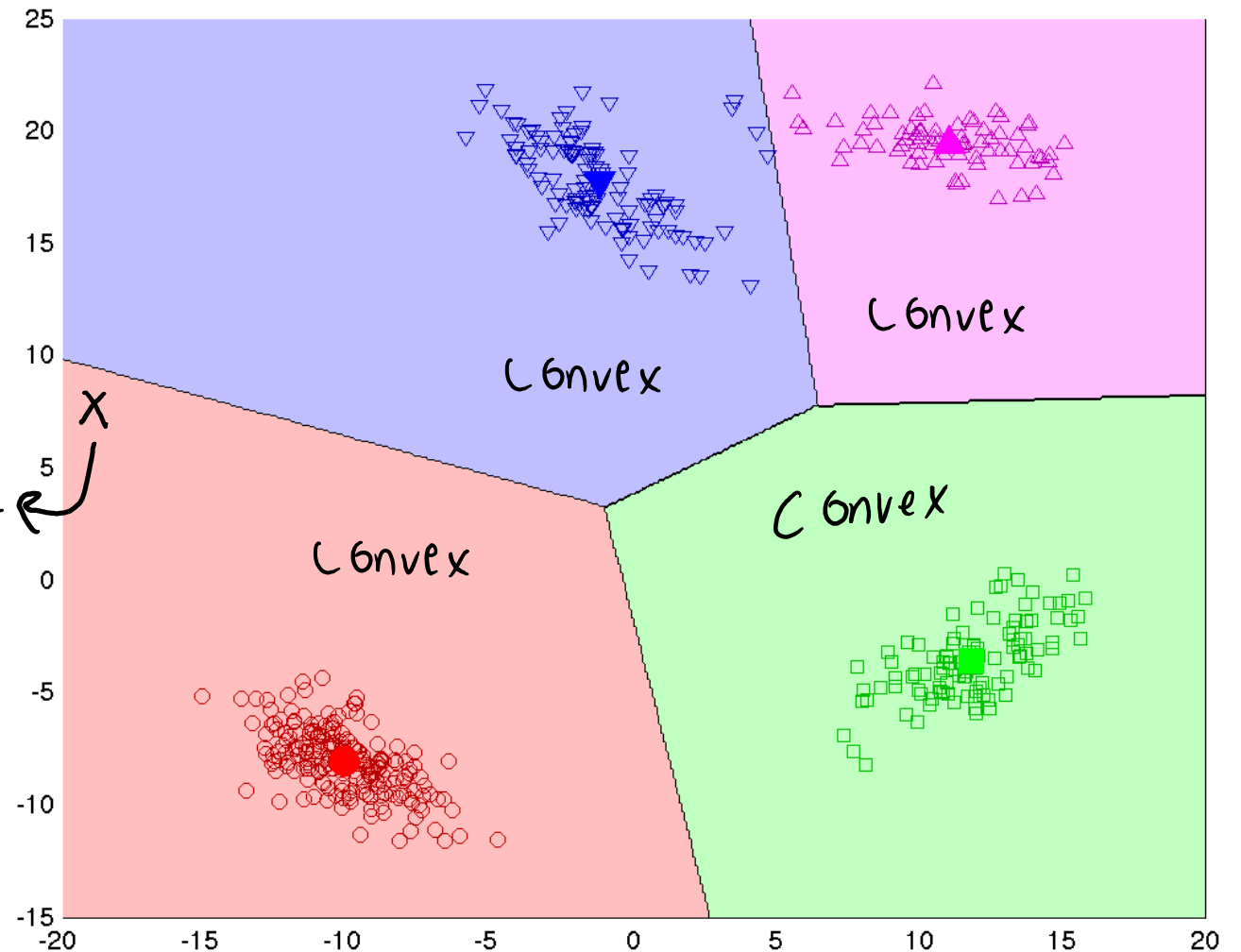


# Shape of K-Means Clusters

Issues with shape of K-means clusters:

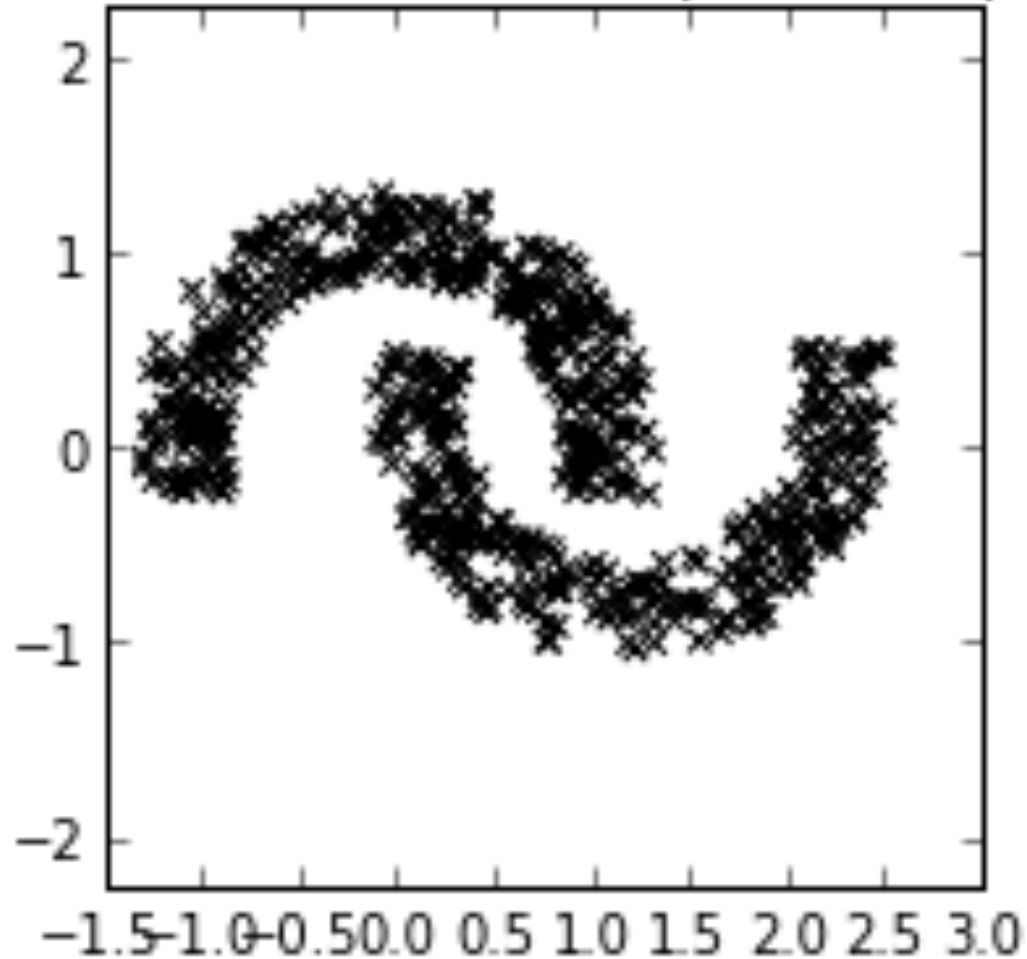
1. Clusters in the data might not be convex.

2. Does this point really belong in red cluster?

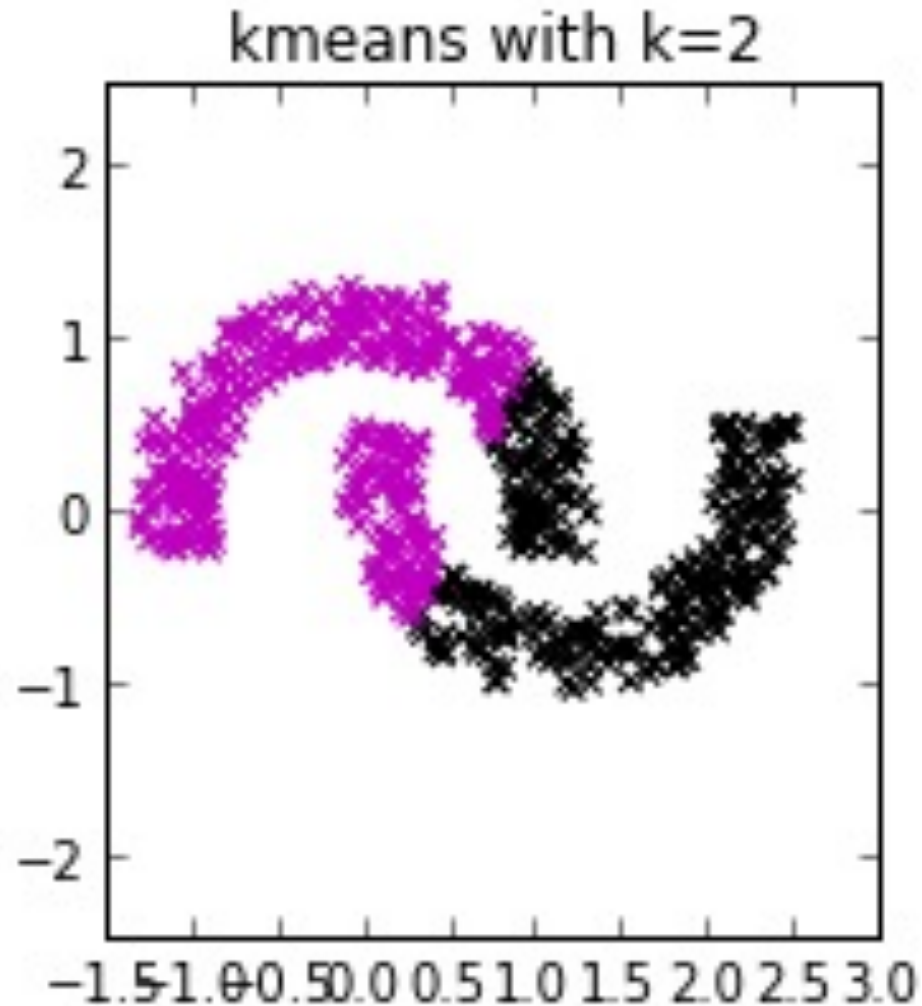


# K-Means with Non-Convex Clusters

Non-convex banana-shaped data points

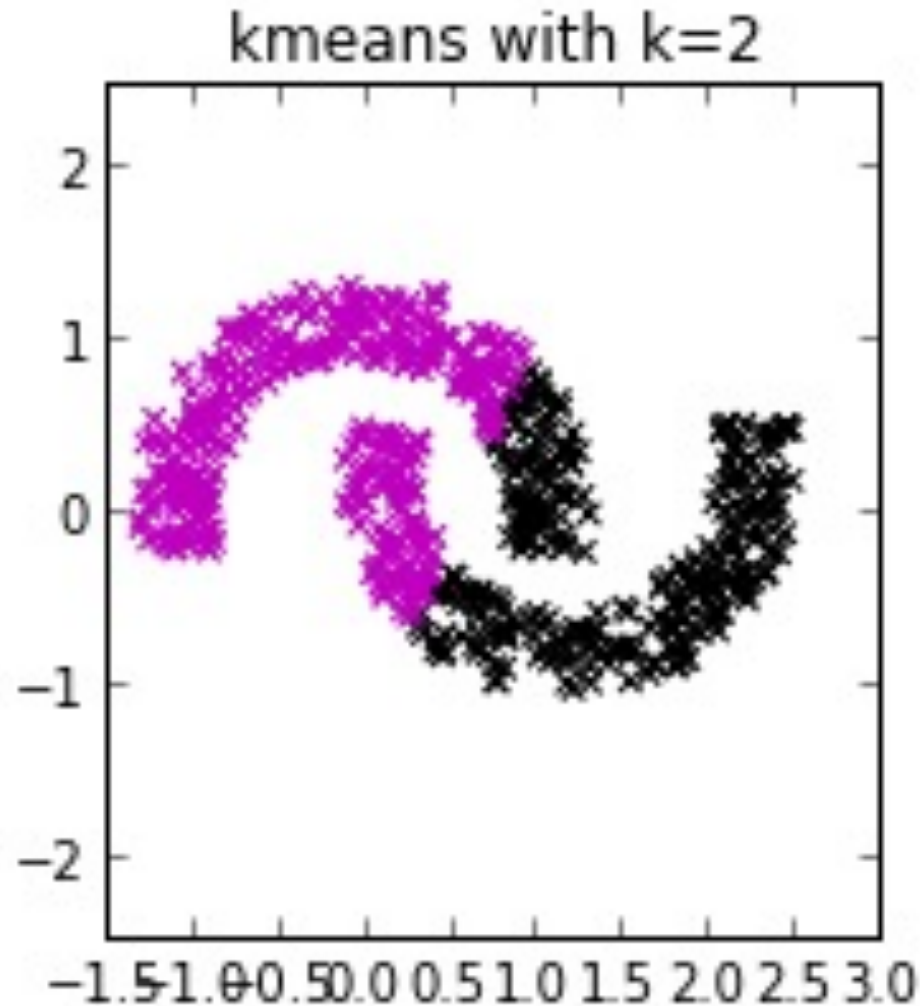


# K-Means with Non-Convex Clusters



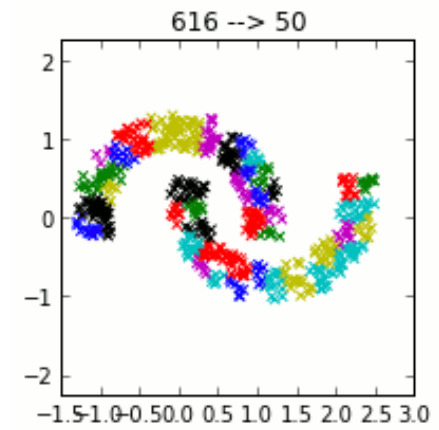
K-means **cannot separate**  
some non-convex clusters

# K-Means with Non-Convex Clusters



K-means **cannot separate**  
some non-convex clusters

Though over-clustering can help  
("hierarchical")



Google john snow [camera icon] [search icon] [grid icon] [2] [M] | All **Images** News Videos Maps More | Settings Tools | View saved SafeSearch

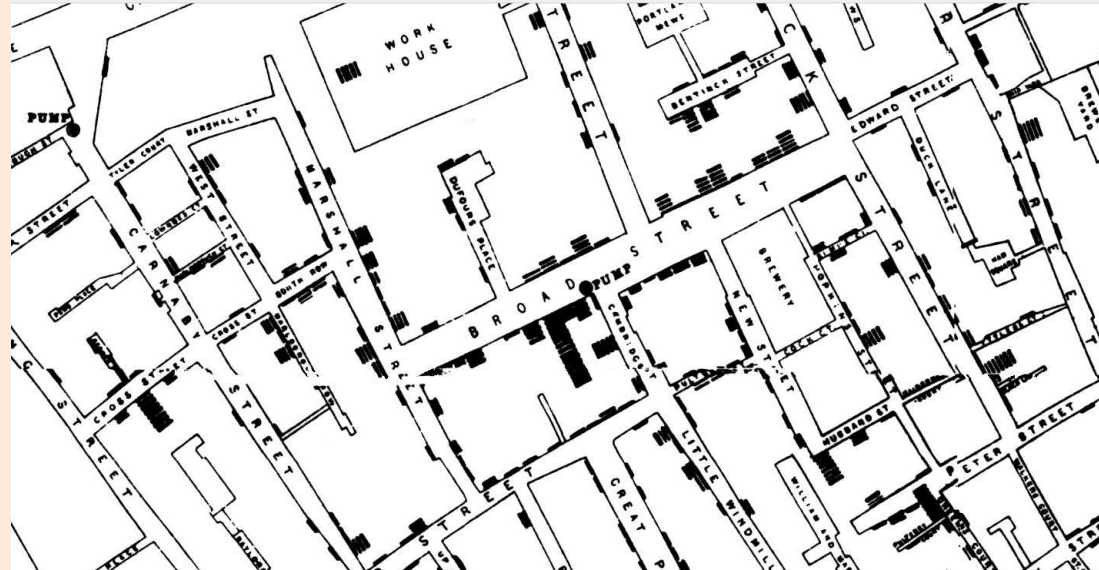
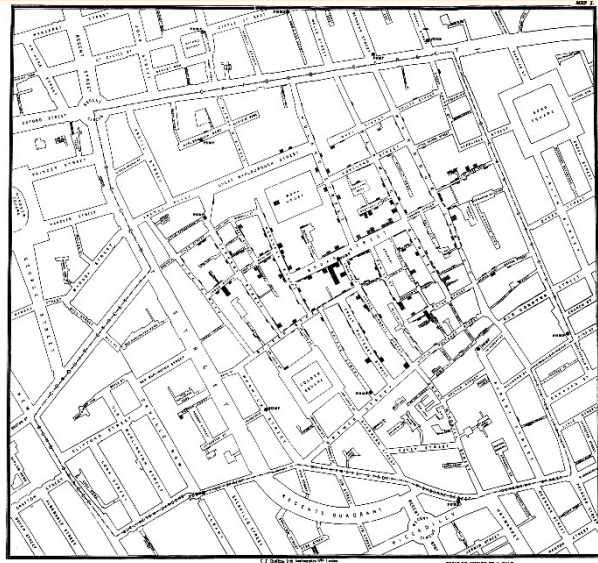
- wallpaper
- funny
- season 6
- winter is coming meme
- genderbent
- let it snow
- real life
- ghost
- wallpaper hd
- the watch meme
- his wolf
- game throne
- simpsons

Did you mean: [jon snow](#)



# John Snow and Cholera Epidemic

- John Snow's 1854 spatial histogram of deaths from cholera:



- Found cluster of cholera deaths around a particular water pump.
  - Went against airborne theory, but pump later found to be contaminated.
  - “Father” of epidemiology.

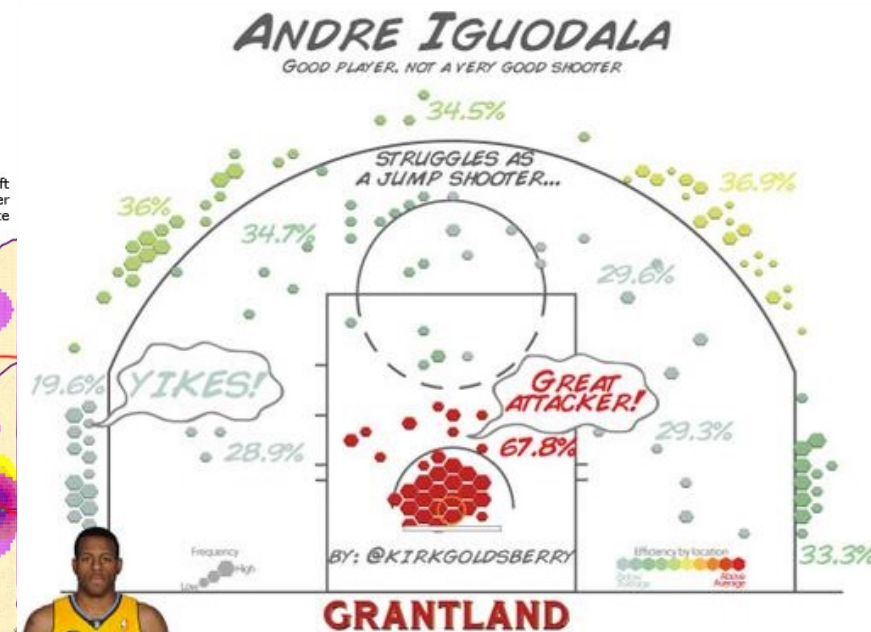
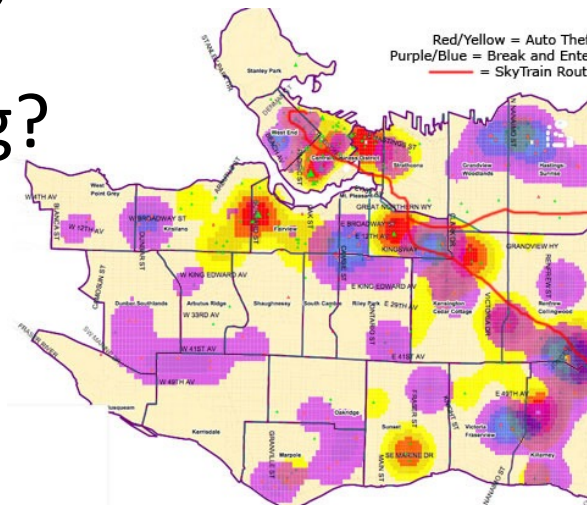
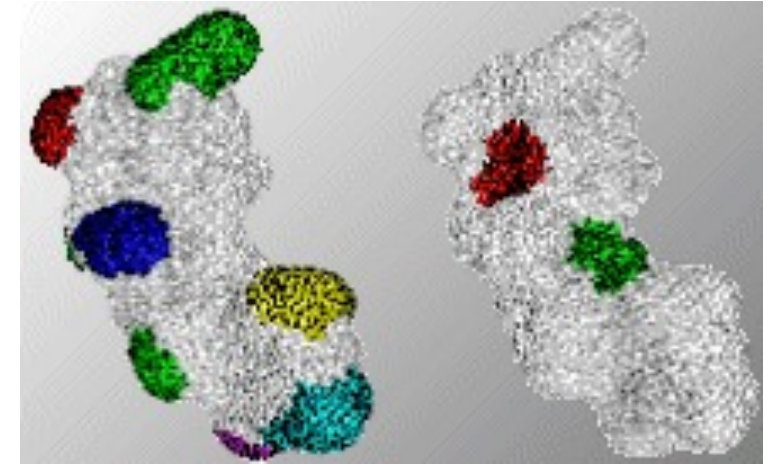
# Motivation for Density-Based Clustering

- **Density-based clustering:**
  - Clusters are **defined by “dense” regions.**
  - Examples in **non-dense regions don’t get clustered.**
    - Not trying to “partition” the space.
- Clusters can be **non-convex:**
  - Elephant clusters affected by vegetation, mountains, rivers, water access, etc.
- It’s a **non-parametric clustering** method:
  - No fixed number of clusters ‘k’.
  - Clusters can become more complicated with more data.



