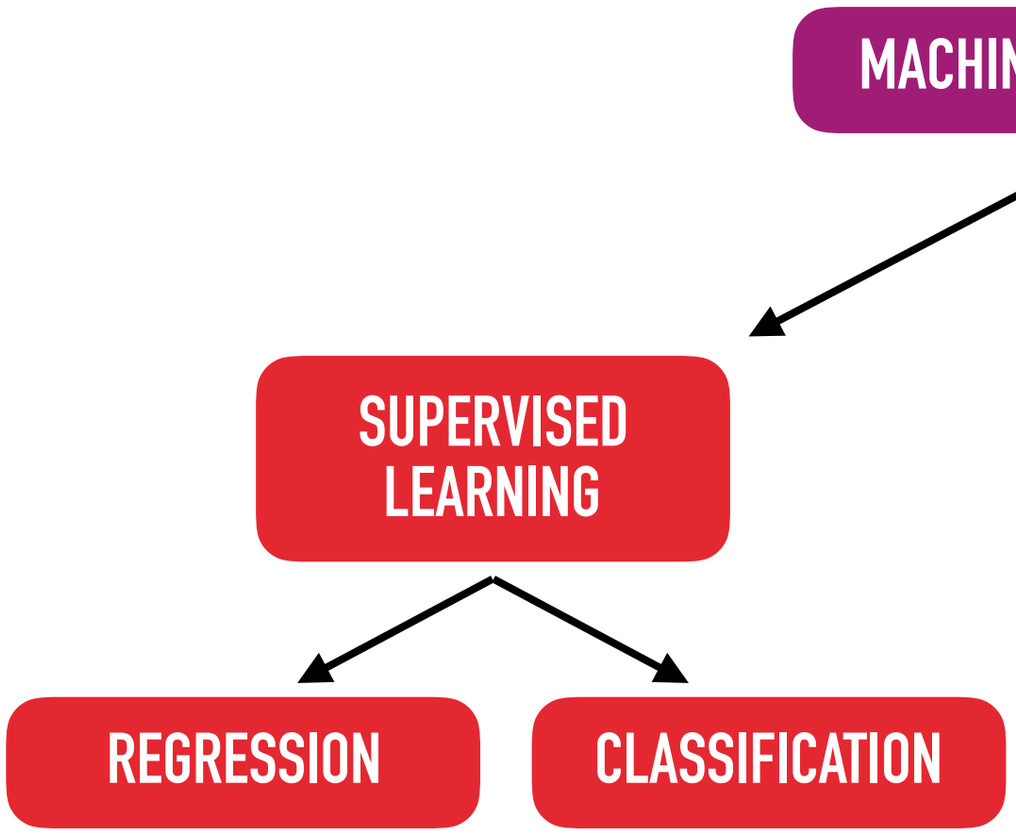
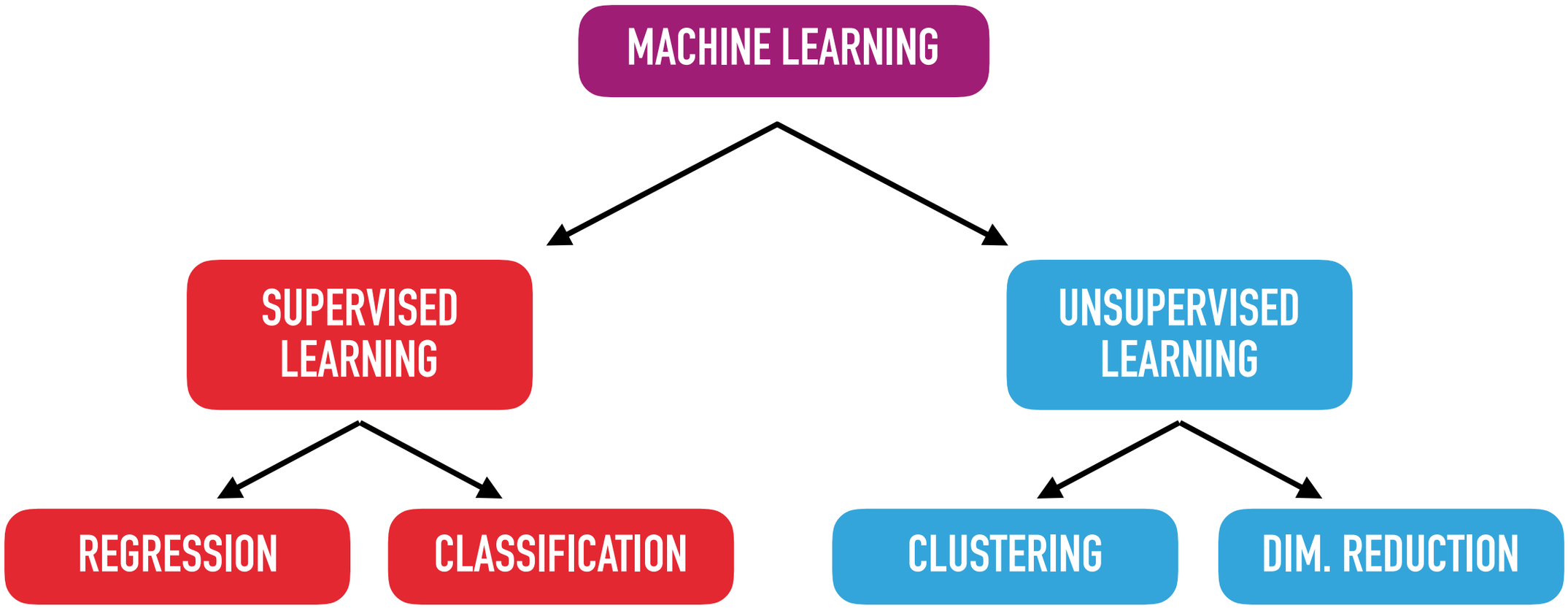
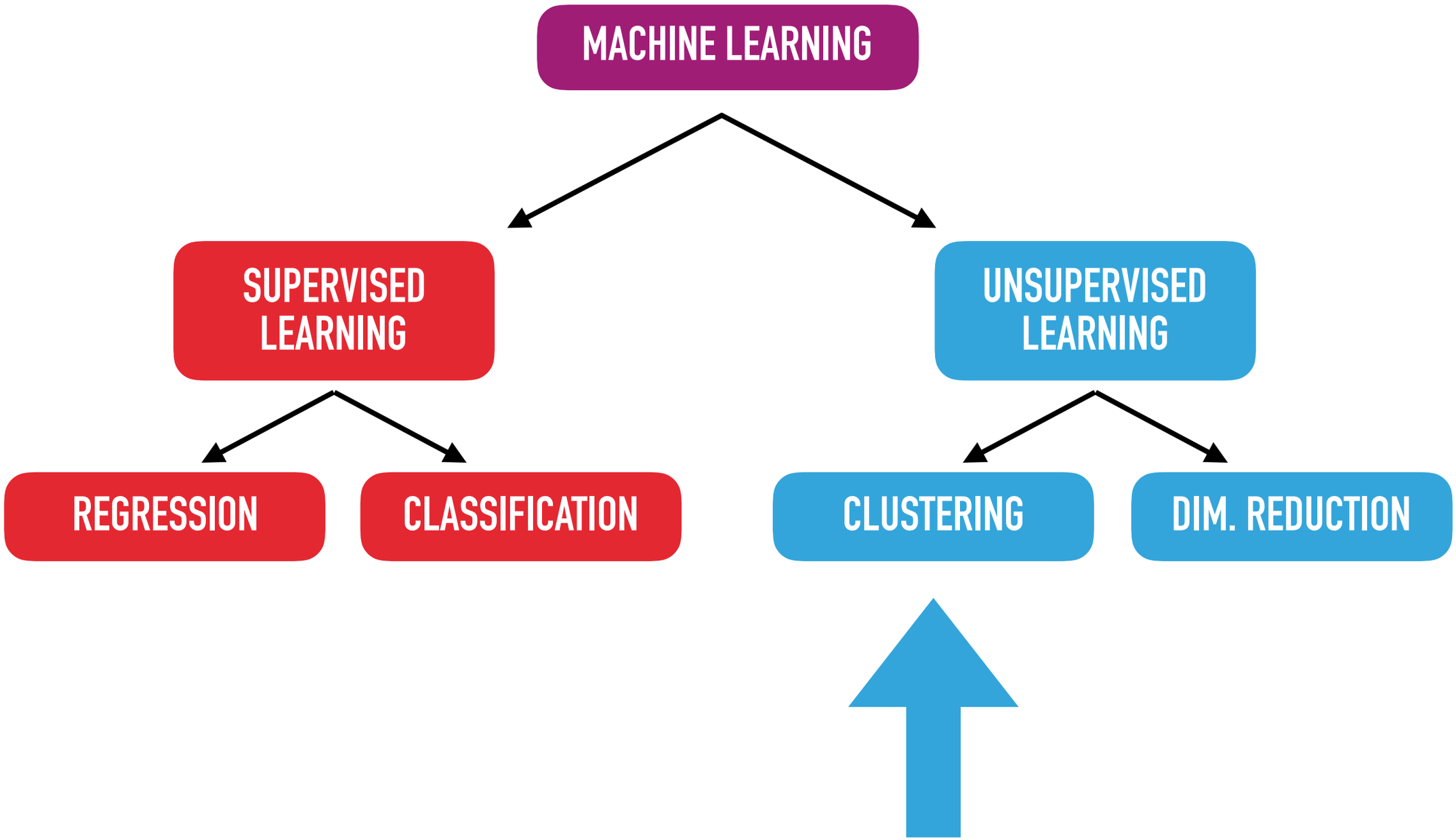
K-MEANS CLUSTERING

KEEP AN EYE OUT FOR THESE TOPICS

- Difference between supervised and unsupervised learning
- What is clustering? Why does it matter?
- How does k-Means work?
- What are some challenges of k-Means?


