

CPSC 340: Machine Learning and Data Mining

Clustering: *k*-means and DBSCAN

Admin

- **Assignment 2** is out
 - Due Friday (a week from today)

Application: Classifying Cancer Types

- “I collected gene expression data for 1000 different types of cancer cells, can you tell me the different classes of cancer?”
- We are not given the class labels y , but want **meaningful labels**.
- An example of **unsupervised learning**.

Unsupervised Learning

- Supervised learning:
 - We have features x_i and class labels y_i .
 - Write a program that produces y_i from x_i .
- Unsupervised learning:
 - We **only have x_i values**, but no explicit target labels.
 - You want to do “something” with them.
- Some unsupervised learning tasks:
 - Outlier detection: Is this a ‘normal’ x_i ?
 - Similarity search: Which examples look like this x_i ?
 - Association rules: Which x^j occur together?
 - Latent-factors: What ‘parts’ are the x_i made from?
 - Data visualization: What does the high-dimensional X look like?
 - Ranking: Which are the most important x_i ?
 - Clustering: **What types of x_i are there?**

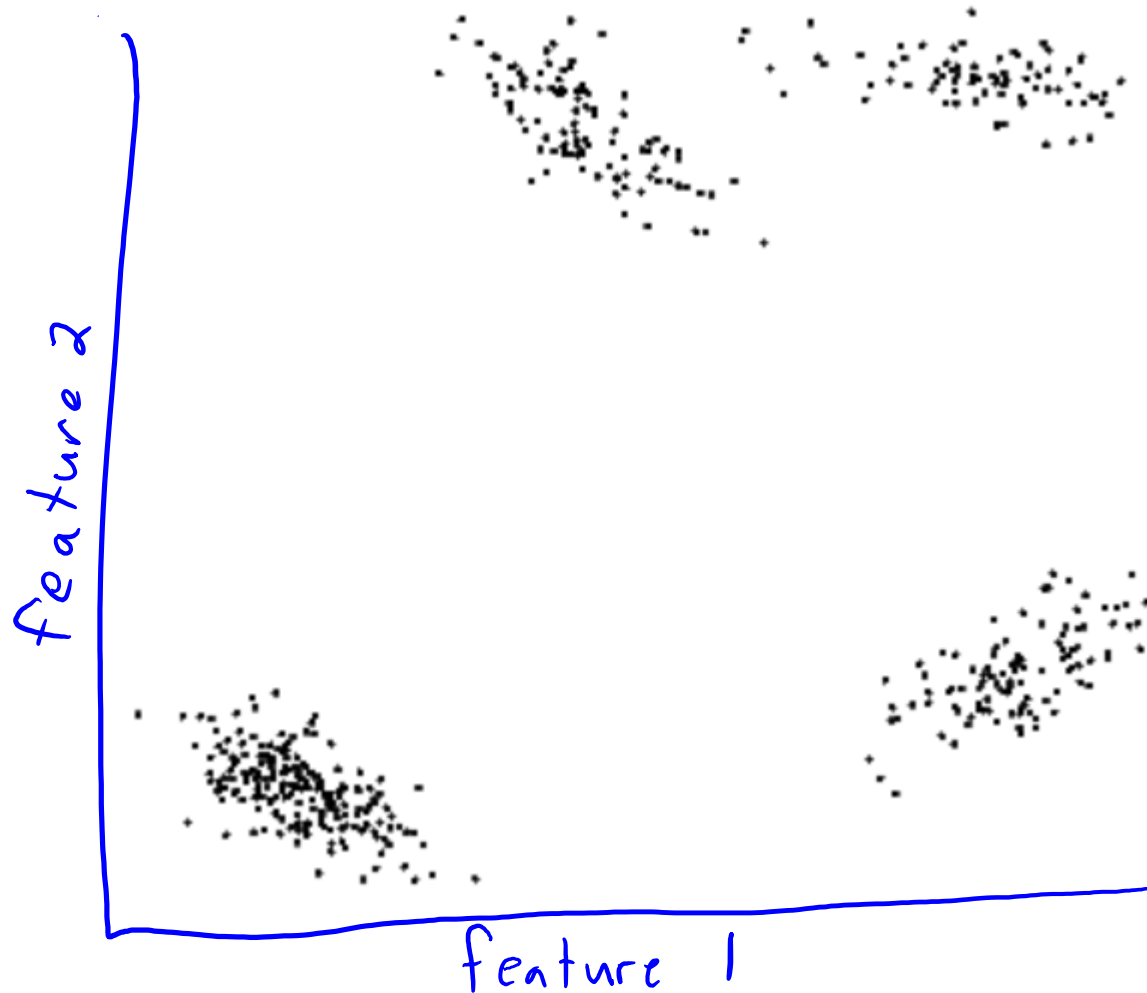
Clustering

- Clustering:
 - Input: set of objects described by features x_i .
 - Output: an assignment of objects to 'groups'.
- Unlike classification, we are not given the 'groups'.
 - Algorithm must discover groups.
- Example of groups we might discover in e-mail spam:
 - 'Lucky winner' group.
 - 'Weight loss' group.
 - 'Nigerian prince' group.
 - 'Russian bride' group.

Clustering Example

Input: data matrix 'X'.

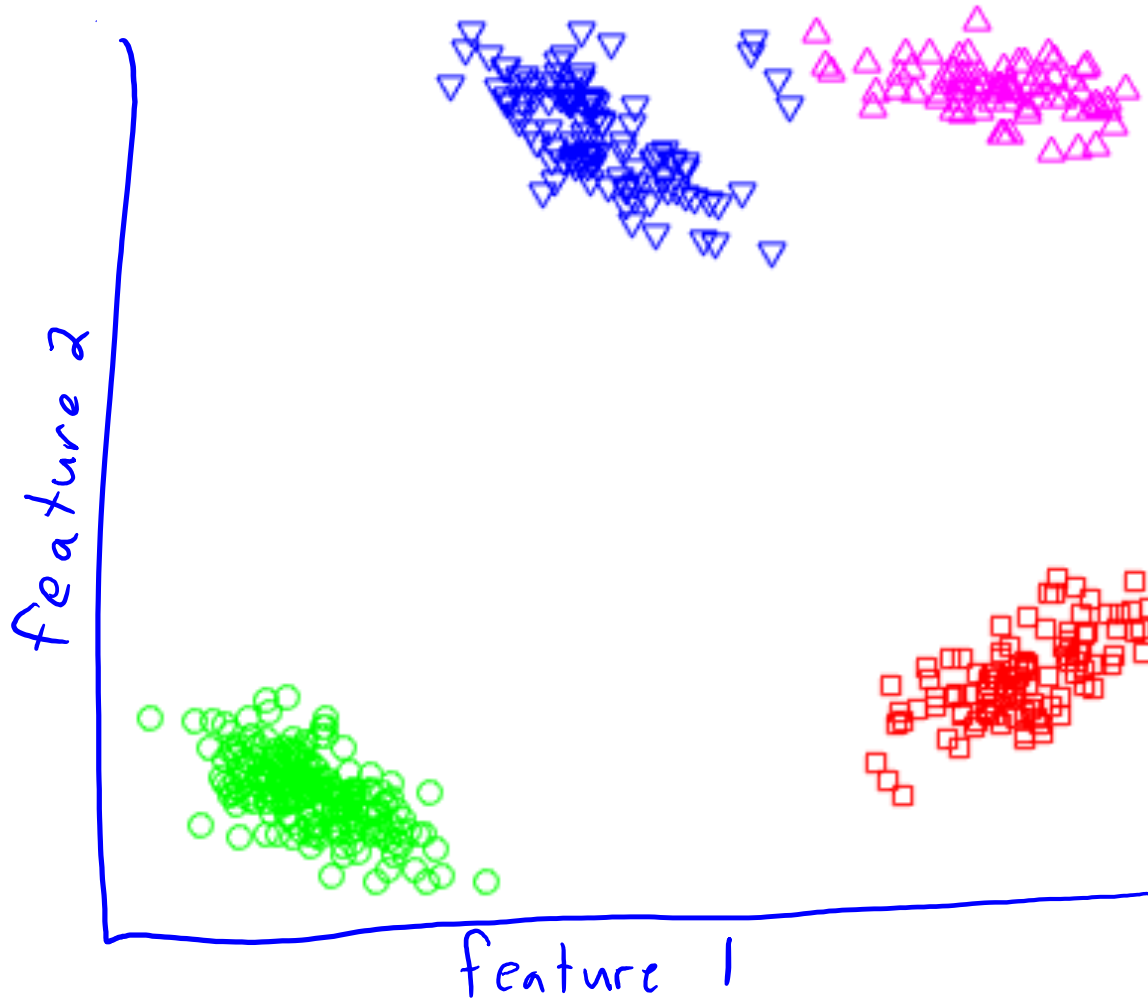
$$X = \begin{bmatrix} -9.0 & -7.3 \\ -10.9 & -9.0 \\ 13.7 & 19.3 \\ 13.8 & 20.4 \\ 12.8 & 20.6 \\ \vdots & \vdots \end{bmatrix}$$



Clustering Example

Input: data matrix 'X'.

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



Output: clusters \hat{y} .

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

Data Clustering

- General goal of clustering algorithms:
 - Objects in the same group should be ‘similar’.
 - Objects in different groups should be ‘different’.
- But the ‘best’ clustering is hard to define:
 - We don’t have a test error.
 - Generally, there is no ‘best’ method in unsupervised learning.
 - So there are lots of methods: we’ll focus on important/representative ones.
- Why cluster?
 - You could want to know what the groups are.
 - You could want a ‘prototype’ example for each group.
 - You could want to find the group for a new example x_i .
 - You could want to find objects related to a new example x_i .

Other Clustering Applications

- NASA: what types of stars are there?
- Biology: are there sub-species?
- Documents: what kinds of articles are there on Wikipedia?
- I can sell 3 flavours of my pasta sauce – what are the 3 types that customers would like?

K-Means

- Most popular clustering method is **k-means**.
- Input:
 - The **number of clusters 'k'** (hyperparameter).
 - Initial guess of the center (the “mean”) of each cluster.
- Algorithm:
 1. **Assign each x_i** to its closest mean.
 2. **Update the means** based on the cluster assignments.
 3. Repeat steps 1-2 until convergence.

Jupyter notebook demo (part 1)

K-Means Issues

- **Guaranteed to converge** when using Euclidean distance.
- Given a new test object: **assign it to the nearest mean** to cluster it.
- Assumes you **know number of clusters 'k'**.
 - Lots of heuristics to pick 'k', none satisfying:
 - https://en.wikipedia.org/wiki/Determining_the_number_of_clusters_in_a_data_set
- Each object is assigned to **one (and only one) cluster**:
 - No possibility for overlapping clusters or leaving objects unassigned.
- It may converge to **sub-optimal solution**...
 - Classic approach to dealing with sensitivity to initialization: **random restarts**.
 - Try several different random starting points, choose the “best”.
 - See bonus slides for a more clever approach called k-means++.
 - This is what scikit-learn's KMeans does by default.

What is K-Means Doing?

- We can interpret K-Means steps as trying to minimize an objective:
 - Total sum of squared distances from object x_i to their centers $w_{\hat{y}_i}$:

$$f(w_1, w_2, \dots, w_K, \hat{y}_1, \hat{y}_2, \dots, \hat{y}_n) = \sum_{i=1}^n \|w_{\hat{y}_i} - x_i\|^2$$

- The k-means steps:
 - Minimize 'f' in terms of the \hat{y}_i (update cluster assignments).
 - Minimize 'f' in terms of the w_c (update means).

- Convergence follows because:
 - Each step does not increase the objective.
 - There are a finite number of assignments to k clusters.

→ Cluster of example 'i',
 $\hat{y}_i \in \{1, 2, \dots, K\}$

$$W = \begin{bmatrix} \text{---} w_1 \text{---} \\ \text{---} w_2 \text{---} \\ \vdots \\ \text{---} w_k \text{---} \end{bmatrix} \left. \vphantom{\begin{bmatrix} \text{---} w_1 \text{---} \\ \text{---} w_2 \text{---} \\ \vdots \\ \text{---} w_k \text{---} \end{bmatrix}} \right\}^k$$

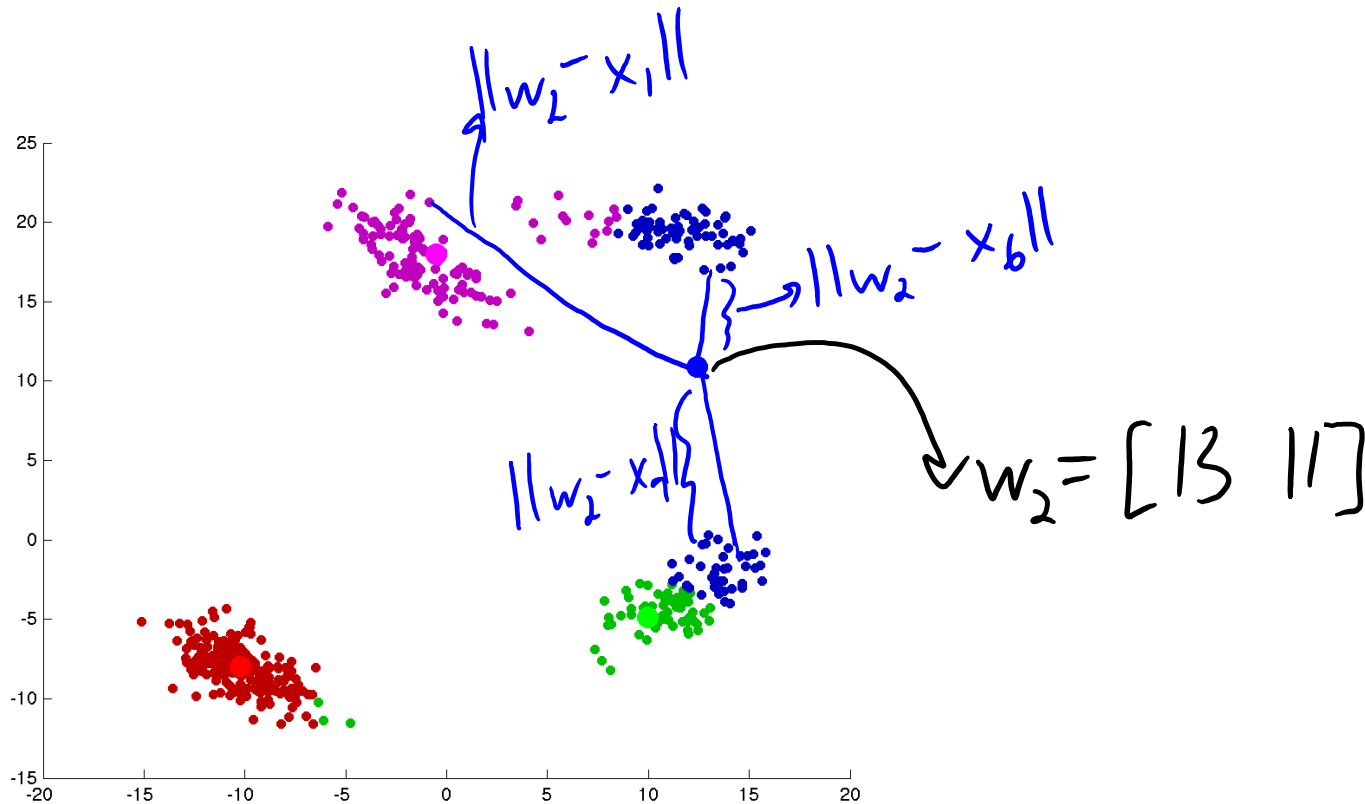
$\underbrace{\hspace{10em}}_d$

Cost of K-means

- Bottleneck is calculating distance from each x_i to each mean w_c :

$$\|w_c - x_i\|^2 = \sum_{j=1}^d (w_{cj} - x_{ij})^2$$

d-dimensional vector



Cost of K-means

- Bottleneck is **calculating distance** from each x_i to each **mean w_c** :

$$\|w_c - x_i\|^2 = \sum_{j=1}^d (w_{cj} - x_{ij})^2$$

d-dimensional vector

- Each time we do this costs $O(d)$.
- We need to compute distance from 'n' objects to 'k' clusters.
- **Total cost of assigning objects to clusters is $O(ndk)$.**
 - Fast if k is not too large.
- Updating means is cheaper: $O(nd)$.

– For each cluster 'c', compute $w_c = \frac{1}{n_c} \sum_{i \in C} x_i$

Object in cluster.

Loop over objects in cluster.

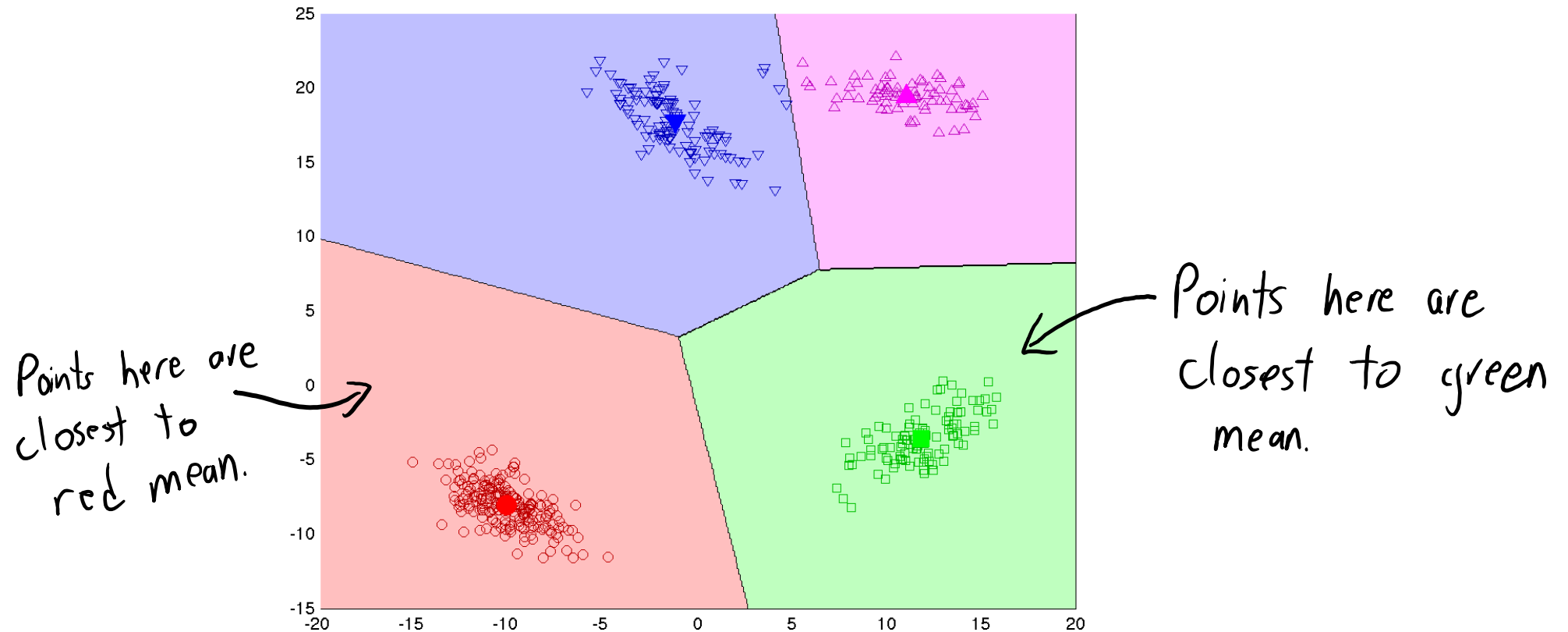
Number of objects in cluster 'c'

Vector Quantization

- K-means originally comes from signal processing.
- Designed for **vector quantization**:
 - Replace objects with the mean of their cluster (“prototype”).
- Example:
 - Facebook places: 1 location summarizes many.
 - What sizes of clothing should I make?
- This is on assignment 2, see also bonus slides.

Shape of K-Means Clusters

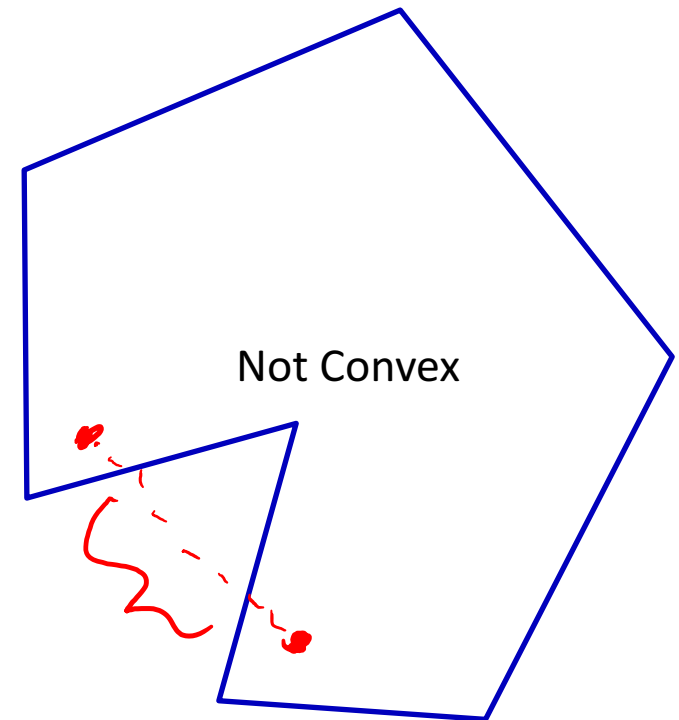
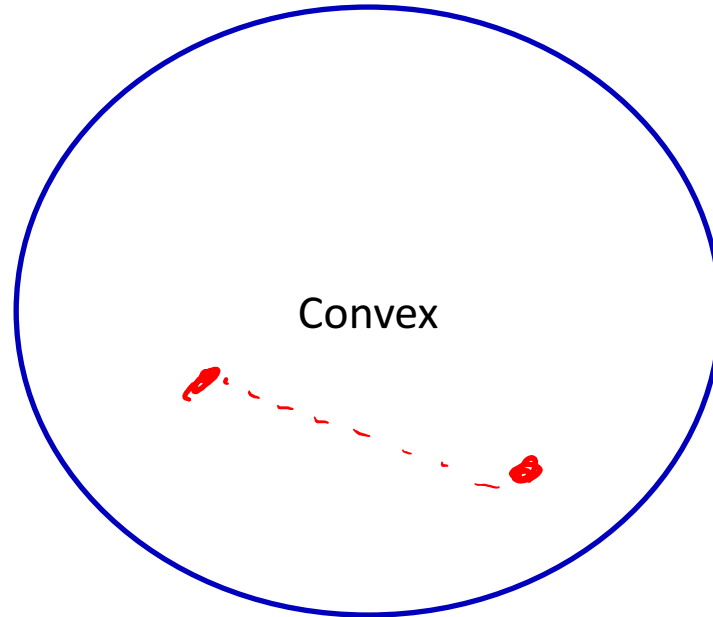
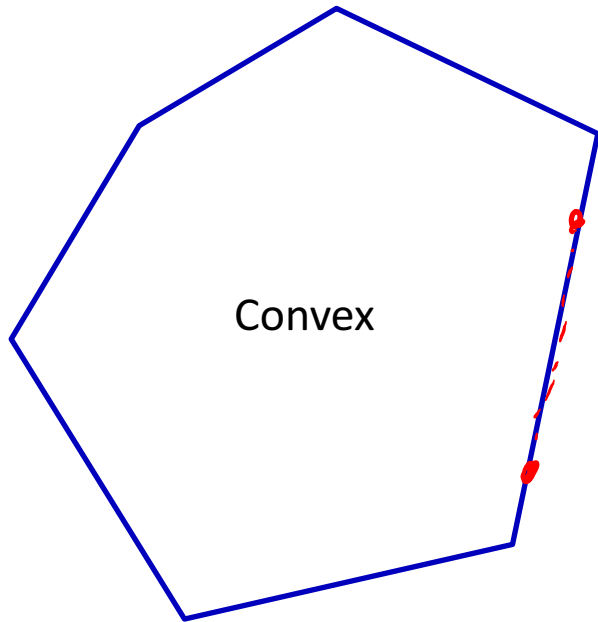
- K-means partitions the space based on the “closest mean”:



- Observe that the clusters are convex regions.