# Other Potential Applications

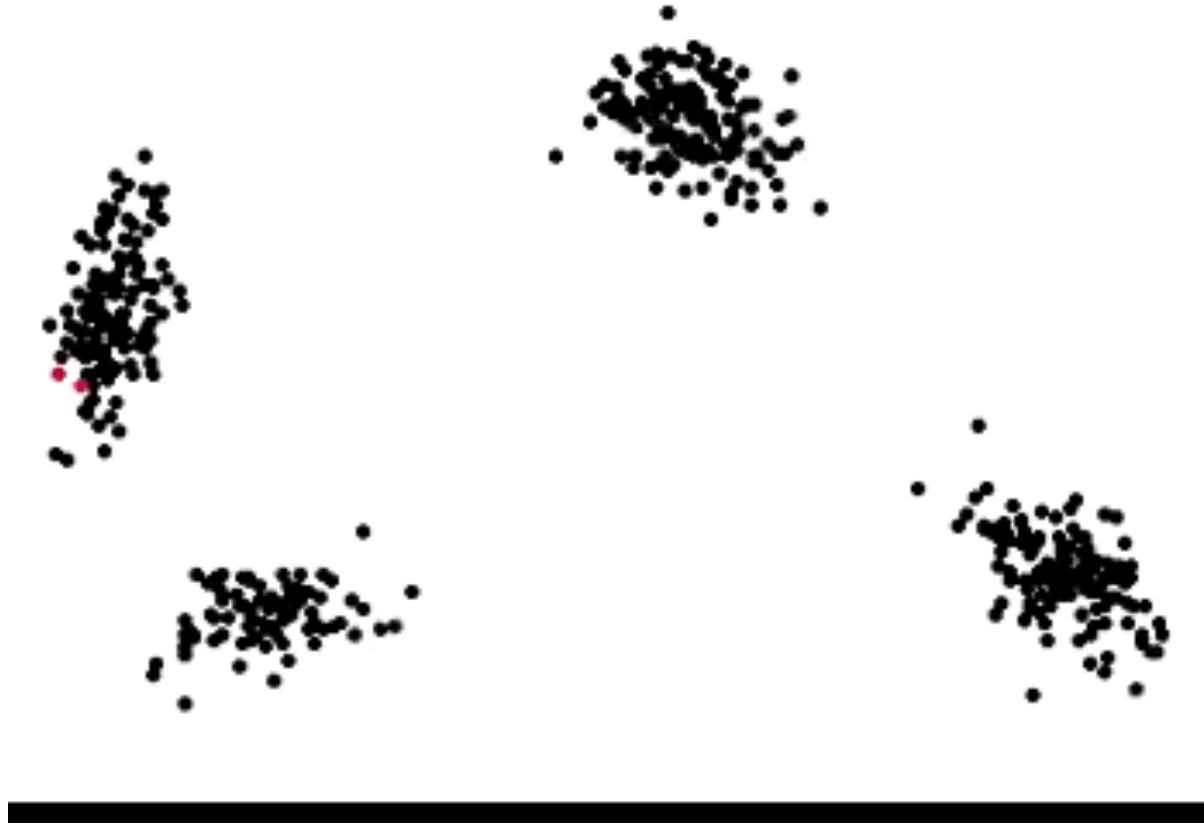
- Where are high crime regions of a city?
- Where should taxis patrol?
- Where does Iguodala make/miss shots?
- Which products are similar to this one?
- Which pictures are in the same place?
- Where can proteins 'dock'?
- Where are people tweeting?





# Density-Based Clustering in Action

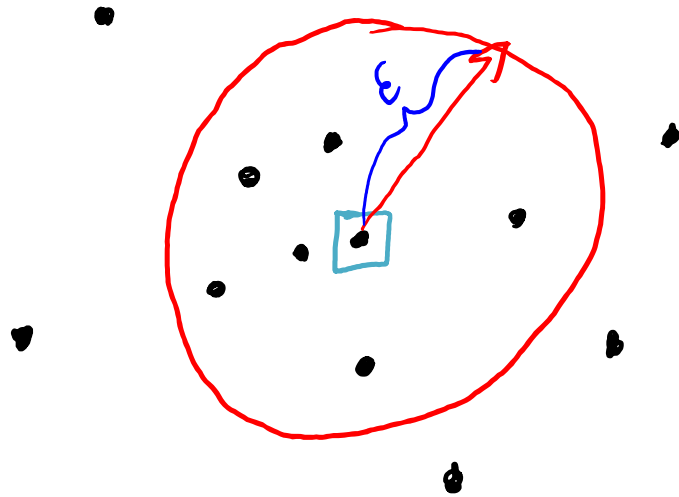
---



# Density-Based Clustering

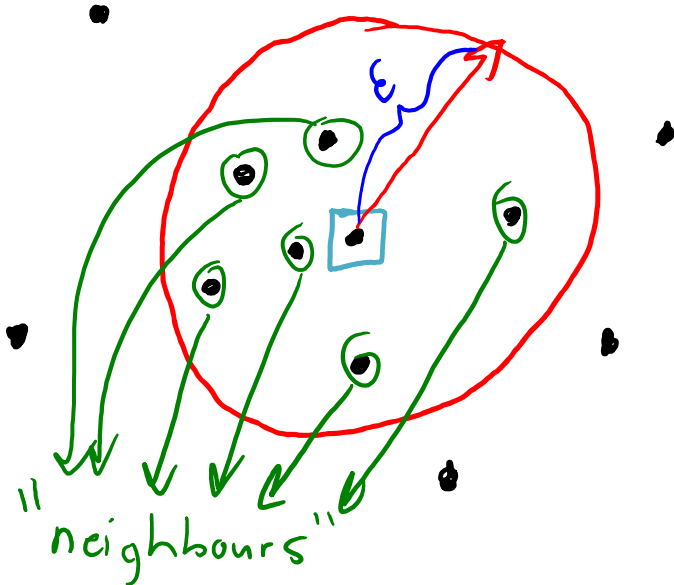
DBSCAN: "Density-Based Spatial Clustering of Applications with Noise."

- Density-based clustering algorithm (DBSCAN) has two hyperparameters:
  - Epsilon ( $\epsilon$ ): distance we use to decide if another point is a "neighbour".



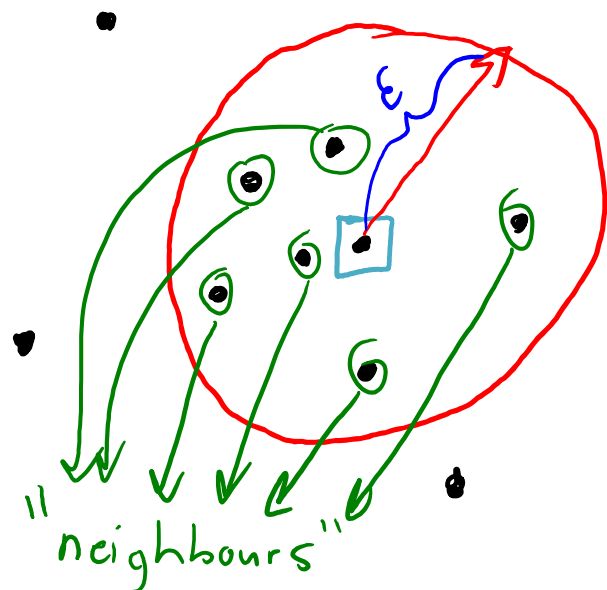
# Density-Based Clustering

- Density-based clustering algorithm (DBSCAN) has two hyperparameters:
  - Epsilon ( $\epsilon$ ): distance we use to decide if another point is a “neighbour”.



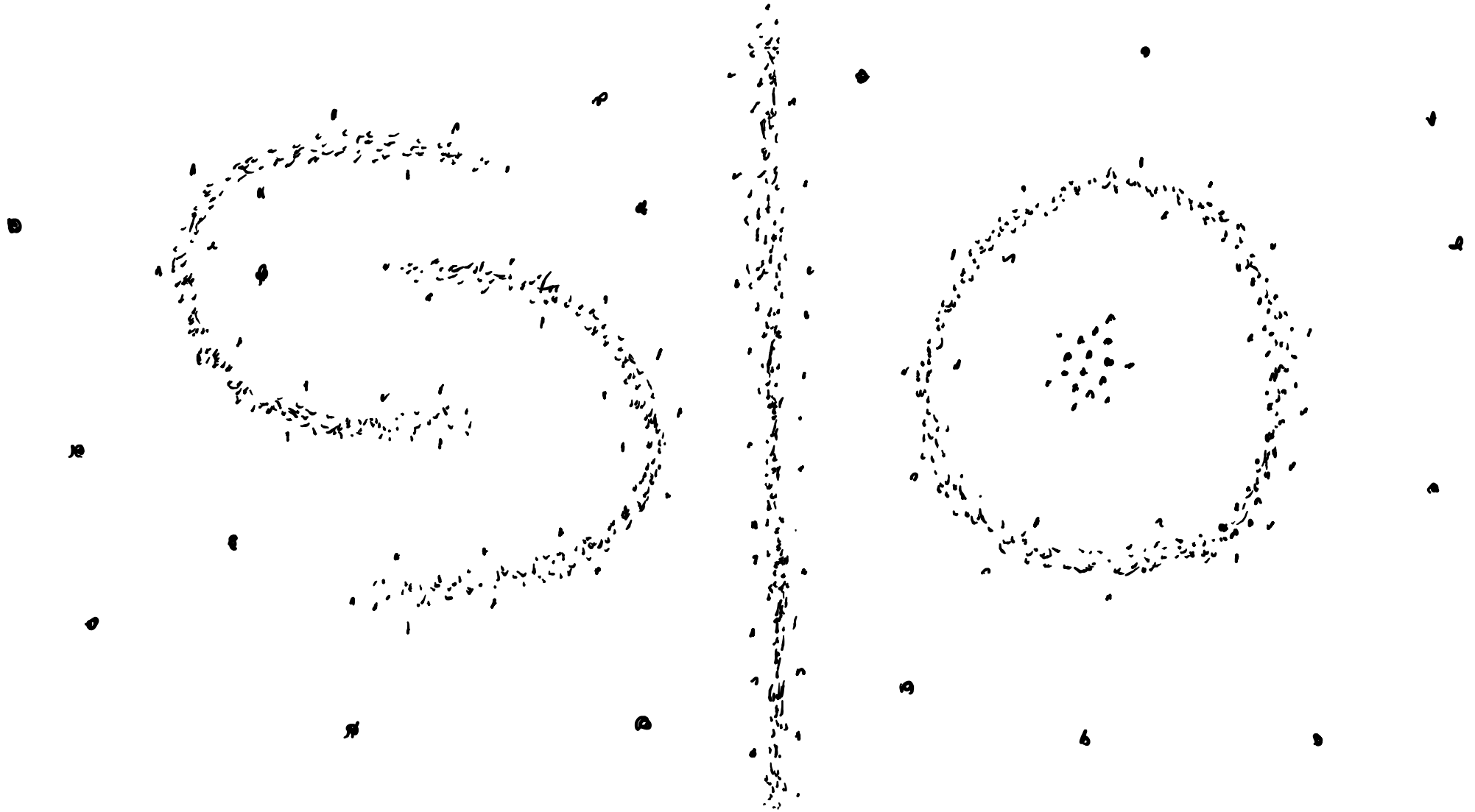
# Density-Based Clustering

- **Density-based clustering** algorithm (DBSCAN) has two hyperparameters:
  - **Epsilon ( $\epsilon$ )**: distance we use to decide if another point is a “**neighbour**”.
  - **MinNeighbours**: **number of neighbours** needed to say a region is “dense”.
    - If you have **at least minNeighbours** “neighbours”, you are called a “**core**” point.
- **Main idea**: **merge all neighbouring core points to form clusters**.

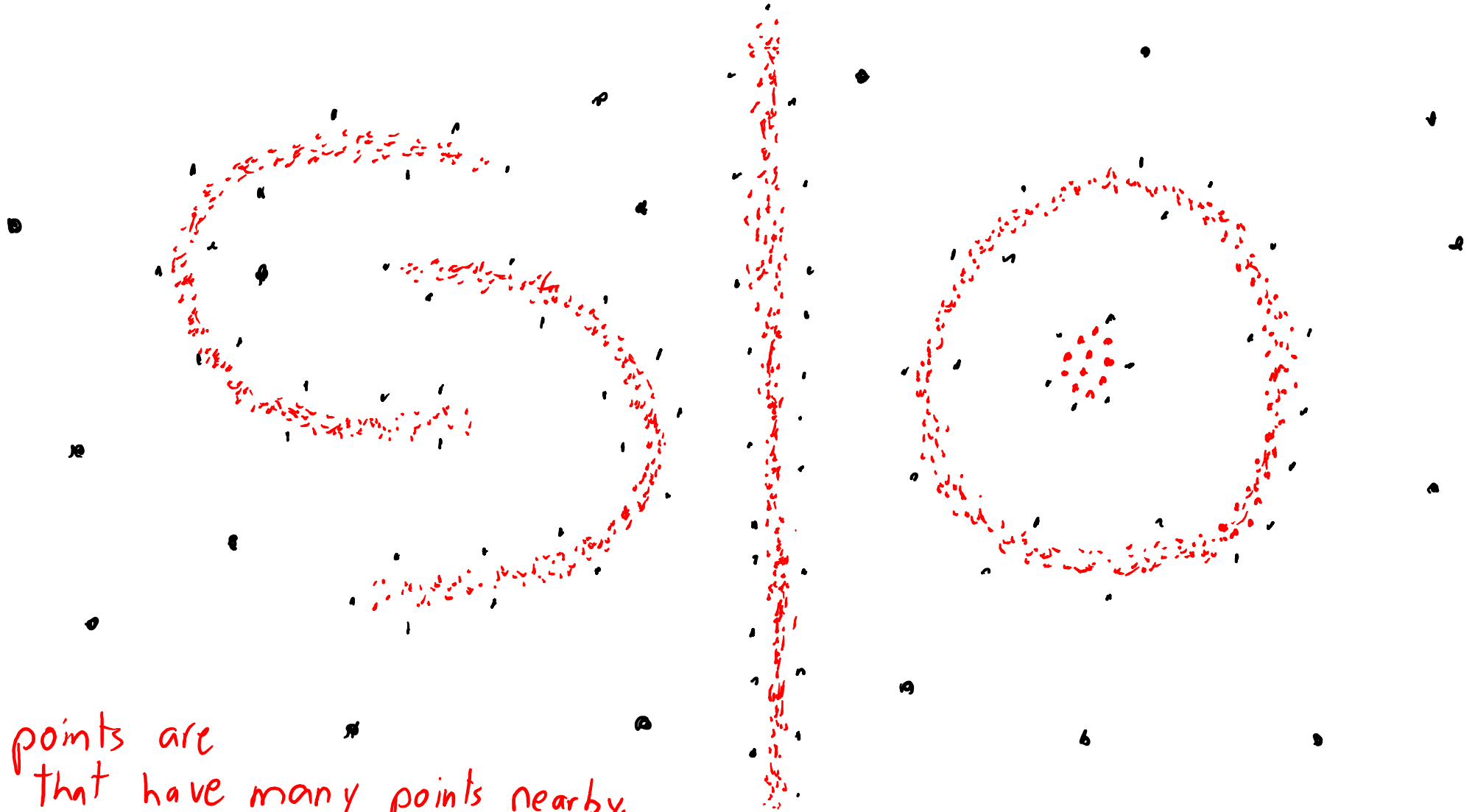


E.g., if  $\text{minNeighbours} = 3$   
then this is a “core”  
point since 6 points are  
“neighbours”

# Density-Based Clustering

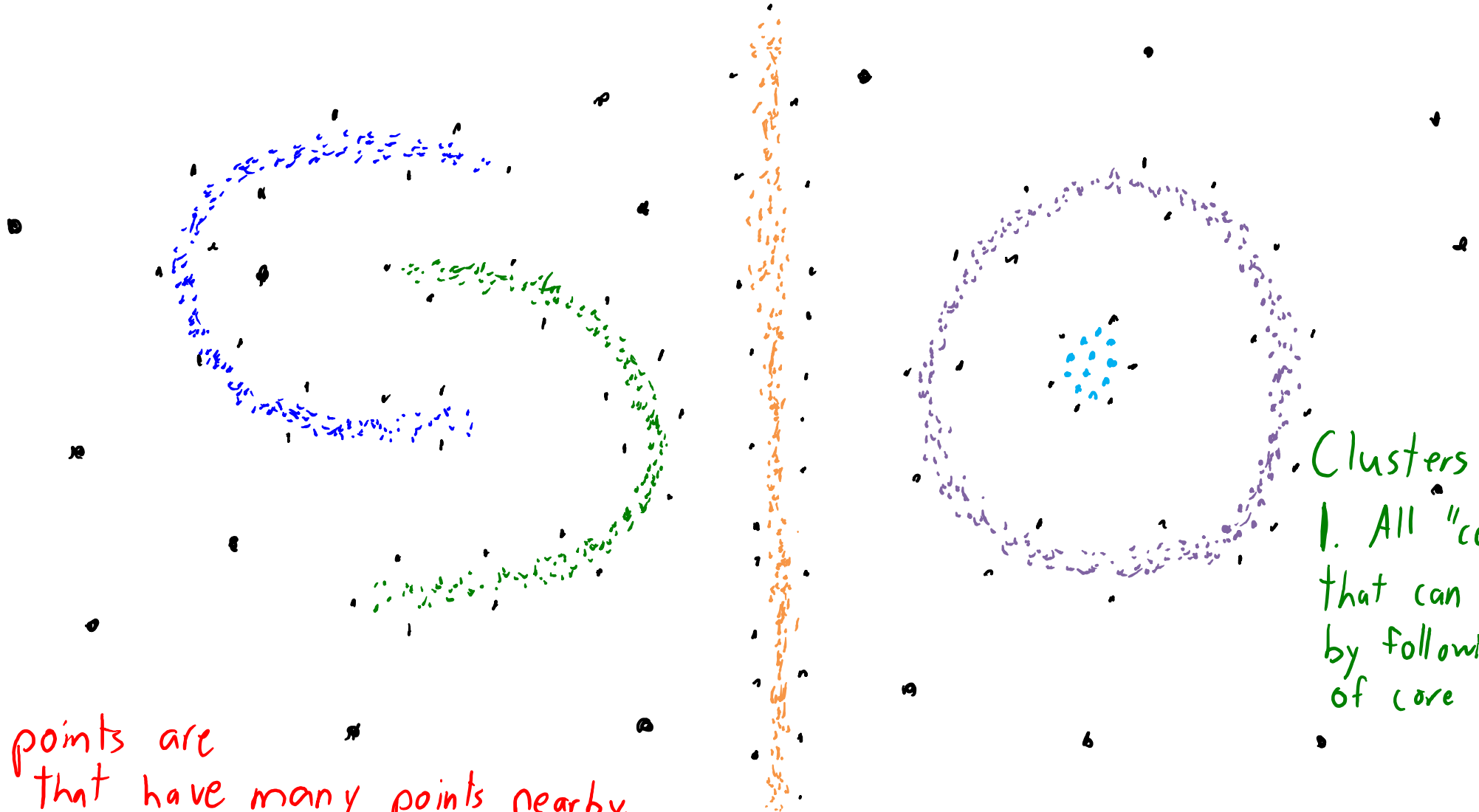


# Density-Based Clustering



"Core" points are points that have many points nearby.

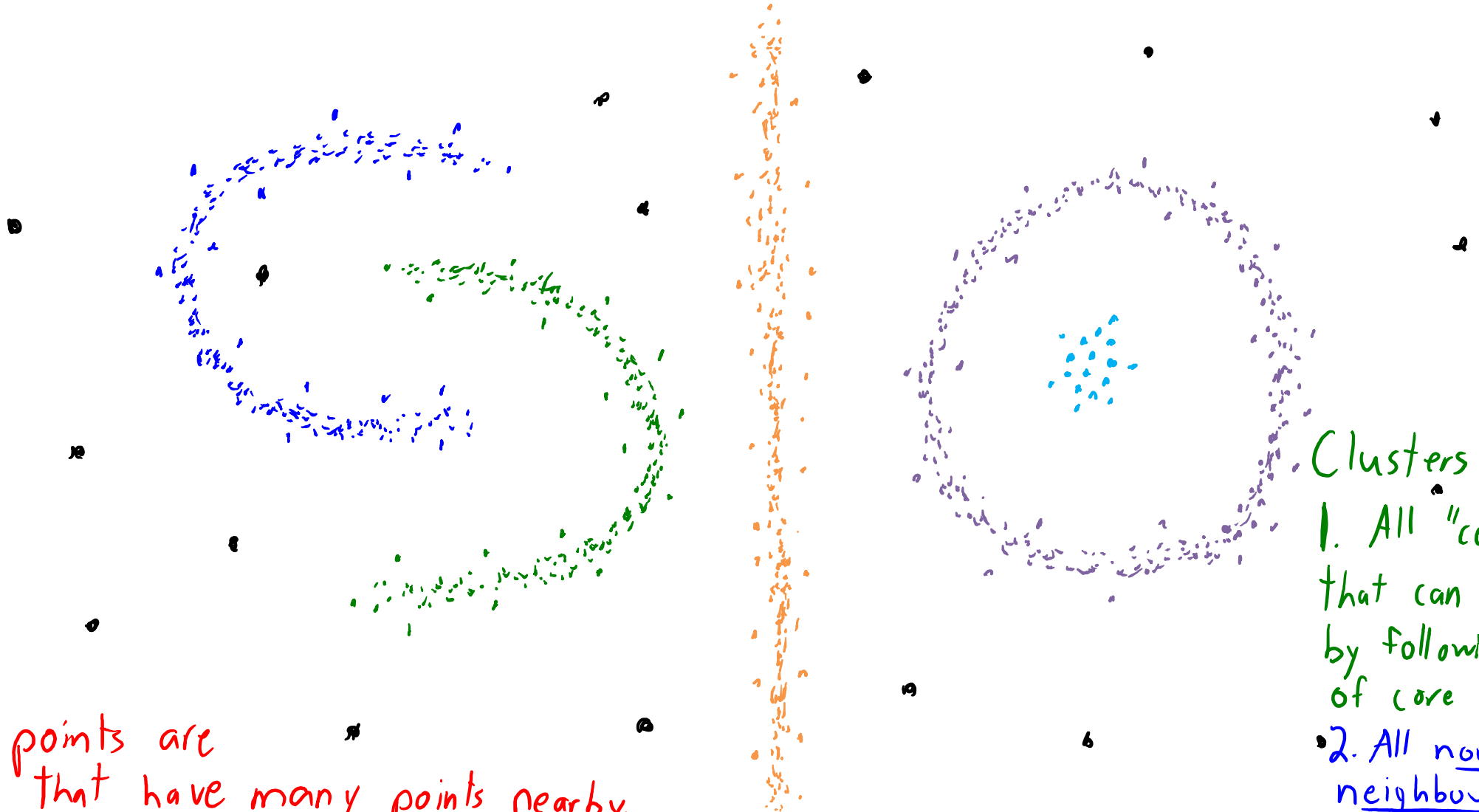
# Density-Based Clustering



Clusters contain:  
1. All "core" points that can be reached by following a sequence of core points.

"Core" points are points that have many points nearby.

# Density-Based Clustering



Clusters contain:

1. All "core" points that can be reached by following a sequence of core points.
2. All non-core neighbours of core points (boundary points)

"Core" points are points that have many points nearby.