MACHINE LEARNING
SUPERVISED LEARNING
REGRESSION
CLASSIFICATION


MACHINE LEARNING
SUPERVISED LEARNING
UNSUPERVISED LEARNING
REGRESSION
CLASSIFICATION
CLUSTERING
DIM. REDUCTION


MACHINE LEARNING
SUPERVISED LEARNING
UNSUPERVISED LEARNING
REGRESSION
CLASSIFICATION
CLUSTERING
DIM. REDUCTION

HOW MANY CLUSTERS?

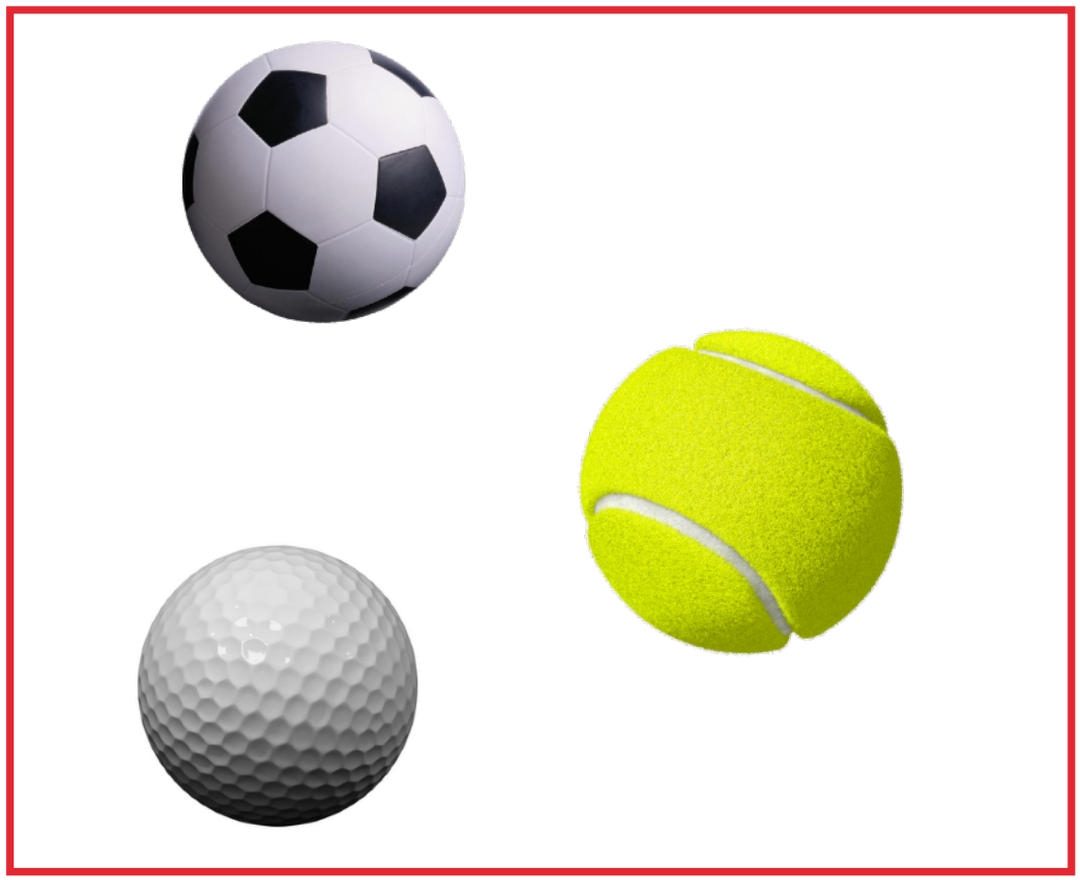
HOW MANY CLUSTERS?

By Location.

HOW MANY CLUSTERS?

By Color.

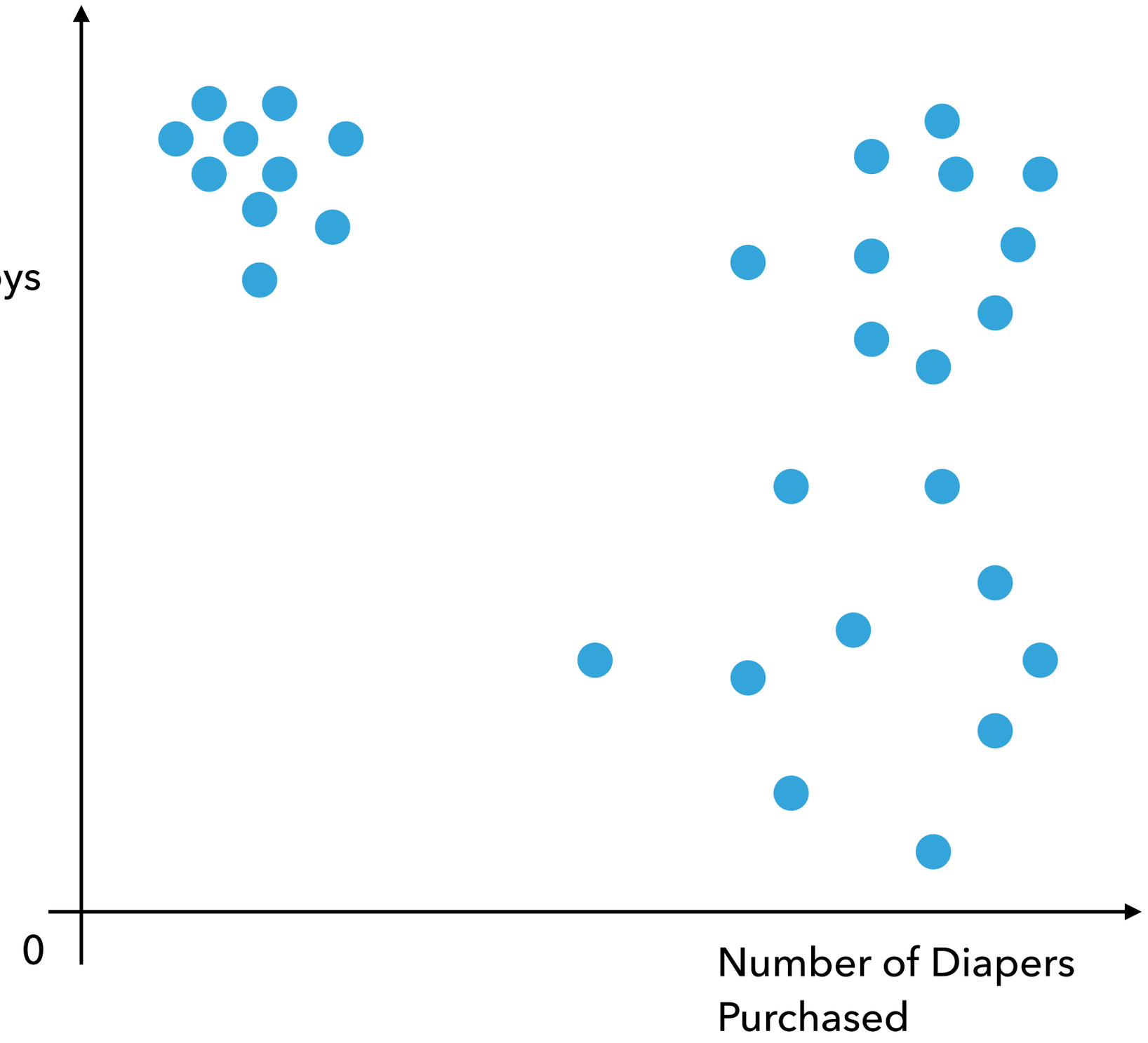
HOW MANY CLUSTERS?