Convex Sets

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

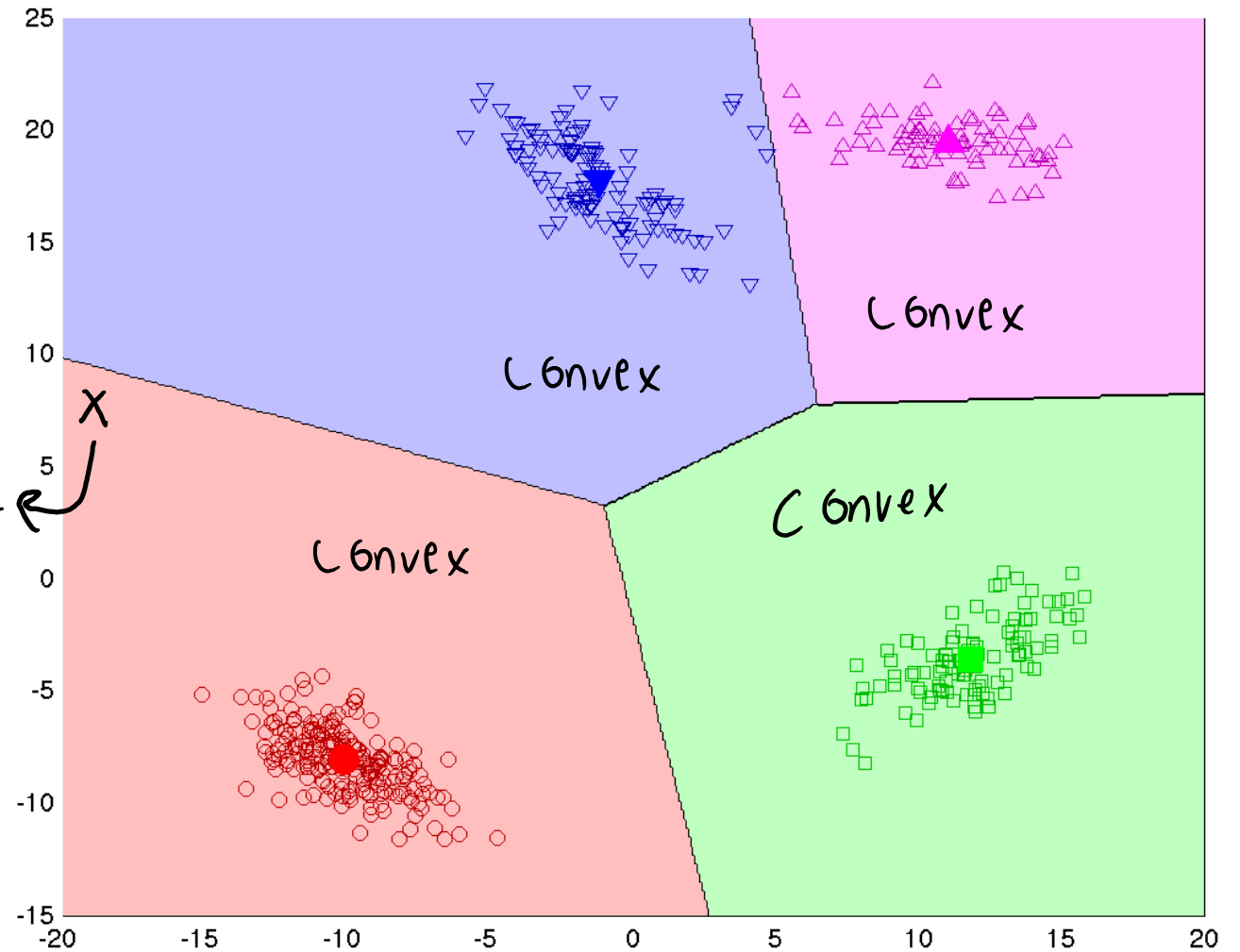


Shape of K-Means Clusters

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?



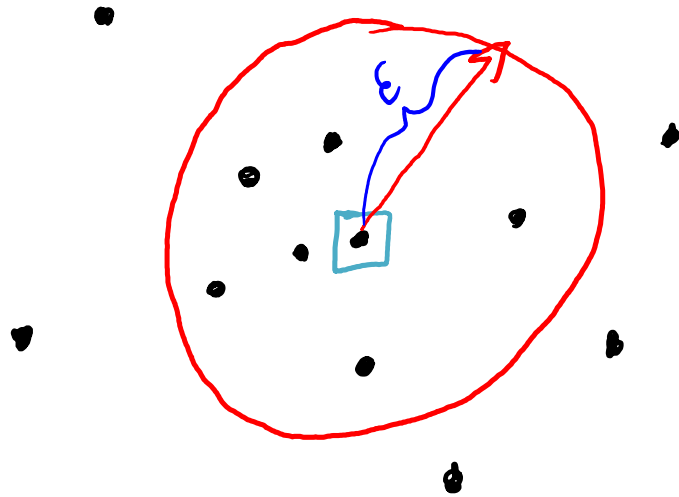
Motivation for Density-Based Clustering

- Density-based clustering:
 - Clusters are defined by “dense” regions.
 - Objects in non-dense regions don’t get clustered.
 - Not trying to “partition” the space.
- Clusters can be non-convex
- It’s a non-parametric clustering method:
 - No fixed number of clusters ‘k’.
 - Clusters can become more complicated with more data.

Jupyter notebook demo (part 2)

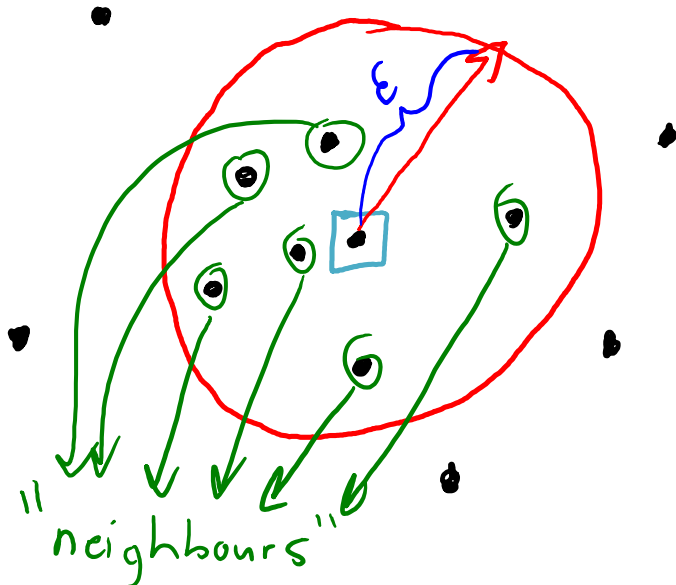
Density-Based Clustering

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



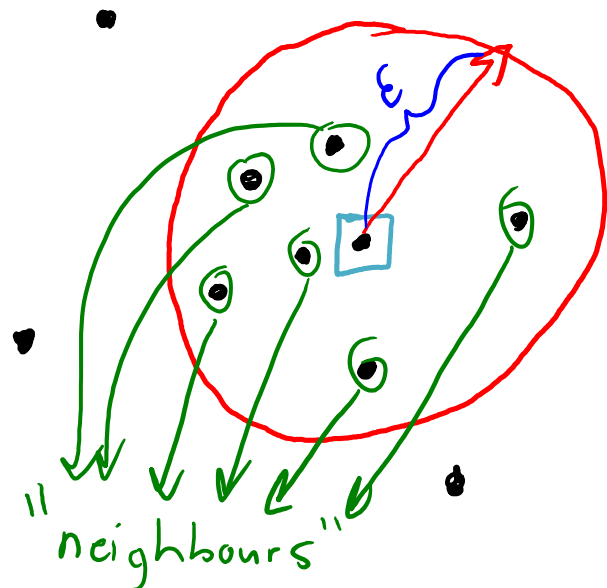
Density-Based Clustering

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



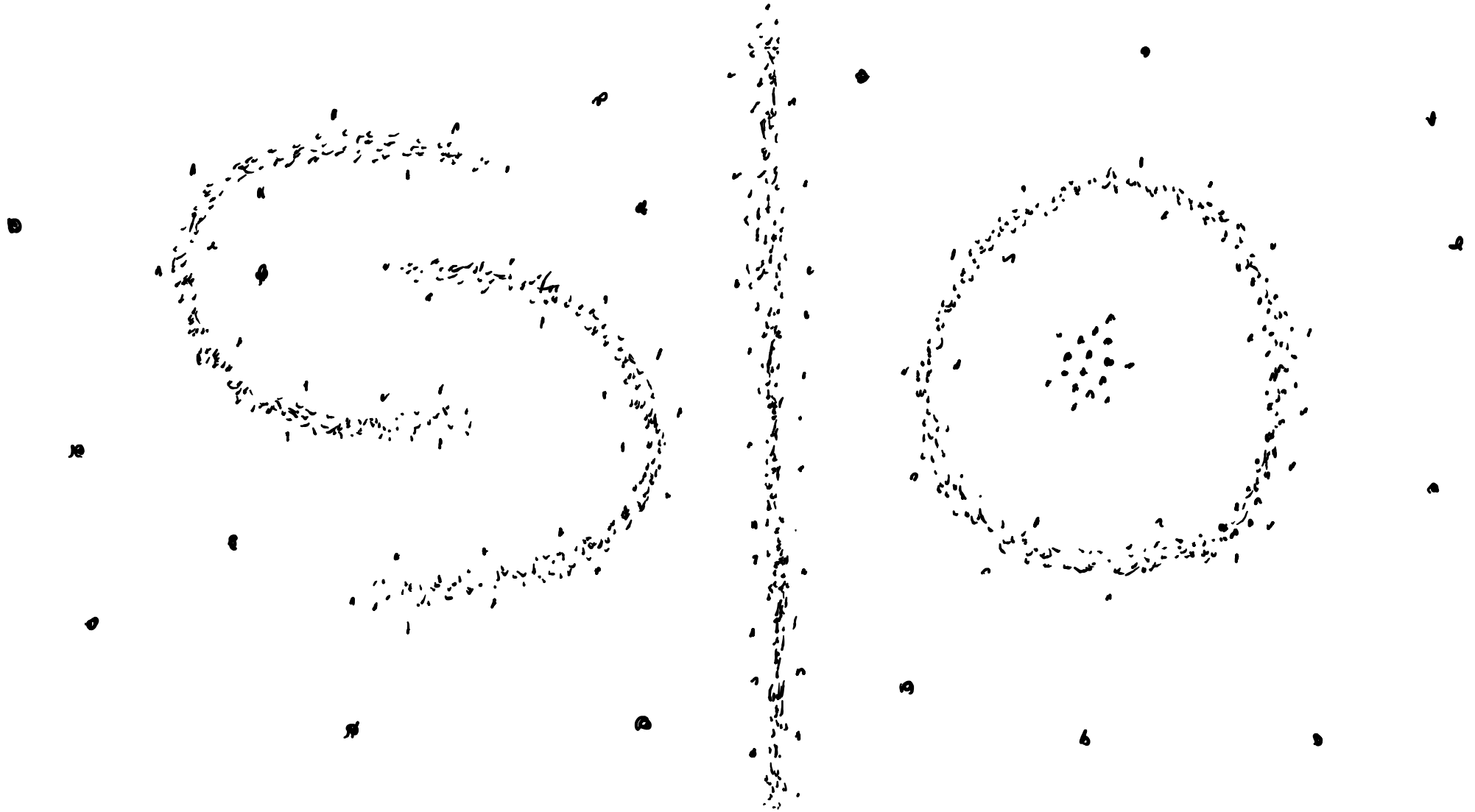
Density-Based Clustering

- Density-based clustering algorithm (DBSCAN) has two hyperparameters:
 - Epsilon (ϵ): 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.

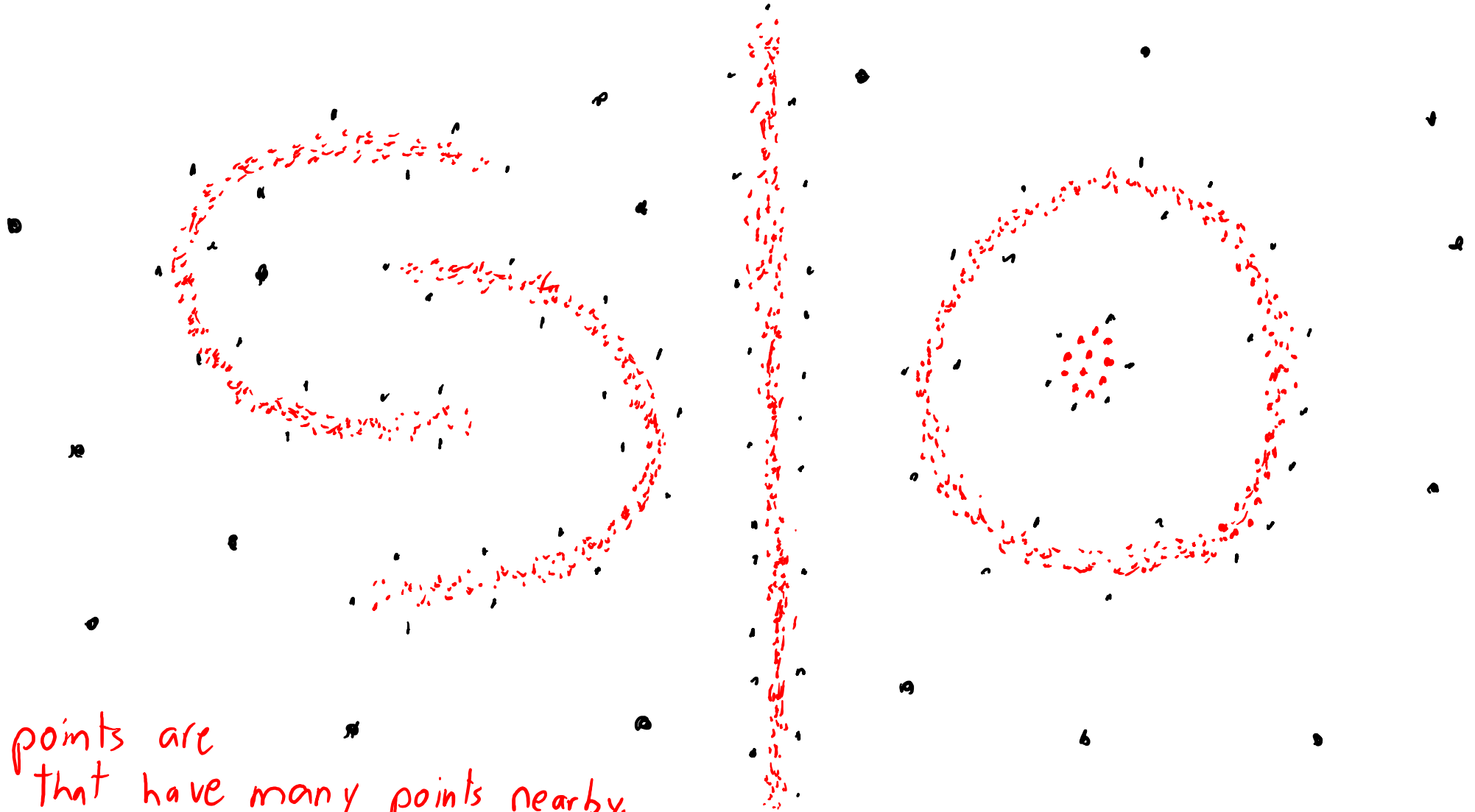


E.g., if $\text{minNeighbours} = 3$
then this is a “core”
point since 6 points are
“neighbours”

Density-Based Clustering

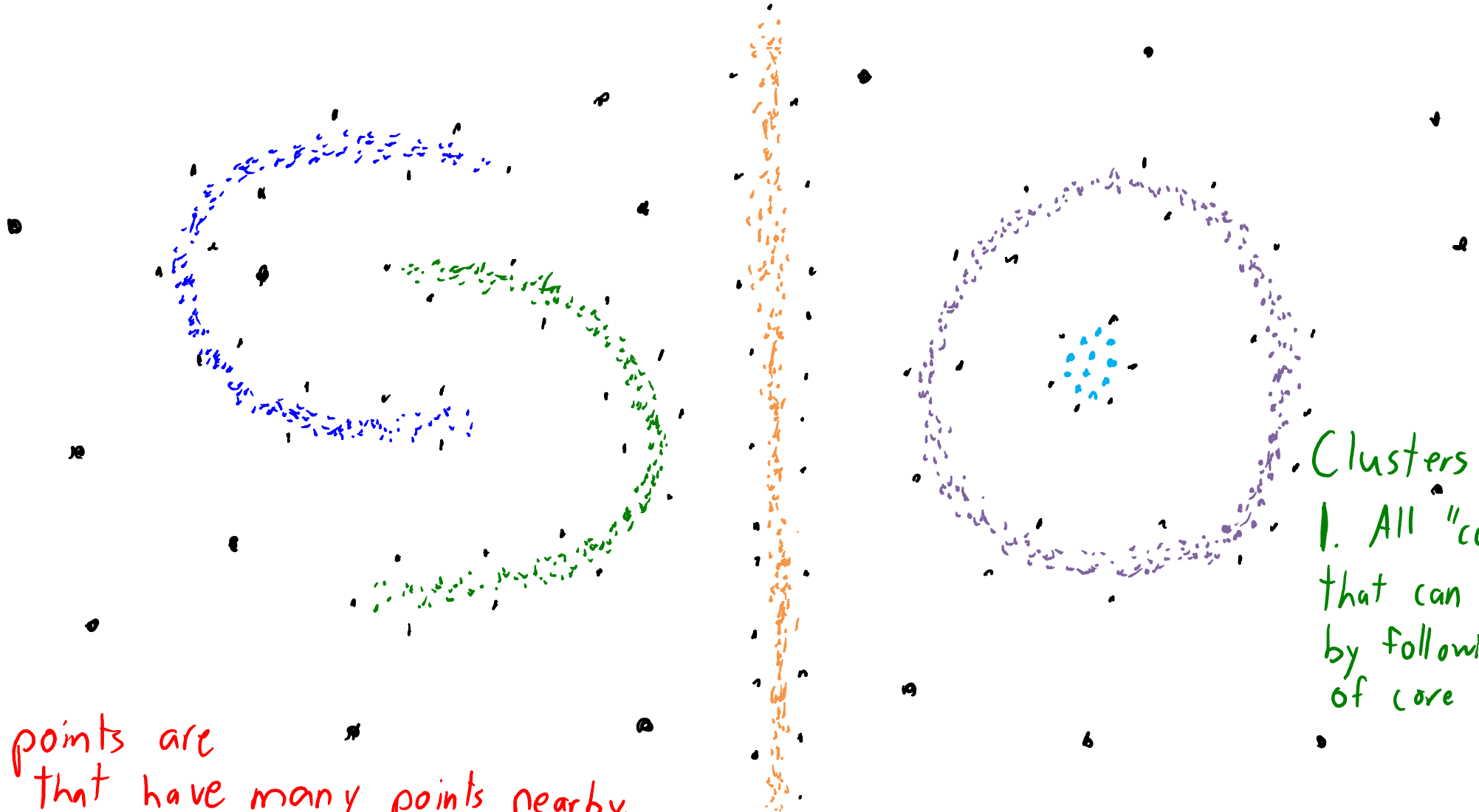


Density-Based Clustering



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

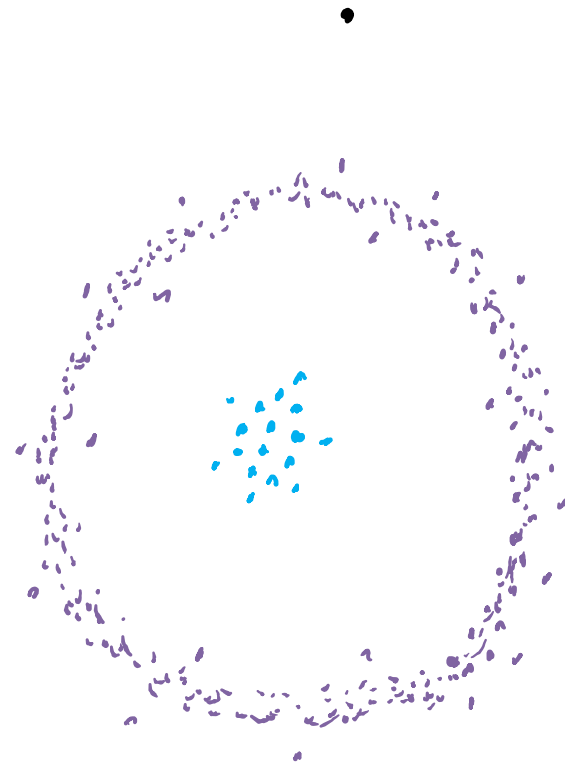
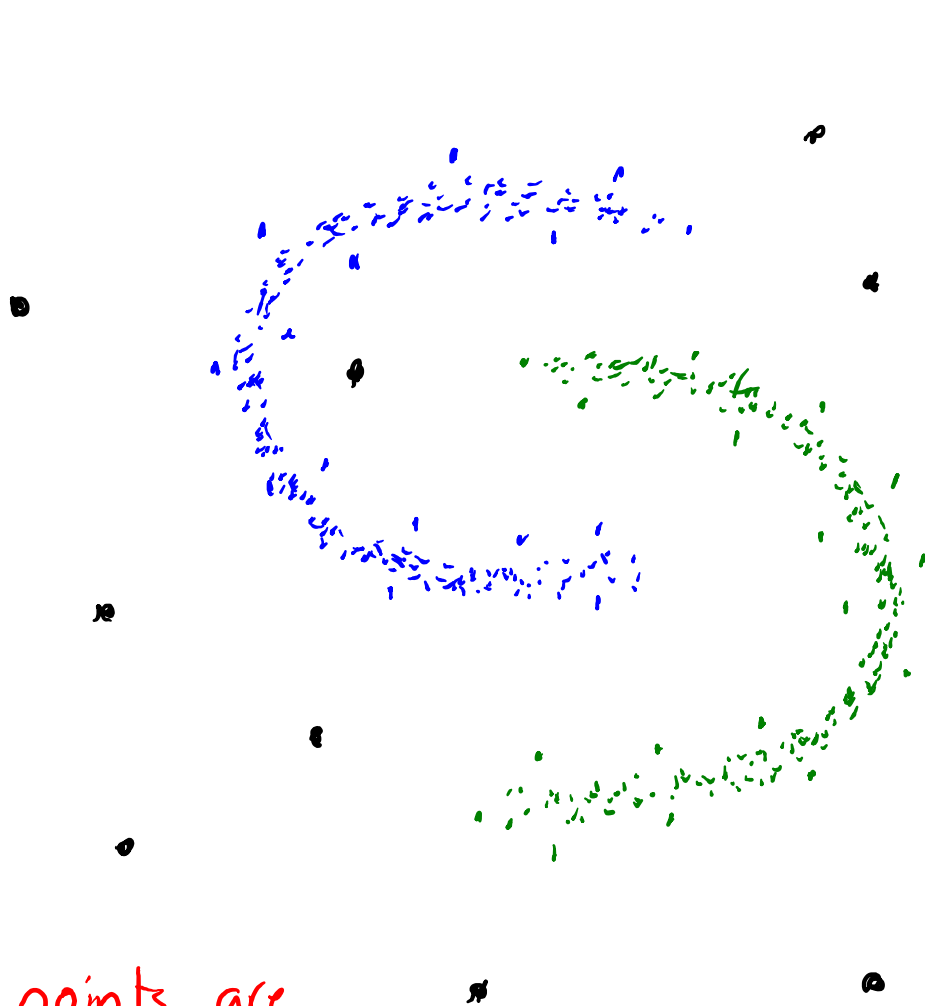
Density-Based Clustering



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:

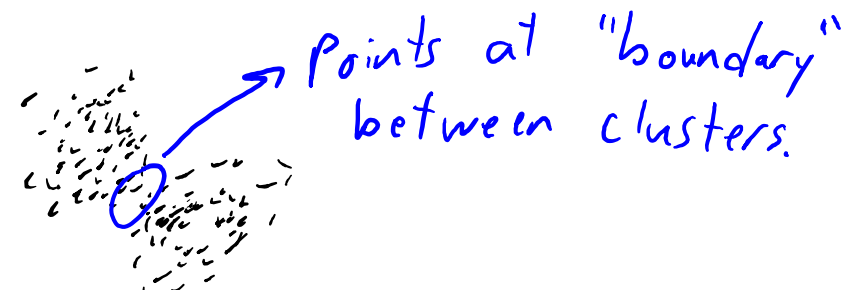
1. All "core" points that can be reached by following a sequence of core points.
2. All non-core neighbours of core points (boundary points)

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

Density-Based Clustering Pseudo-Code

- For each example x_i :
 - If x_i is already assigned to a cluster, do nothing.
 - Test whether x_i is a ‘core’ point ($\geq \text{minNeighbours}$ examples within ‘ ϵ ’).
 - If x_i is not core point, do nothing (this could be an outlier).
 - If x_i is a core point, “expand” cluster.
- “Expand” cluster function:
 - Assign all x_j within distance ‘ ϵ ’ of core point x_i to cluster.
 - For each newly-assigned neighbour x_j that is a core point, “expand” cluster.