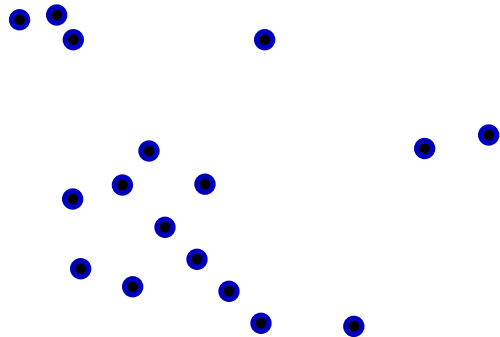
# Density-Based Clustering Pseudo-Code

- Intuitively, density-based clustering **algorithm** implements a “chain reaction” throughout the dense areas.
- For each example  $x_i$ :
  - If  $x_i$  is already assigned to a cluster, do nothing.
  - Else: Test whether  $x_i$  is a ‘core’ point ( $\geq$  minNeighbours examples within ‘ $\epsilon$ ’)
    - If  $x_i$  is not core point, do nothing (this could be an outlier).
    - If  $x_i$  is a core point, make a **new cluster** and call the “**expand cluster**” function.
      - Which spreads the “reaction” to nearby points.

# Density-Based Clustering Pseudo-Code

- “Expand cluster” function:
  - Assign to this cluster all  $x_j$  within distance ‘ $\epsilon$ ’ of core point  $x_i$  to this cluster.
  - For each new “core” point found, call “expand cluster” (recursively).

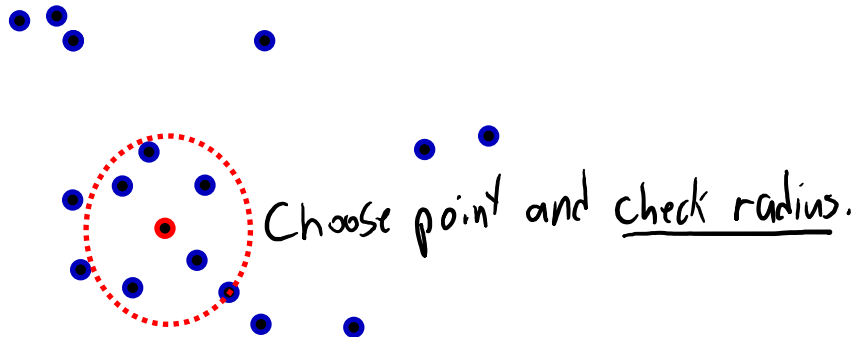
minNeighbors = 2



# Density-Based Clustering Pseudo-Code

- “Expand cluster” function:
  - Assign to this cluster all  $x_j$  within distance ‘ $\epsilon$ ’ of core point  $x_i$  to this cluster.
  - For each new “core” point found, call “expand cluster” (recursively).

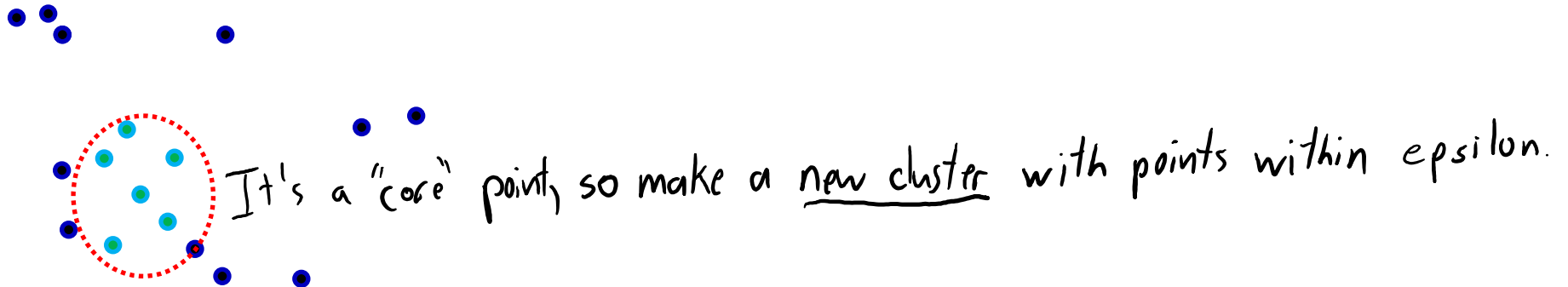
minNeighbors = 2



# Density-Based Clustering Pseudo-Code

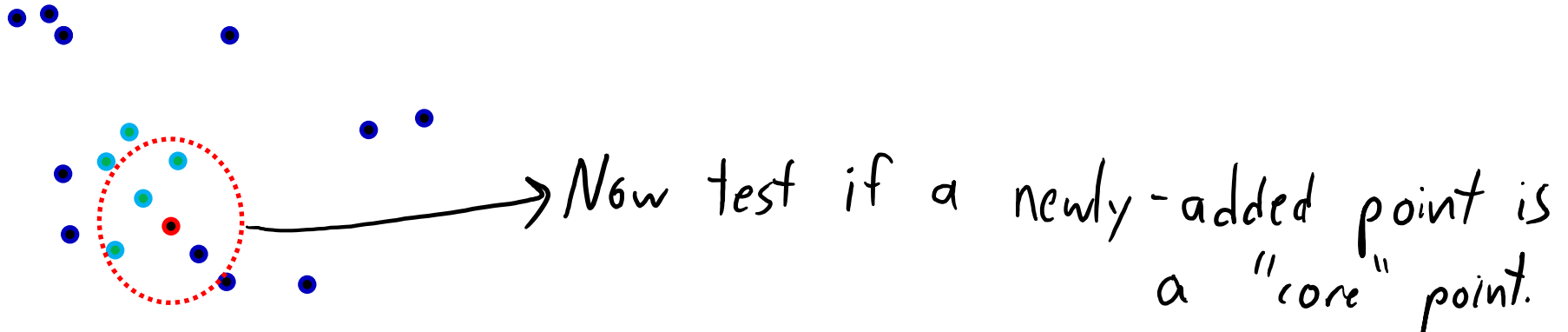
- “Expand cluster” function:
  - Assign to this cluster all  $x_j$  within distance ‘ $\epsilon$ ’ of core point  $x_i$  to this cluster.
  - For each new “core” point found, call “expand cluster” (recursively).

minNeighbors = 2



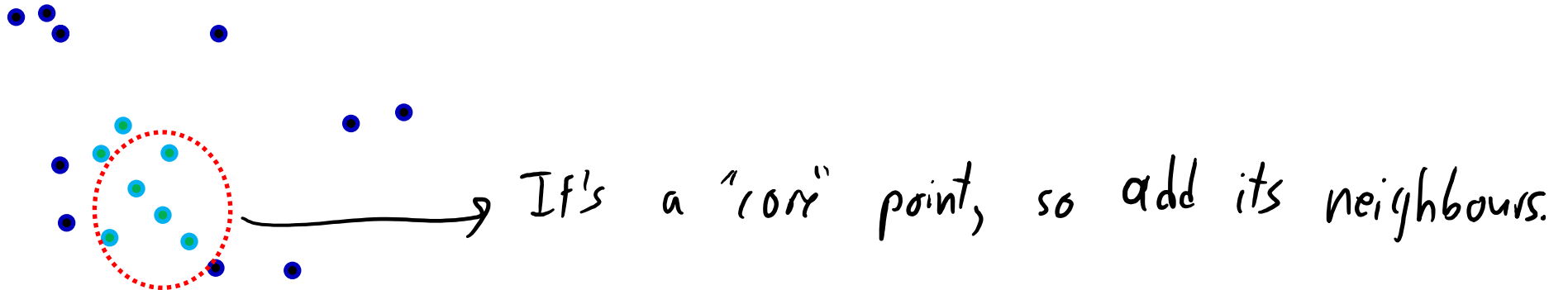
# Density-Based Clustering Pseudo-Code

- “Expand cluster” function:
  - Assign to this cluster all  $x_j$  within distance ‘ $\epsilon$ ’ of core point  $x_i$  to this cluster.
  - For each new “core” point found, call “expand cluster” (recursively).



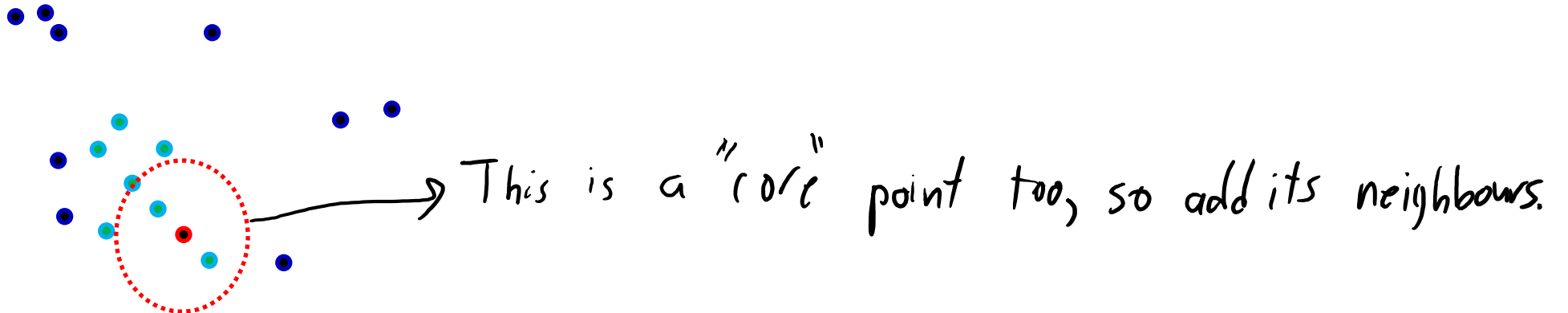
# Density-Based Clustering Pseudo-Code

- “Expand cluster” function:
  - Assign to this cluster all  $x_j$  within distance ‘ $\epsilon$ ’ of core point  $x_i$  to this cluster.
  - For each new “core” point found, call “expand cluster” (recursively).



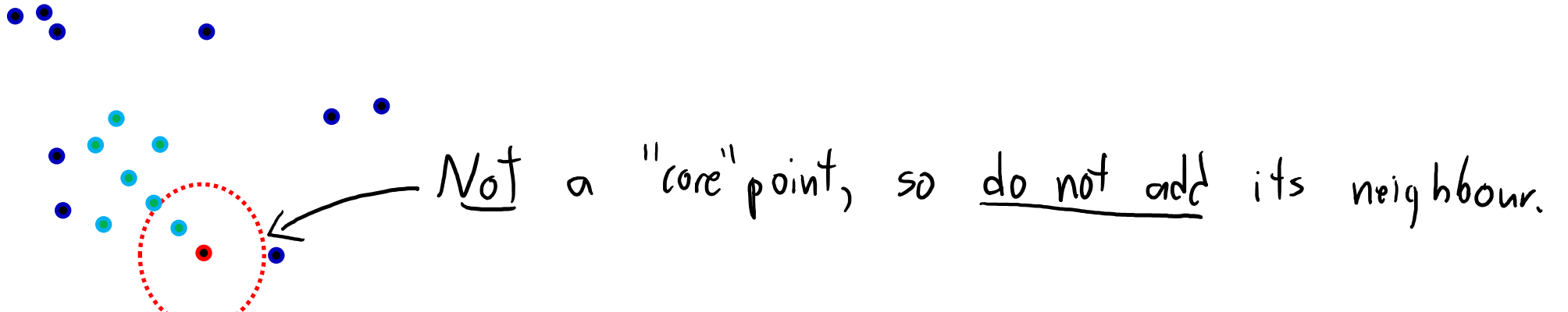
# Density-Based Clustering Pseudo-Code

- “Expand cluster” function:
  - Assign to this cluster all  $x_j$  within distance ‘ $\epsilon$ ’ of core point  $x_i$  to this cluster.
  - For each new “core” point found, call “expand cluster” (recursively).



# Density-Based Clustering Pseudo-Code

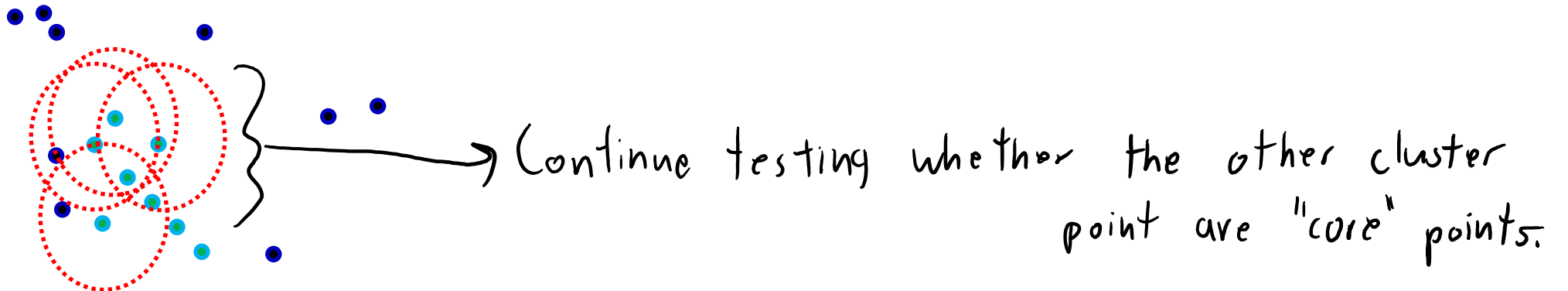
- “Expand cluster” function:
  - Assign to this cluster all  $x_j$  within distance ‘ $\epsilon$ ’ of core point  $x_i$  to this cluster.
  - For each new “core” point found, call “expand cluster” (recursively).





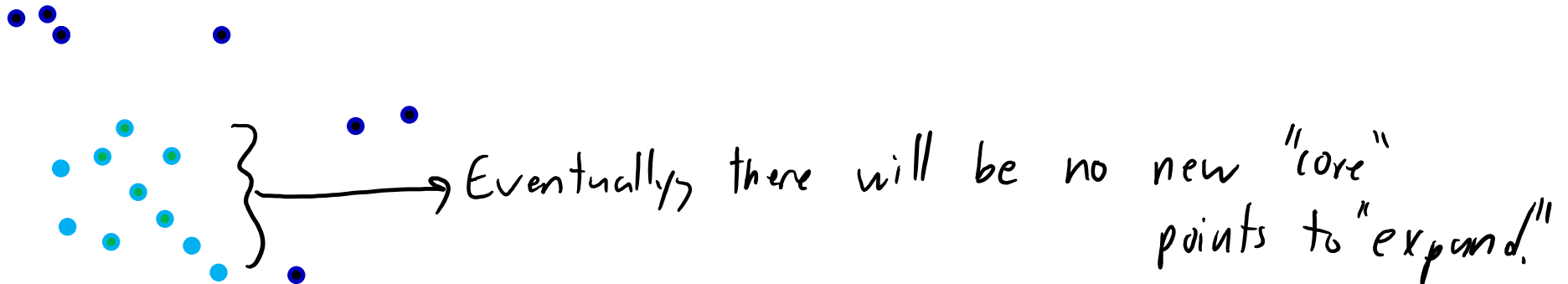
# Density-Based Clustering Pseudo-Code

- “Expand cluster” function:
  - Assign to this cluster all  $x_j$  within distance ‘ $\epsilon$ ’ of core point  $x_i$  to this cluster.
  - For each new “core” point found, call “expand cluster” (recursively).



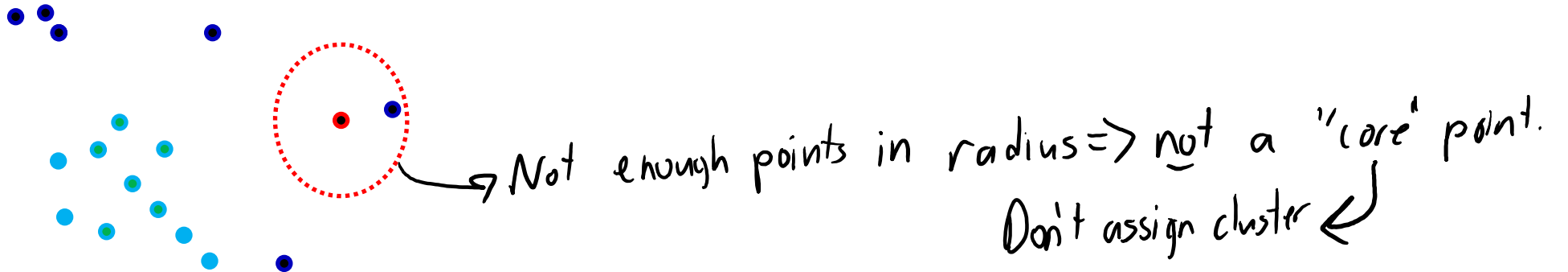
# Density-Based Clustering Pseudo-Code

- “Expand cluster” function:
  - Assign to this cluster all  $x_j$  within distance ‘ $\epsilon$ ’ of core point  $x_i$  to this cluster.
  - For each new “core” point found, call “expand cluster” (recursively).



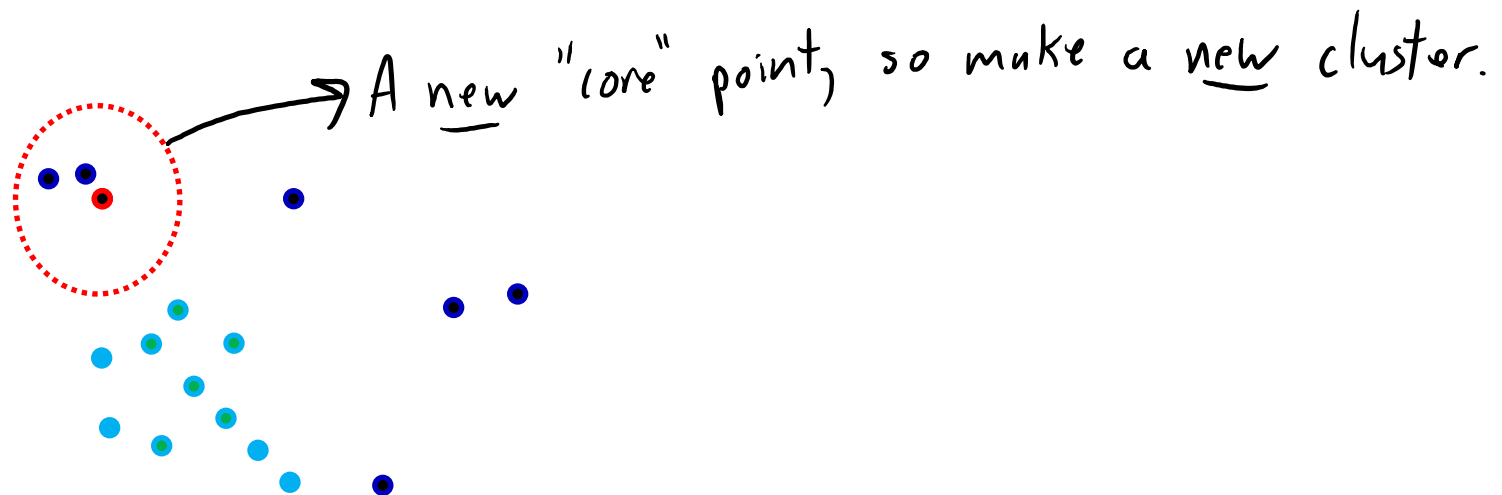
# Density-Based Clustering Pseudo-Code

- “Expand cluster” function:
  - Assign to this cluster all  $x_j$  within distance ‘ $\epsilon$ ’ of core point  $x_i$  to this cluster.
  - For each new “core” point found, call “expand cluster” (recursively).



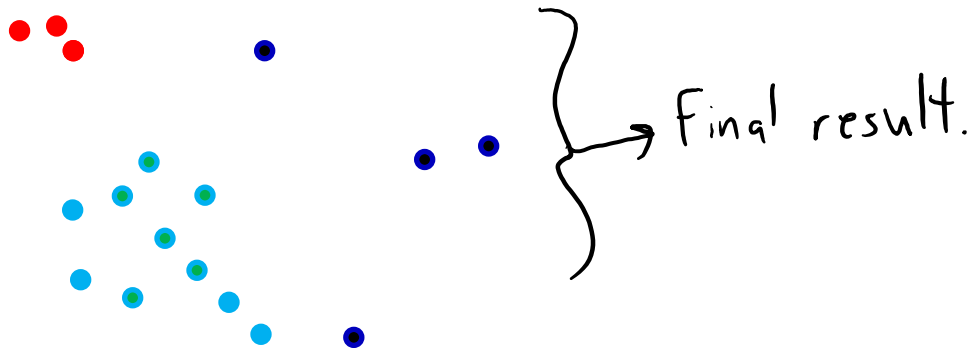
# Density-Based Clustering Pseudo-Code

- “Expand cluster” function:
  - Assign to this cluster all  $x_j$  within distance ‘ $\epsilon$ ’ of core point  $x_i$  to this cluster.
  - For each new “core” point found, call “expand cluster” (recursively).

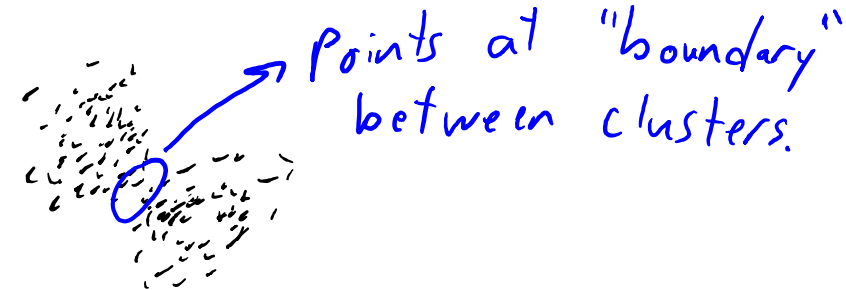


# Density-Based Clustering Pseudo-Code

- “Expand cluster” function:
  - Assign to this cluster all  $x_j$  within distance ‘ $\epsilon$ ’ of core point  $x_i$  to this cluster.
  - For each new “core” point found, call “expand cluster” (recursively).