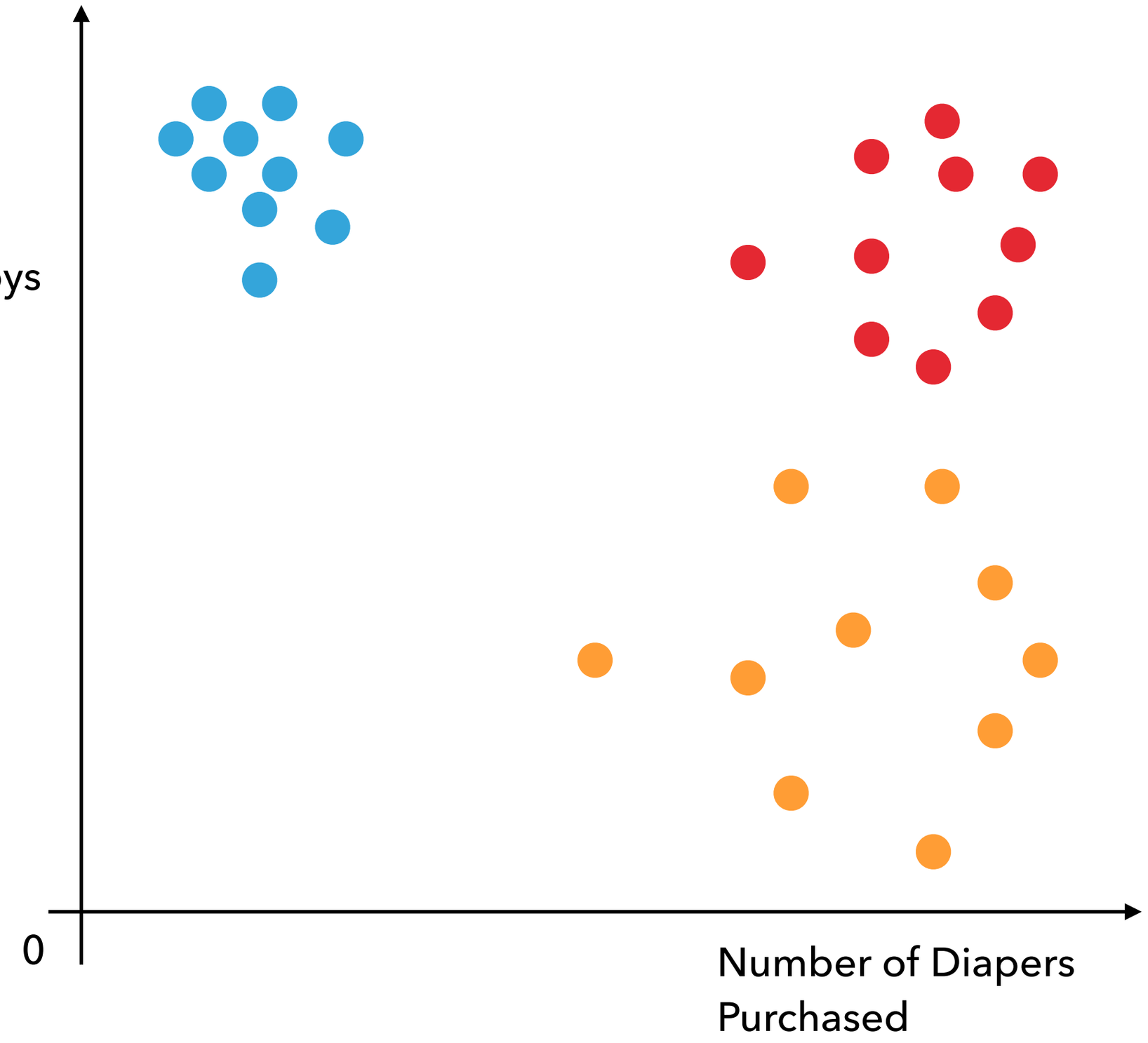
By Shape.

We want an algorithm that can take information about a set and decide which should be grouped together.

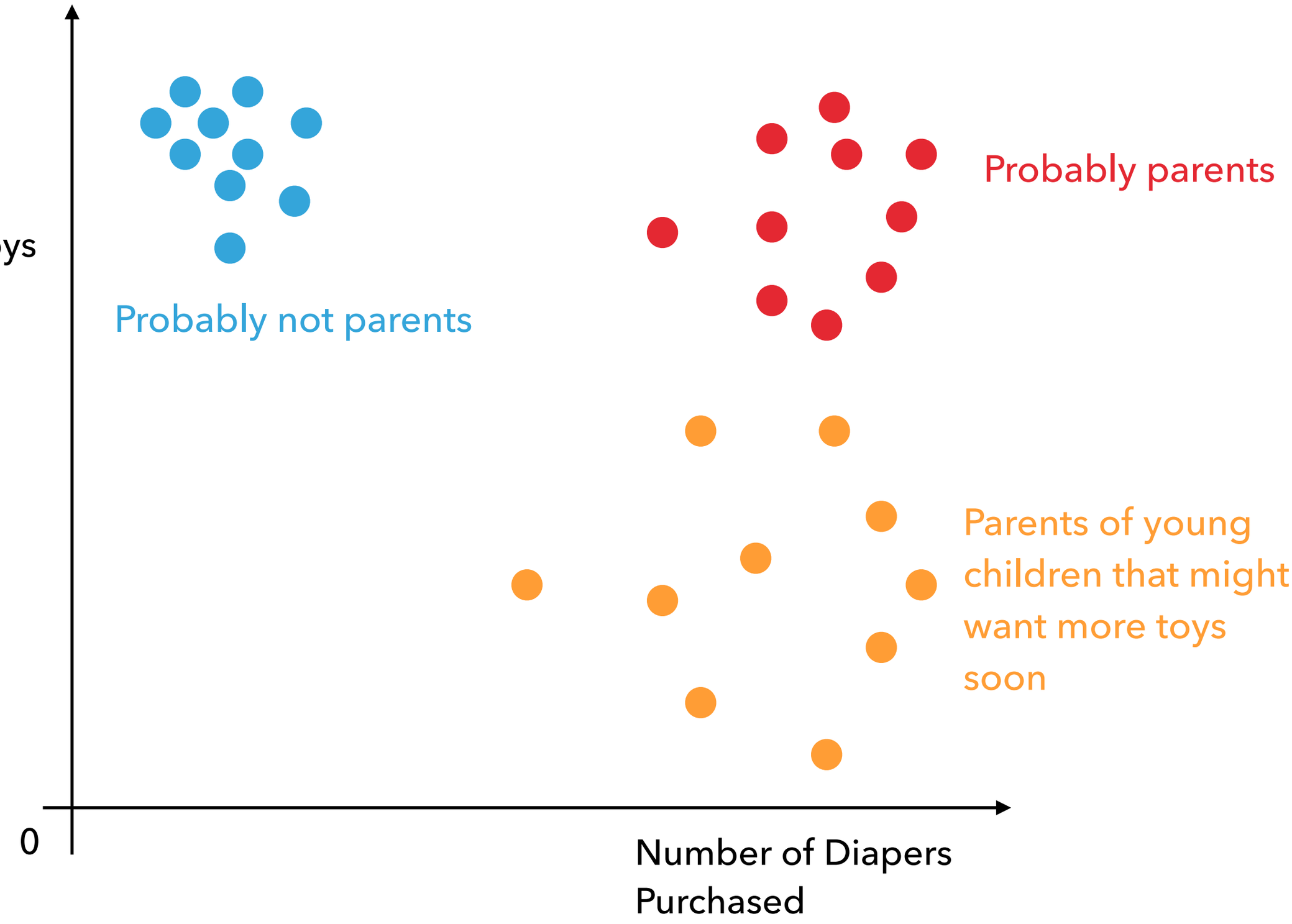
No true labels - so how can we make marketing decisions?



No true labels - so how can we make marketing decisions?



No true labels - so how can we make marketing decisions?