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:
 - Otherwise, not sensitive to initialization (except for boundary points).
- Sensitive to the choice of ϵ and minNeighbours.
 - Need to compute distances to training points.
- If you get a new example, finding cluster is expensive.
 - Need to compute distances to training points.
- In high-dimensions, need a lot of points to ‘fill’ the space.

Ensemble Clustering

? question ☆

stop following 23 views

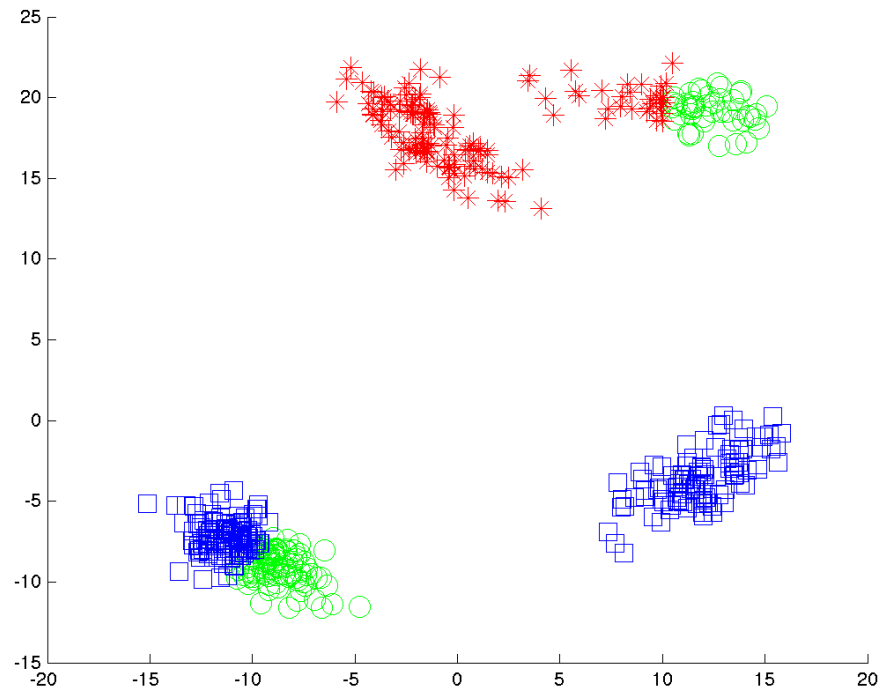
Multiple random runs of K means

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

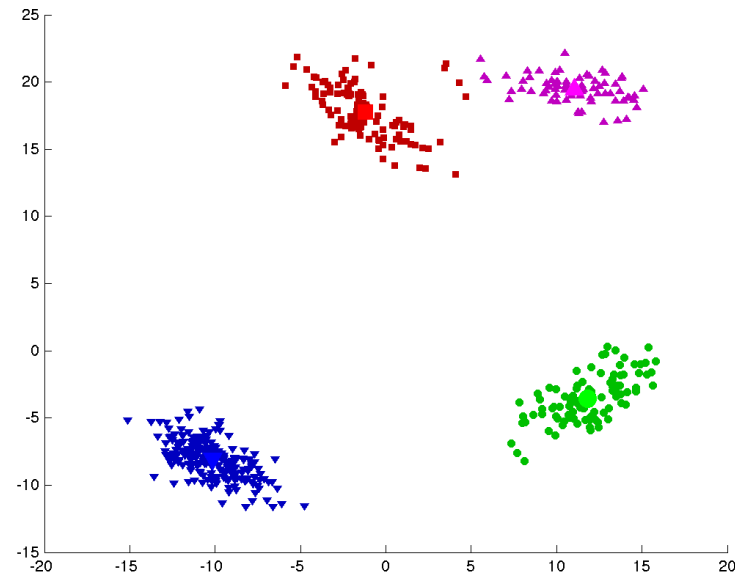
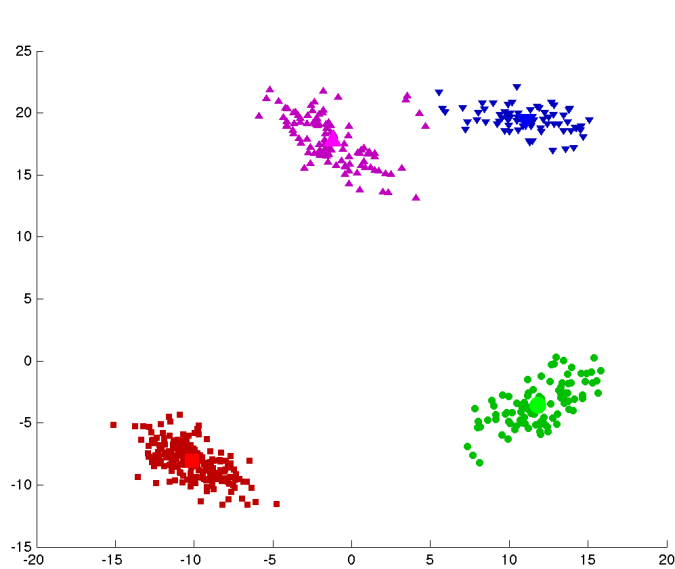
- E.g., run k-means 20 times and then cluster using the mode of each \hat{y}_i .
- Normally, averaging across models doing different things is good.



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

Label Switching Problem

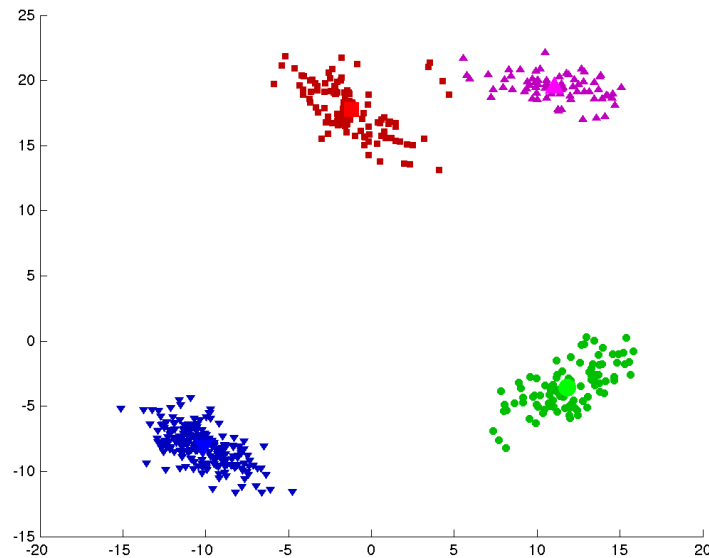
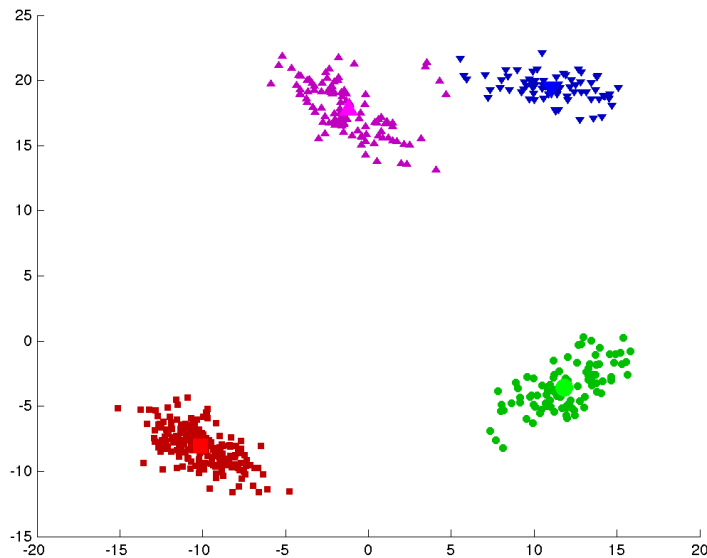
- This doesn't work because of “label switching” problem:
 - The cluster labels \hat{y}_i are meaningless.
 - We could get same clustering with permuted labels:



- 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.

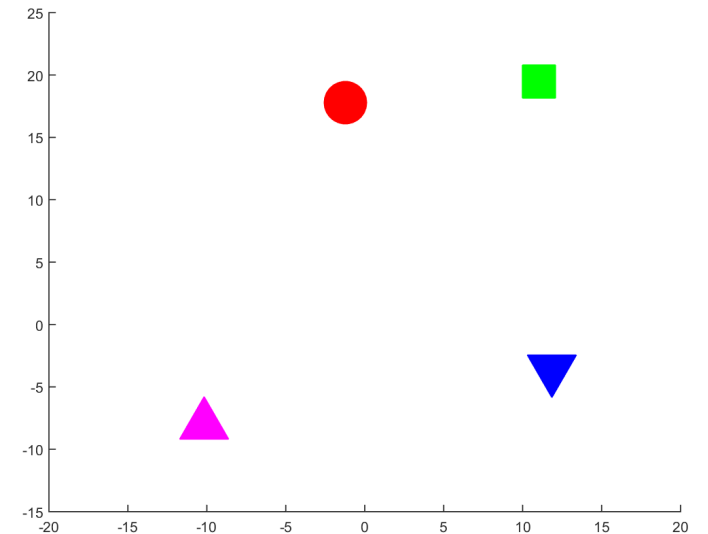
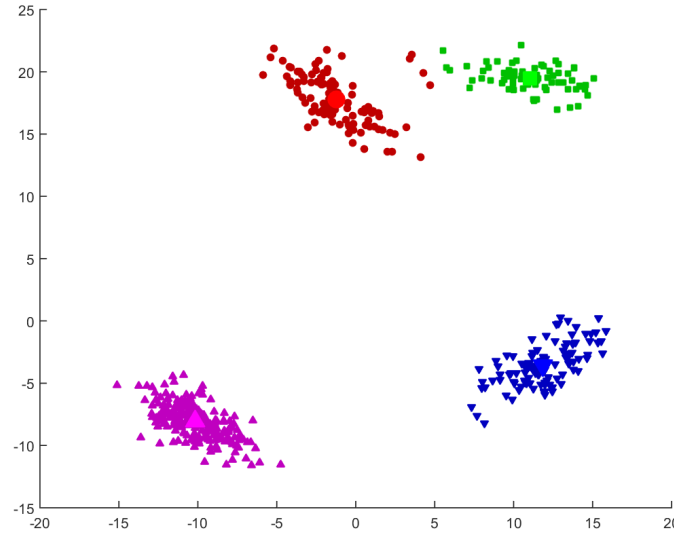
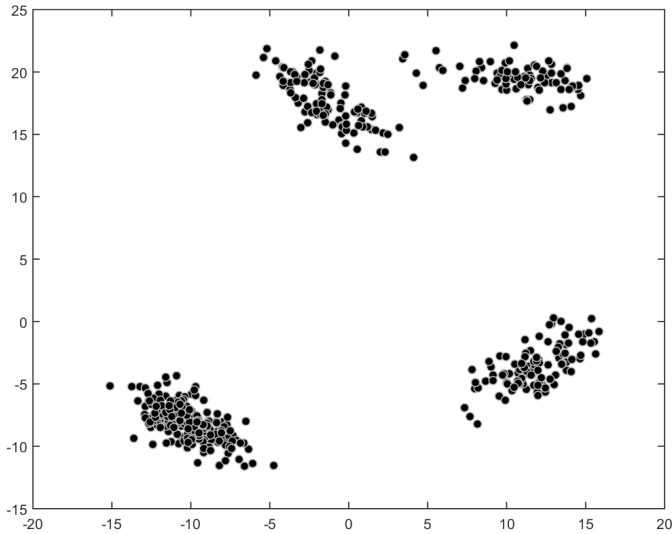


- Bonus slides give an example method ("UBClustering").

Summary

- **Unsupervised learning**: fitting data without explicit labels.
- **Clustering**: finding ‘groups’ of related objects.
- **K-means**: simple iterative clustering strategy.
 - Fast but sensitive to initialization.
 - Partitions space into convex sets.
- **Density-based clustering**:
 - “Expand” and “merge” dense regions of points to find clusters.
 - Not 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**.

Vector Quantization



$n \times d$

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

Run k-means

$k \times d$

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

$n \times 1$

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

Approximate objects with means.

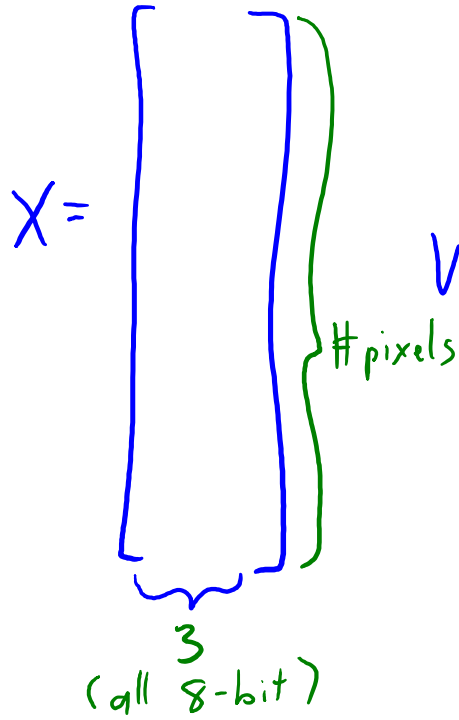
$$\hat{X} = \begin{bmatrix} -10.2 & -8.0 \\ -10.2 & -8.0 \\ 11.0 & 19.5 \\ 11.0 & 19.5 \\ -1.2 & 17.8 \\ \vdots & \vdots \end{bmatrix}$$

Vector Quantization: Image Colors

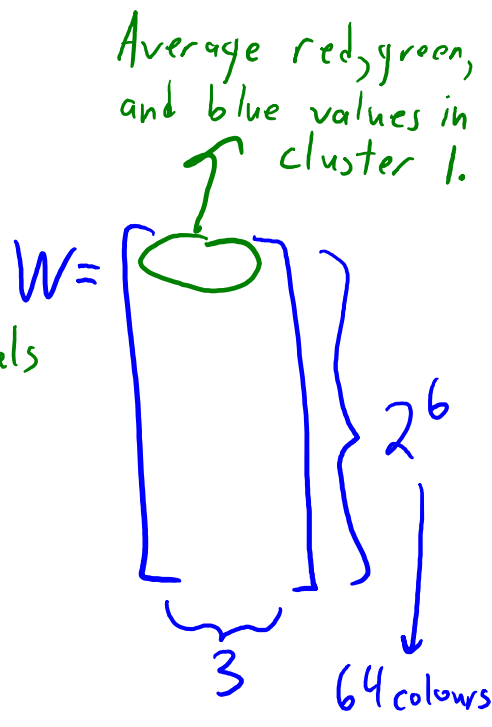
- Usual RGB representation of a pixel's color: three 8-bit numbers.
 - For example, [241 13 50] = ■.
 - Can apply k-means to find set of prototype colours.



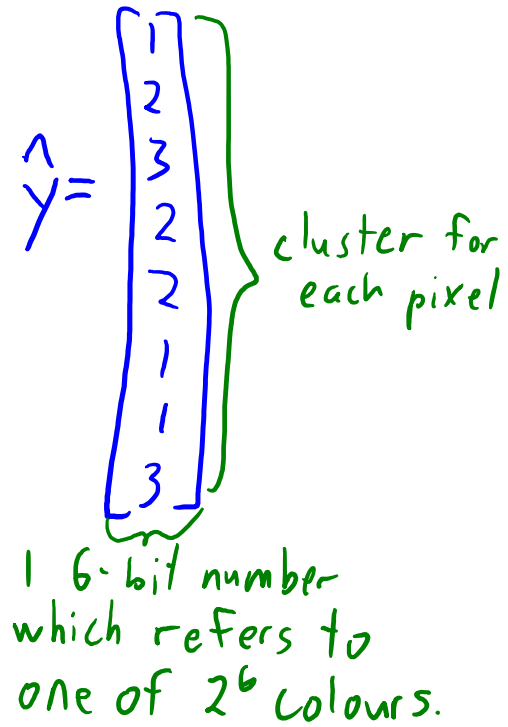
Original:
(24-bits/pixel)



Run k-means with
 2^6 clusters:



K-means predictions:
(6-bits/pixel)

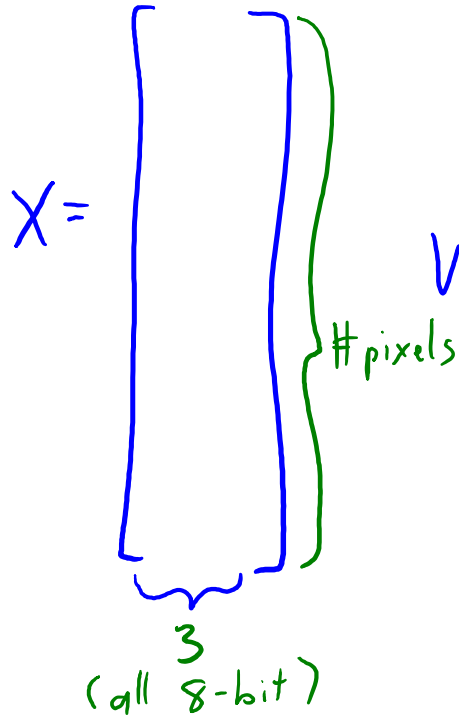


Vector Quantization: Image Colors

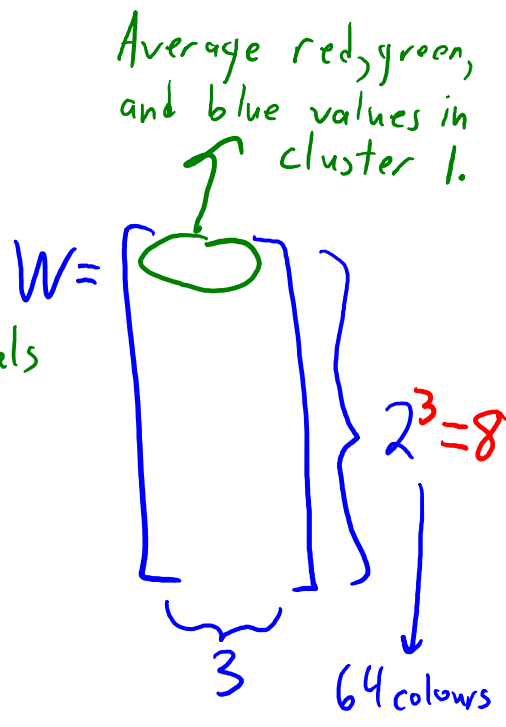
- Usual RGB representation of a pixel's color: three 8-bit numbers.
 - For example, [241 13 50] = ■.
 - Can apply k-means to find set of prototype colours.



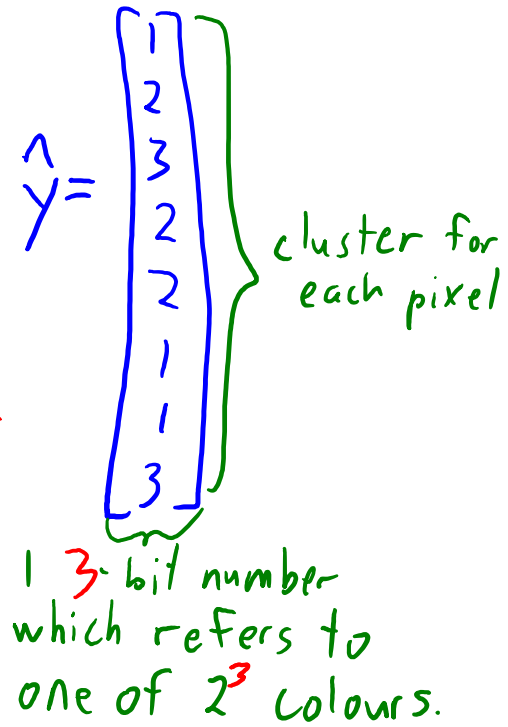
Original:
(24-bits/pixel)



Run k-means with
 2^6 clusters:



K-means predictions:
(3-bits/pixel)

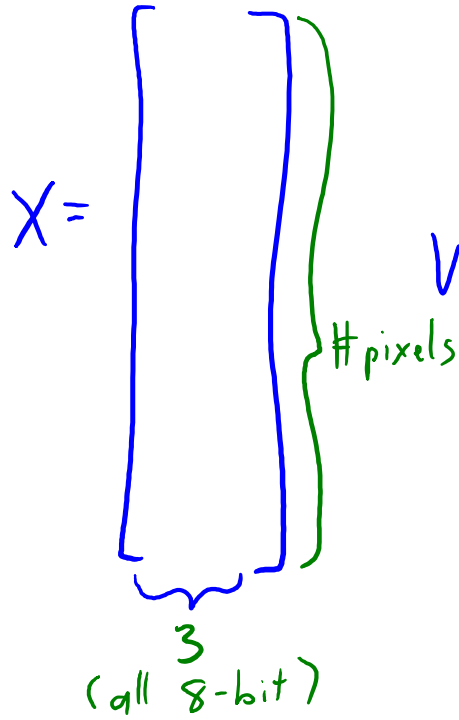


Vector Quantization: Image Colors

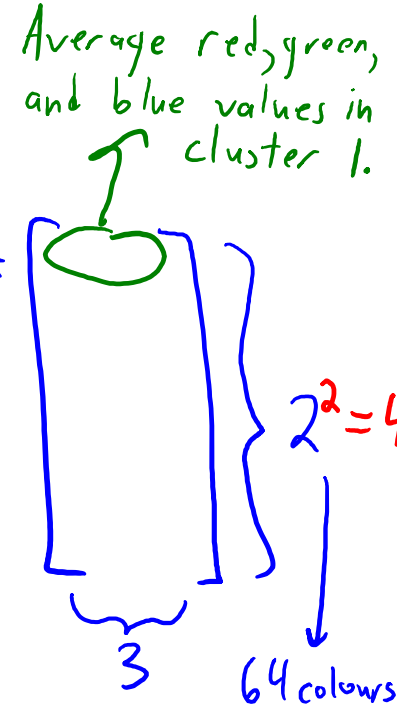
- Usual RGB representation of a pixel's color: three 8-bit numbers.
 - For example, [241 13 50] = ■.
 - Can apply k-means to find set of prototype colours.