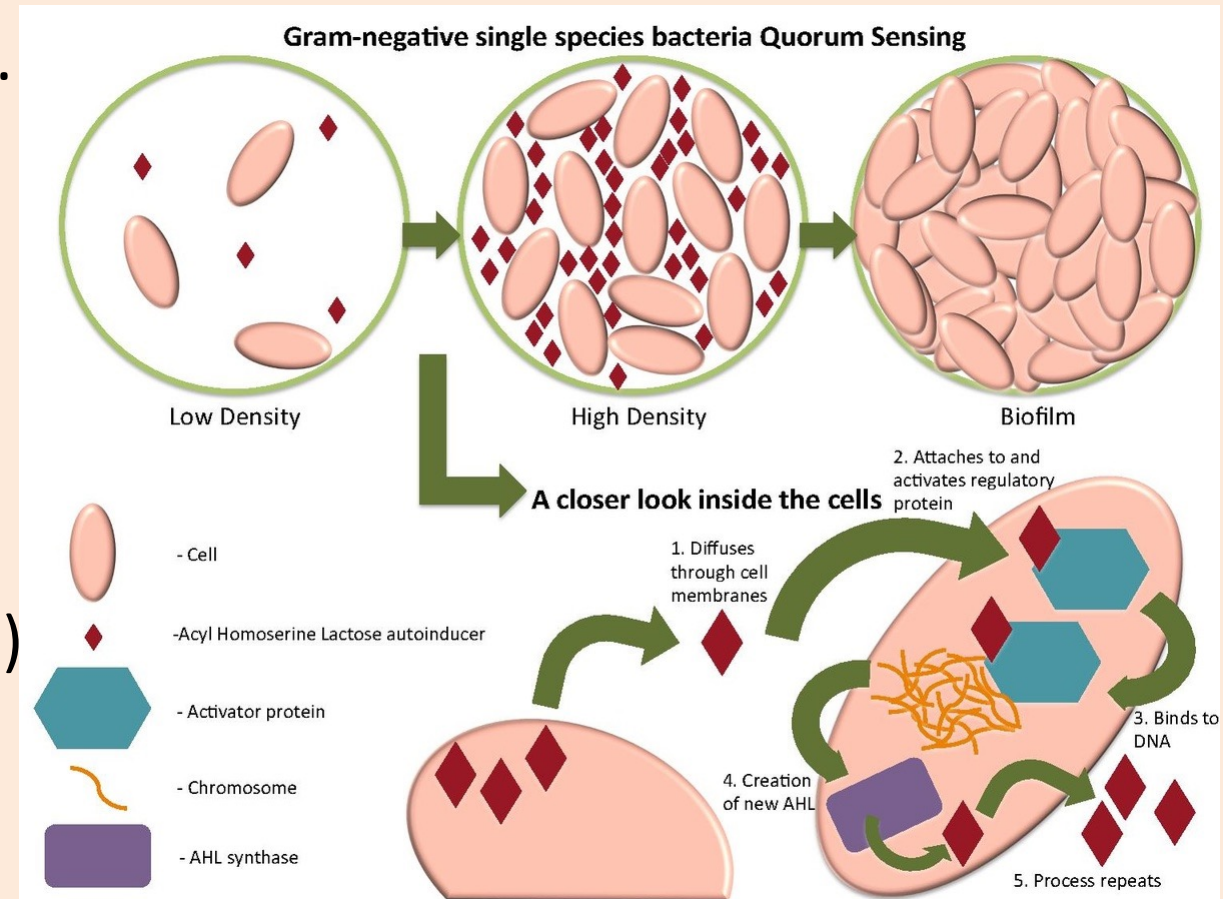


# Density-Based Clustering Issues

- Some points are not assigned to a cluster.
  - Good or bad, depending on the application.
- Ambiguity of “non-core” (boundary) points:
- Sensitive to the choice of  $\epsilon$  and minNeighbours.
  - Original paper proposed an “elbow” method (see bonus slide).
  - Otherwise, not sensitive to initialization (except for boundary points).
- If you get a new example, finding cluster is expensive.
  - Need to compute distances to core points (or maybe all training points).
- In high-dimensions, need a lot of points to ‘fill’ the space.

# Density-Based Clustering in Bacteria

- Quorum sensing:
  - Bacteria continuously release a particular molecule.
  - They have sensors for this molecule.
- If sensors become very active:
  - It means cell density is high.
  - Causes cascade of changes in cells. (Some cells “stick together” to form a physical cluster via “biofilm”.)



# Density-Based Clustering in People

- “High density leading a chain reaction” can also happen in people.

## COVID-19 cluster reported at Port Coquitlam Costco; outbreak at VGH declared

BY DEAN RECKSIDLER AND HANA MAE NASSAR

Posted Mar 15, 2021 9:38 am PDT Last Updated Mar 16, 2021 at 1:09 am PDT



- “Social distancing”: try to reduce the number of people within  $\epsilon$ .
- “Wearing masks”: try to increase the  $\epsilon$  needed for a chain reaction.



Next Topic: Ensemble Clustering

# Ensemble Clustering

? question ☆

stop following 23 views

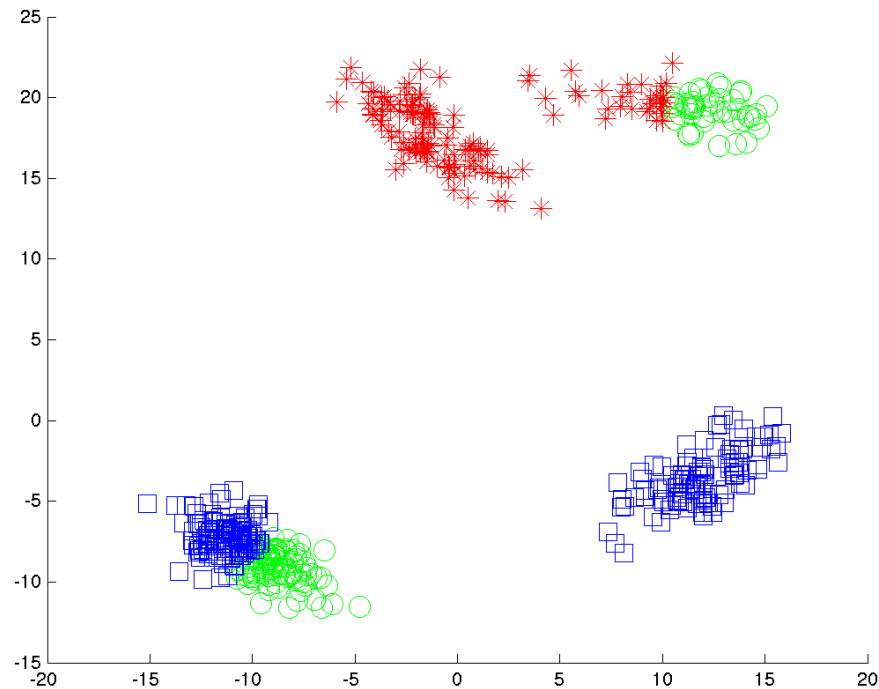
## Multiple random runs of K means

I was wondering how running K Means (original version, not K means ++ ) several times with random initializations can help us make an accurate model. K Means outputs the class labels of all the samples. We definitely can't use mode of all the labels it got in different runs because class labels from different runs don't make any sense when compared. We somehow have to see what points are coming in the same cluster in a lot of runs..I am not sure, how do we do it?

- We can consider **ensemble methods** for clustering.
  - “Consensus clustering”
- It's a good/important idea:
  - **Bootstrapping** is widely-used.
  - “Do clusters change if the data was slightly different?”
- But we **need to be careful** about how we combine models.

# Ensemble Clustering

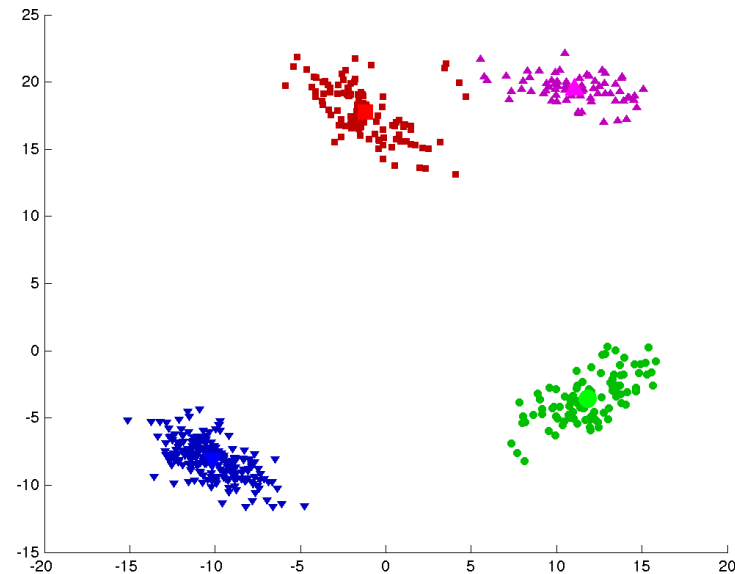
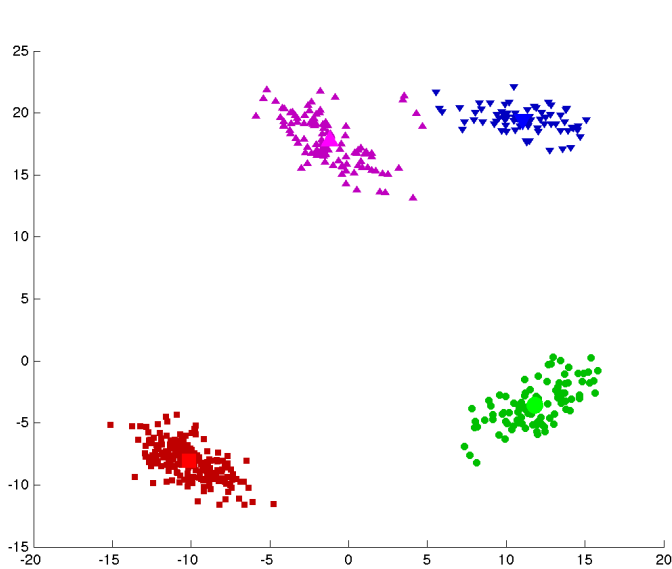
- E.g., run k-means 20 times and then cluster using the mode of each  $\hat{y}_i$ .
- Normally, averaging across models doing different things is good.



- But this is a bad ensemble method: **worse than k-means on its own.**

# Label Switching Problem

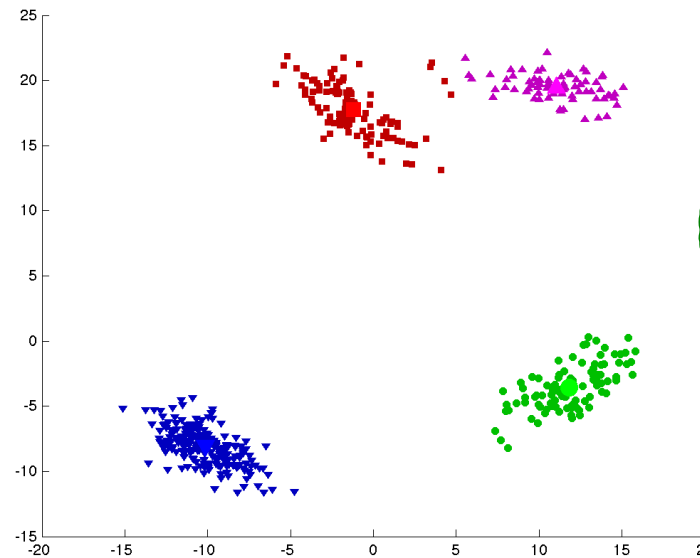
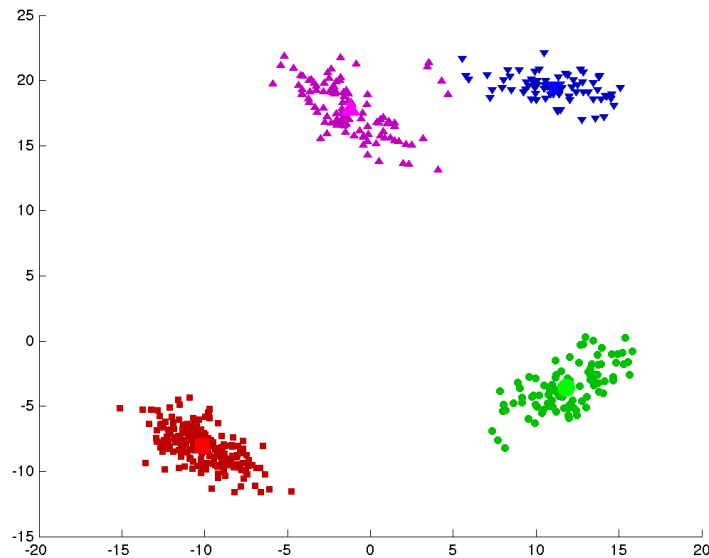
- This doesn't work because of “label switching” problem:
  - The cluster labels  $\hat{y}_i$  are meaningless.
  - We could get same clustering with permuted labels (“exchangeable”):



- All  $\hat{y}_i$  become equally likely as number of initializations increases.

# Addressing Label Switching Problem

- Ensembles can't depend on label "meaning":
  - Don't ask "is point  $x_i$  in red square cluster?", which is meaningless.
  - Ask "is point  $x_i$  in the same cluster as  $x_j$ ?", which is meaningful.



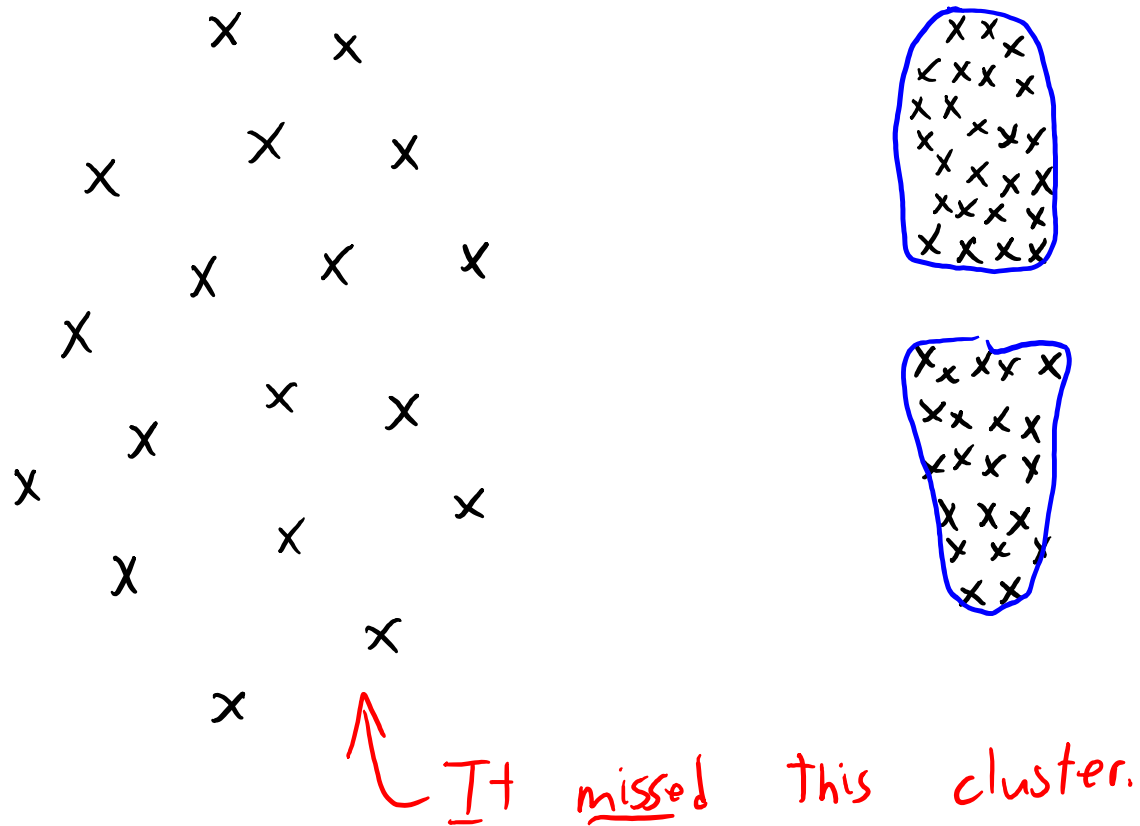
Different permutation  
of labels but  
same groups  
of points.

- Bonus slides give an example method ("UBClustering").

Next Topic: Hierarchical Clustering

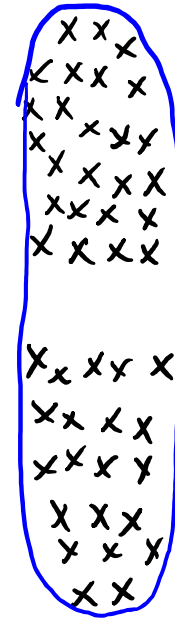
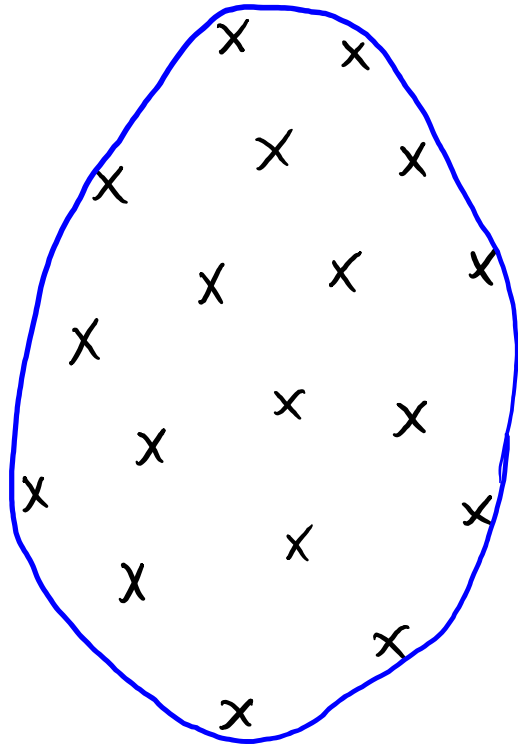
# Differing Densities

- Consider density-based clustering on this data:



# Differing Densities

- Increase epsilon and run it again:



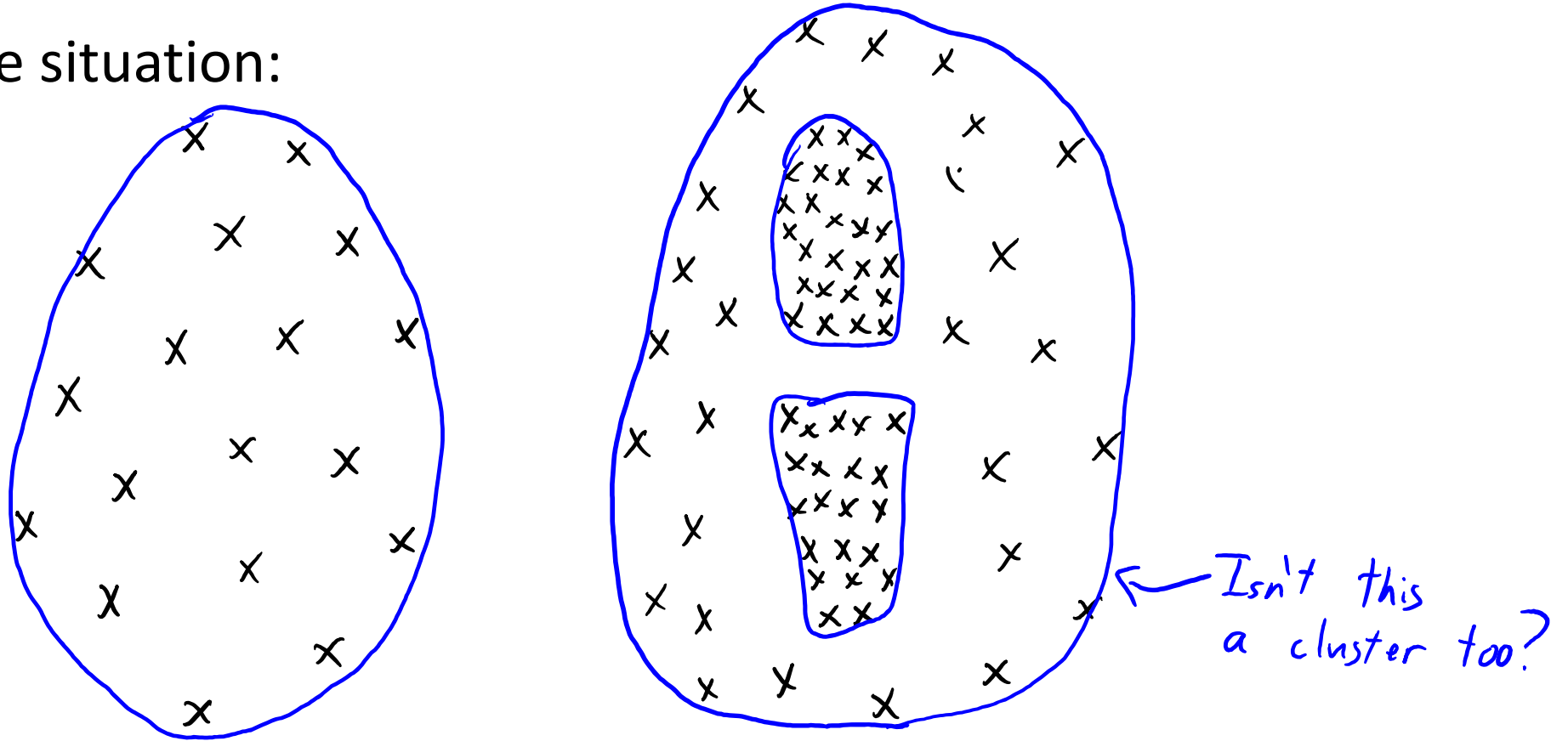
These 2 clusters  
are now "close."

- There may be **no density-level that gives you 3 clusters.**



# Differing Densities

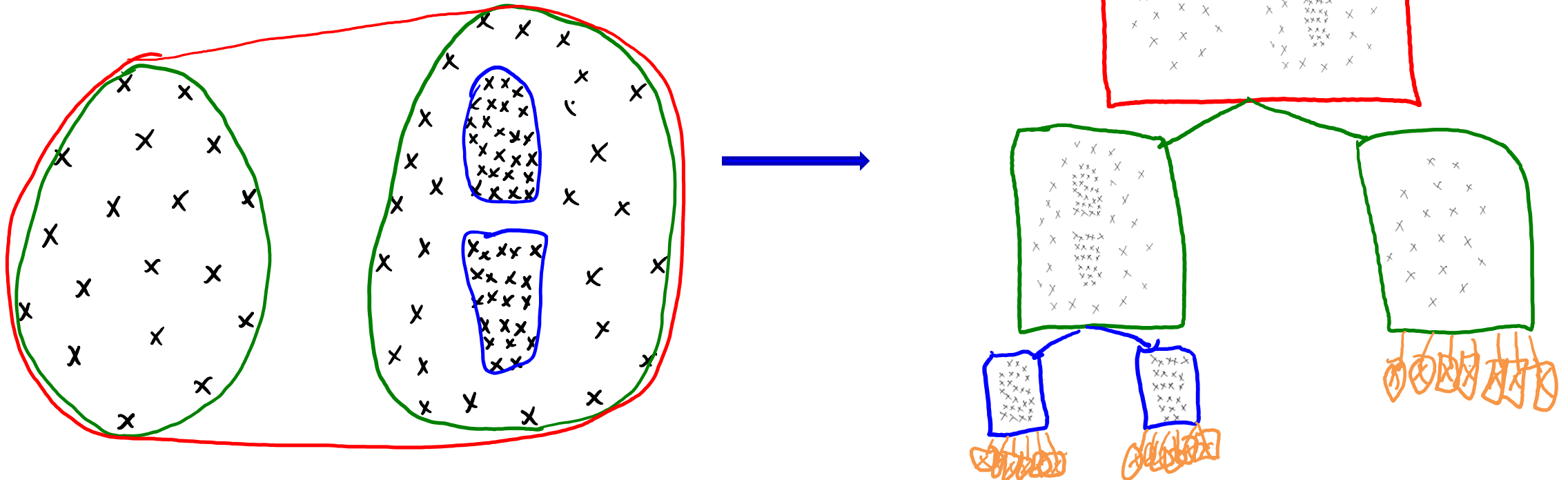
- Here is a worse situation:



- Now you need to choose between coarse/fine clusters.
- Instead of fixed clustering, we often want **hierarchical clustering**.