Several options for how to approach the problem

- Partition the space (draw lines between groups)
- Merge together things that are most alike
- Look for high density groupings of data

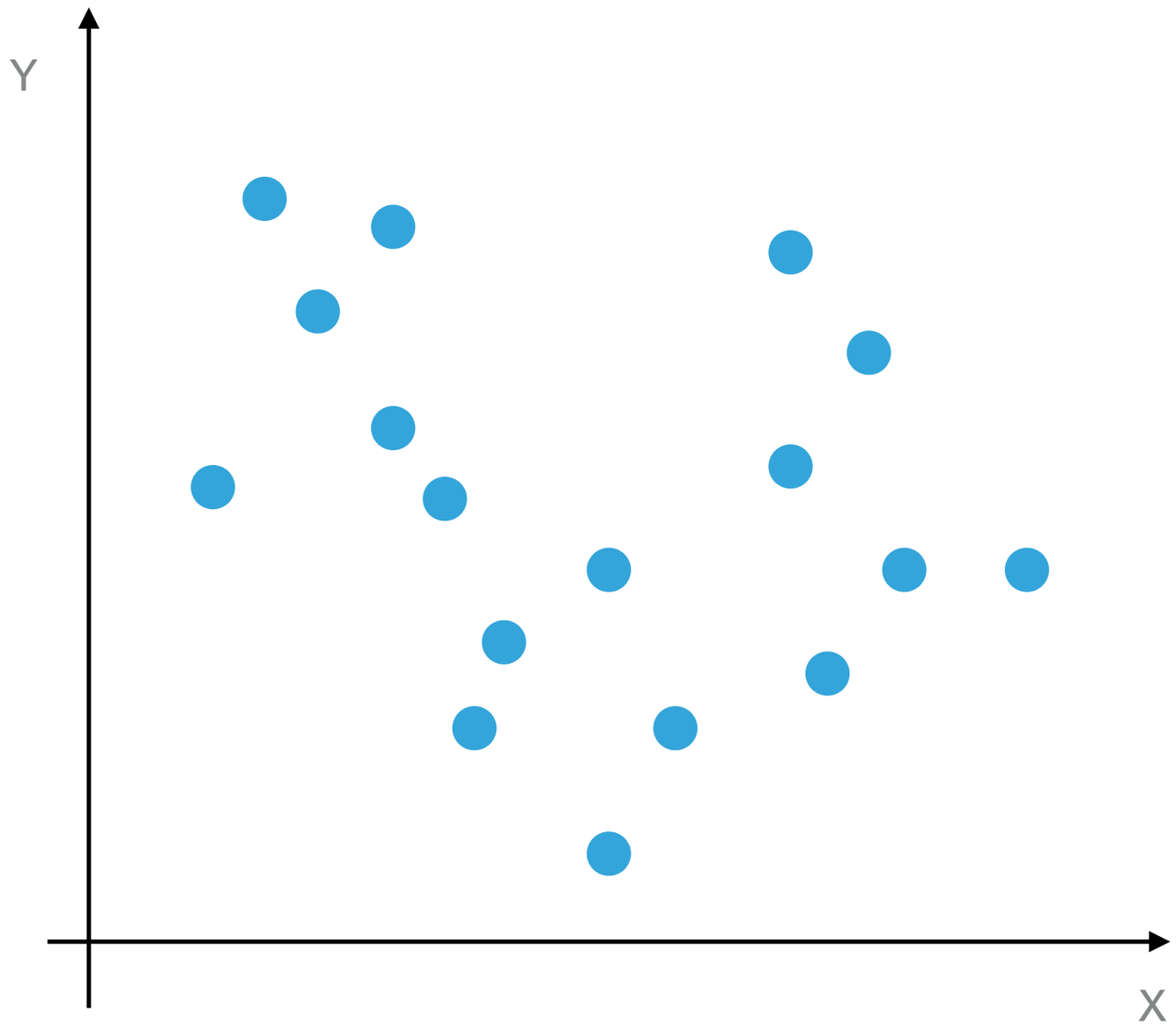
Several options for how to approach the problem

- Partition the space (draw lines between groups)
- Merge together things that are most alike
- Look for high density groupings of data

Our focus. Today, we'll learn about a method called ***k-Means***.

Let's visualize the algorithm:

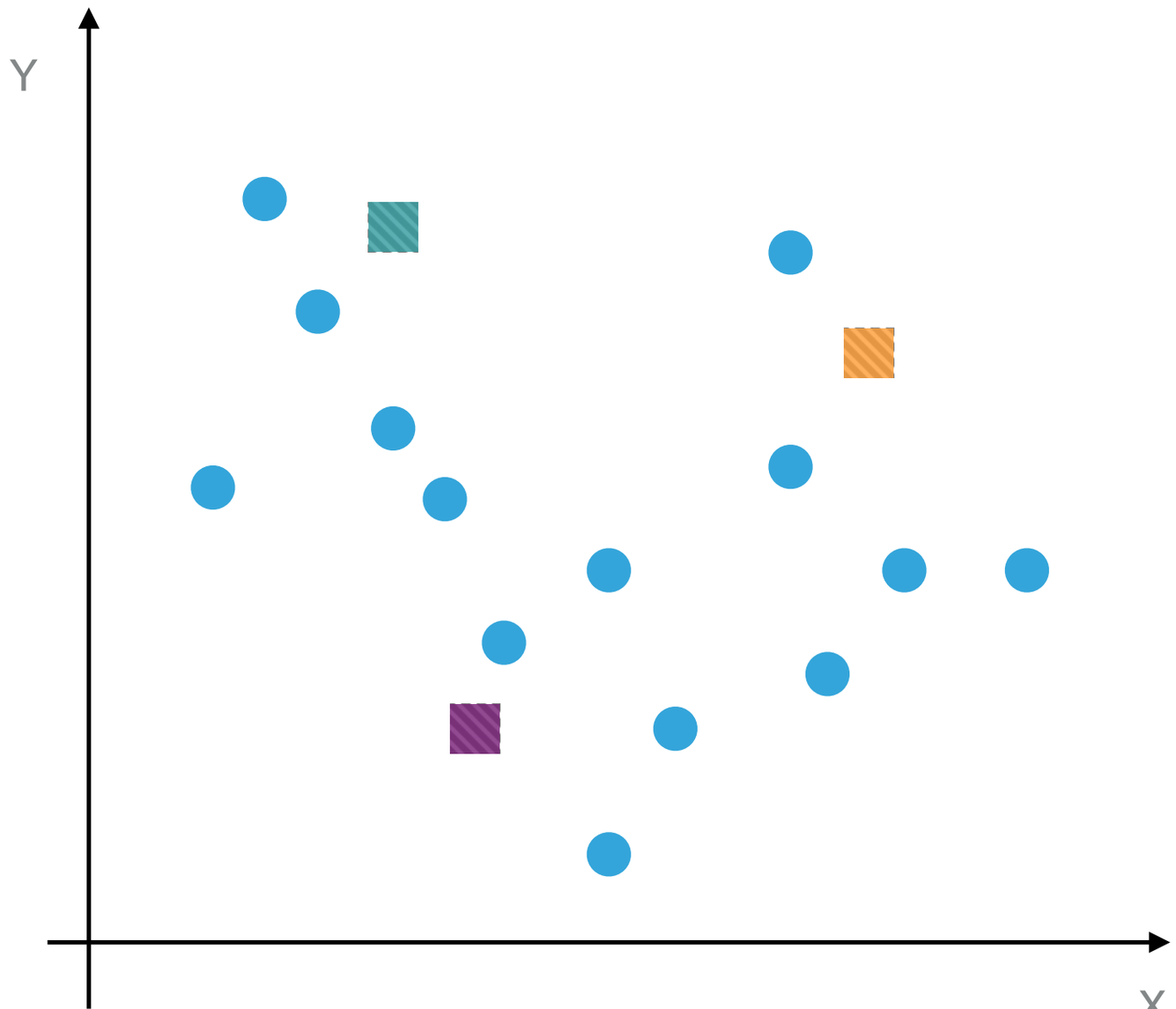
STAGE 1: INITIALIZATION



To initialize the algorithm, we need to choose the number of clusters. For this example, let's say k = 3. (more on choosing k later).

Let's visualize the algorithm:

STAGE 1: INITIALIZATION



We randomly choose members of the set to start the clusters. This is called Forgy initialization. There are smarter ways to do this, but we can get to that later.