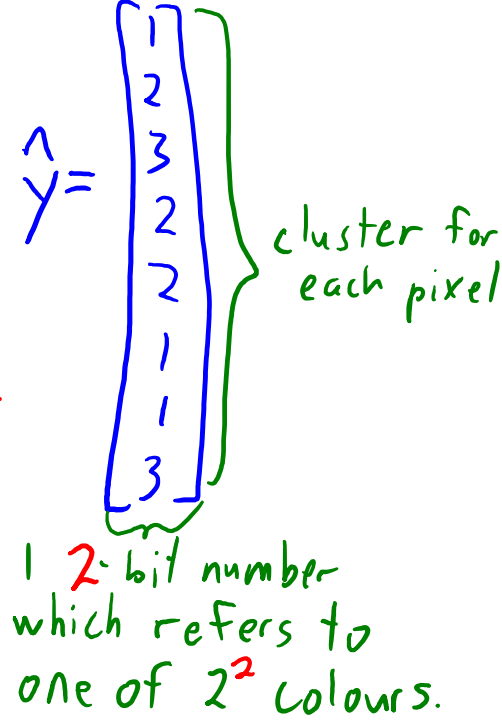
Original:
(24-bits/pixel)



Run k-means with
 2^6 clusters:



K-means predictions:
(2-bits/pixel)

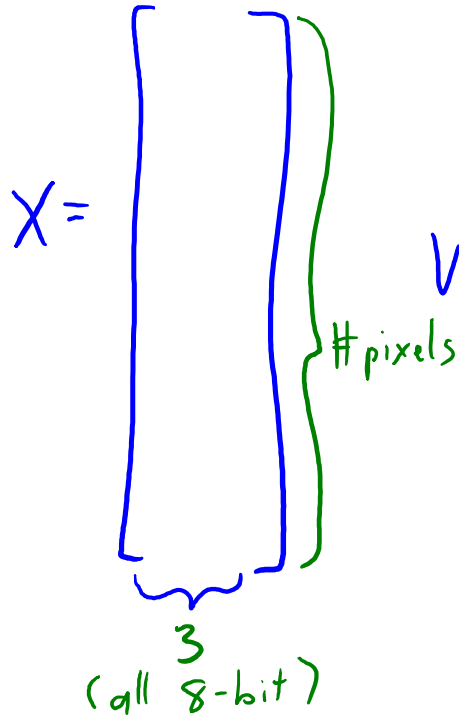


Vector Quantization: Image Colors

- Usual RGB representation of a pixel's color: three 8-bit numbers.
 - For example, [241 13 50] = ■.
 - Can apply k-means to find set of prototype colours.

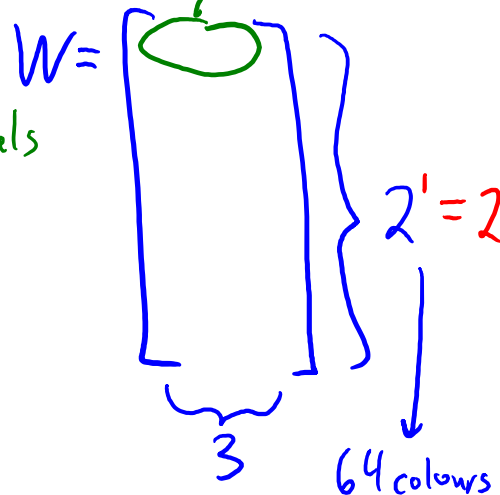


Original:
(24-bits/pixel)

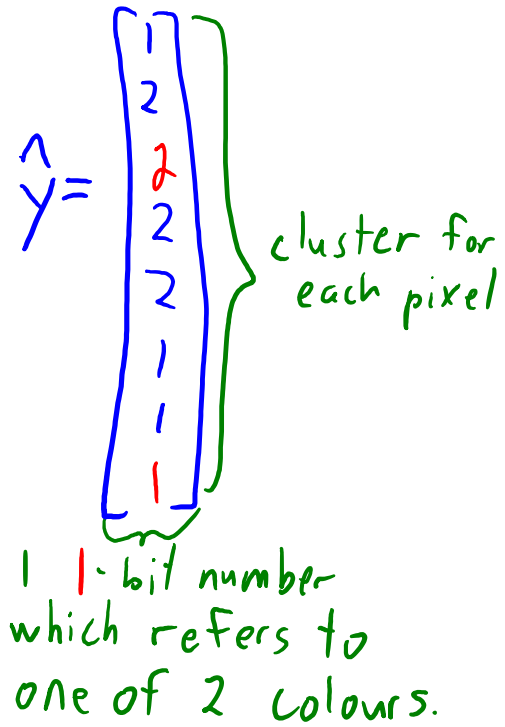


Run k-means with
 2^6 clusters:

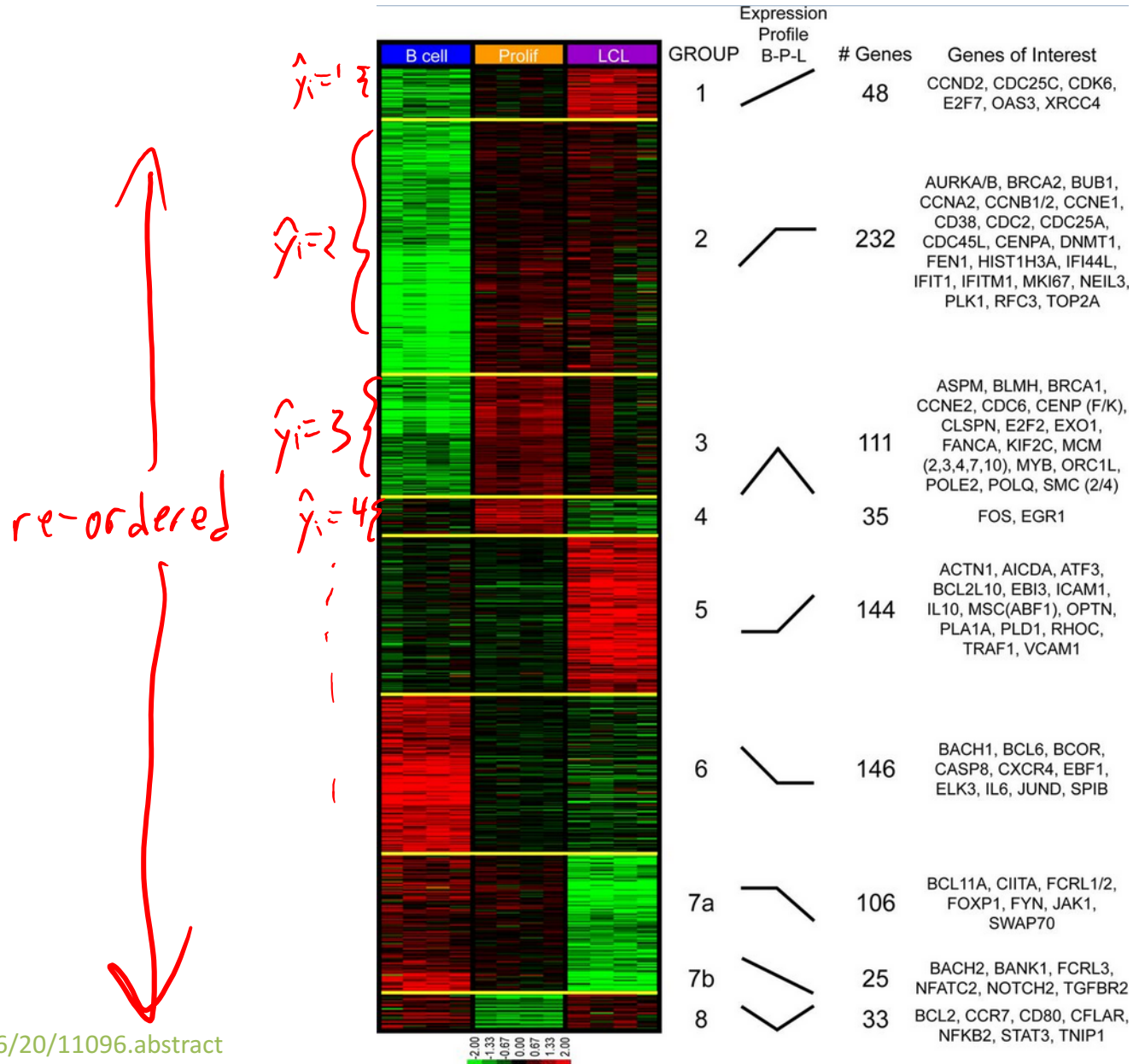
Average red, green,
and blue values in
cluster 1.



K-means predictions:
(1-bit/pixel)



Clustering of Epstein-Barr Virus



What is K-Means Doing?

- How are are k-means step decreasing this objective?

$$f(w_1, w_2, \dots, w_k, \hat{y}_1, \hat{y}_2, \dots, \hat{y}_n) = \sum_{i=1}^n \|w_{\hat{y}_i} - x_i\|^2$$

- If we just write as function of a particular \hat{y}_i , we get:

$$f(\hat{y}_i) = \|w_{\hat{y}_i} - x_i\|^2 + (\text{constant})$$

- The “constant” includes all other terms, and doesn’t affect location of min.
- We can minimize in terms of \hat{y}_i by setting it to the ‘c’ with w_c closest to x_i .

What is K-Means Doing?

- How are are k-means step decreasing this objective?

$$f(w_1, w_2, \dots, w_k, \hat{y}_1, \hat{y}_2, \dots, \hat{y}_n) = \sum_{i=1}^n \|w_{\hat{y}_i} - x_i\|^2$$

- If we just write as function of a particular w_{c_j} we get:

$$f(w_{c_j}) = \sum_{i \in C_j} \sum_{j=1}^d (w_{c_j} - x_{ij})^2 + (\text{constant})$$

$\underbrace{\hspace{10em}}_{\text{set of examples with } \hat{y}_i = c}$

- Derivative is given by: $f'(w_{c_j}) = 2 \sum_{i \in C_j} (w_{c_j} - x_{ij})$

- Setting equal to 0 and solving for w_{c_j} gives: $\sum_{i \in C_j} w_{c_j} = \sum_{i \in C_j} x_{ij}$ or $w_{c_j} * n_{c_j} = \sum_{i \in C_j} x_{ij}$
or $w_{c_j} = \frac{1}{n_{c_j}} \sum_{i \in C_j} x_{ij}$

K-Medians Clustering

- With **other distances k-means may not converge**.
 - But we can **make it converge by changing the updates** so that they are minimizing an objective function.
- E.g., we can use the L1-norm objective: $\sum_{i=1}^n \|w_{y_i} - x_i\|_1$
- Minimizing the L1-norm objective gives the ‘k-medians’ algorithm:
 - Assign points to clusters by finding “mean” with smallest L1-norm distance.
 - Update ‘means’ as median value (dimension-wise) of each cluster.
 - This minimizes the L1-norm distance to all the points in the cluster.
- This approach is more robust to outliers.

↑ k-means will put a cluster here.



What is the “L1-norm and median” connection?

- Point that minimizes the sum of squared L2-norms to all points:

$$f(w) = \sum_{i=1}^n \|w - x_i\|^2$$

- Is given by the **mean** (just take derivative and set to 0):

$$w = \frac{1}{n} \sum_{i=1}^n x_i$$

- Point that minimizes the sum of L1-norms to all all points:

$$f(w) = \sum_{i=1}^n \|w - x_i\|_1$$

- Is given by the **median** (derivative of absolute value is +1 if positive and -1 if negative, so any point with half of points larger and half of points smaller is a solution).

K-Medoids Clustering

- A disadvantage of k-means in some applications:
 - The **means might not be valid data** points.
 - May be important for vector quantization.
- E.g., consider bag of words features like $[0,0,1,1,0]$.
 - We have words 3 and 4 in the document.
- A mean from k-means might look like $[0.1\ 0.3\ 0.8\ 0.2\ 0.3]$.
 - What does it mean to have 0.3 of word 2 in a document?
- Alternative to k-means is **k-medoids**:
 - Same algorithm as k-means, except the means must be data points.
 - Update the means by finding example in cluster minimizing squared L2-norm distance to all points in the cluster.

K-Means Initialization

- K-means is fast but **sensitive to initialization**.
- Classic approach to initialization: **random restarts**.
 - Run to convergence using different random initializations.
 - Choose the one that minimizes average squared distance of data to means.
- Newer approach: **k-means++**
 - Random initialization that **prefers means that are far apart**.
 - Yields **provable bounds** on expected approximation ratio.

K-Means++

- Steps of **k-means++**:

1. Select initial mean w_1 as a **random x_i** .

2. **Compute distance d_{ic}** of each object x_i to each mean w_c .

$$d_{ic} = \sqrt{\sum_{j=1}^d (x_{ij} - w_{cj})^2} = \|x_i - w_c\|_2$$

3. For each object 'i' **set d_i to the distance to the closest mean.**

$$d_i = \min_c \{d_{ic}\}$$

4. Choose next mean by **sampling an example 'i' proportional to $(d_i)^2$.**

$$p_i \propto d_i^2 \Rightarrow p_i = \frac{d_i^2}{\sum_{j=1}^n d_j^2}$$

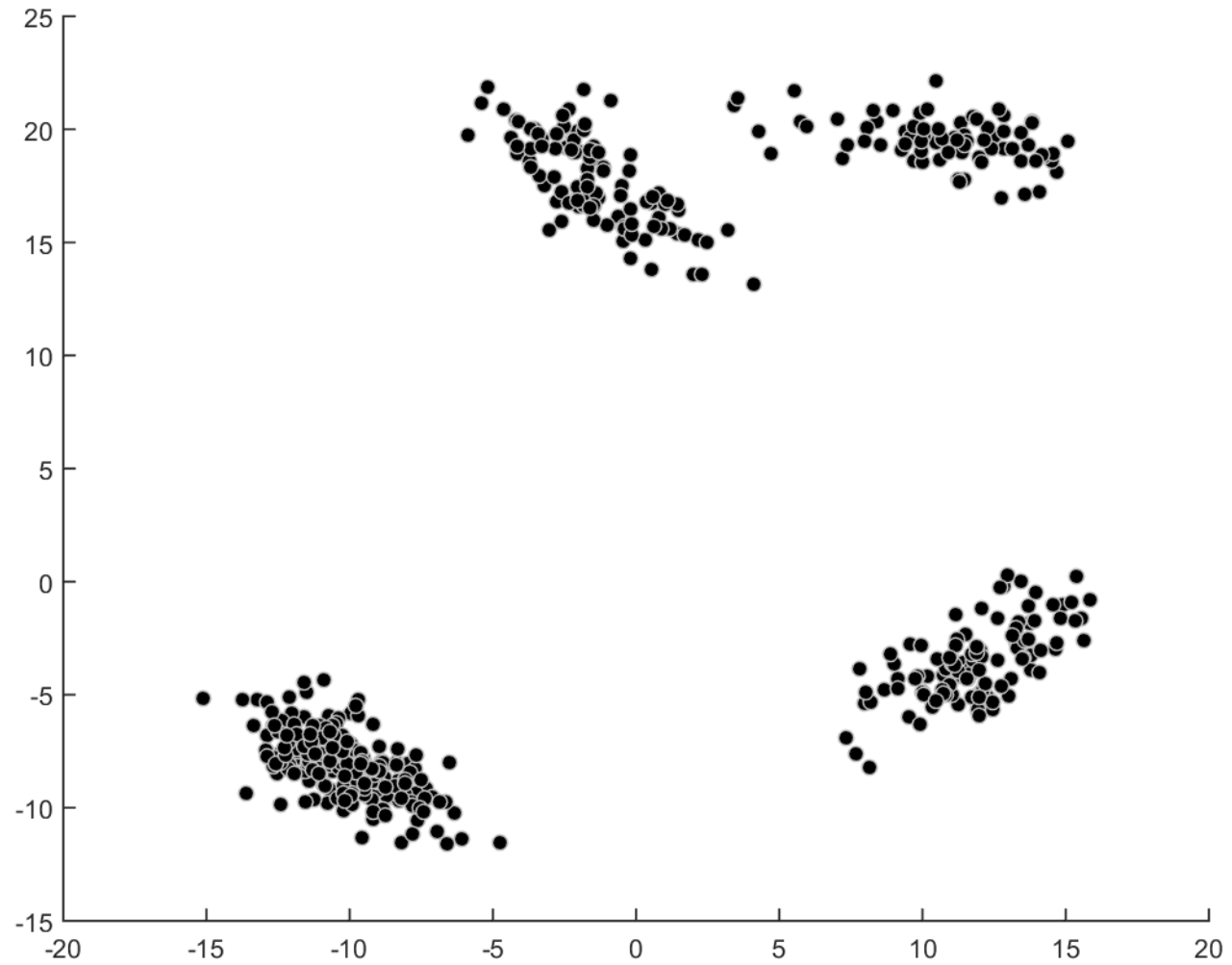
Can be
done in
 $O(n)$.

5. Keep returning to step 2 until we have k-means.

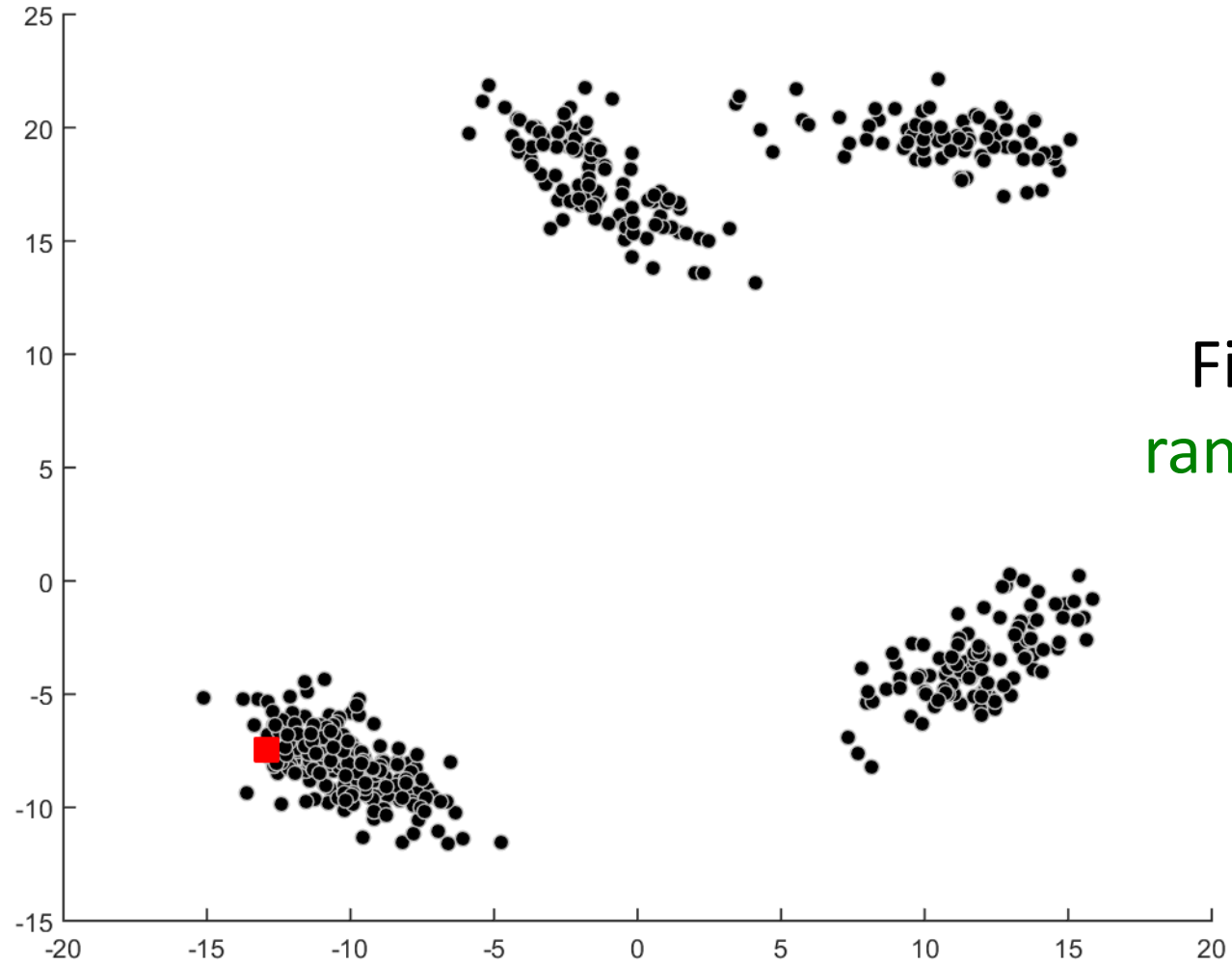
- Expected approximation ratio is $O(\log(k))$.

"probability that we
choose x_i as next mean"

K-Means++

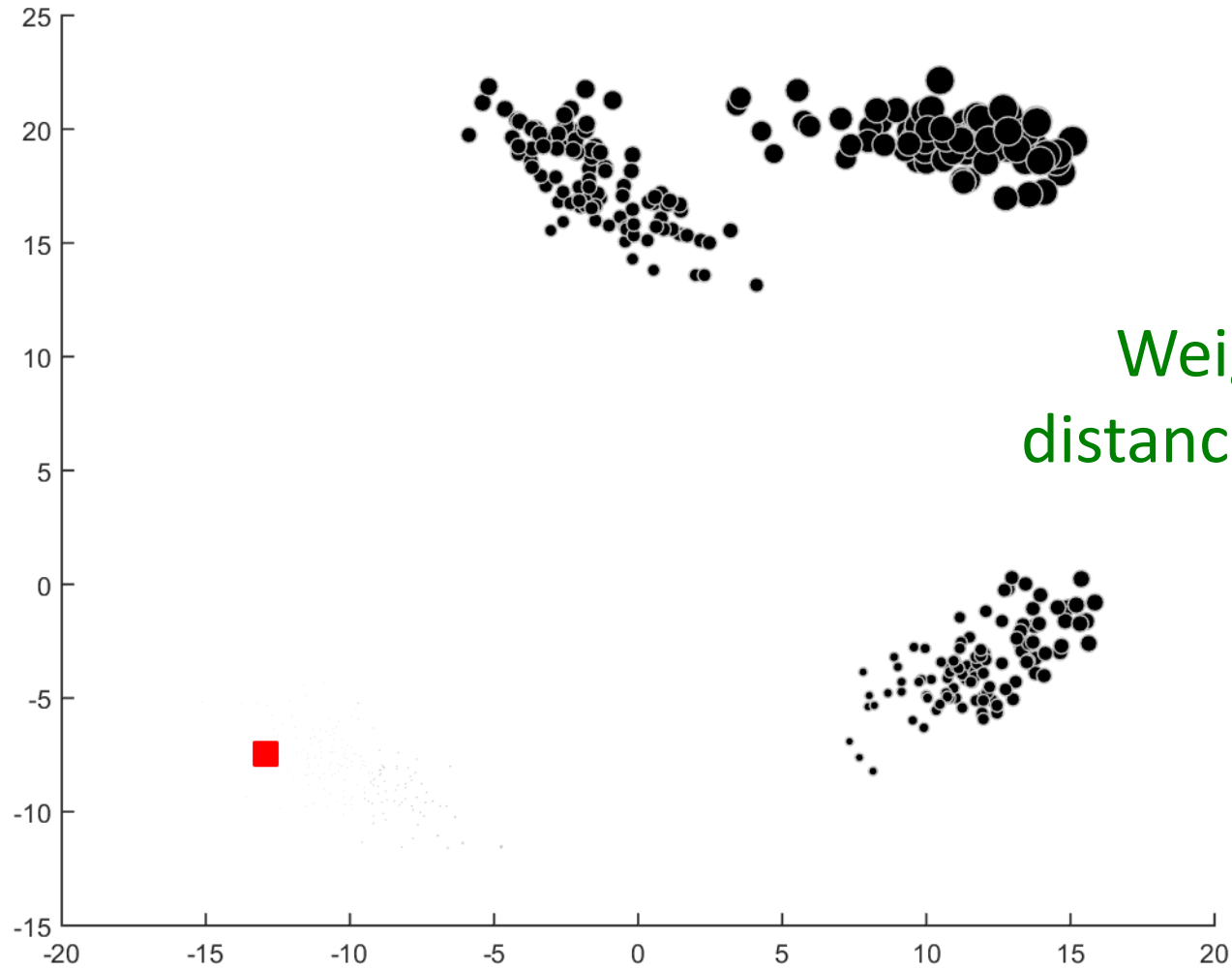


K-Means++



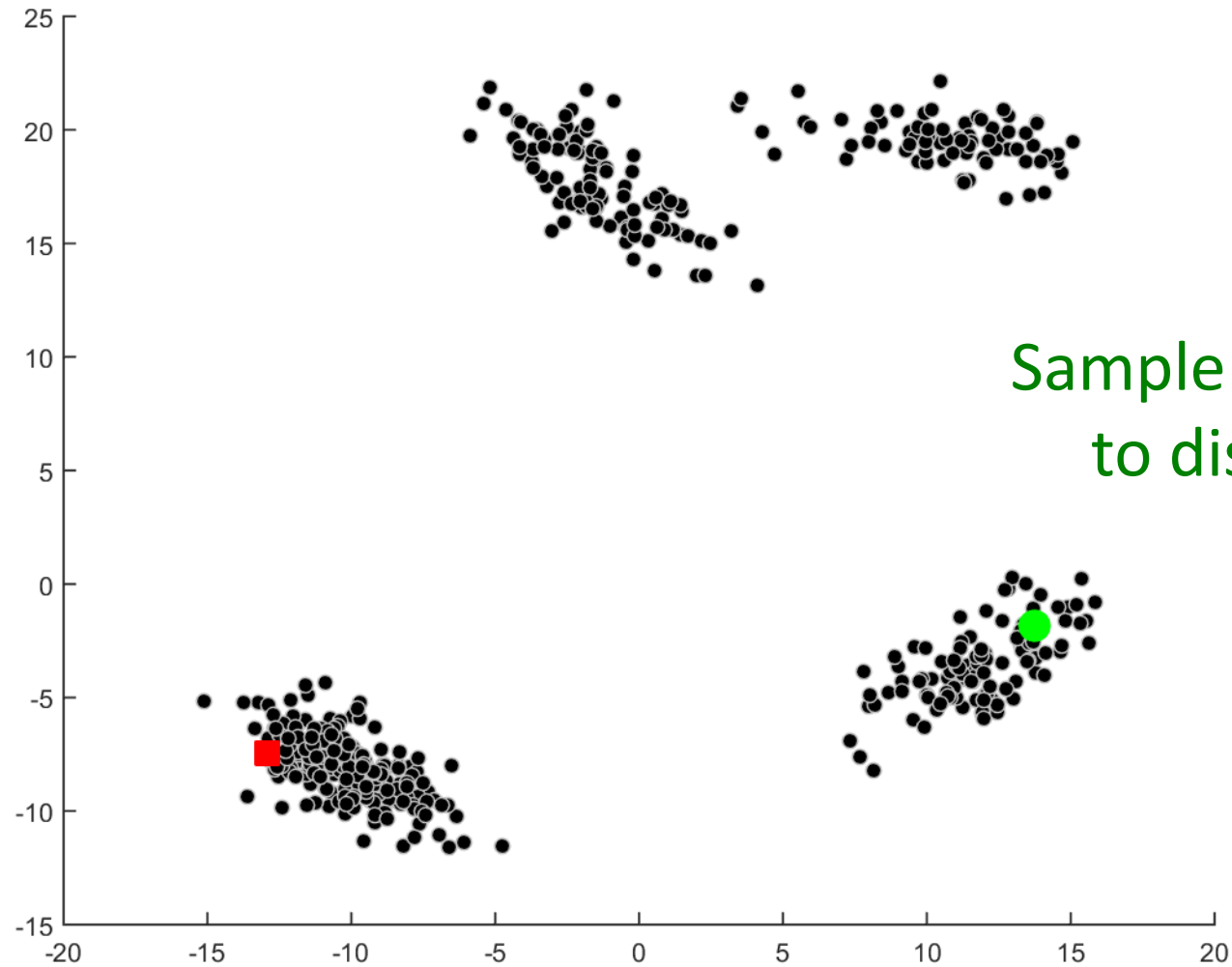
First mean is a
random example.