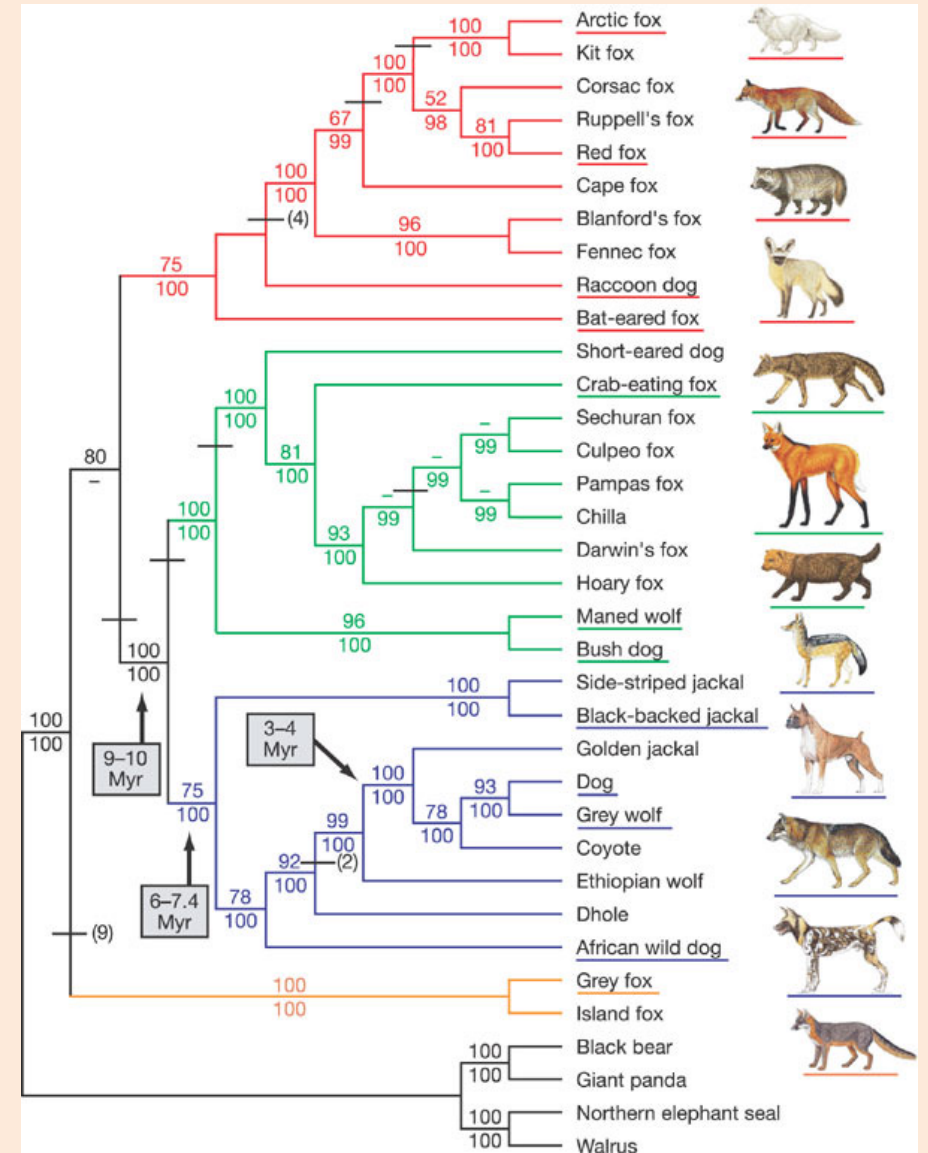
# Hierarchical Clustering

- Hierarchical clustering produces a tree of clusterings.
  - Each node in the tree splits the data into 2 or more clusters.
  - Much more information than using a fixed clustering.
  - Often have individual data points as leaves.



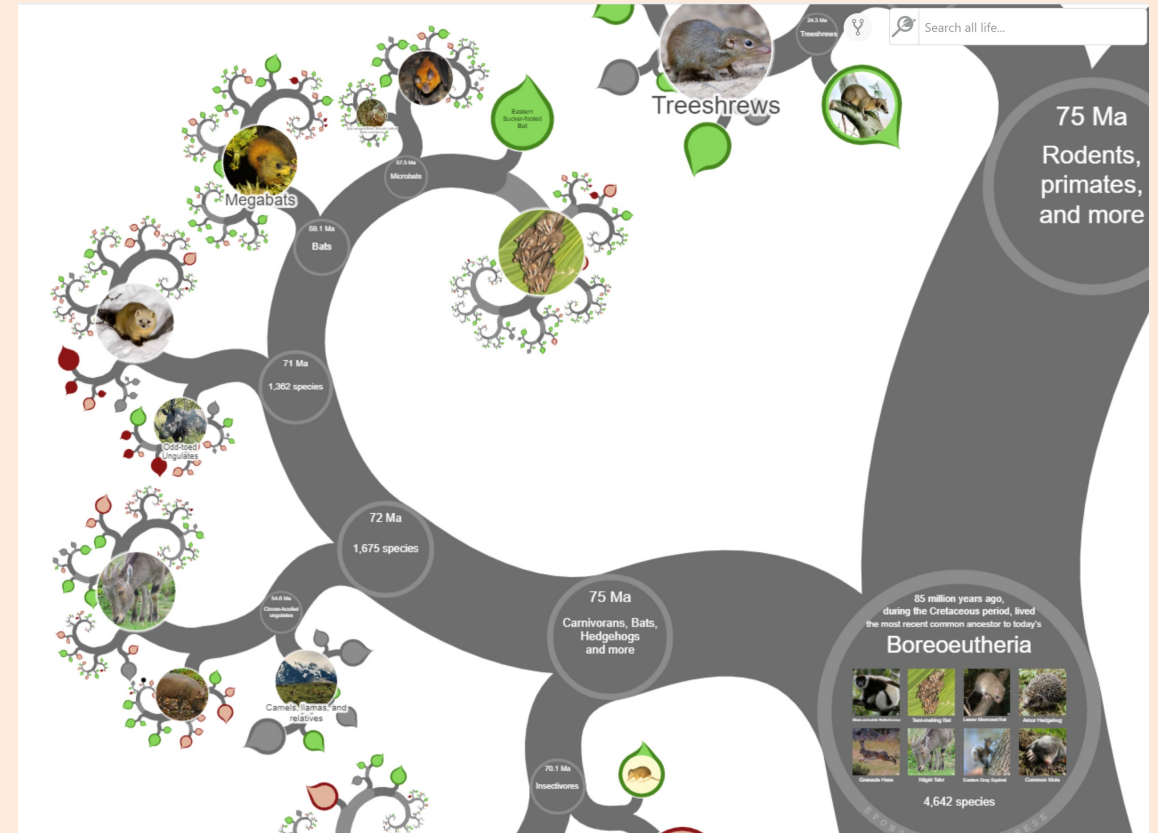
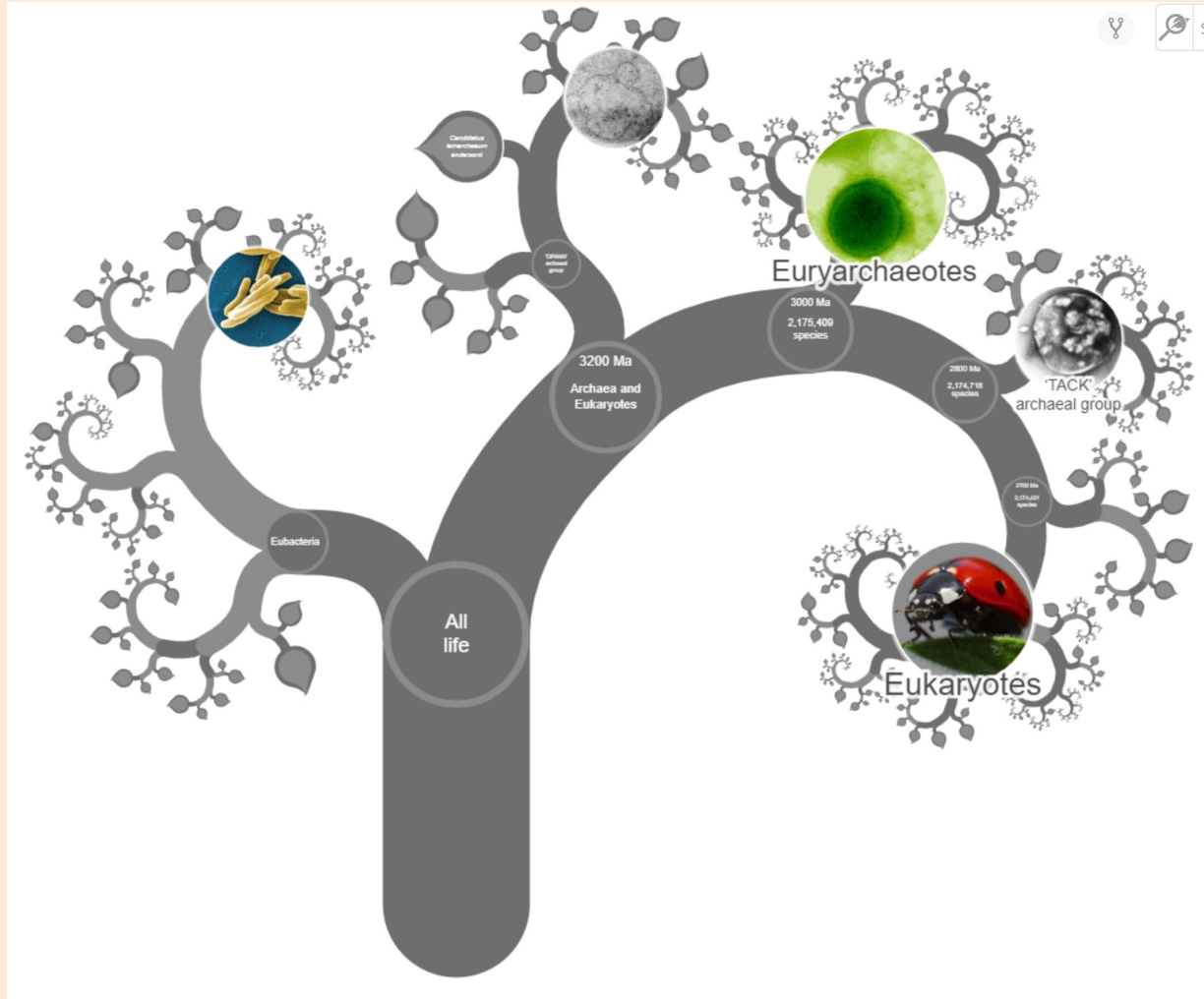
# Application: Phylogenetics

- We sequence genomes of a set of organisms.
- Can we construct the “tree of life”?
- Comments on this application:
  - On the right are individuals.
  - As you go left, clusters merge.
  - Merges are ‘common ancestors’.
- More useful information in the plot:
  - Line lengths: chosen here to approximate time.
  - Numbers: #clusterings across bootstrap samples.
  - ‘Outgroups’ (walrus, panda) are a sanity check.



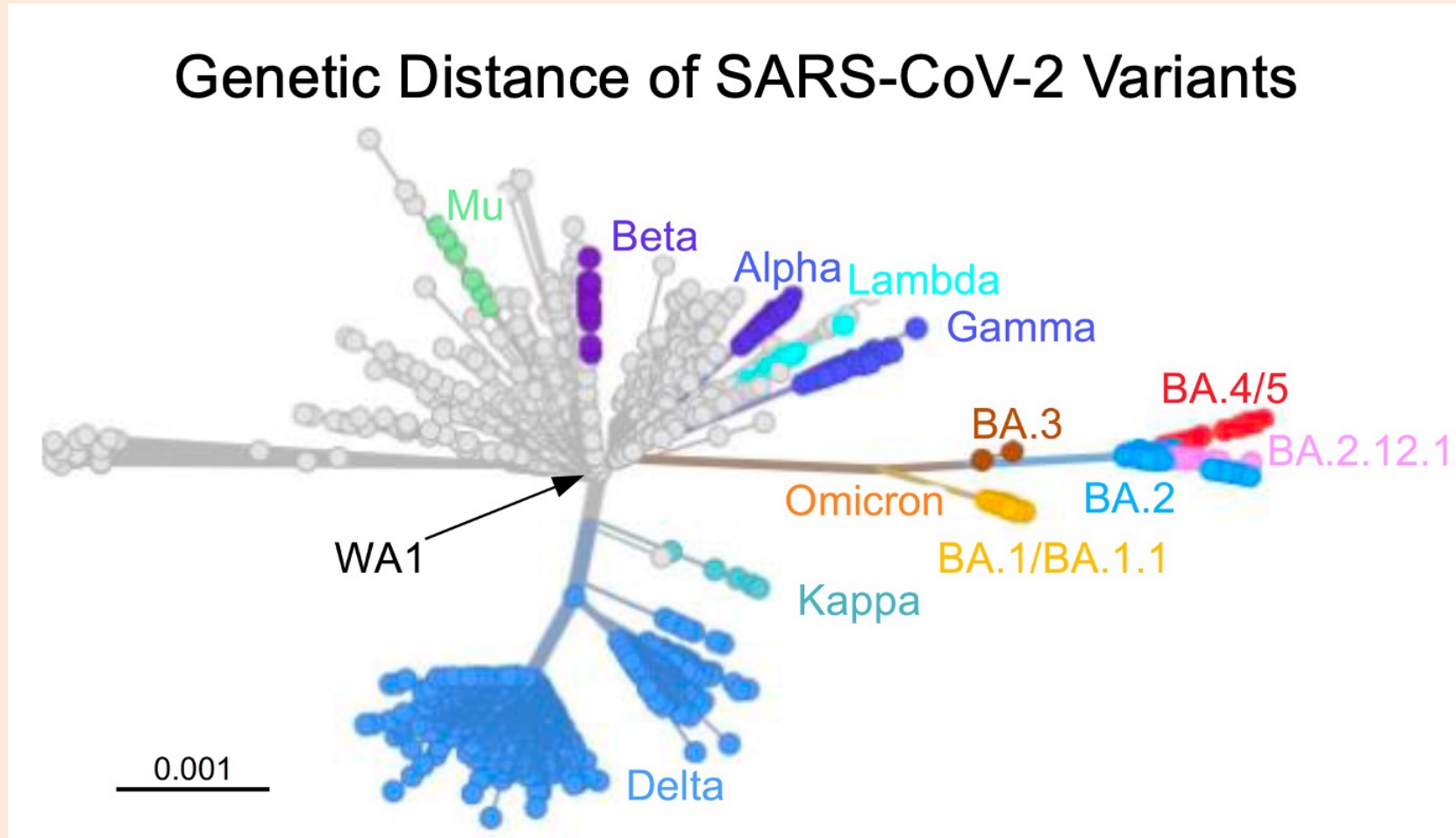
# Application: Phylogenetics

- Interactive demo of model of full tree of life: [www.onezoom.org](http://www.onezoom.org)



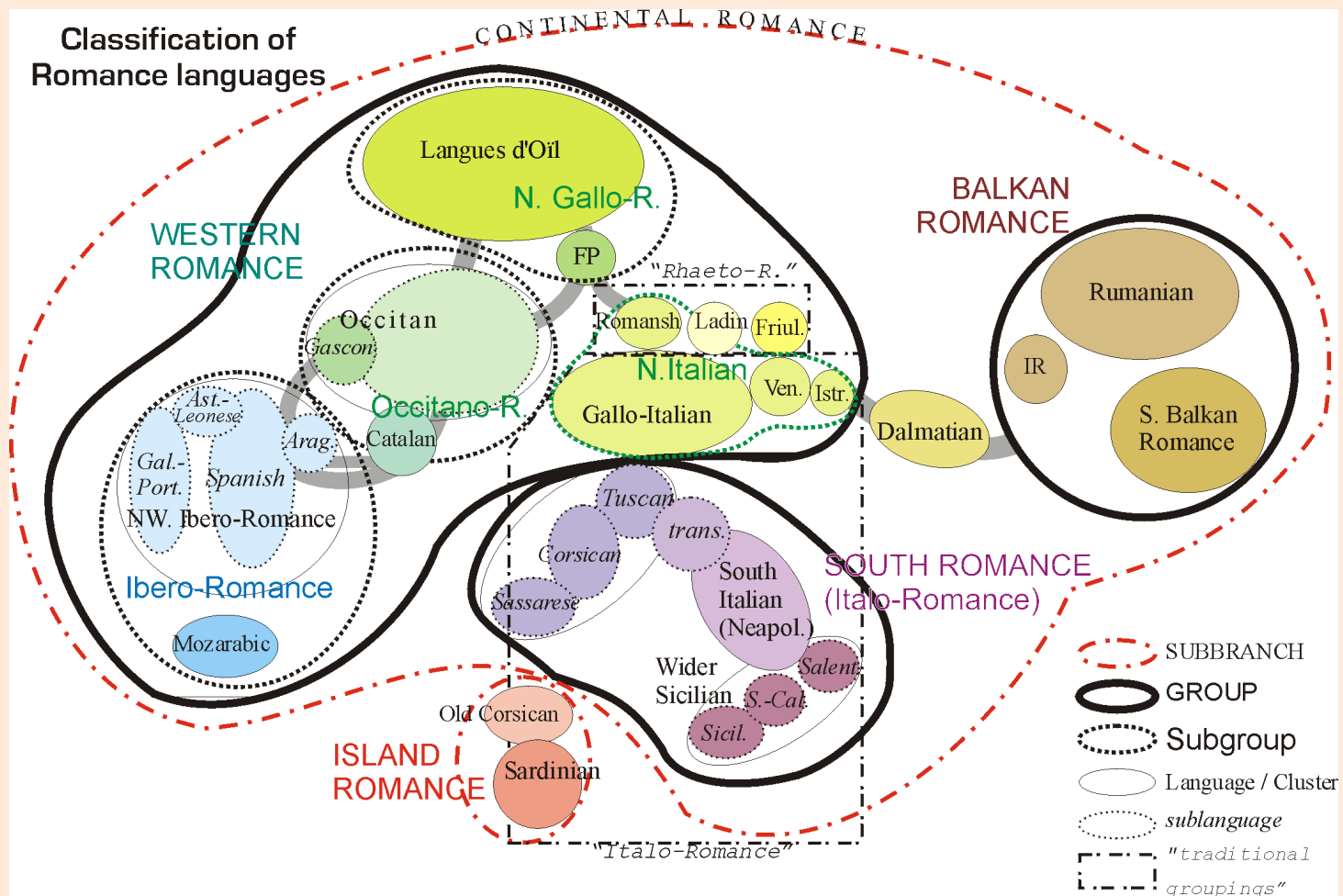
# Application: Phylogenetics

- Model of Covid-19 variants:



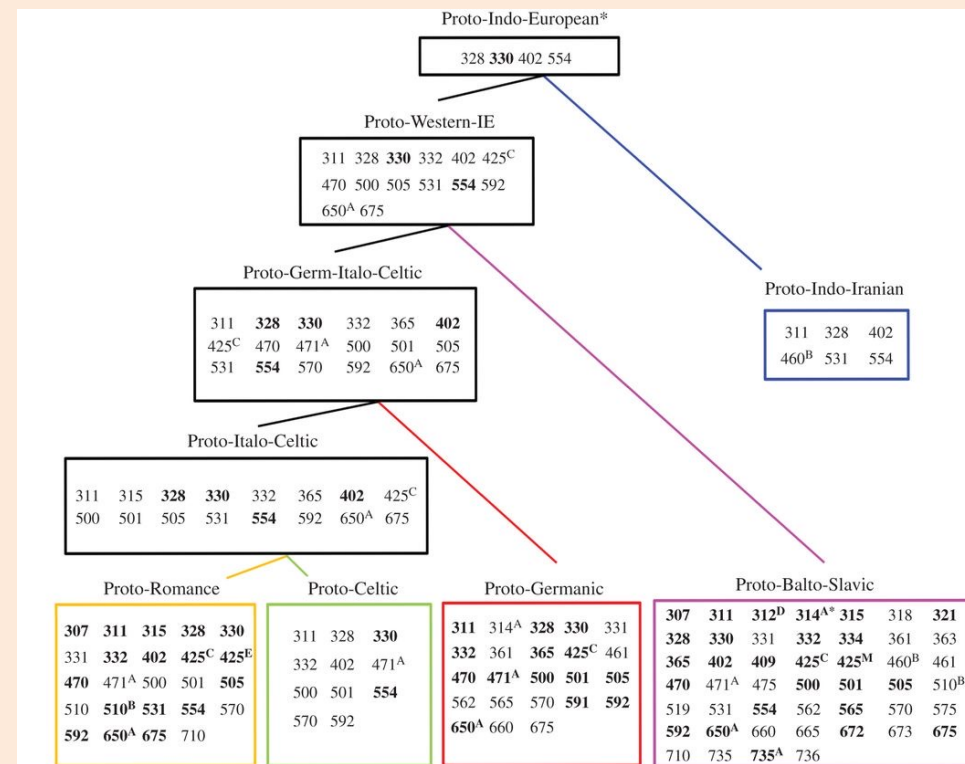
# Application: Phylogenetics

- Comparative method in linguistics studies evolution of languages:



# Application: Phylogenetics

- January 2016: evolution of fairy tales.
  - Evidence that “Devil and the Smith” goes back to bronze age.
  - “Beauty and the Beast” published in 1740, but might be 2500-6000 years old.