Notation:

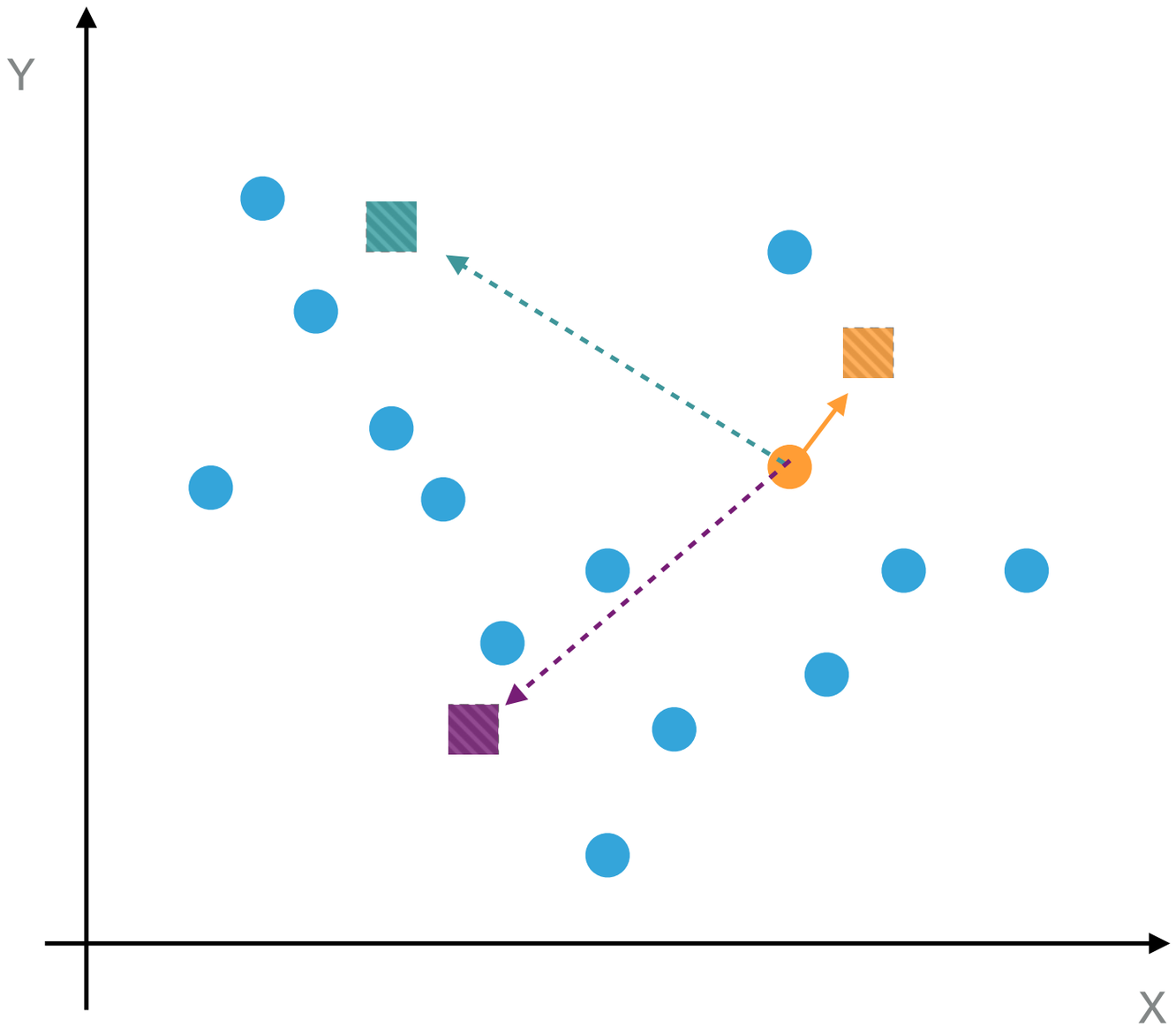
= Cluster Mean/Seed (midpoint of all the data)

= Data point in the set.

Colors represent different clusters.

Let's visualize the algorithm:

STAGE 1: INITIALIZATION

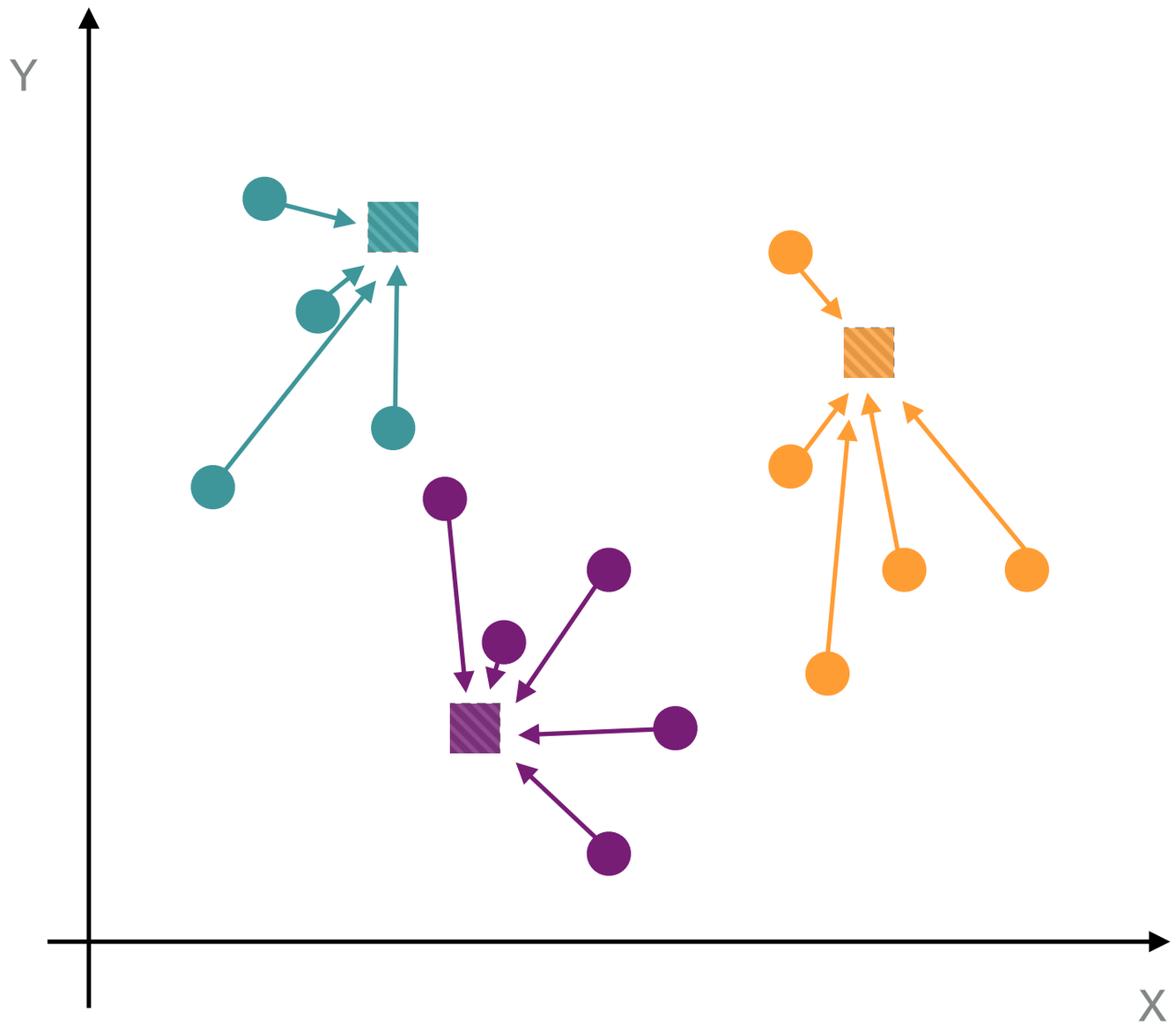


Now we compute the distance from each point to each mean and assign the point to the cluster of the closest seed.

$$\sum_{i=1}^{N} \sqrt{(x_i - \mu)^2}$$

Let's visualize the algorithm:

STAGE 1: INITIALIZATION

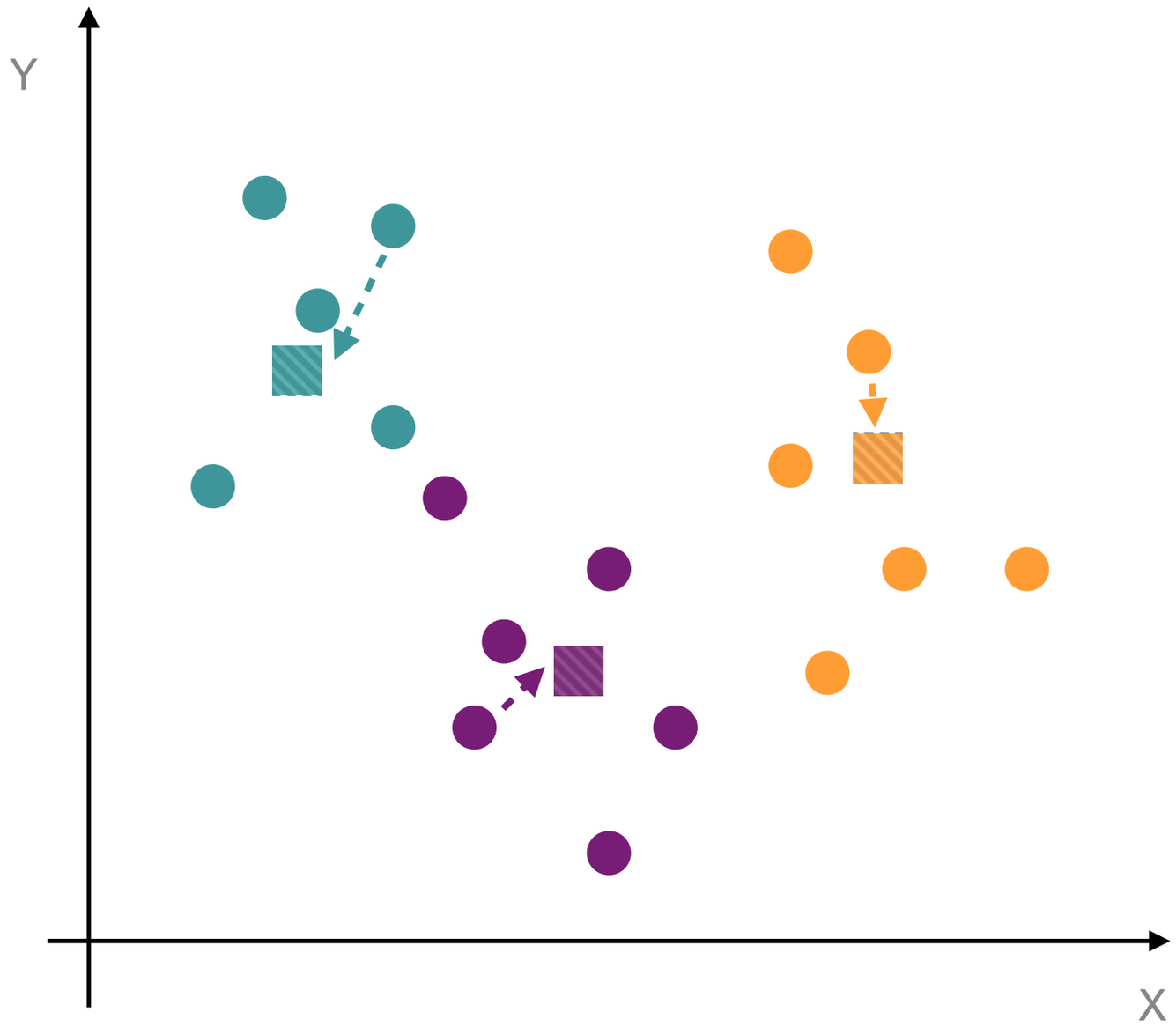


Now we compute the distance from each point to each mean and assign the point to the cluster of the closest seed.

$$\sum_{i=1}^{N} \sqrt{(x_i - \mu)^2}$$

Let's visualize the algorithm:

STAGE 1: INITIALIZATION

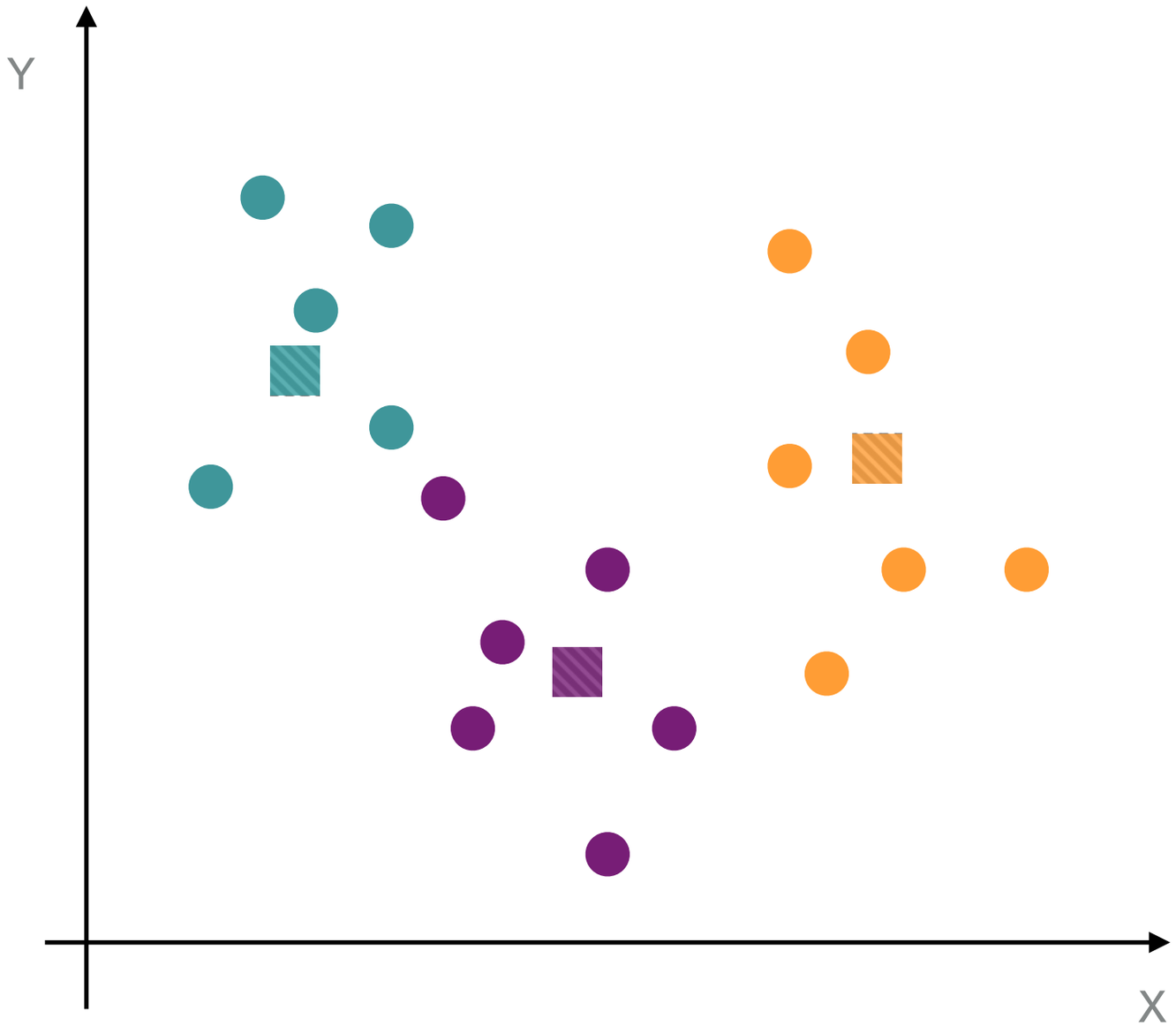


Finally, compute the mean position for each cluster and reassign the cluster seed to that point.

Now we've initialized all 3 clusters.

Let's visualize the algorithm:

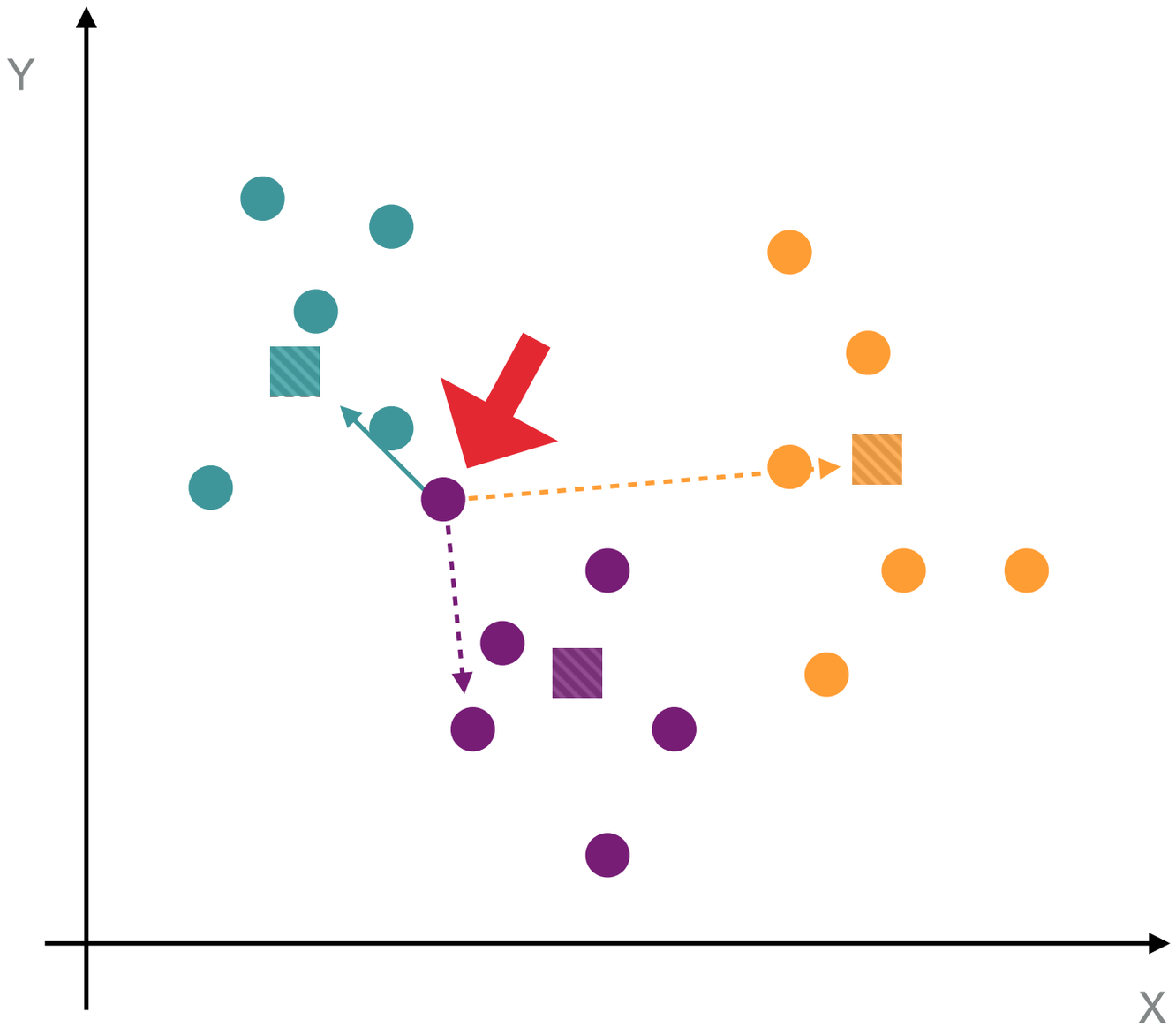
STAGE 2: MINIMIZATION



Our cluster seeds have moved. So now some of the points are closer to a different cluster seed than they used to be. Repeat the process!

Let's visualize the algorithm:

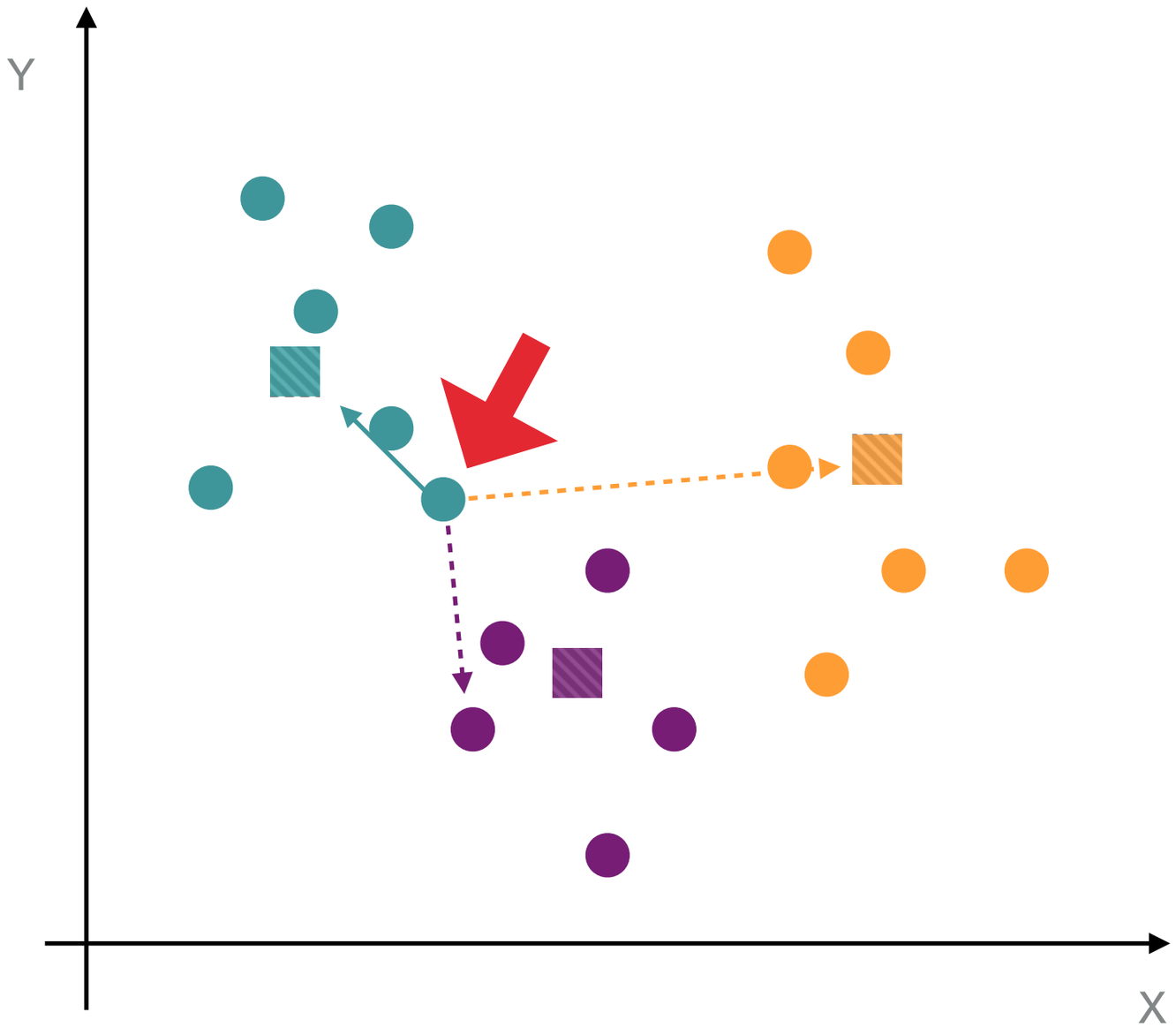
STAGE 2: MINIMIZATION



Our cluster seeds have moved. So now some of the points are closer to a different cluster seed than they used to be. Repeat the process!

Let's visualize the algorithm:

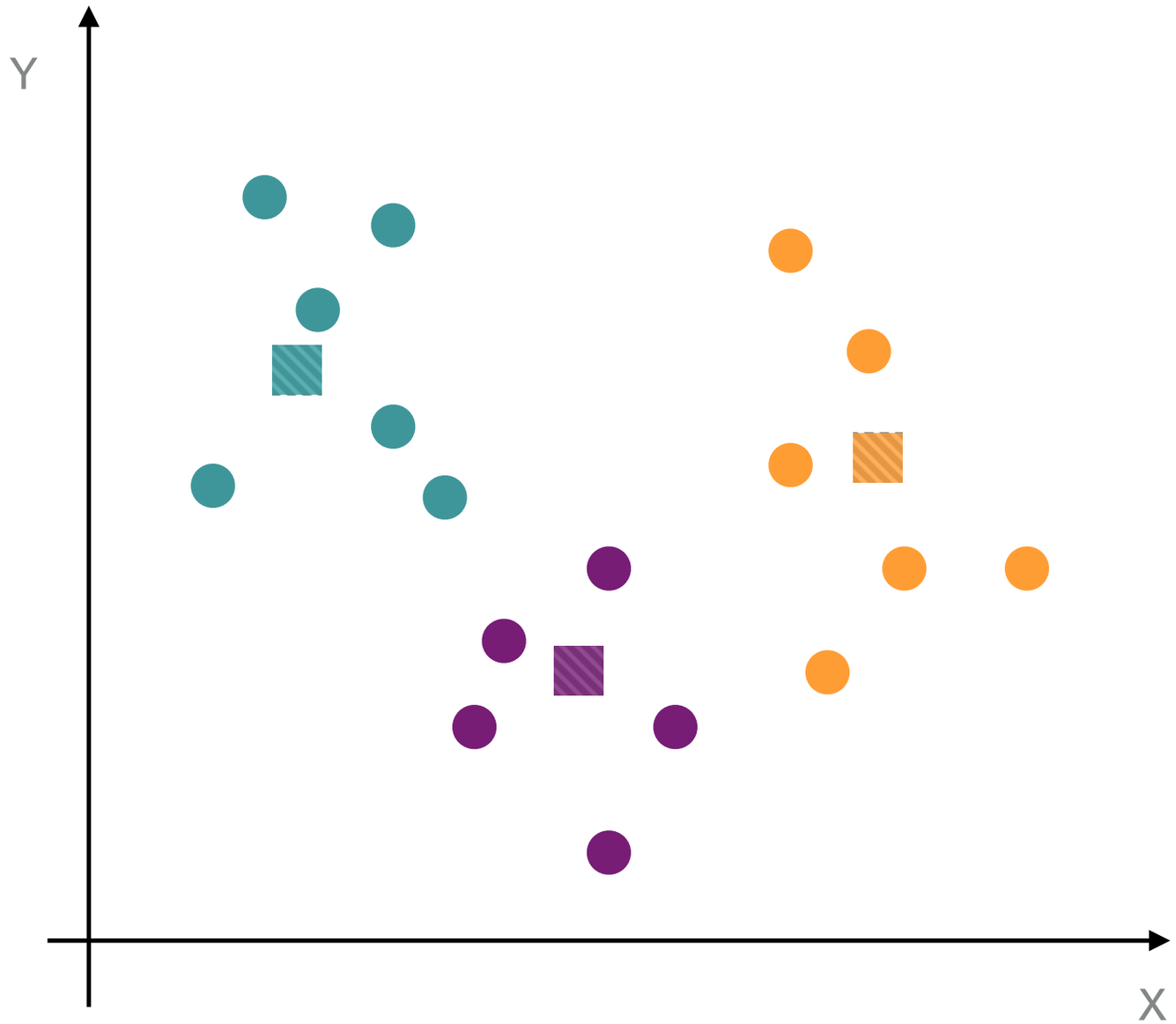
STAGE 2: MINIMIZATION



Our cluster seeds have moved. So now some of the points are closer to a different cluster seed than they used to be. Repeat the process!