K-Means++



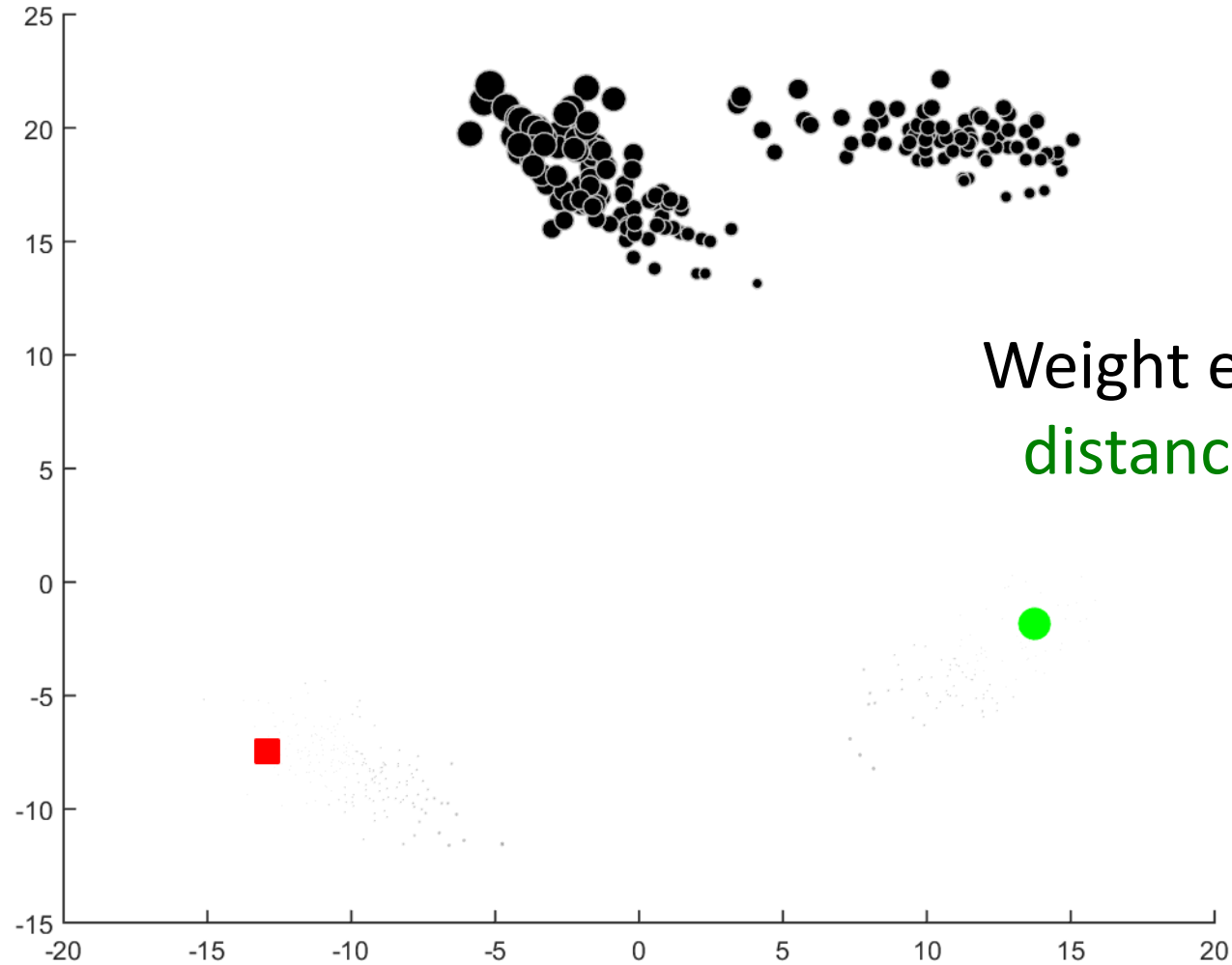
Weight examples by
distance to mean squared.

K-Means++



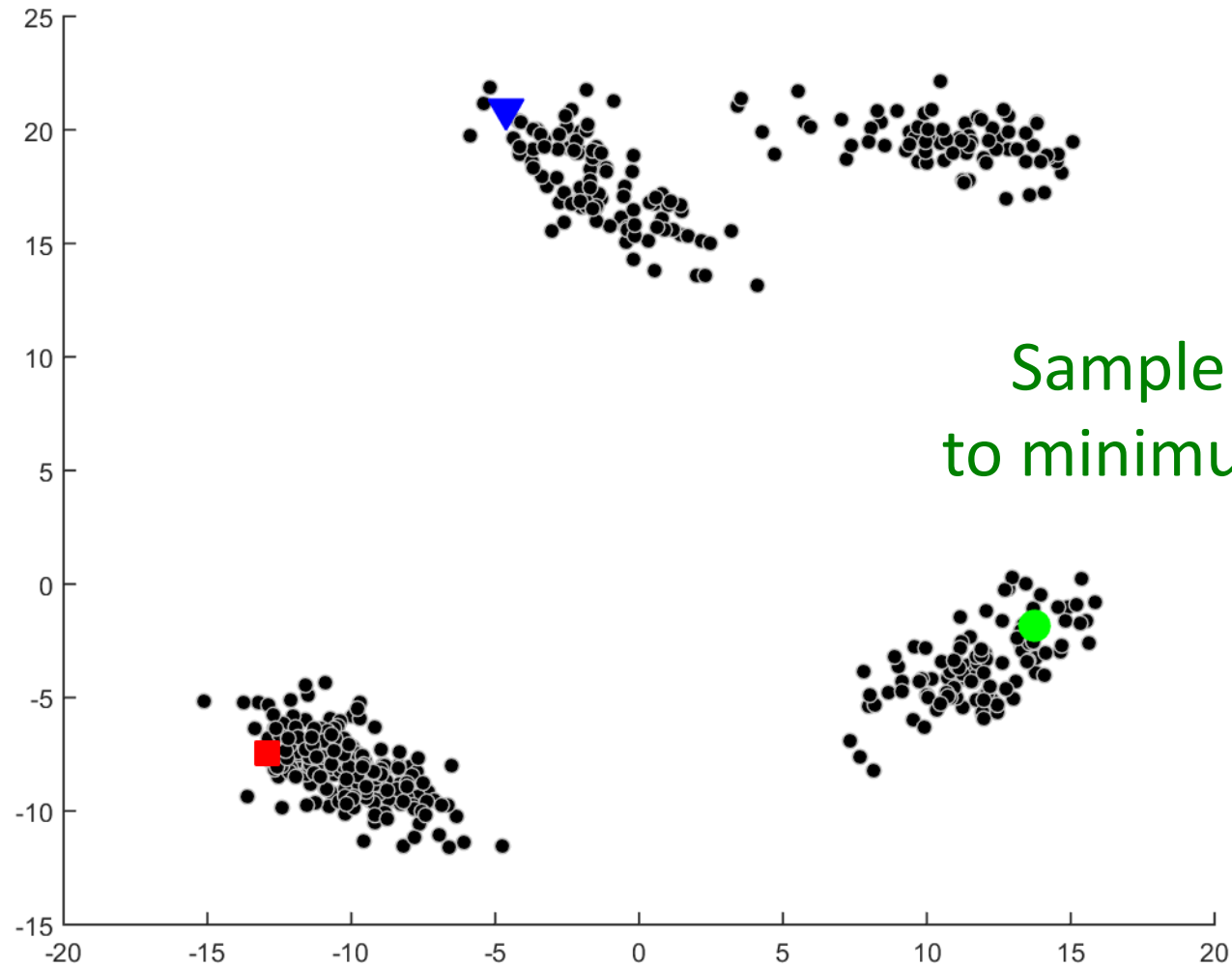
Sample mean proportional
to distances squared.

K-Means++



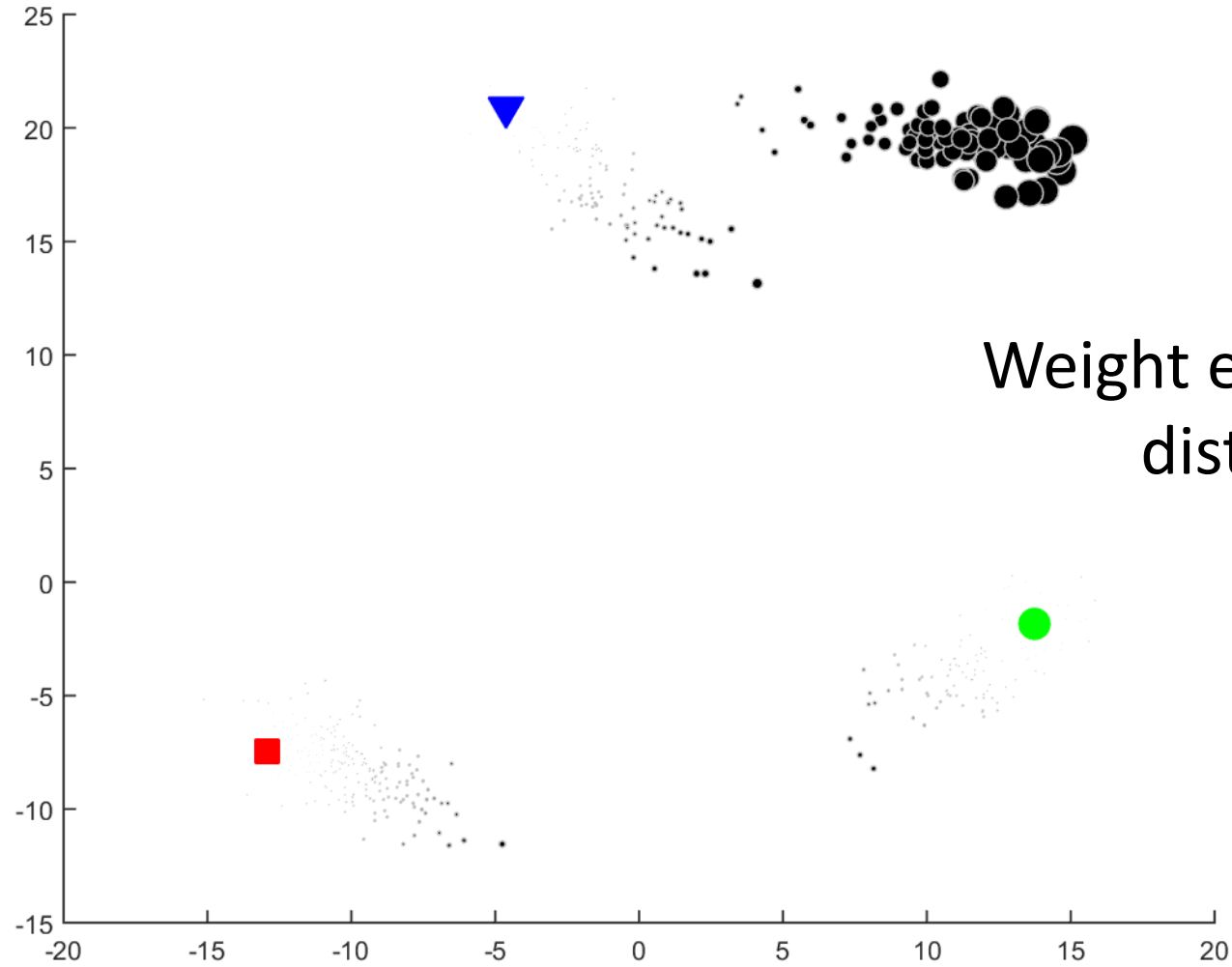
Weight examples by squared
distance to nearest mean.

K-Means++



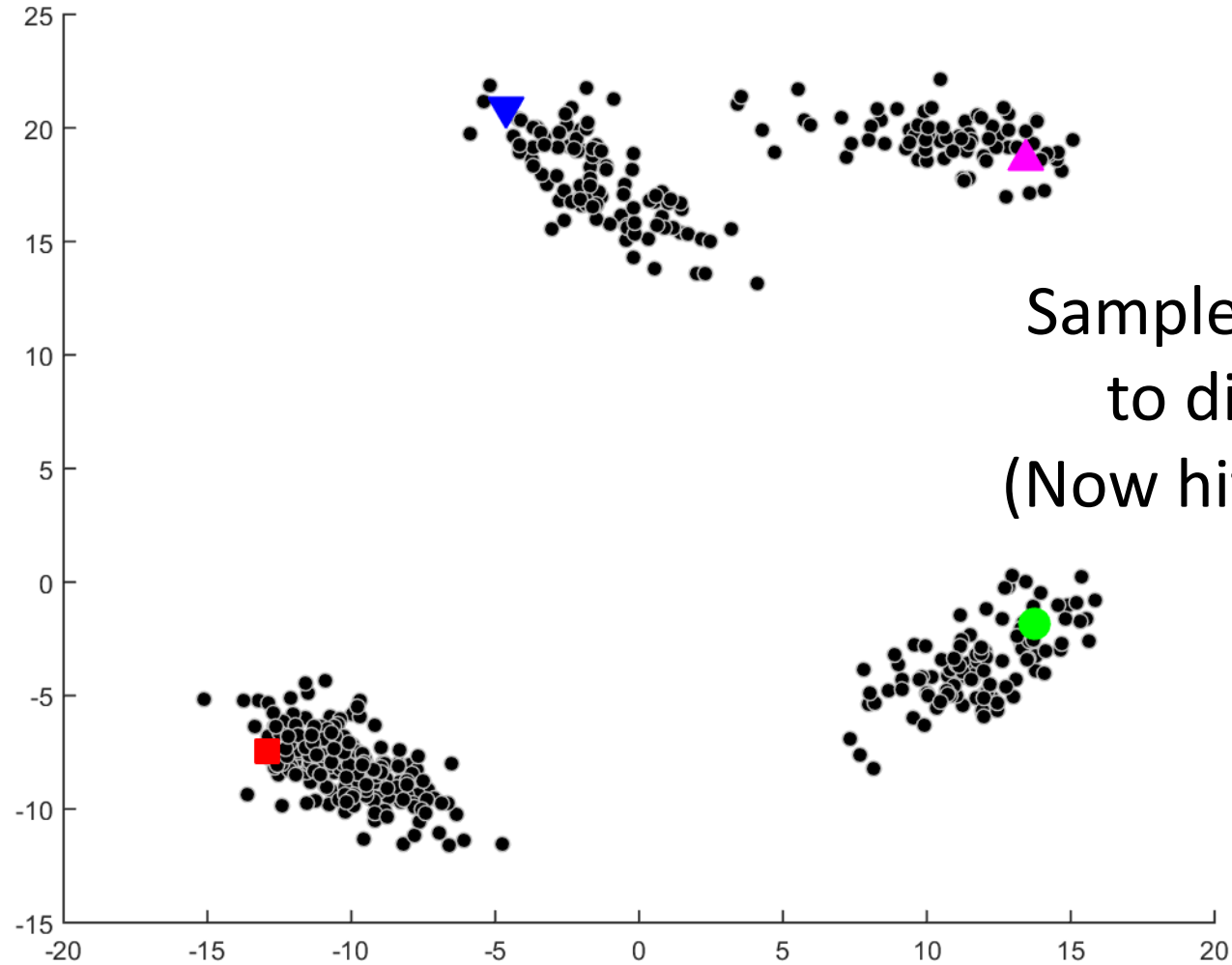
Sample mean proportional
to minimum distances squared

K-Means++



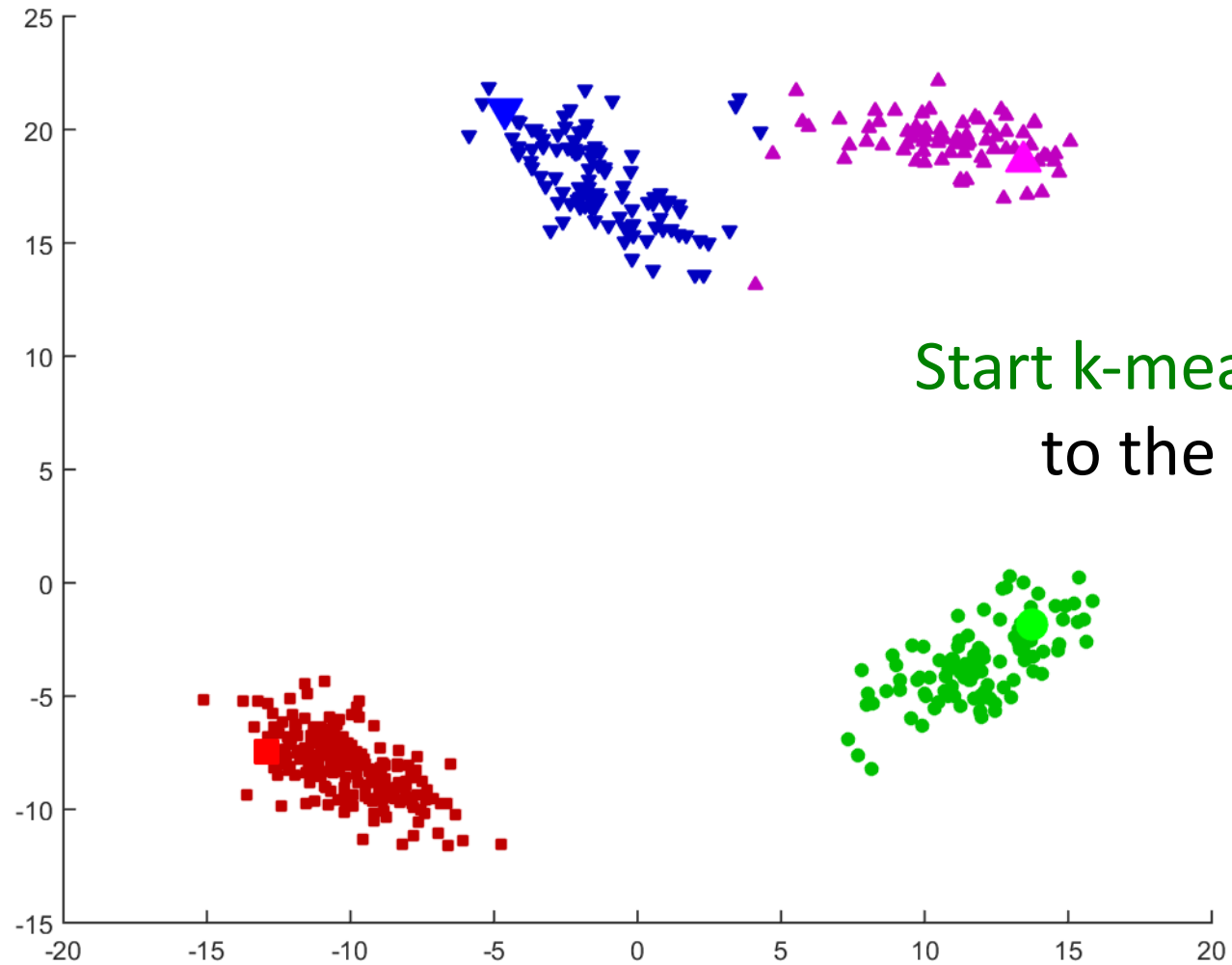
Weight examples by squared distance to mean.

K-Means++



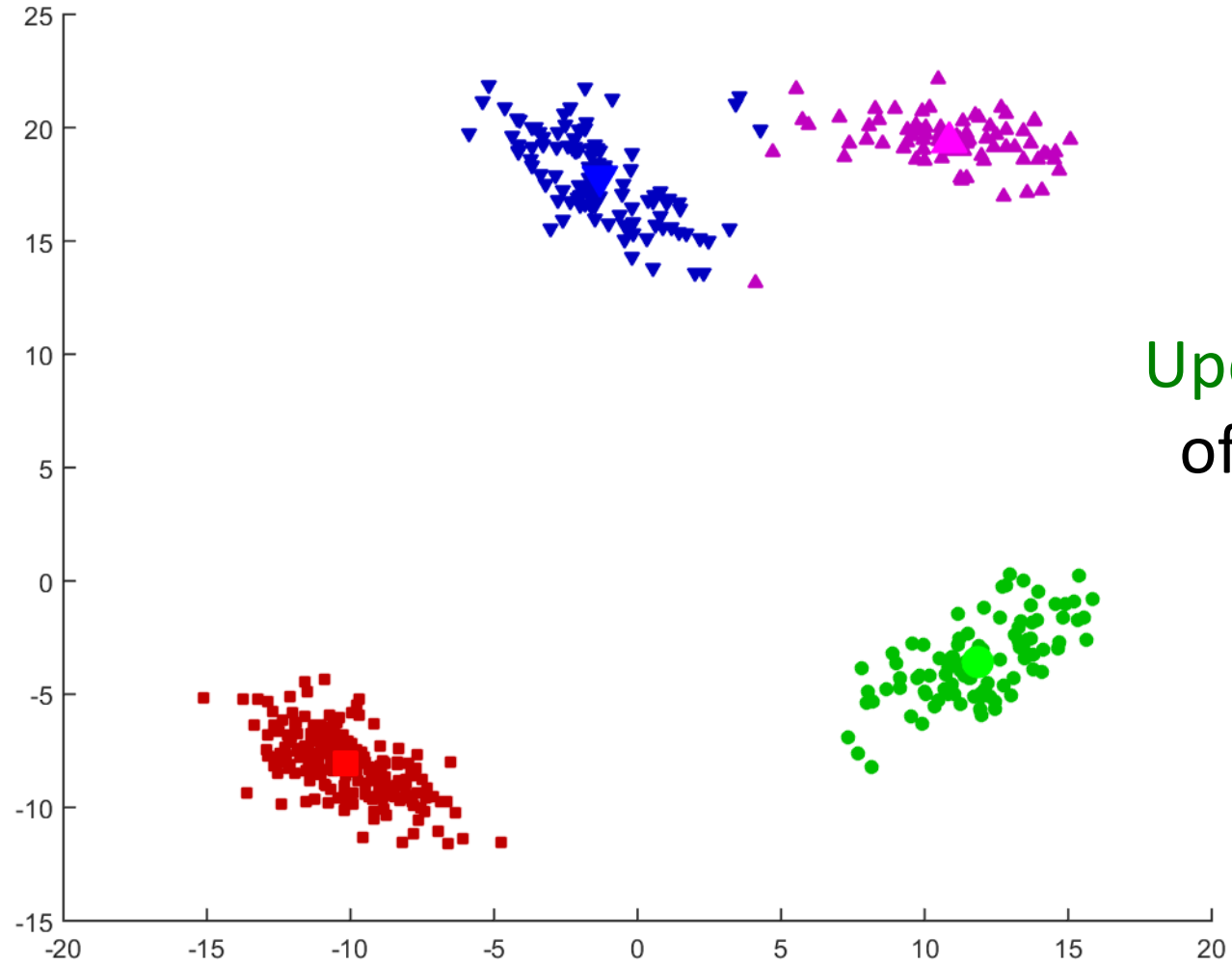
Sample mean proportional
to distances squared.
(Now hit chosen target $k=4$.)