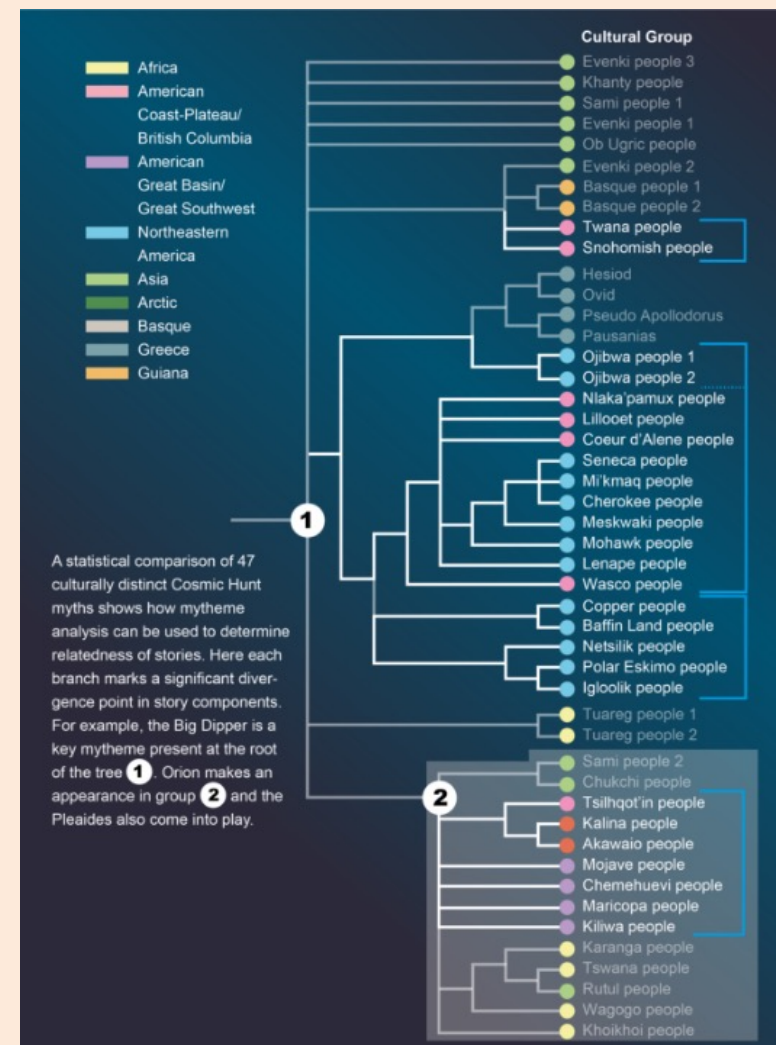


International tale types

307	The Princess in the Coffin	409	The Girl as Wolf	562	The Spirit in the Blue Light
311	Rescue by Sister	425 <sup>C</sup>	Beauty and the Beast	565	The Magic Mill
312 <sup>D</sup>	Rescue by the Brother	425 <sup>E</sup>	The Enchanted Husband	570	The Rabbit-Herd
314 <sup>A</sup>	The Shepherd and the Giants	425 <sup>M</sup>	The Snake Bridegroom	575	The Prince's Wings
314 <sup>A*</sup>	Animal Helper in the Flight	460 <sup>B</sup>	The Journey	591	The Thieving Pot
315	The Faithless Sister	461	Three Hairs	592	The Dance Among Thorns
318	The Faithless Wife	470	Friends in Life and Death	650 <sup>A</sup>	Strong John
321	Eyes Recovered from Witch	471 <sup>A</sup>	The Monk and the Bird	660	The Three Doctors
328	The Boy Steals Ogre's Treasure	475	The Man as the Heater	665	The Man who Flew and Swam
330	The Smith and the Devil	500	Supernatural Helper	672	The Serpent's Crown
331	The Spirit in the Bottle	501	The Three Old Spinning Women	673	The White Serpent's Flesh
332	Godfather Death	505	The Grateful Dead	675	The Lazy Boy
334	Household of the Witch	510	Cinderella and Peau d'Âne	710	Our Lady's Child
361	Bear Skin	510 <sup>B</sup>	Peau d'Asne	735	The Rich and the Poor Man
363	The Corpse-Eater	519	The Strong Woman as Bride	735 <sup>A</sup>	Bad Luck Imprisoned
365	The Dead Bridegroom	531	The Clever Horse	736	Luck and Wealth
402	The Animal Bride	554	The Grateful Animals		

# Application: Phylogenetics

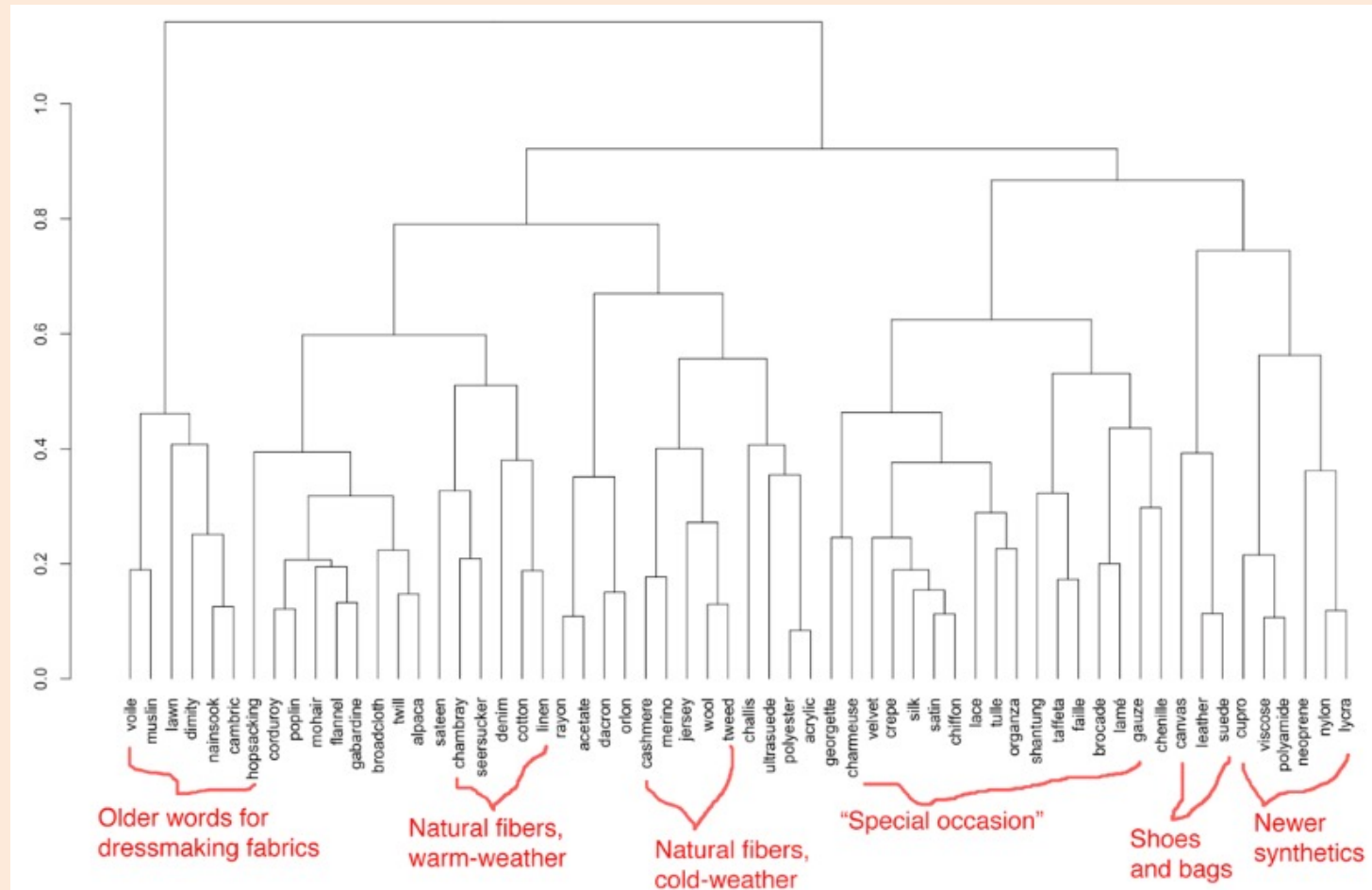
- January 2016: evolution of fairy tales.
  - Evidence that “Devil and the Smith” goes back to bronze age.
  - “Beauty and the Beast” published in 1740, but might be 2500-6000 years old.
- September 2016: evolution of myths.
  - “Cosmic hunt” story:
    - Person hunts animal that becomes constellation.
      - Previously known to be at least 15,000 years old.
    - May go back to paleolithic period.





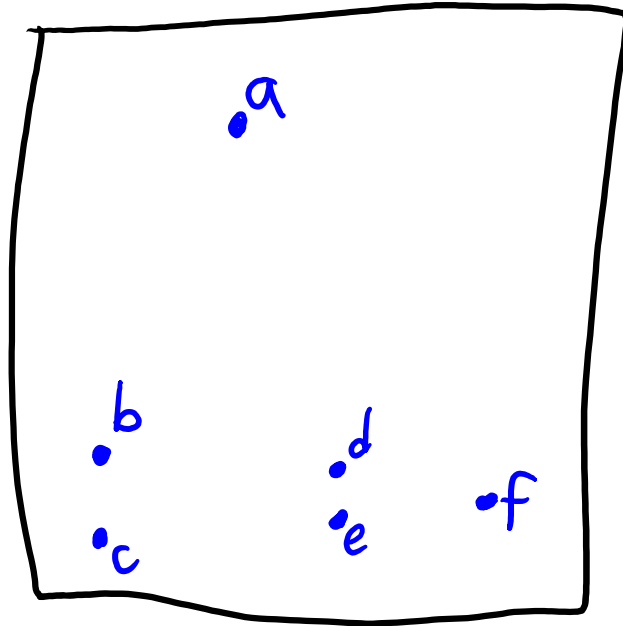
# Application: Fashion?

- Hierarchical clustering of clothing material words in Vogue:



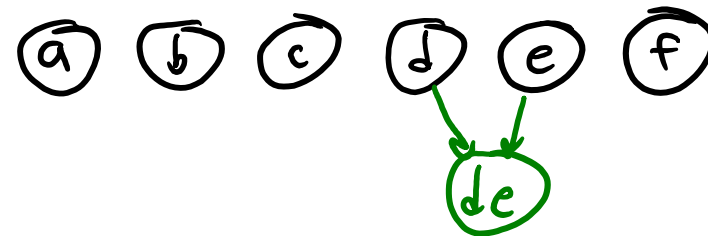
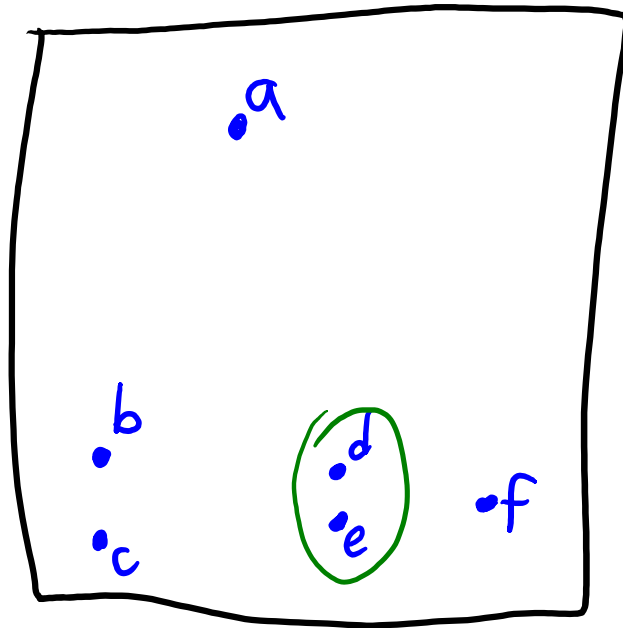
# Agglomerative (Bottom-Up) Clustering

- Most common hierarchical method: **agglomerative clustering**.
  1. Starts with **each point in its own cluster**.



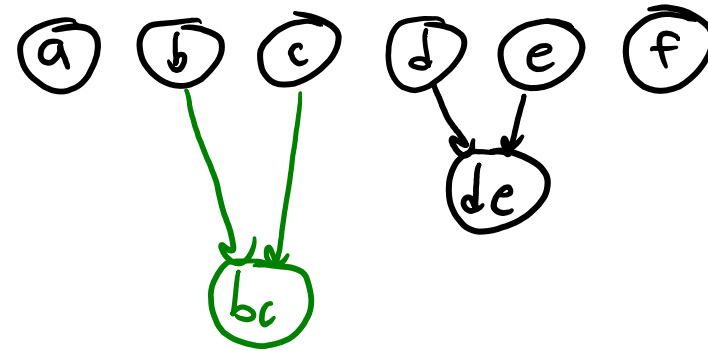
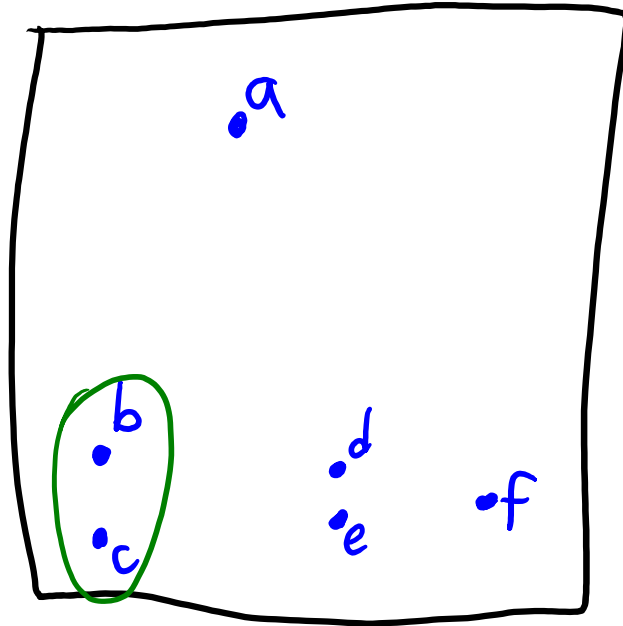
# Agglomerative (Bottom-Up) Clustering

- Most common hierarchical method: **agglomerative clustering**.
  1. Starts with **each point in its own cluster**.
  2. Each step **merges the two "closest" clusters**.



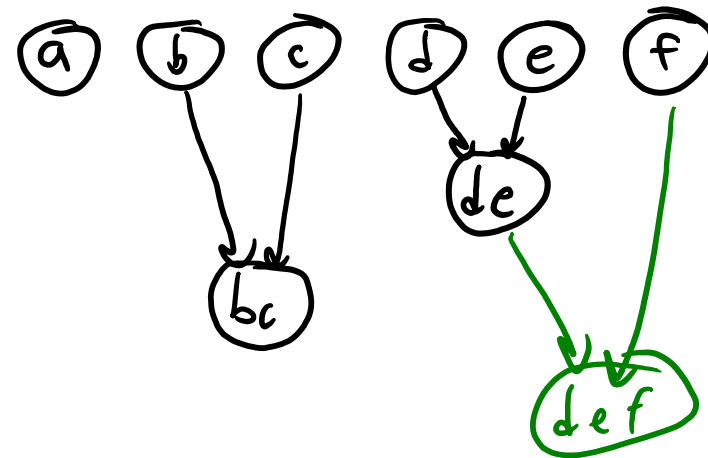
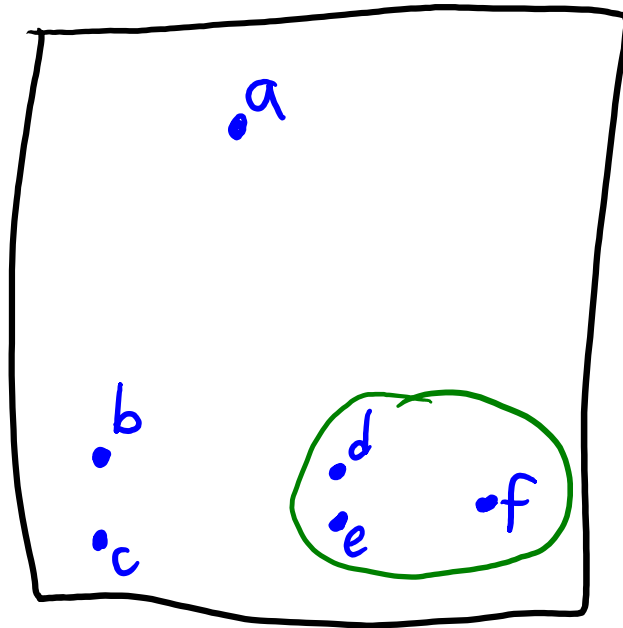
# Agglomerative (Bottom-Up) Clustering

- Most common hierarchical method: **agglomerative clustering**.
  1. Starts with **each point in its own cluster**.
  2. Each step **merges the two "closest" clusters**.



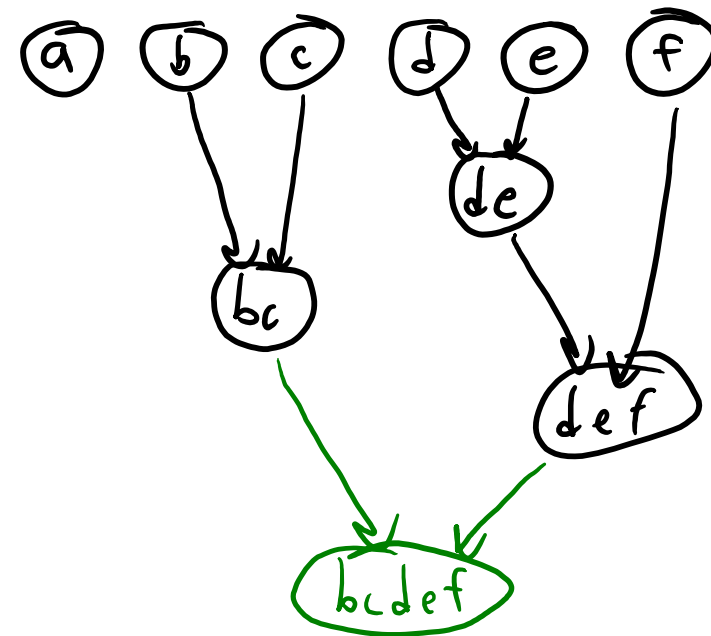
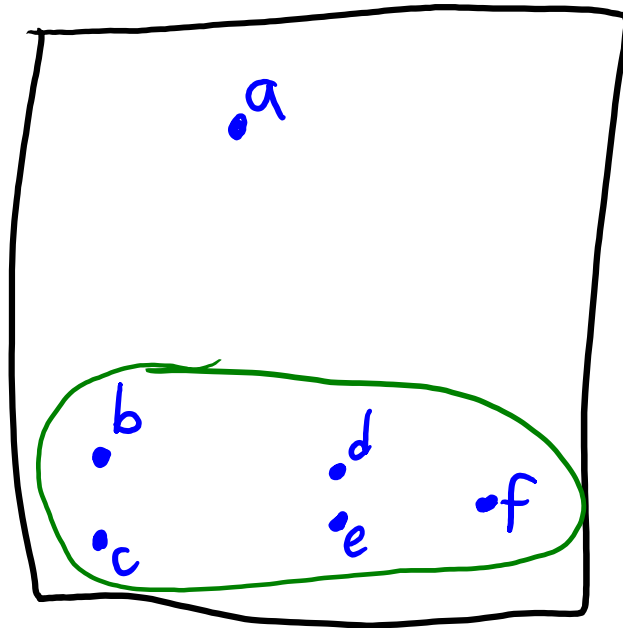
# Agglomerative (Bottom-Up) Clustering

- Most common hierarchical method: **agglomerative clustering**.
  1. Starts with **each point in its own cluster**.
  2. Each step **merges the two "closest" clusters**.



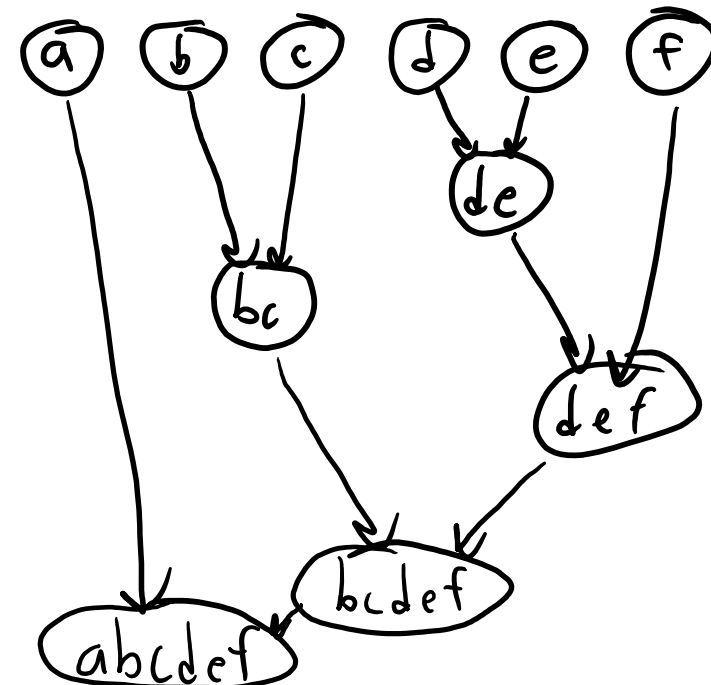
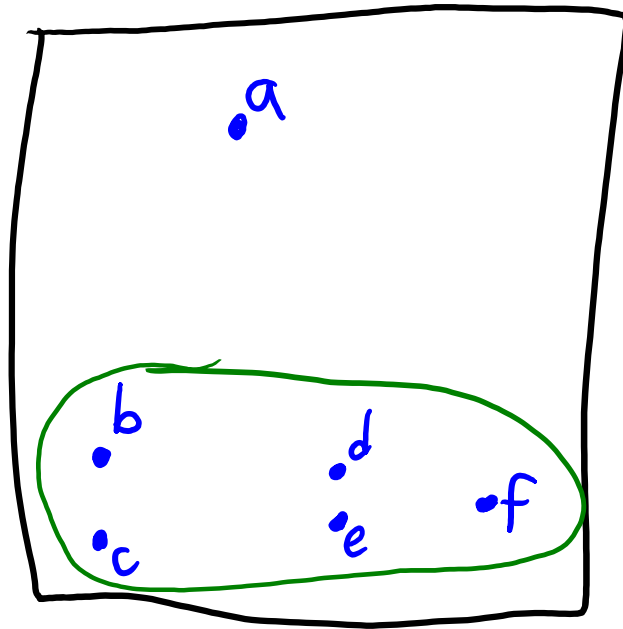
# Agglomerative (Bottom-Up) Clustering

- Most common hierarchical method: **agglomerative clustering**.
  1. Starts with **each point in its own cluster**.
  2. Each step **merges the two "closest" clusters**.



# Agglomerative (Bottom-Up) Clustering

- Most common hierarchical method: **agglomerative clustering**.
  1. Starts with **each point in its own cluster**.
  2. Each step **merges the two "closest" clusters**.
  3. **Stop with one big cluster** that has all points.



Output is  
the tree.