K-Means++



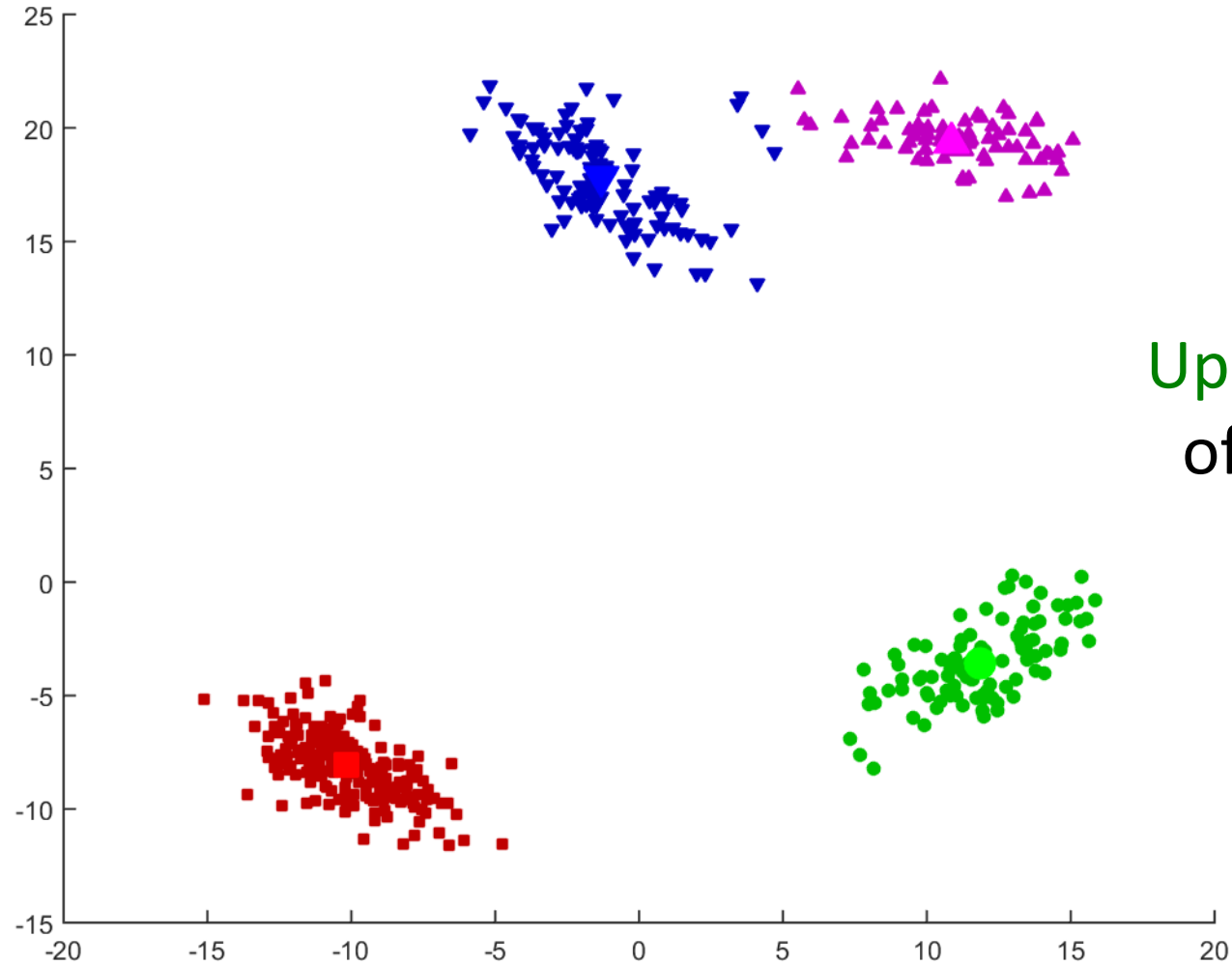
Start k-means: assign objects to the closest mean.

K-Means++



Update the mean
of each cluster.

K-Means++



Update the mean
of each cluster.

In this case: just 2 iterations!

Discussion of K-Means++

- Recall the objective function k-means tries to minimize:

$$f(W, c) = \sum_{i=1}^n \|x_i - w_{c(i)}\|_2^2$$

all means all assignments

- The initialization of 'W' and 'c' given by k-means++ satisfies:

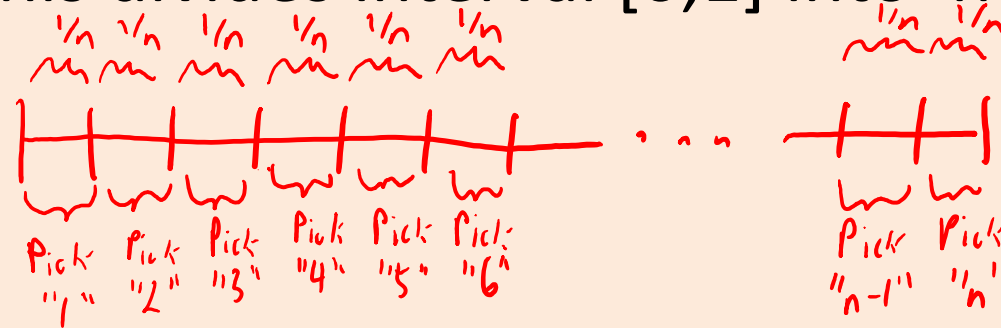
$$\underbrace{E [f(W, c)]}_{\substack{\text{expectation over} \\ \text{random samples}}} = O(\log(k))$$

$f(w^*, c^*)$ "Best" mean and clustering according to objective.

- Get good clustering with high probability by re-running.
- However, there is no guarantee that c^* is a good clustering.

Uniform Sampling

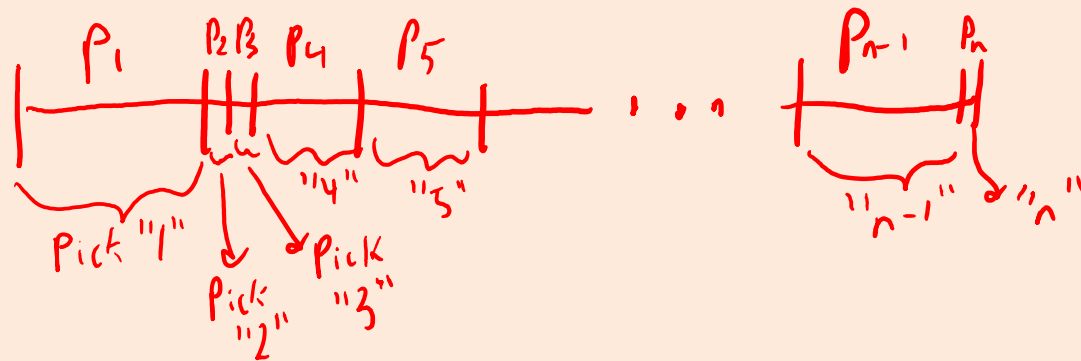
- Standard approach to generating a random number from $\{1,2,\dots,n\}$:
 1. Generate a uniform random number 'u' in the interval $[0,1]$.
 2. Return the largest index 'i' such that $u \leq i/n$.
- Conceptually, this divides interval $[0,1]$ into 'n' equal-size pieces:



- This assumes $p_i = 1/n$ for all 'i'.
↑ probability of picking number 'i'.

Non-Uniform Sampling

- Standard approach to generating a random number for general p_i .
 1. Generate a uniform random number 'u' in the interval [0,1].
 2. Return the largest index 'i' such that $u \leq \sum_{j=1}^i p_j$
- Conceptually, this divides interval [0,1] into non-equal-size pieces:



- Can sample from a generic discrete probability distribution in $O(n)$.
- If you need to generate 'm' samples:
 - Cost is $O(n + m \log(n))$ with binary search and storing cumulative sums.

How many iterations does k-means take?

- Each update of the ' \hat{y}_i ' or ' w_c ' does not increase the objective 'f'.
- And there are k^n possible assignments of the \hat{y}_i to 'k' clusters.
- So within k^n iterations you cannot improve the objective by changing \hat{y}_i , and the algorithm stops.

- Tighter-but-more-complicated “smoothed” analysis:
 - <https://arxiv.org/pdf/0904.1113.pdf>

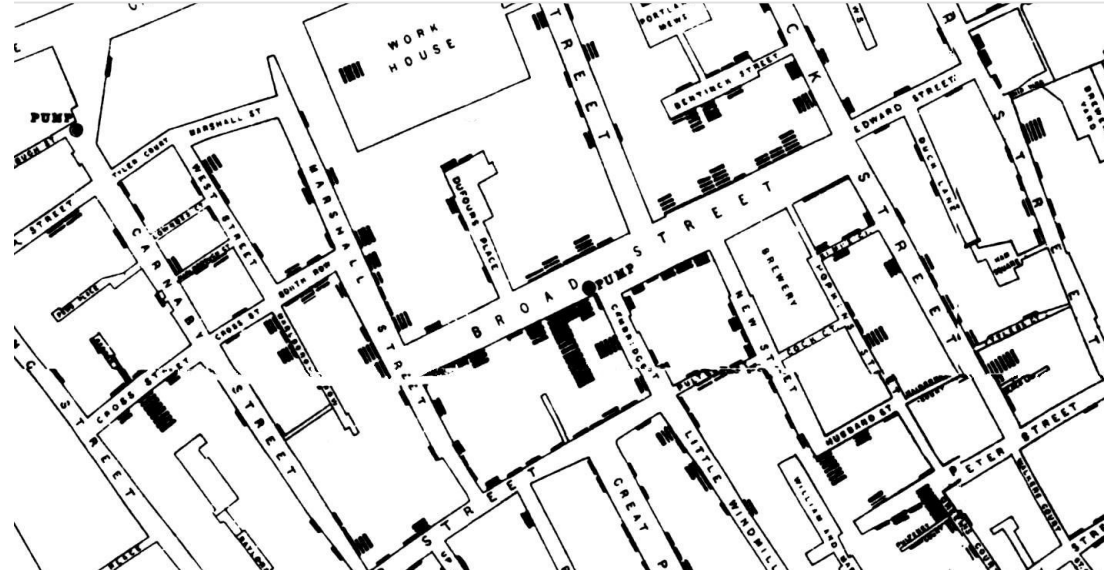
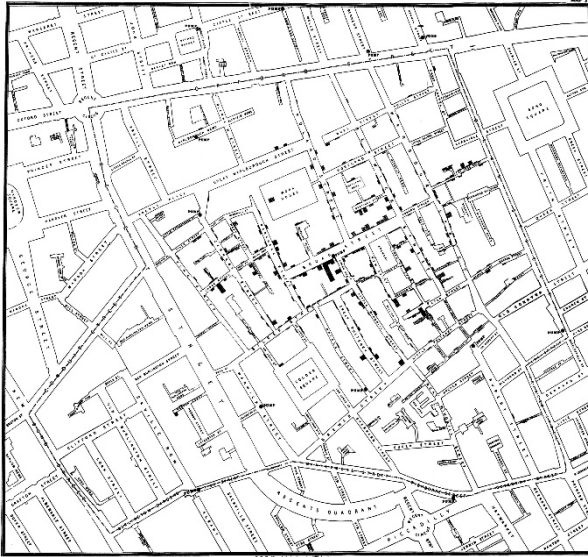
- wallpaper
- funny
- season 6
- winter is coming meme
- genderbent
- let it snow
- real life
- ghost
- wallpaper hd
- the watch meme
- his wolf
- game throne
- simpsons

Did you mean: [jon snow](#)



John Snow and Cholera Epidemic

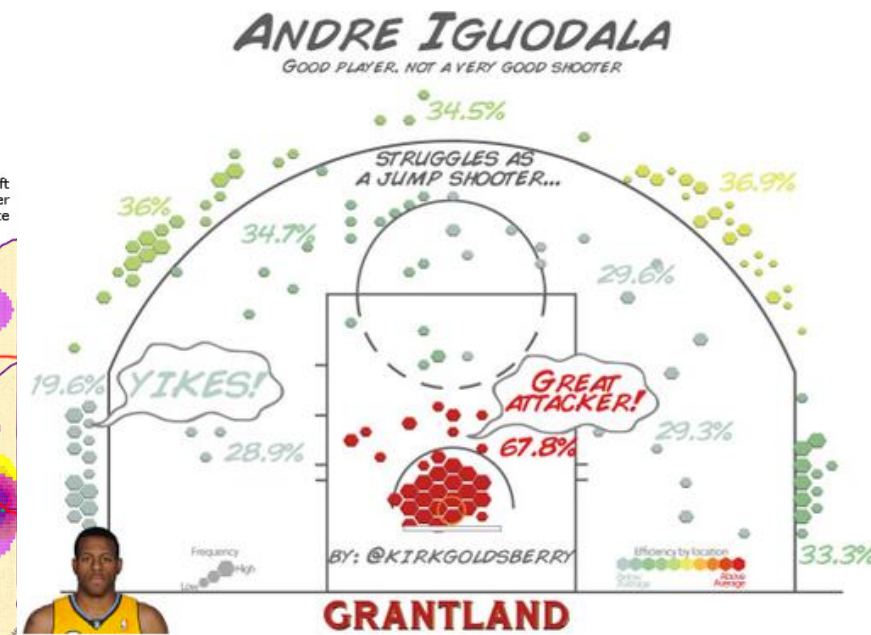
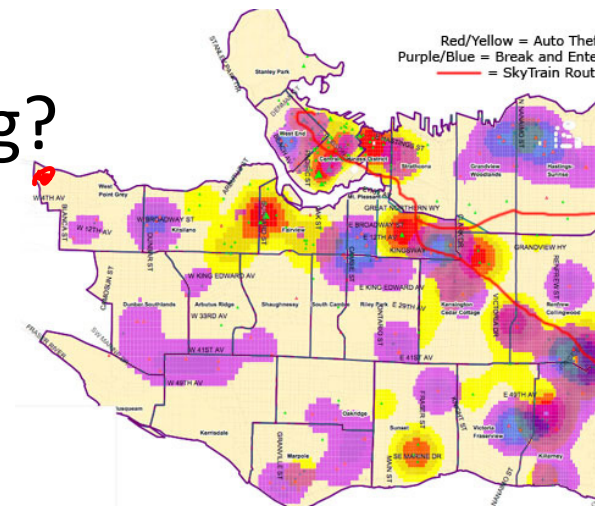
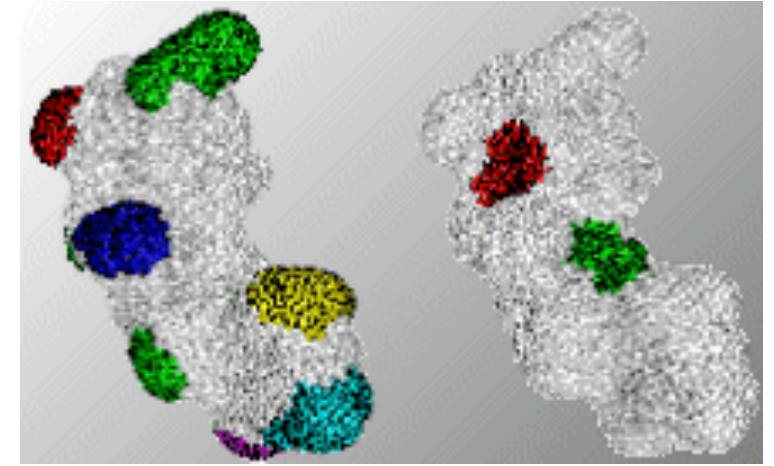
- John Snow's 1854 spatial histogram of deaths from cholera:



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

Other Potential Applications

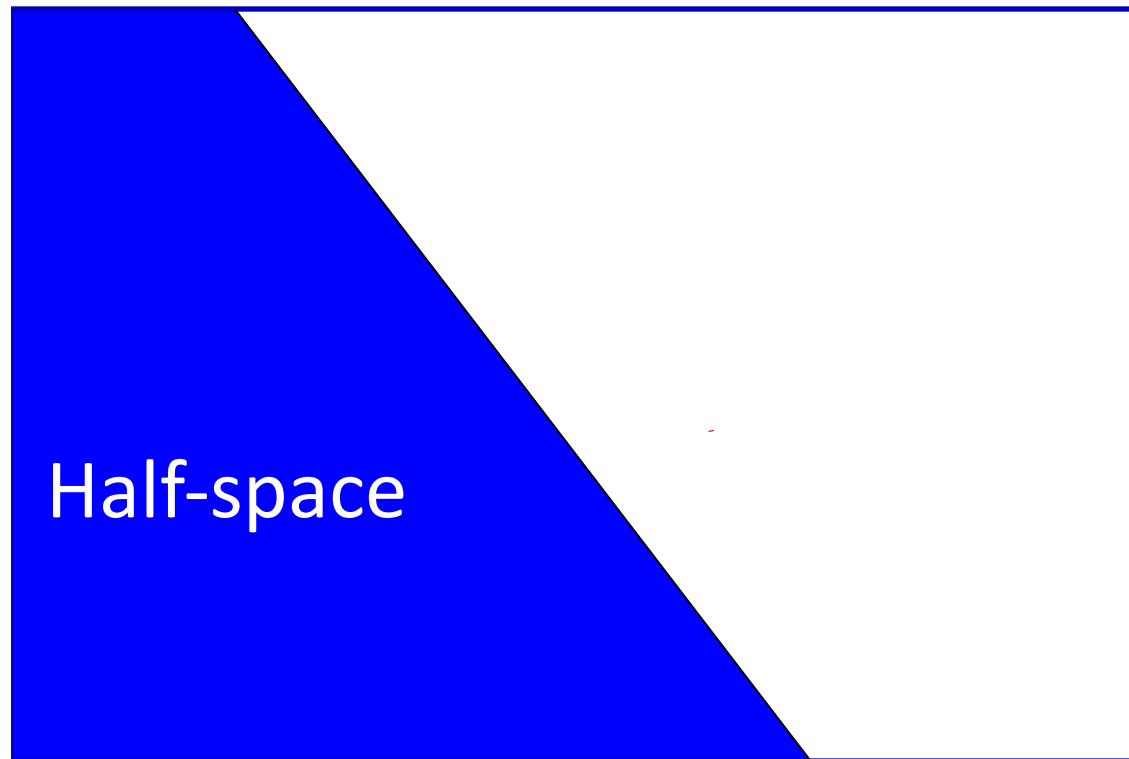
- Where are high crime regions of a city?
- Where should taxis patrol?
- Where does Iguodala make/miss shots?
- Which products are similar to this one?
- Which pictures are in the same place?
- Where can protein 'dock'?
- Where are people tweeting?



Why are k-means clusters convex?

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

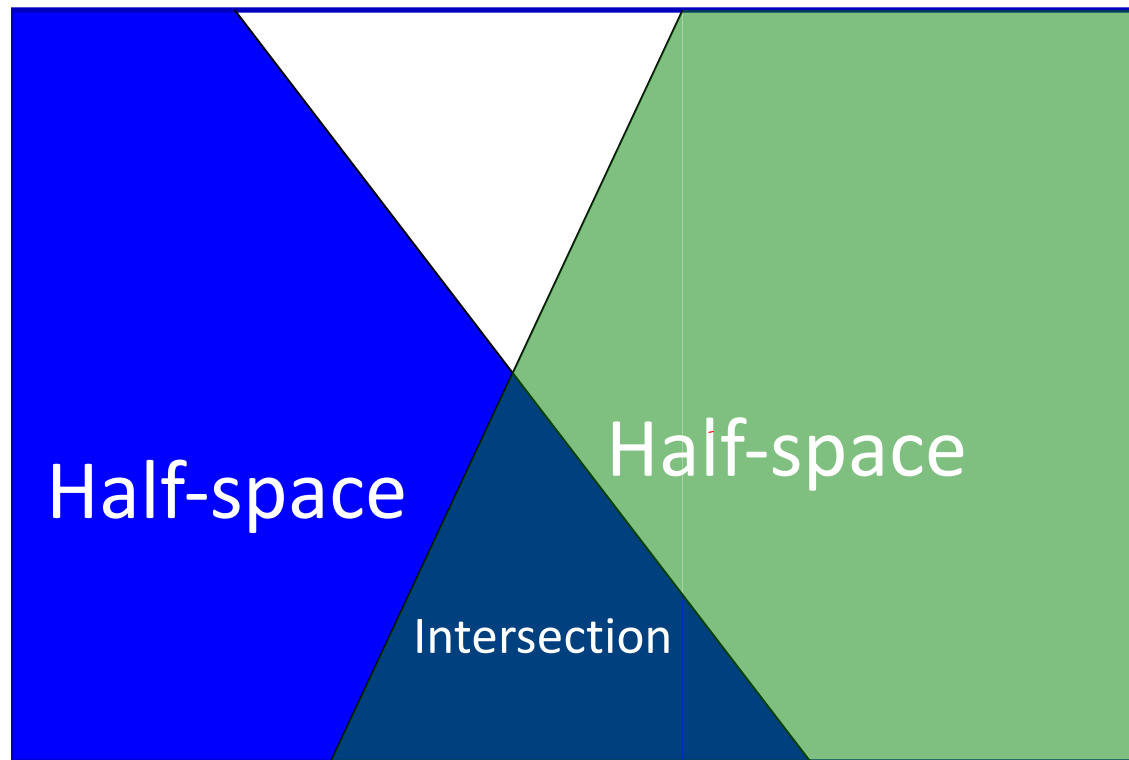
Half-space is Set of points (satisfying a linear inequality), like $\sum_{j=1}^d a_j x_j \leq b$



Why are k-means clusters convex?

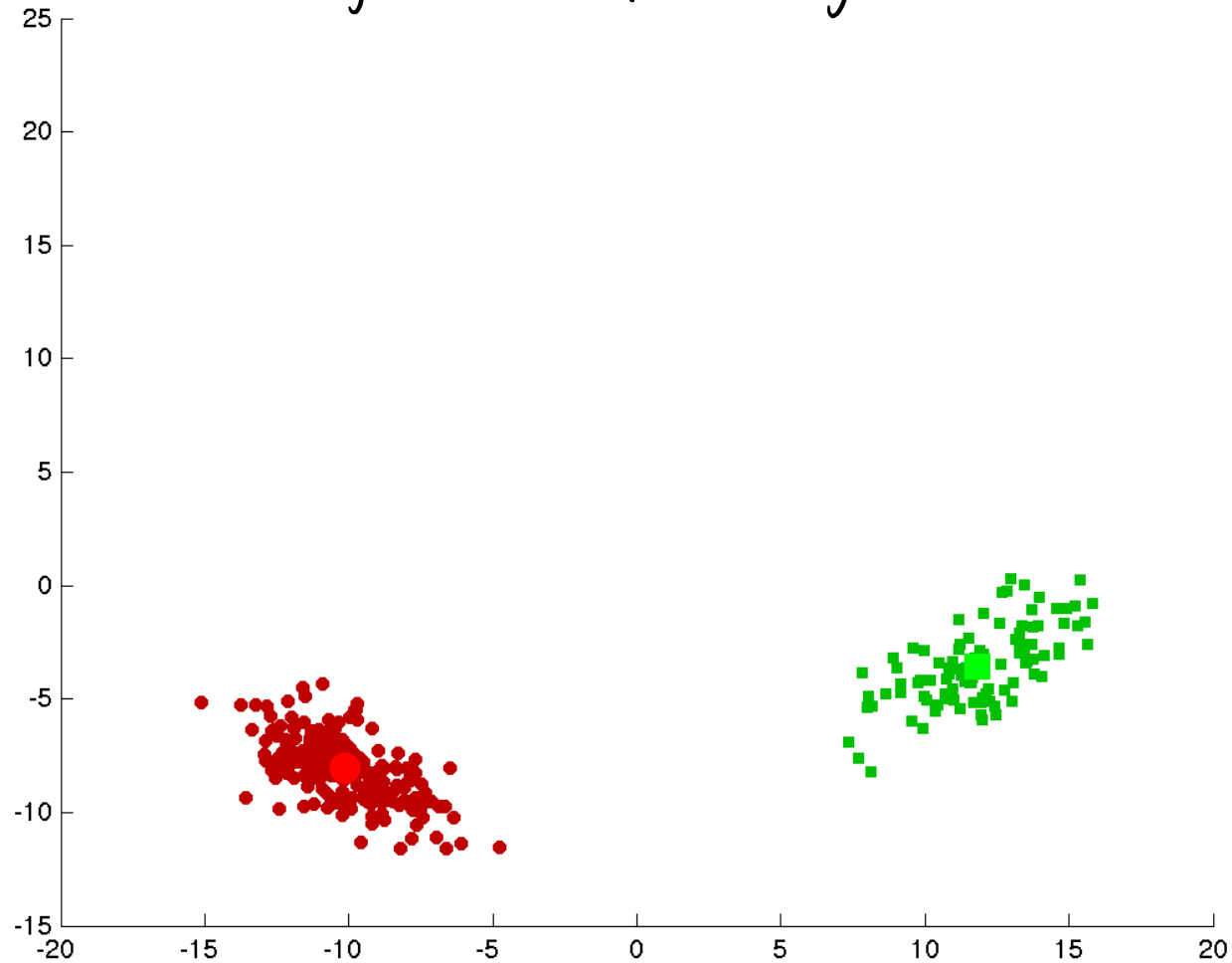
- K-means clusters are formed by the **intersection** of **half-spaces**.

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



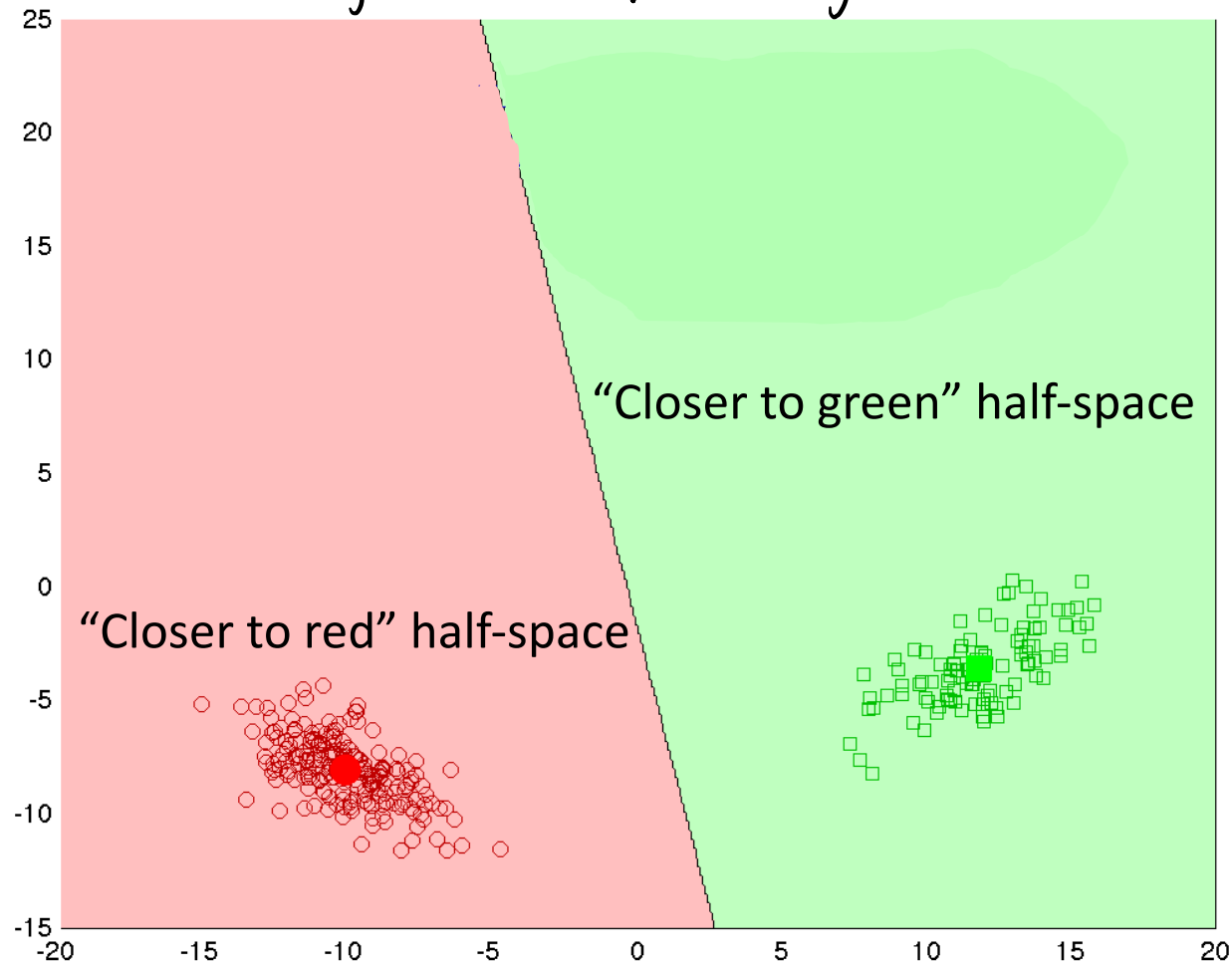
Why are k-means clusters convex?

Which regions are put in green cluster?

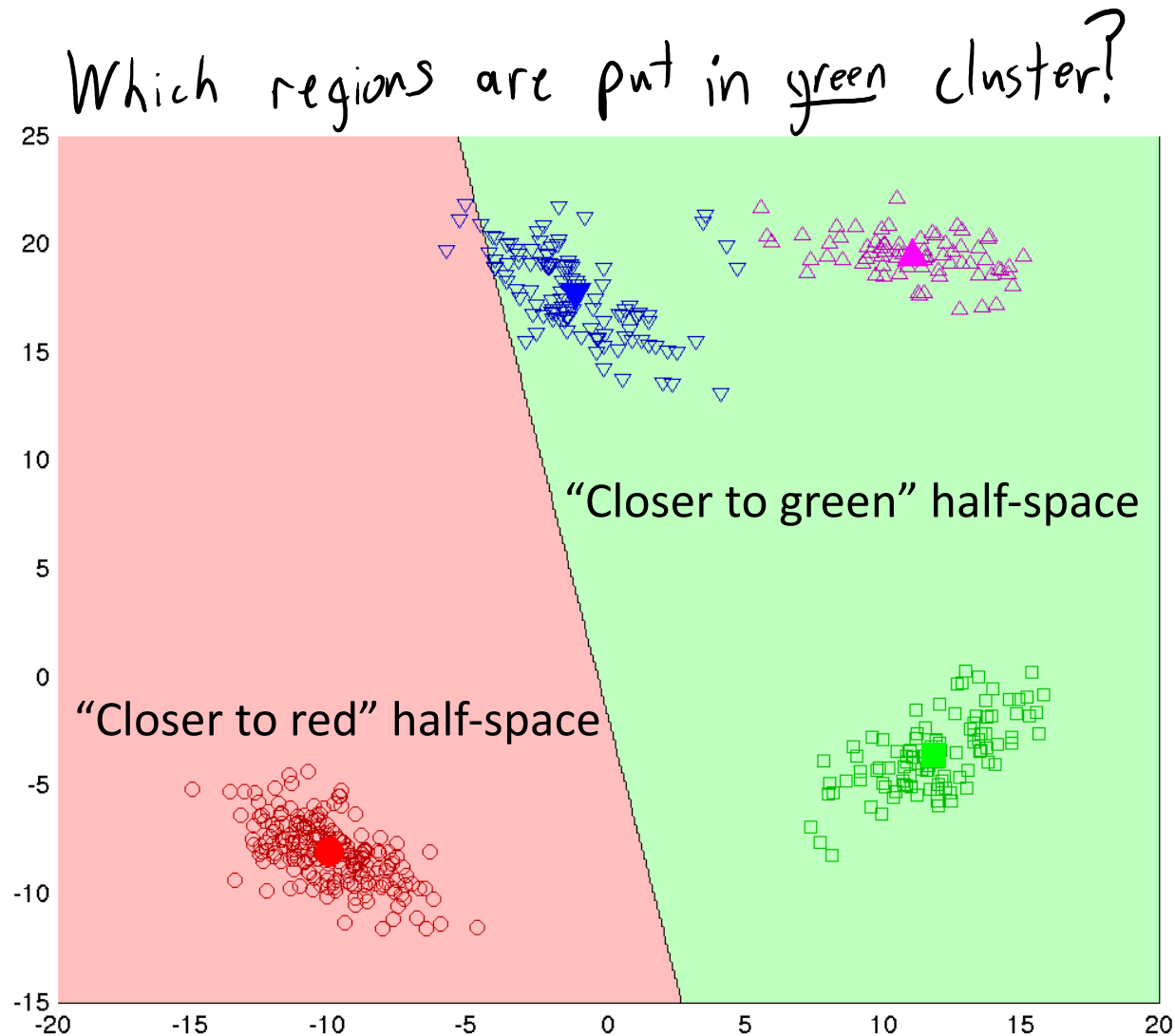


Why are k-means clusters convex?

Which regions are put in green cluster?

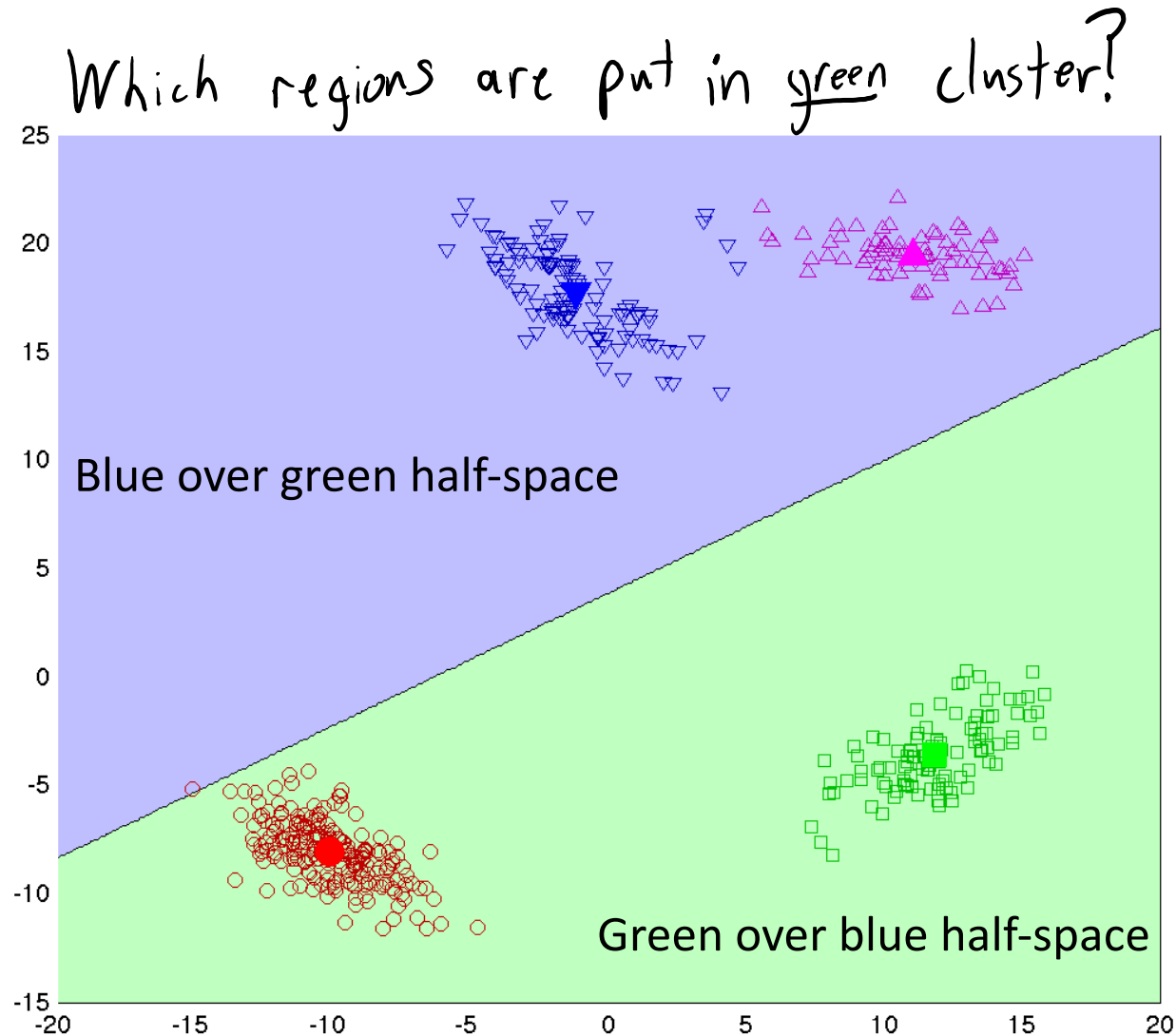


Why are k-means clusters convex?



Red vs. green
decision stays the
same with more clusters.

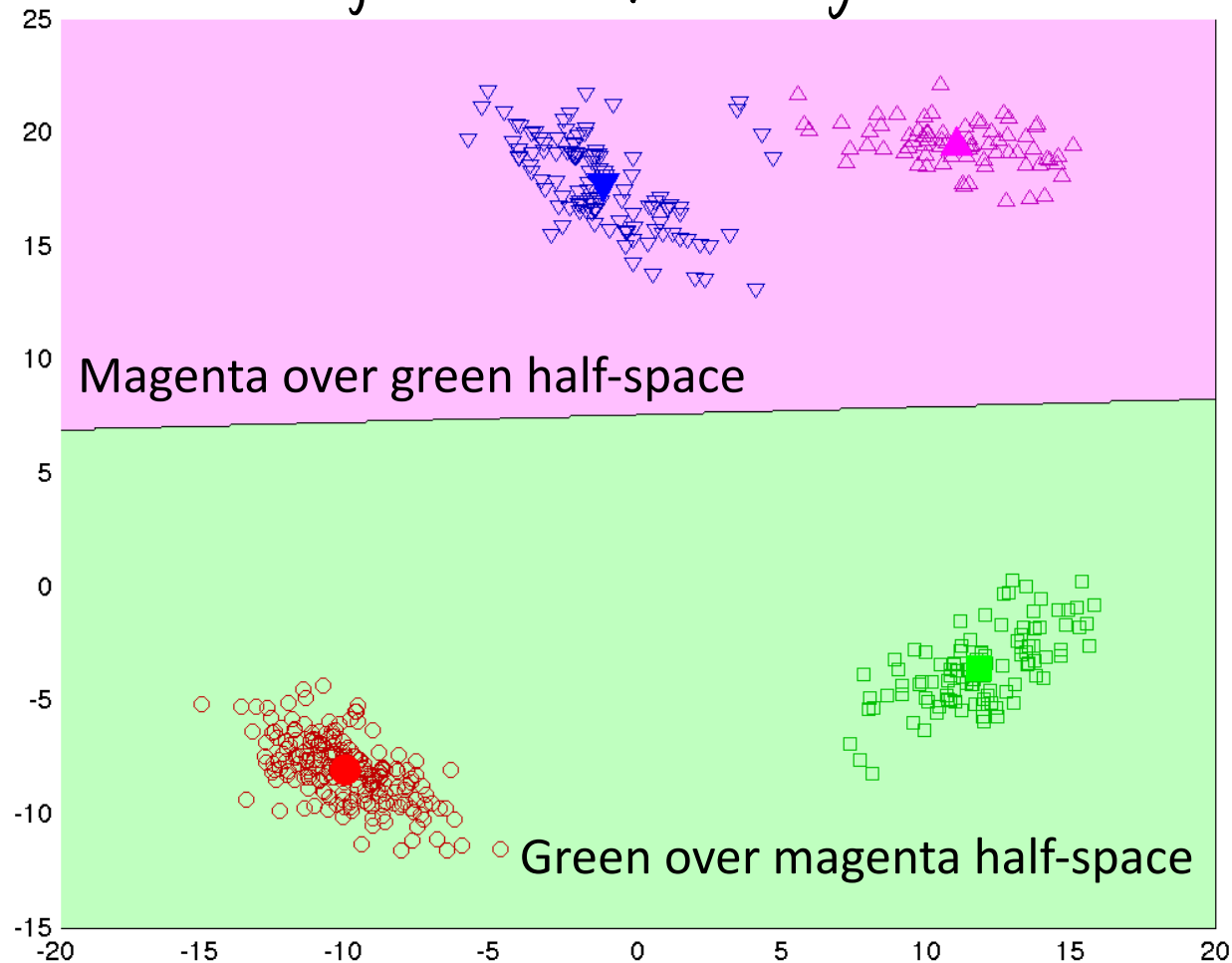
Why are k-means clusters convex?



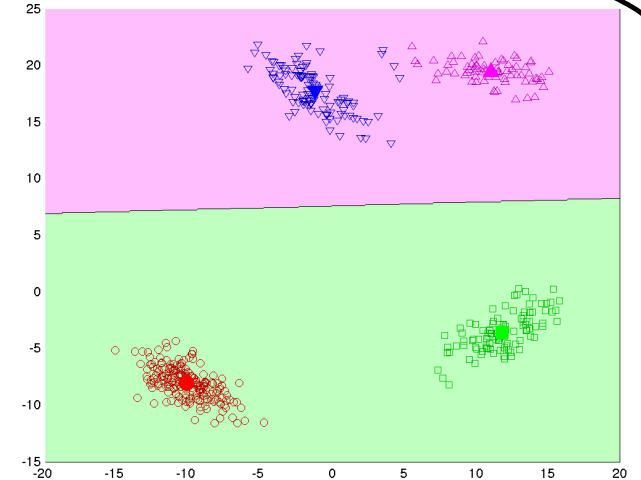
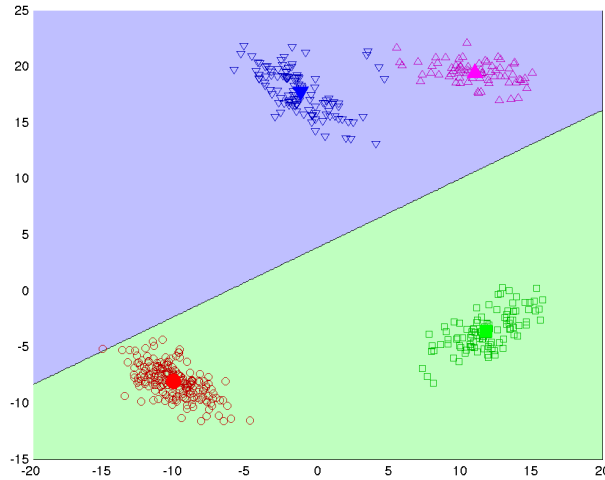
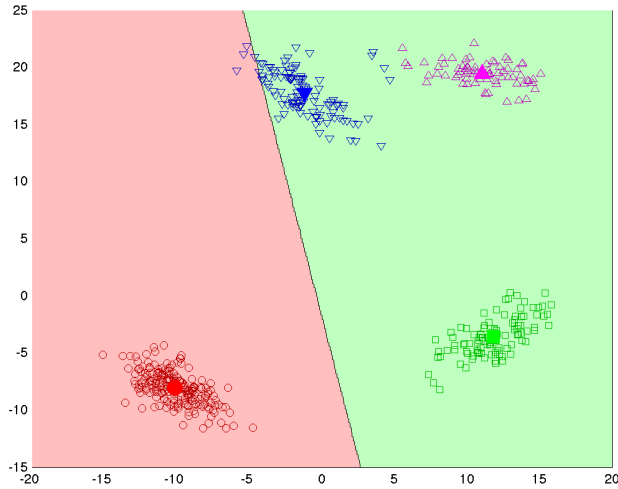
Blue vs. green decision
defines different
half-spaces.

Why are k-means clusters convex?

Which regions are put in green cluster?

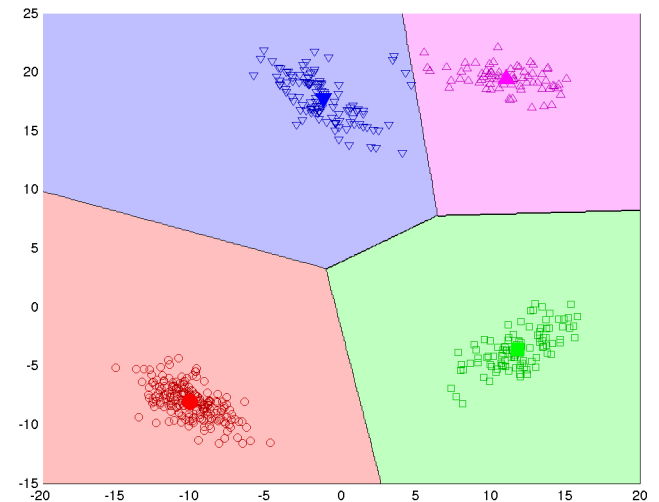


Why are k-means clusters convex?



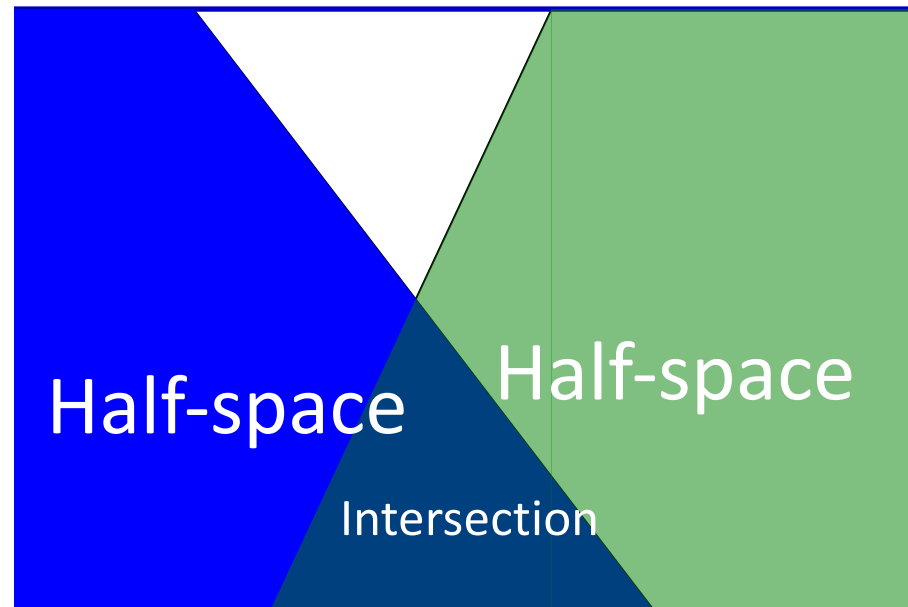
→ Green "cluster" is the intersection of these three half-spaces.

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.
- So intersection of half-spaces is convex.



Why are k-means clusters convex?

- Formal **proof** that "cluster 1" is convex (so all clusters are convex):

Let x_i and x_j be arbitrary points in cluster 1.