[Animation](#)

# Agglomerative (Bottom-Up) Clustering

- Reinvented by different fields under different names (“UPGMA”).
- Needs a “distance” between two clusters.
- A standard choice: distance between means of the clusters.
  - Not necessarily the best, many choices exist (bonus slide).
- Cost is  $O(n^3d)$  for basic implementation.
  - Each step costs  $O(n^2d)$ , and each step might only cluster 1 new point.



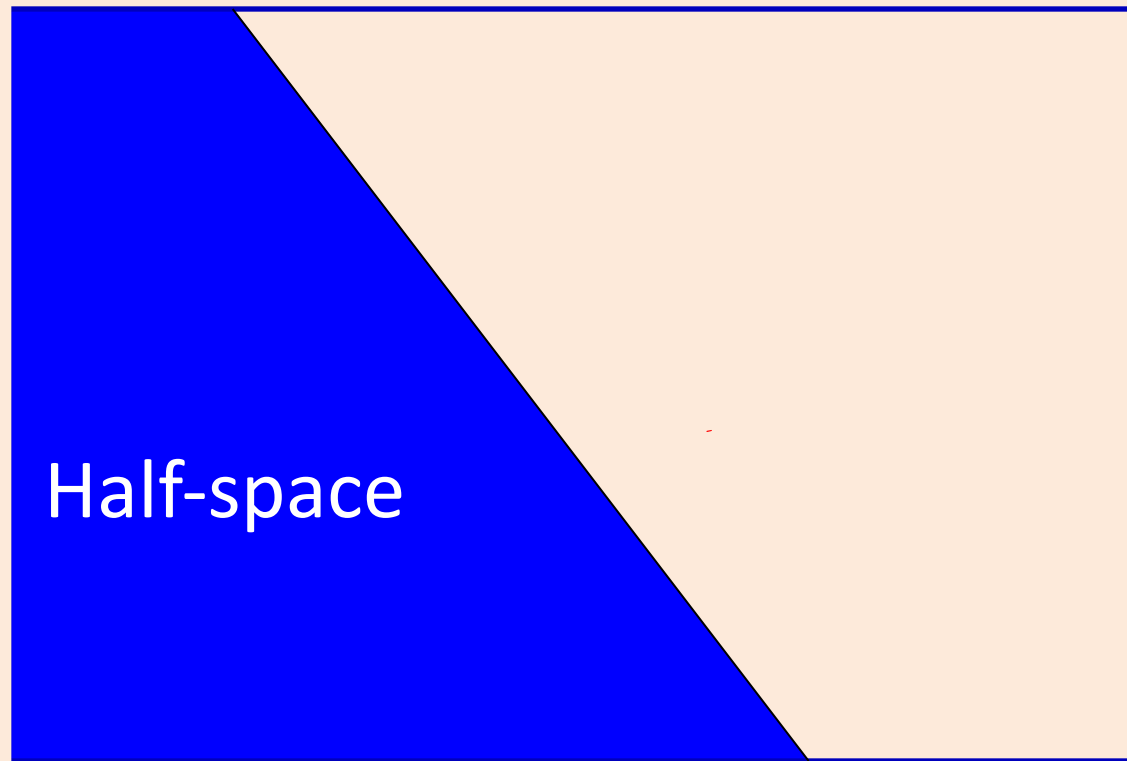
# Summary

- **Vector quantization:**
  - Compressing examples by replacing them with the mean of their cluster.
- **Shape of K-means clusters:**
  - Partitions space into convex sets.
- **Density-based clustering:**
  - “Expand” and “merge” dense regions of points to find clusters.
  - Not very sensitive to initialization or outliers.
  - Useful for finding non-convex connected clusters.
- **Ensemble clustering:** combines multiple clusterings.
  - Can work well but need to account for **label switching**.
- **Hierarchical clustering:** more informative than fixed clustering.
- **Agglomerative clustering:** standard hierarchical clustering method.
  - Each point starts as a cluster, sequentially merge clusters.

# Why are k-means clusters convex?

- K-means clusters are formed by the **intersection** of **half-spaces**.

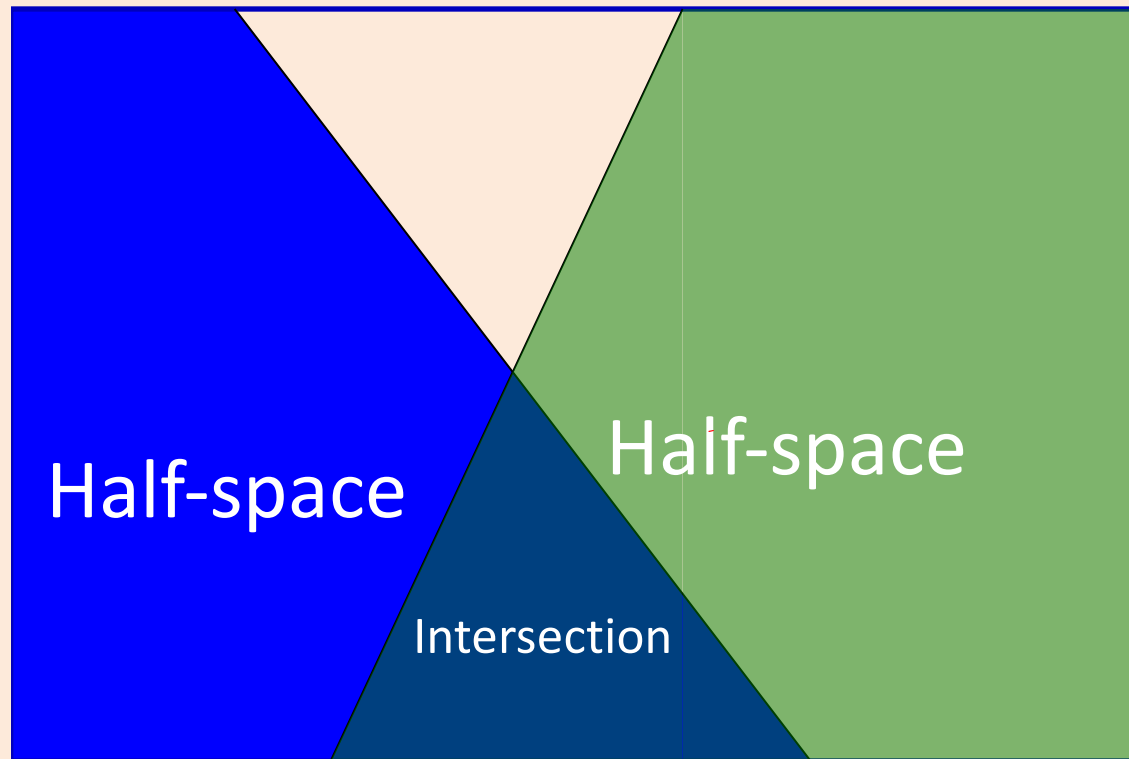
Half-space is Set of points satisfying a linear inequality, like  $\sum_{j=1}^d a_j x_j \leq b$



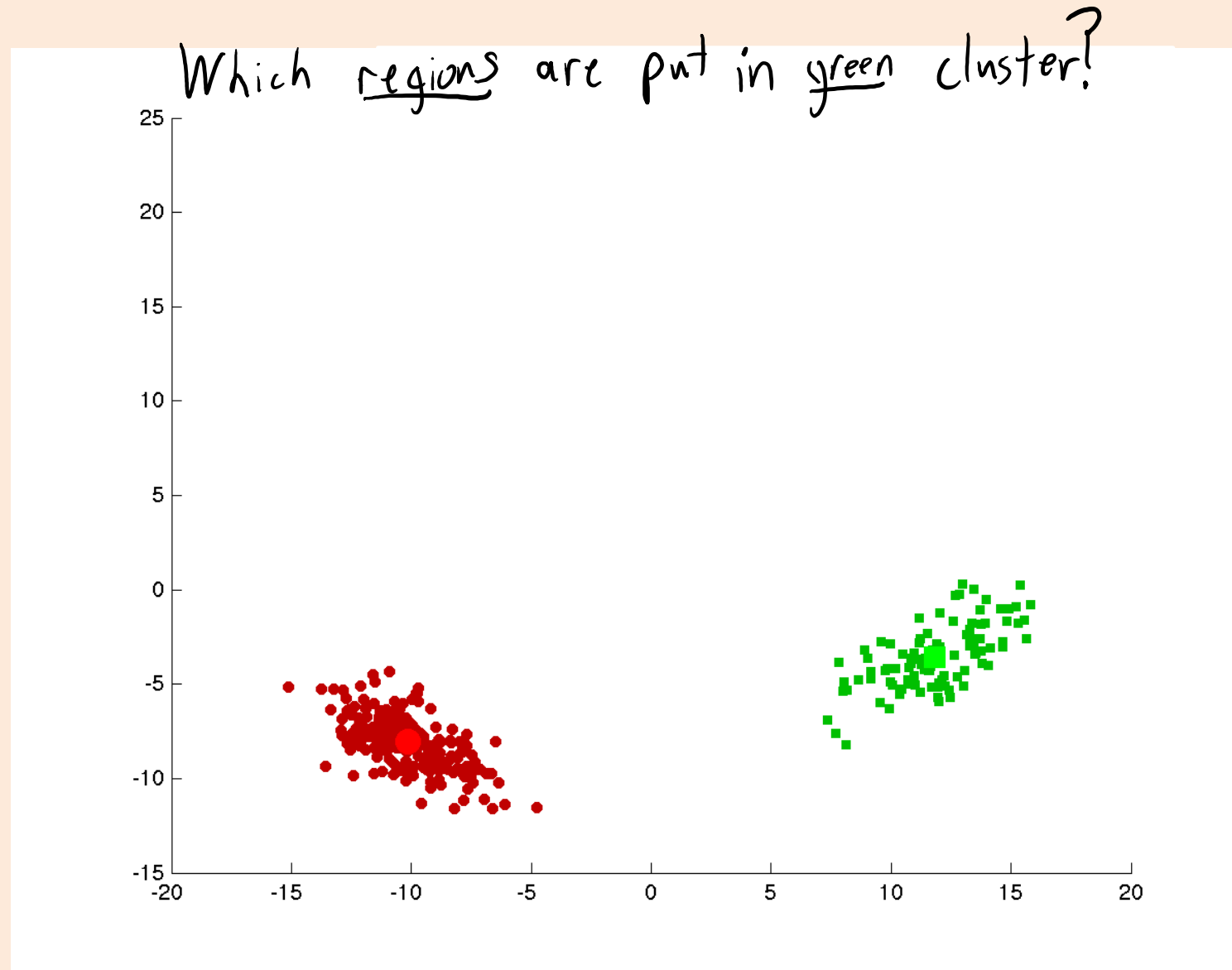
# Why are k-means clusters convex?

- K-means clusters are formed by the **intersection** of **half-spaces**.

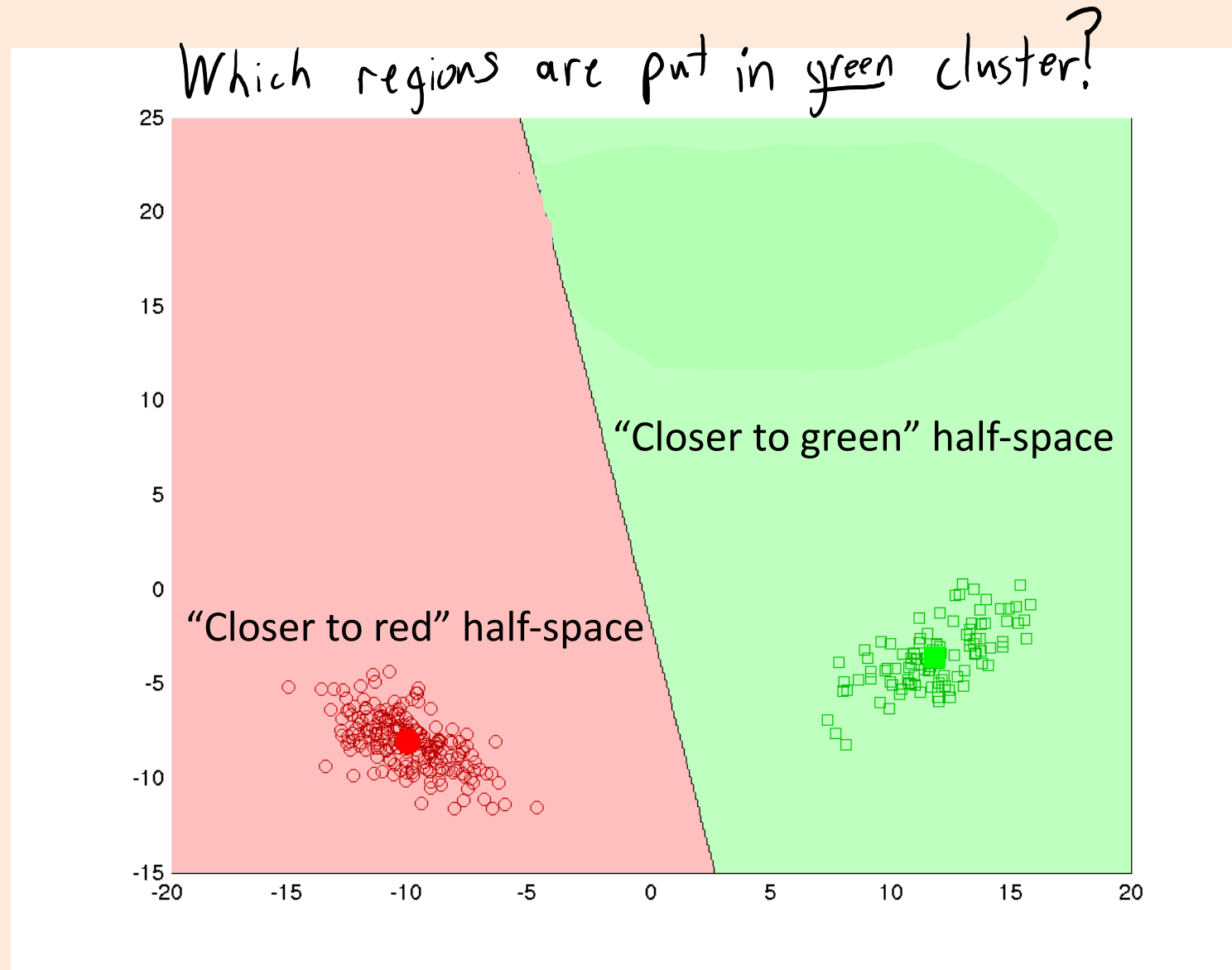
Half-space is Set of points  $\left\{ \begin{array}{l} \text{satisfying a} \\ \text{linear inequality} \end{array} \right.$ , like  $\sum_{j=1}^d a_j x_j \leq b$



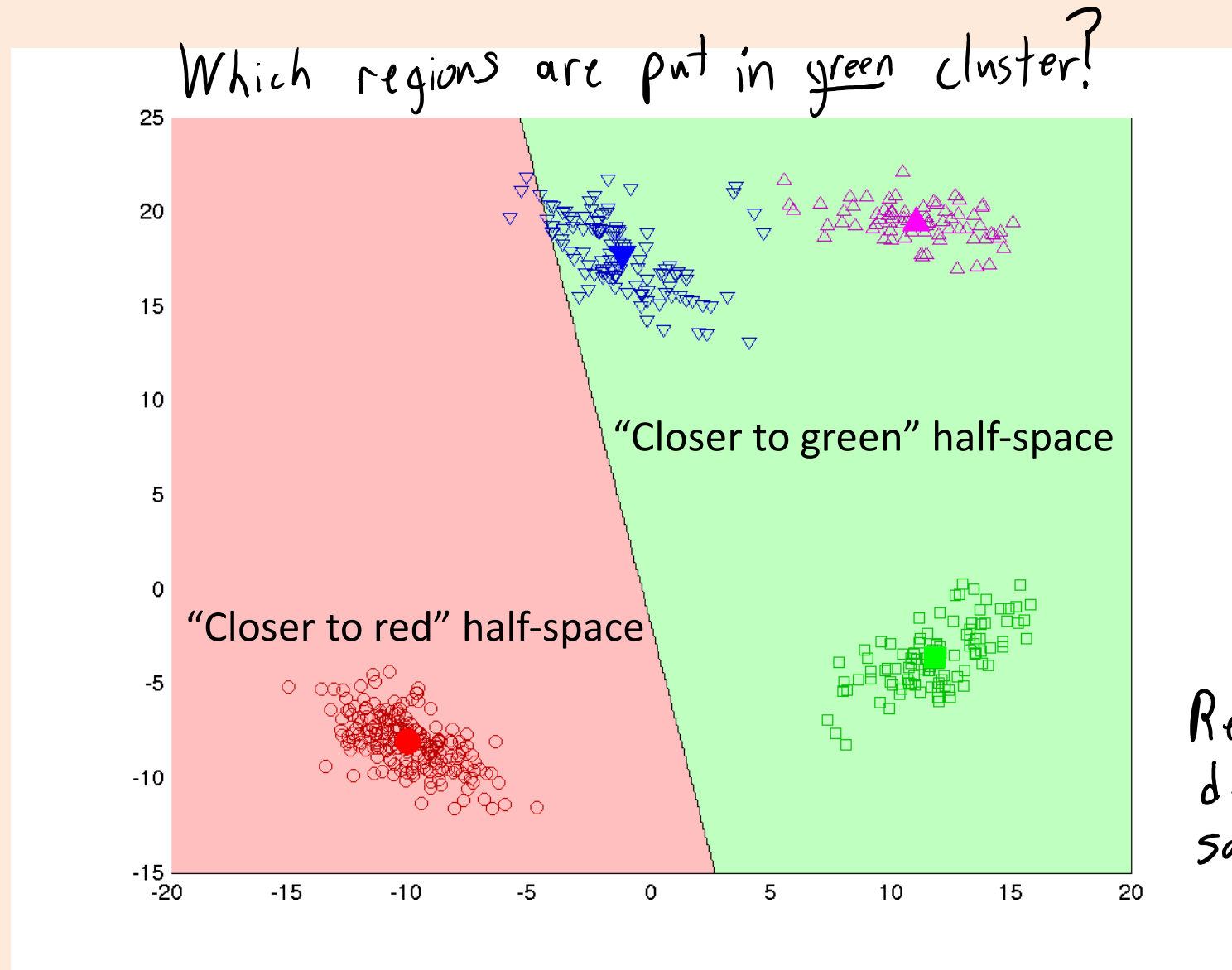
# Why are k-means clusters convex?



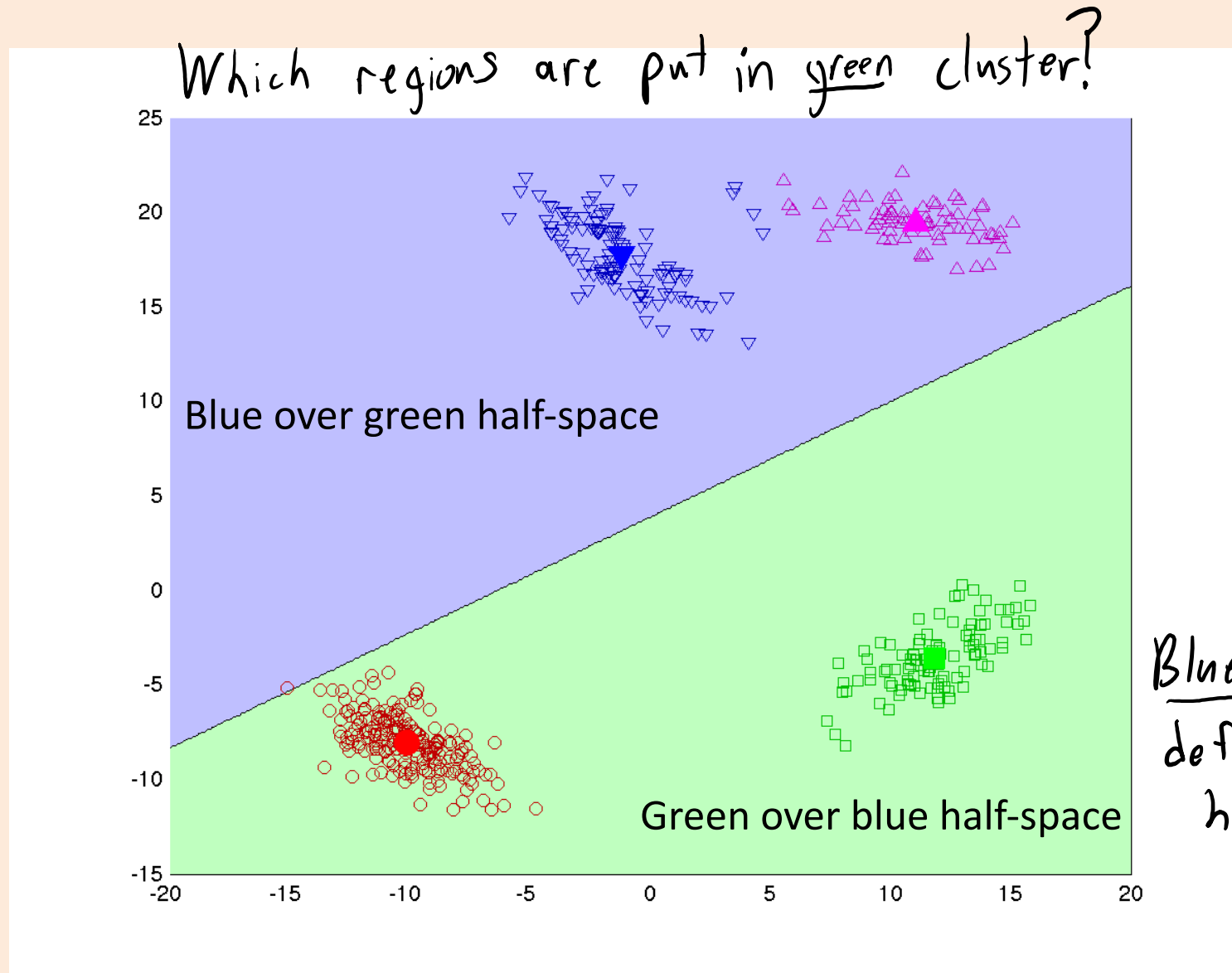
# Why are k-means clusters convex?