→ By def'n of cluster 1, $\|x_i - w_1\| \leq \|x_i - w_c\|$ for all 'c' } equality
 $\|x_j - w_1\| \leq \|x_j - w_c\|$ for all 'c' } for $c=1$

→ Let x_m be an arbitrary point between x_i and x_j .

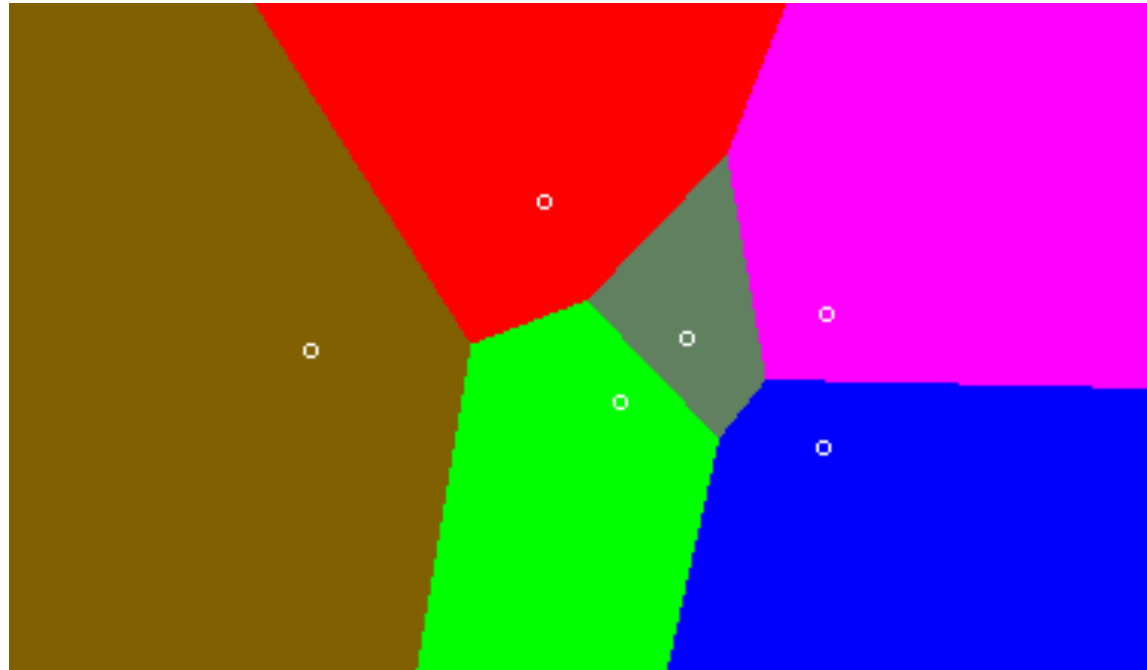
→ So we can write it as $x_m = \theta x_i + (1-\theta)x_j$ for some $\theta \in [0, 1]$

$$\begin{aligned} \text{Then } \|x_m - w_1\| &= \|\theta x_i + (1-\theta)x_j - (\theta w_1 + (1-\theta)w_1)\| && (w_1 = \theta w_1 + (1-\theta)w_1) \\ &\leq \|\theta x_i - \theta w_1\| + \|(1-\theta)x_j - (1-\theta)w_1\| && (\text{triangle inequality}) \end{aligned}$$

$$\begin{aligned} &= \theta \|x_i - w_1\| + (1-\theta) \|x_j - w_1\| && (\text{homogeneity of norms}) \\ \left. \begin{array}{l} x_i \text{ and } x_j \\ \text{are in cluster 1} \end{array} \right\} &\leq \theta \|x_i - w_c\| + (1-\theta) \|x_j - w_c\| = \|x_j - w_c\| && \text{so } x_m \text{ is in cluster 1!} \end{aligned}$$

Voronoi Diagrams

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

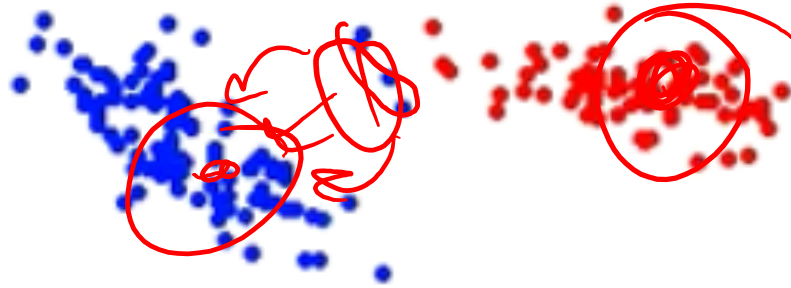


- Can be a useful visualization of “nearest available” problems.
 - E.g., [nearest tube station in London](#).

UBClustering Algorithm

- Let's define a new ensemble clustering method: **UBClustering**.
 1. Run k-means with 'm' different random initializations.
 2. For each object 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.

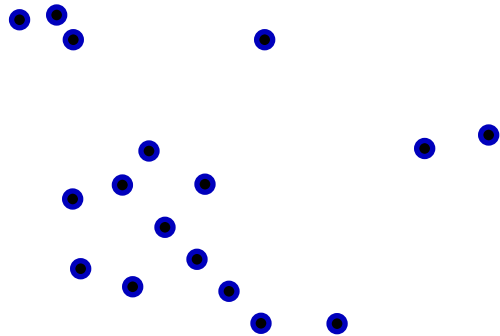
UBClustering Algorithm



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

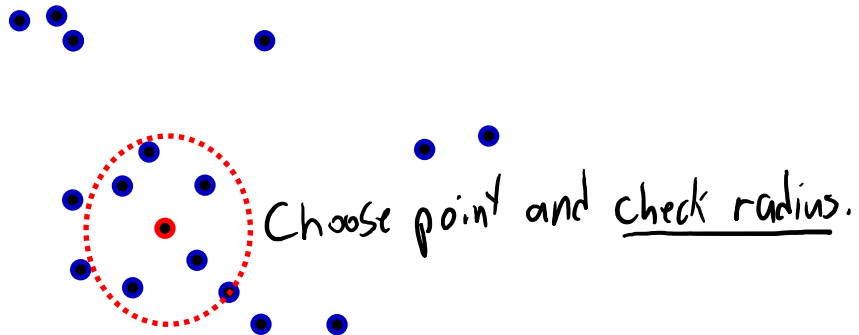
Density-Based Clustering (Example)

- Each “core” point defines a cluster:
 - Consisting of “core” point and all its “neighbours”.
- Merge clusters if “core” points are “neighbours” of each other.



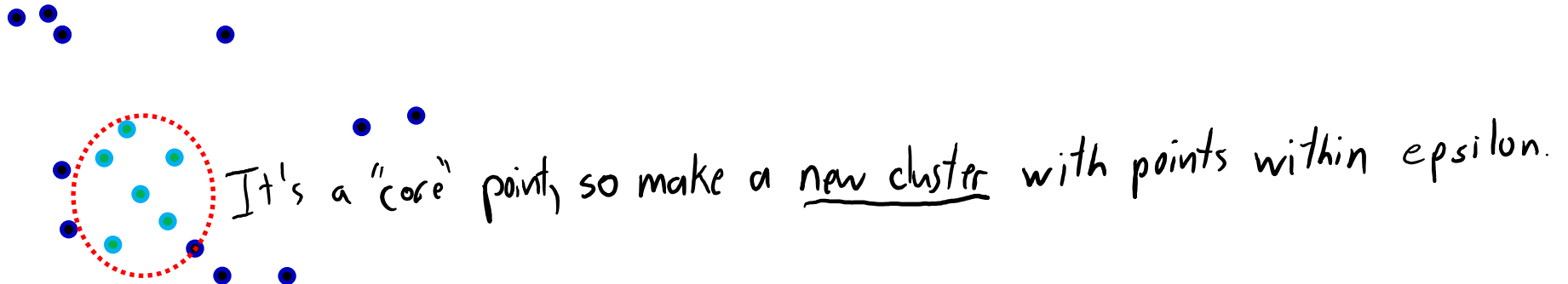
Density-Based Clustering (Example)

- Each “core” point defines a cluster:
 - Consisting of “core” point and all its “neighbours”.
- Merge clusters if “core” points are “neighbours” of each other.



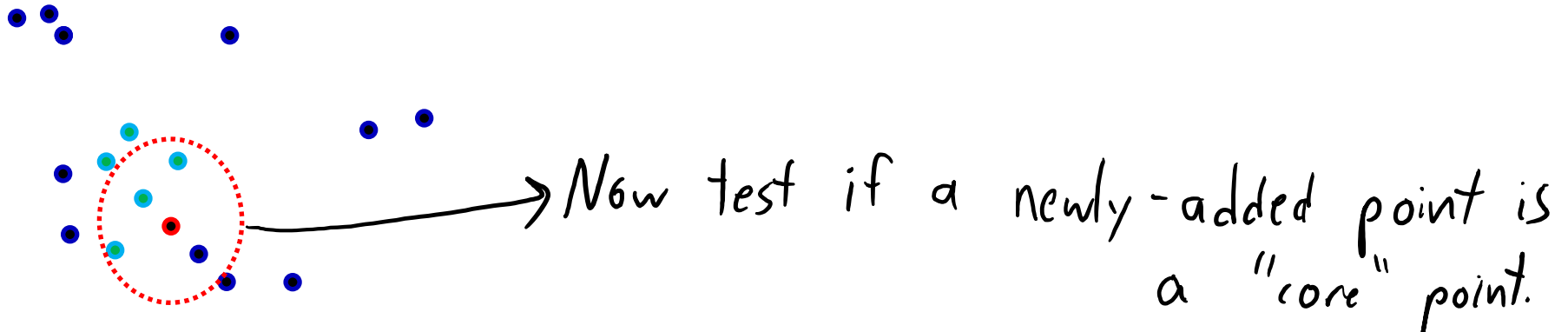
Density-Based Clustering (Example)

- Each “core” point defines a cluster:
 - Consisting of “core” point and all its “neighbours”.
- Merge clusters if “core” points are “neighbours” of each other.



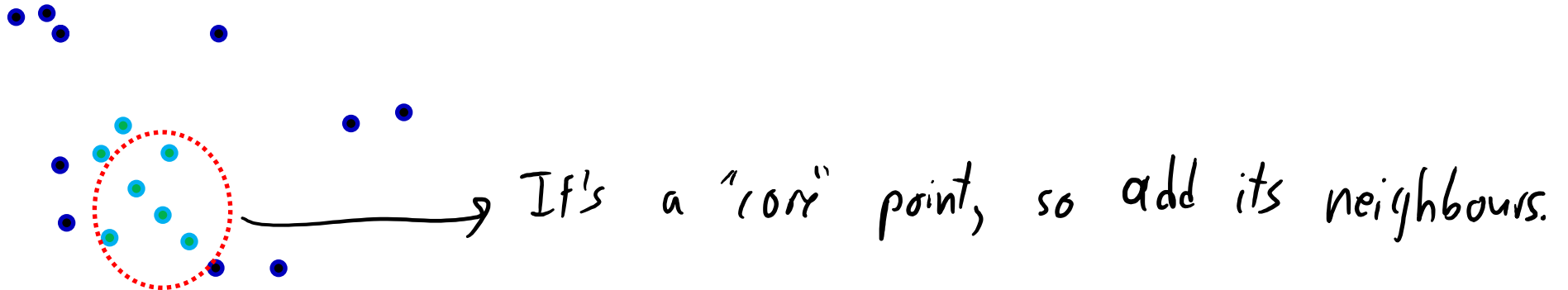
Density-Based Clustering (Example)

- Each “core” point defines a cluster:
 - Consisting of “core” point and all its “neighbours”.
- Merge clusters if “core” points are “neighbours” of each other.



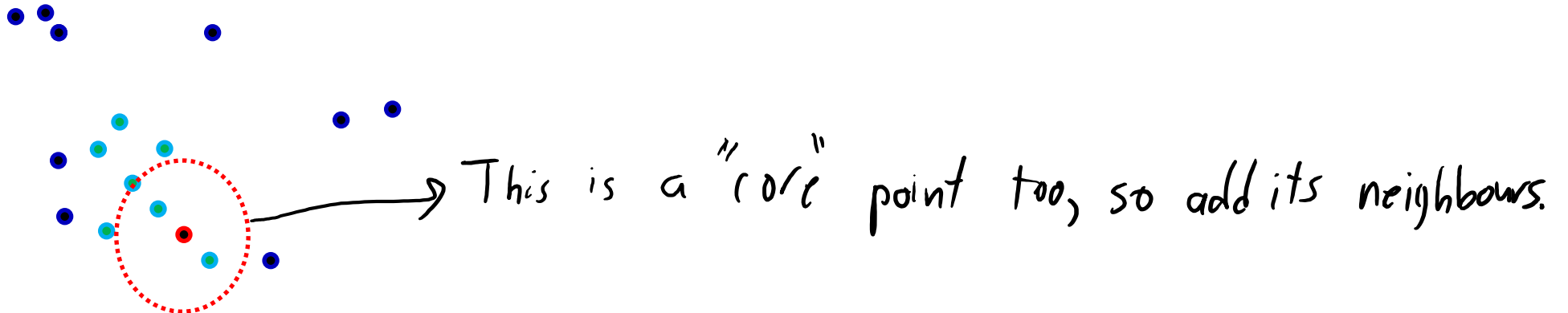
Density-Based Clustering (Example)

- Each “core” point defines a cluster:
 - Consisting of “core” point and all its “neighbours”.
- Merge clusters if “core” points are “neighbours” of each other.



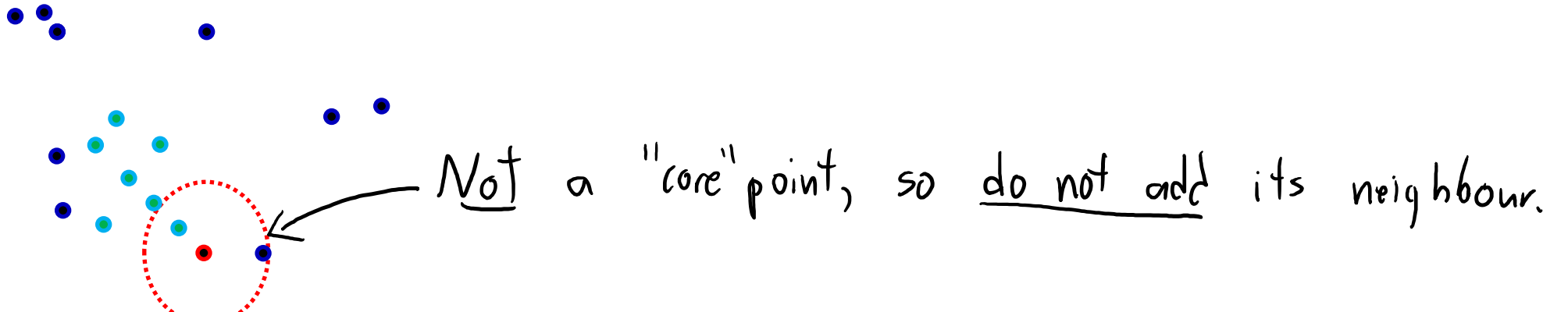
Density-Based Clustering (Example)

- Each “core” point defines a cluster:
 - Consisting of “core” point and all its “neighbours”.
- Merge clusters if “core” points are “neighbours” of each other.



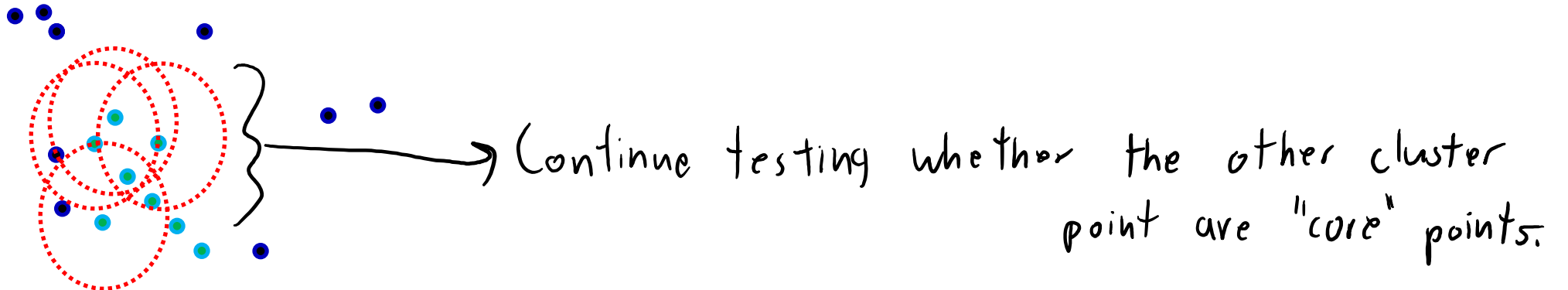
Density-Based Clustering (Example)

- Each “core” point defines a cluster:
 - Consisting of “core” point and all its “neighbours”.
- Merge clusters if “core” points are “neighbours” of each other.



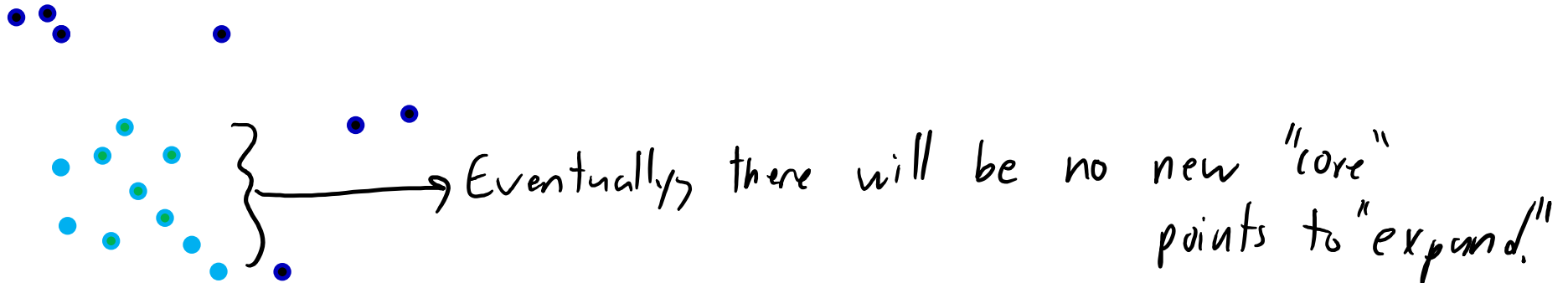
Density-Based Clustering (Example)

- Each “core” point defines a cluster:
 - Consisting of “core” point and all its “neighbours”.
- Merge clusters if “core” points are “neighbours” of each other.



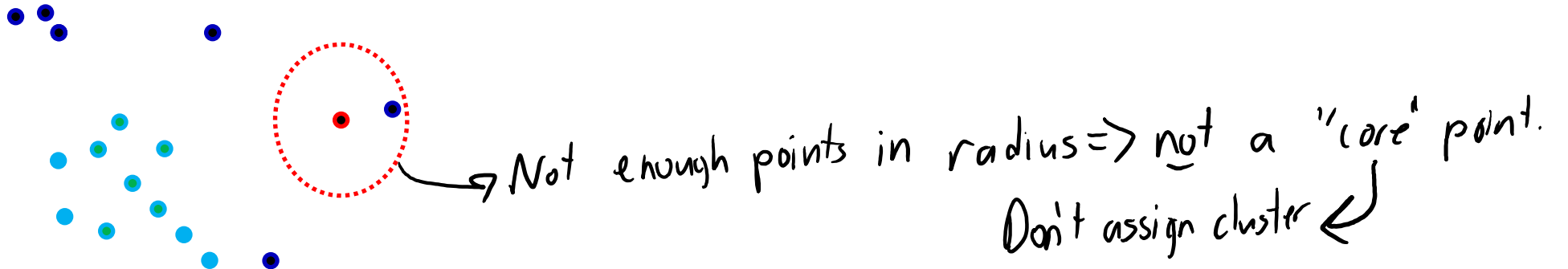
Density-Based Clustering (Example)

- Each “core” point defines a cluster:
 - Consisting of “core” point and all its “neighbours”.
- Merge clusters if “core” points are “neighbours” of each other.



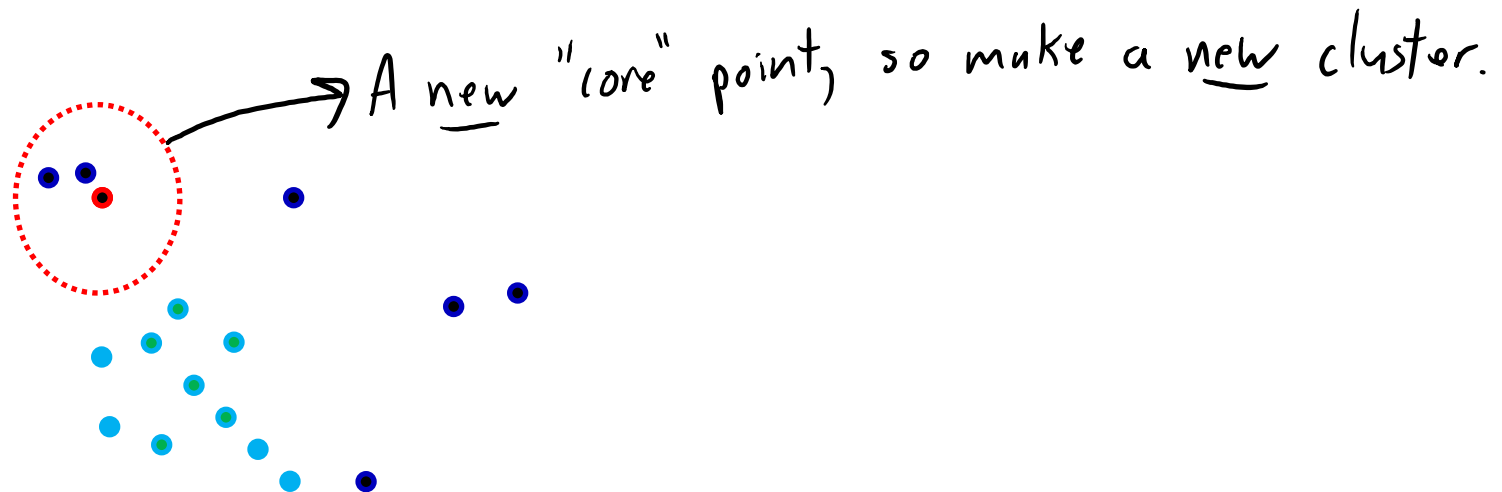
Density-Based Clustering (Example)

- Each “core” point defines a cluster:
 - Consisting of “core” point and all its “neighbours”.
- Merge clusters if “core” points are “neighbours” of each other.



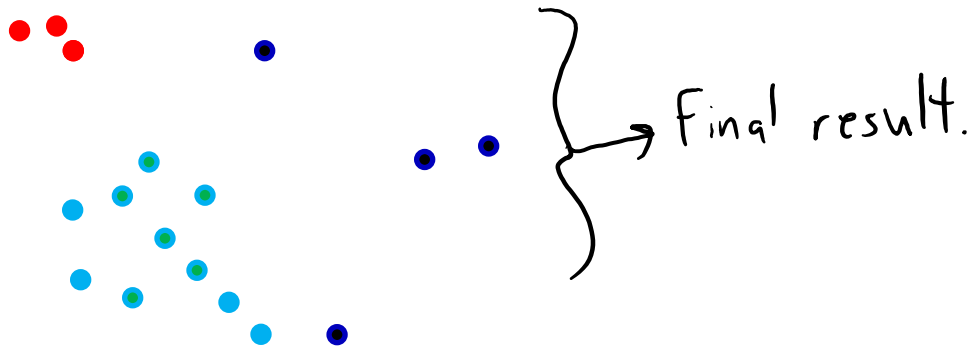
Density-Based Clustering (Example)

- Each “core” point defines a cluster:
 - Consisting of “core” point and all its “neighbours”.
- Merge clusters if “core” points are “neighbours” of each other.



Density-Based Clustering (Example)

- Each “core” point defines a cluster:
 - Consisting of “core” point and all its “neighbours”.
- Merge clusters if “core” points are “neighbours” of each other.



Density-Based Clustering in Action