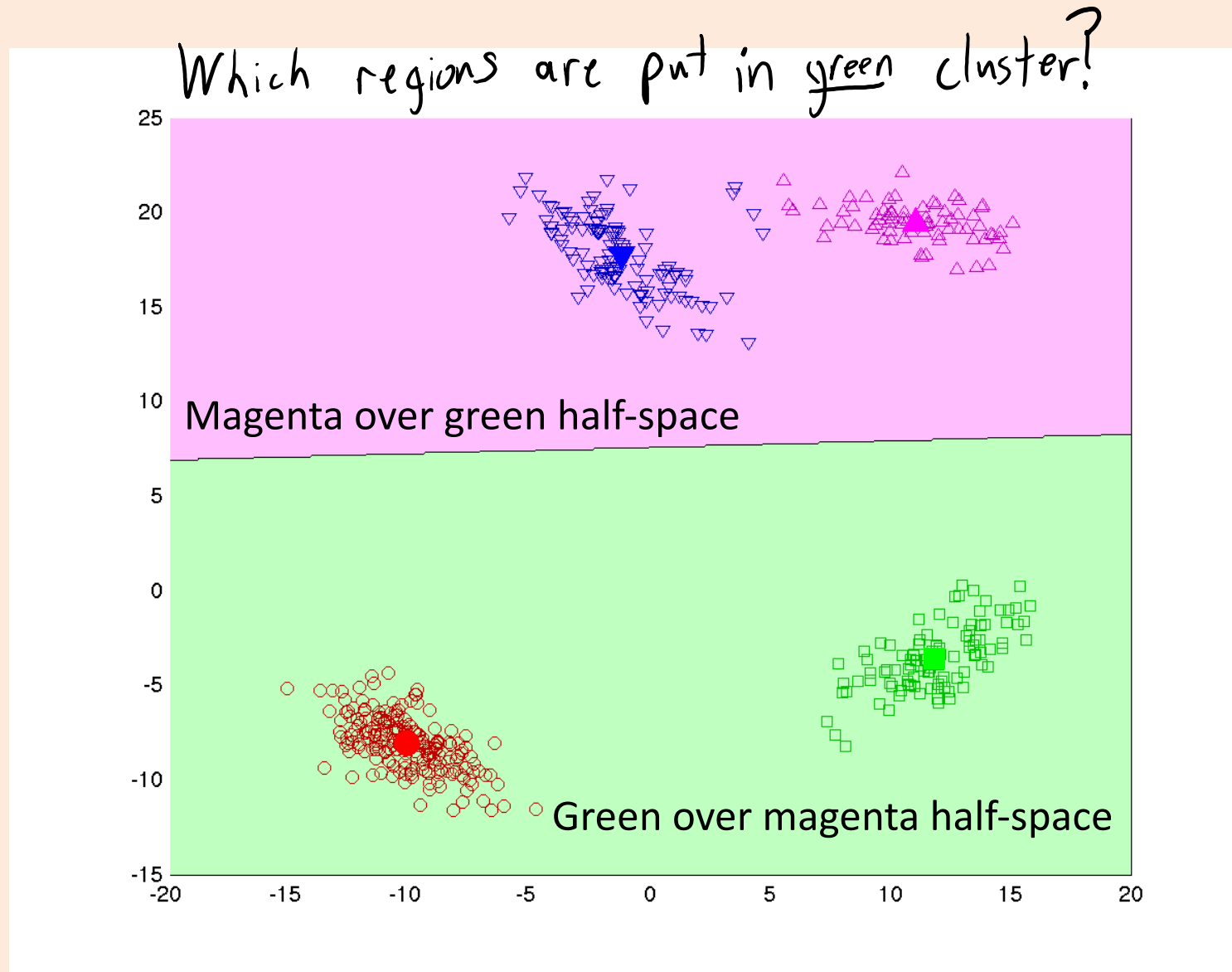
# Why are k-means clusters convex?



# Why are k-means clusters convex?

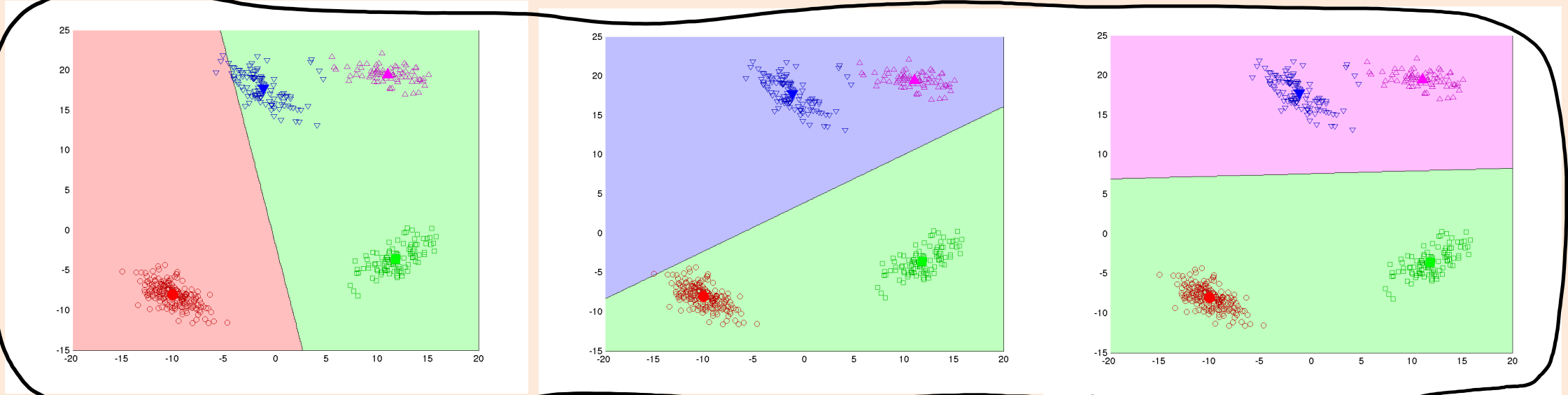


# Why are k-means clusters convex?



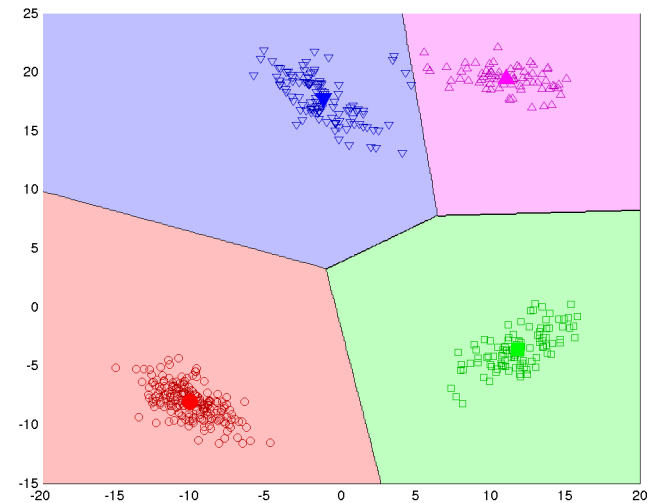


# Why are k-means clusters convex?



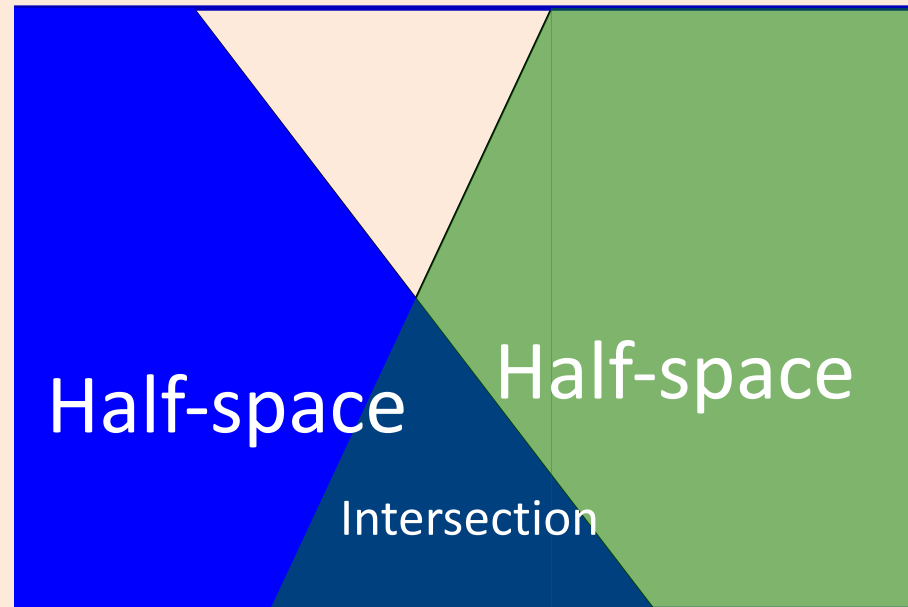
Green "cluster" is the intersection of these three half-spaces.

Here is what the four clusters look like:



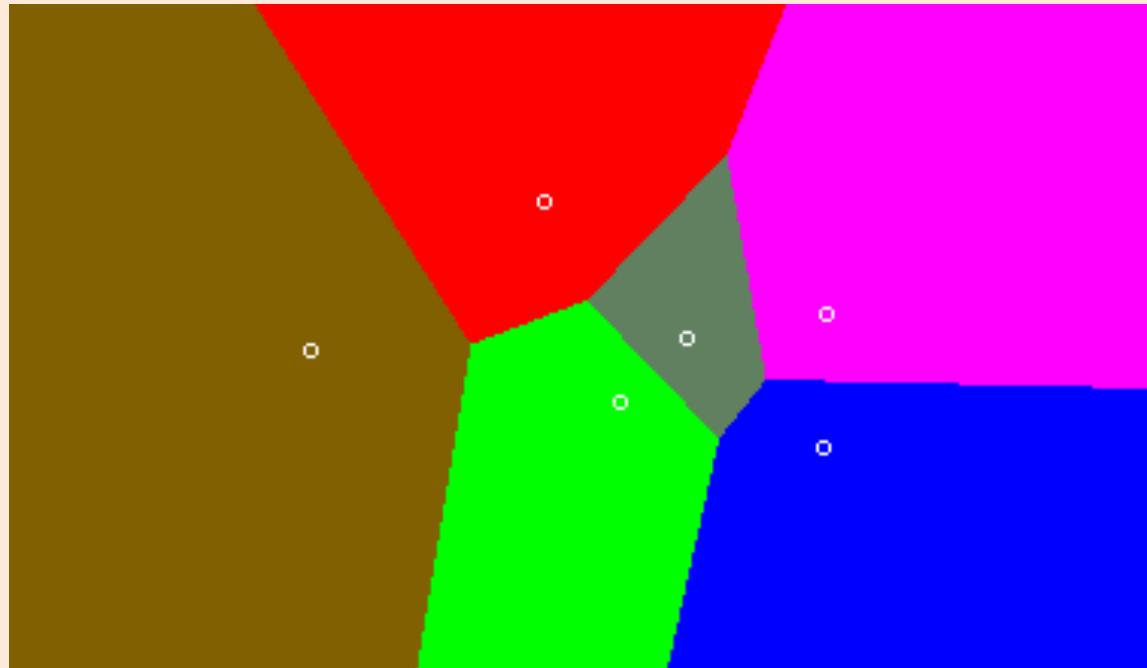
# Why are k-means clusters convex?

- Half-spaces are convex sets.
- Intersection of convex sets is a convex set.
  - Line segment between points in each set are still in each set.
- So intersection of half-spaces is convex.



# Voronoi Diagrams

- The k-means partition can be visualized as a [Voronoi diagram](#):



- Can be a useful visualization of “nearest available” problems.
  - E.g., [nearest tube station in London](#).

# Density-Based Clustering Runtime

? question ☆

stop following

72 views

Actions ▾

## DBSCAN Training time & Testing time

This is a follow-up inquiry post with Mike about the DBScan, would like to know:

1. Training runtime of DBScan, under  $k$  iterations (training set  $X$  has  $n$  examples and  $d$  features)
2. Testing runtime for a single example in DBScan; Testing runtime for test set of size  $t$  in DBScan,

**i** the instructors' answer, *where instructors collectively construct a single answer*

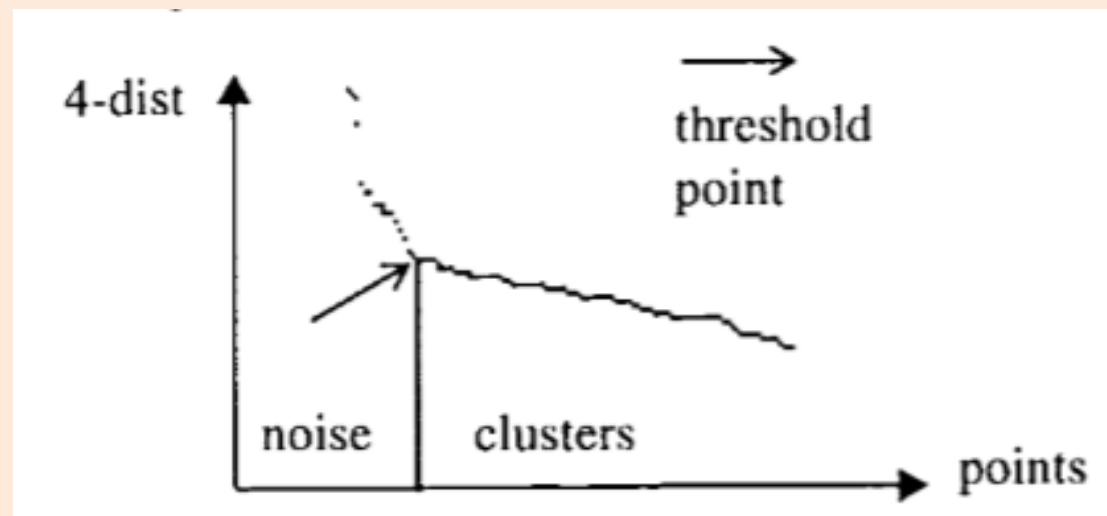
For training, you'll check that each point is a core point exactly once. This check costs  $O(nd)$  since you measure the distance to each other point, leading to a total training cost of  $O(n^2d)$ .

(There are ways to speed this up, like grid-based pruning.)

We didn't define how to apply the DBSCAN model to test data. But a plausible way is to test if the new point is a neighbor of any existing core points. If you have  $m$  core points, you would be able to do this in  $O(md)$ .

# “Elbow” Method for Density-Based Clustering

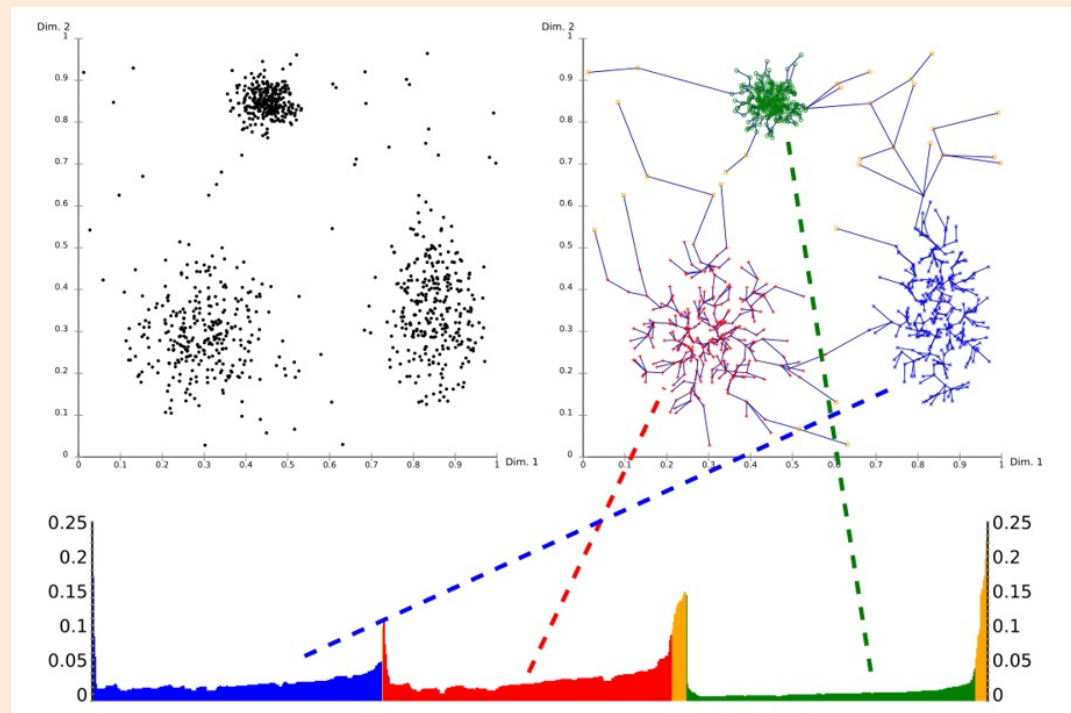
- From the original DBSCAN paper:
  - Choose some ‘k’ (they suggest 4) and set minNeighbours=k.
  - Compute distance of each points to its ‘k’ nearest neighbours.
  - Sort the points based on these distances and plot the distances:



- Look for an “elbow” to choose  $\epsilon$ .

# OPTICS

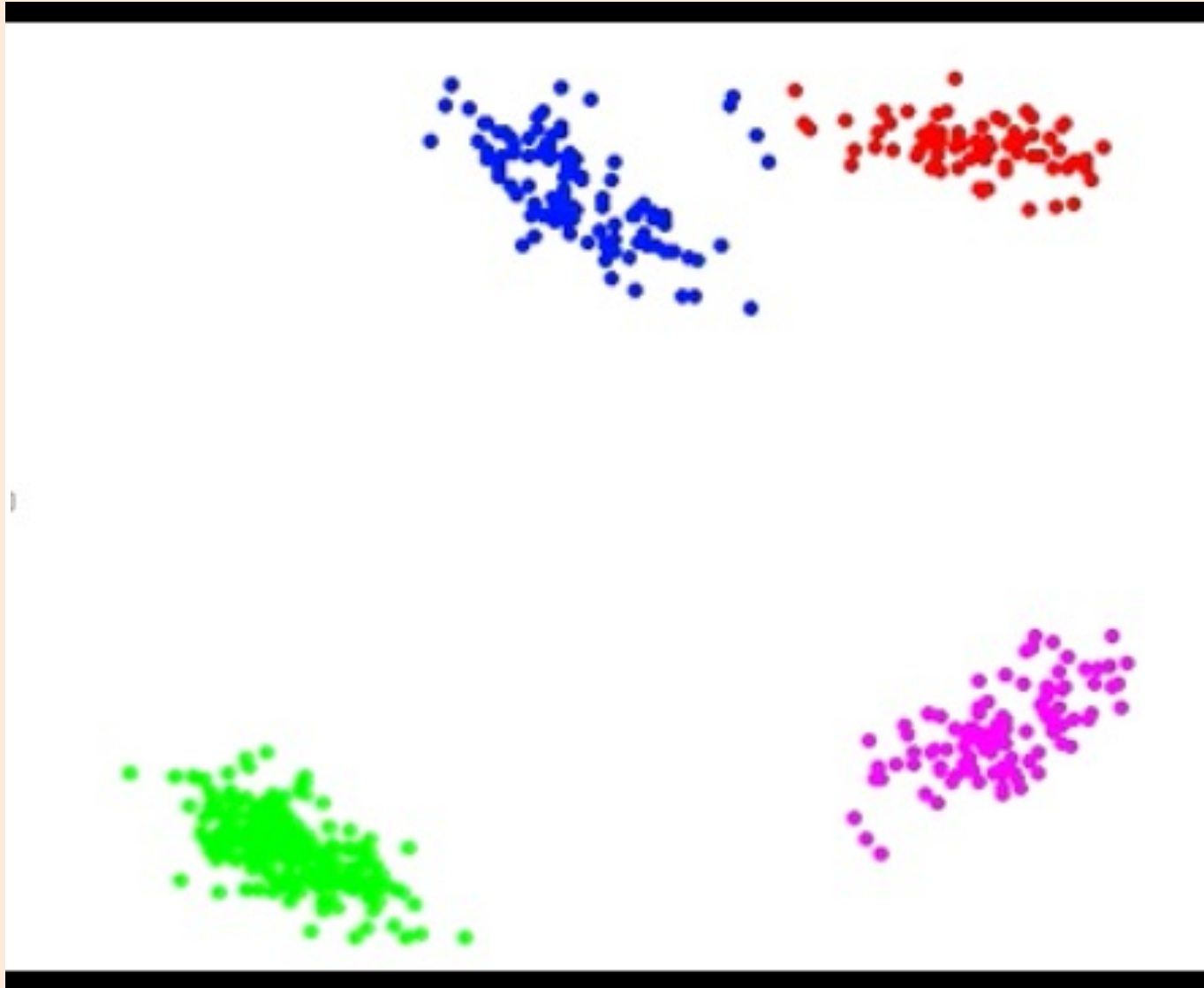
- Related to the DBSCAN “elbow” is “OPTICS”.
  - Sort the points so that neighbours are close to each other in the ordering.
  - Plot the distance from each point to the next point.
  - Clusters should correspond to sequencers with low distance.



# UBClustering Algorithm

- Let's define a new ensemble clustering method: **UBClustering**.
  1. Run k-means with 'm' different random initializations.
  2. For each example  $i$  and  $j$ :
    - Count the number of times  $x_i$  and  $x_j$  are in the same cluster.
    - Define  $p(i,j) = \text{count}(x_i \text{ in same cluster as } x_j)/m$ .
  3. Put  $x_i$  and  $x_j$  in the same cluster if  $p(i,j) > 0.5$ .
- Like DBSCAN **merge clusters** in step 3 if  $i$  or  $j$  are already assigned.
  - You can implement this with a DBSCAN code (just changes "distance").
  - Each  $x_i$  has an  $x_j$  in its cluster with  $p(i,j) > 0.5$ .
  - Some points are not assigned to any cluster.

# UBClustering Algorithm



It looks like DBSCAN, but far-away points will be assigned to a cluster if they always appear in same cluster as other points.



# Distances between Clusters

- Other choices of the distance between two clusters:
  - “Single-link”: minimum distance between points in clusters.
  - “Average-link”: average distance between points in clusters.
  - “Complete-link”: maximum distance between points in clusters.
  - Ward’s method: minimize within-cluster variance.
  - “Centroid-link”: distance between a representative point in the cluster.
    - Useful for distance measures on non-Euclidean spaces (like Jaccard similarity).
    - “Centroid” often defined as point in cluster minimizing average distance to other points.

# Cost of Agglomerative Clustering

- One step of agglomerative clustering costs  $O(n^2d)$ :
  - We need to do the  $O(d)$  distance calculation between up to  $O(n^2)$  points.
  - This is assuming the standard distance functions.
- We do at most  $O(n)$  steps:
  - Starting with 'n' clusters and merging 2 clusters on each step, after  $O(n)$  steps we'll only have 1 cluster left (though typically it will be much smaller).
- This gives a total cost of  $O(n^3d)$ .
- This can be reduced to  $O(n^2d \log n)$  with a priority queue:
  - Store distances in a sorted order, only update the distances that change.
- For single- and complete-linkage, you can get it down to  $O(n^2d)$ .
  - “SLINK” and “CLINK” algorithms.

# Bonus Slide: Divisive (Top-Down) Clustering

- Start with all examples in one cluster, then start dividing.
- E.g., run k-means on a cluster, then run again on resulting clusters.
  - A clustering analogue of decision tree learning.