Let's visualize the algorithm:

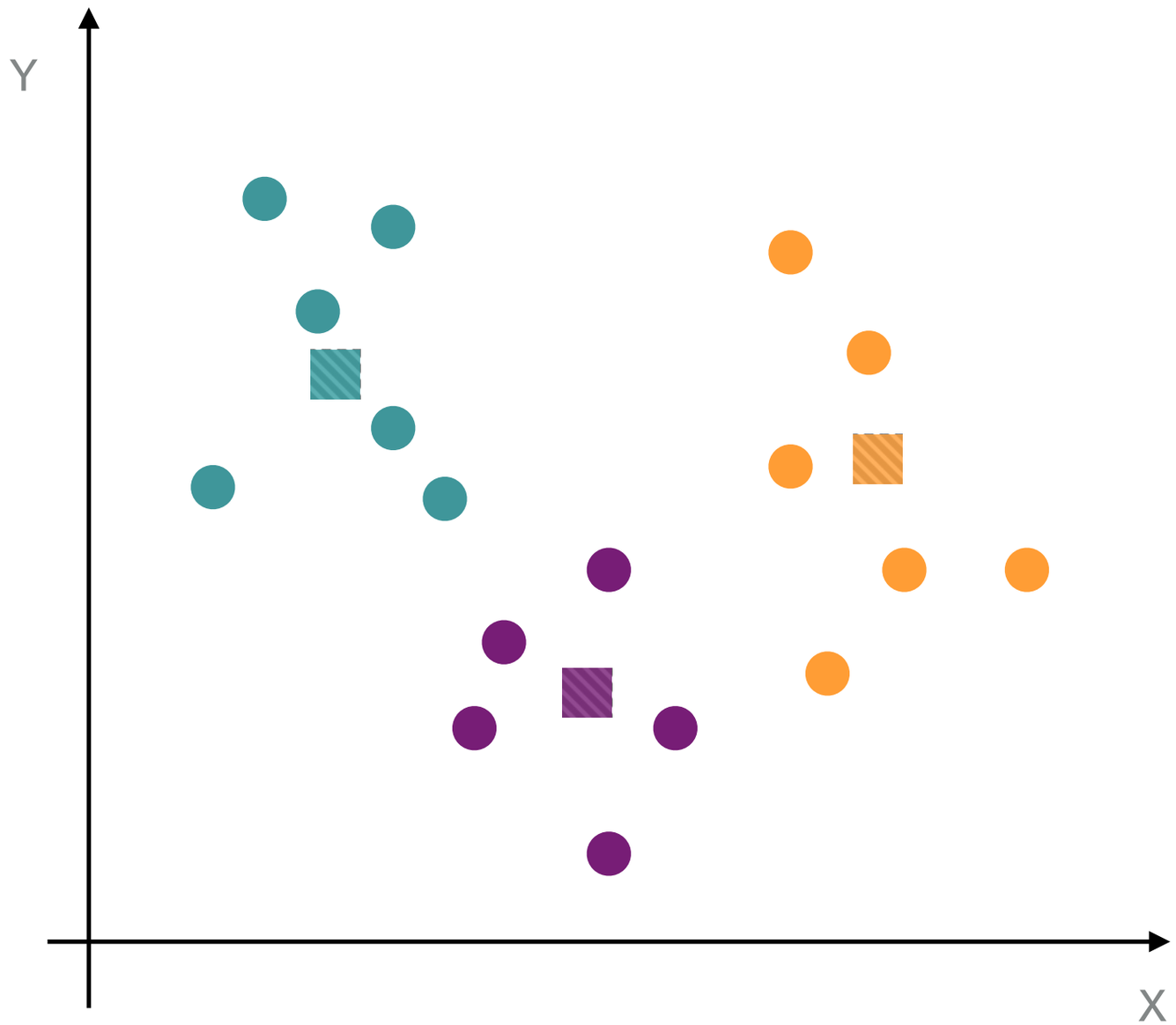
STAGE 2: MINIMIZATION



Our cluster seeds have moved. So now some of the points are closer to a different cluster seed than they used to be. Repeat the process!

Let's visualize the algorithm:

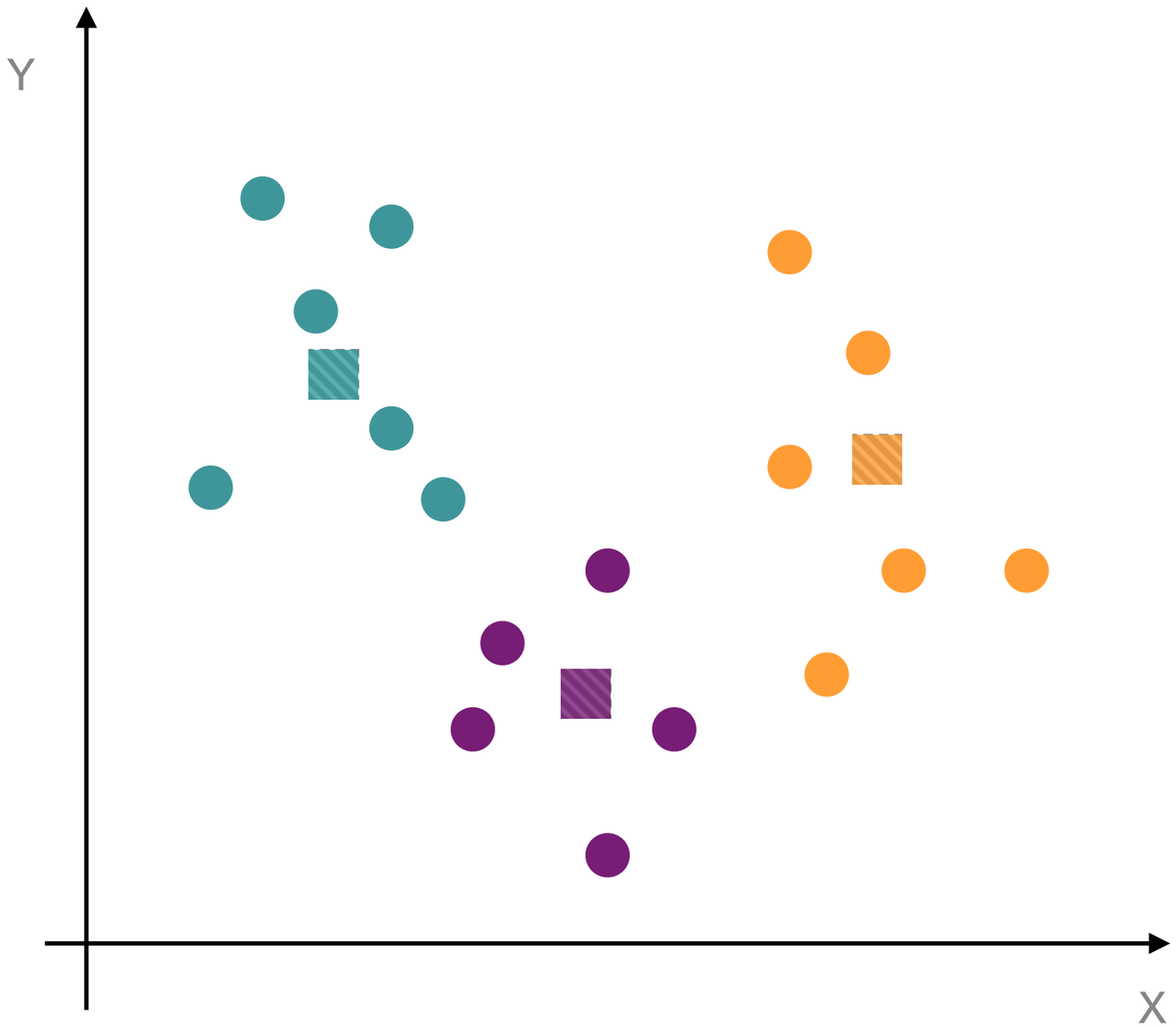
STAGE 2: MINIMIZATION



Now that a point has switched clusters, the means aren't correct any more. Update the means, and try again.

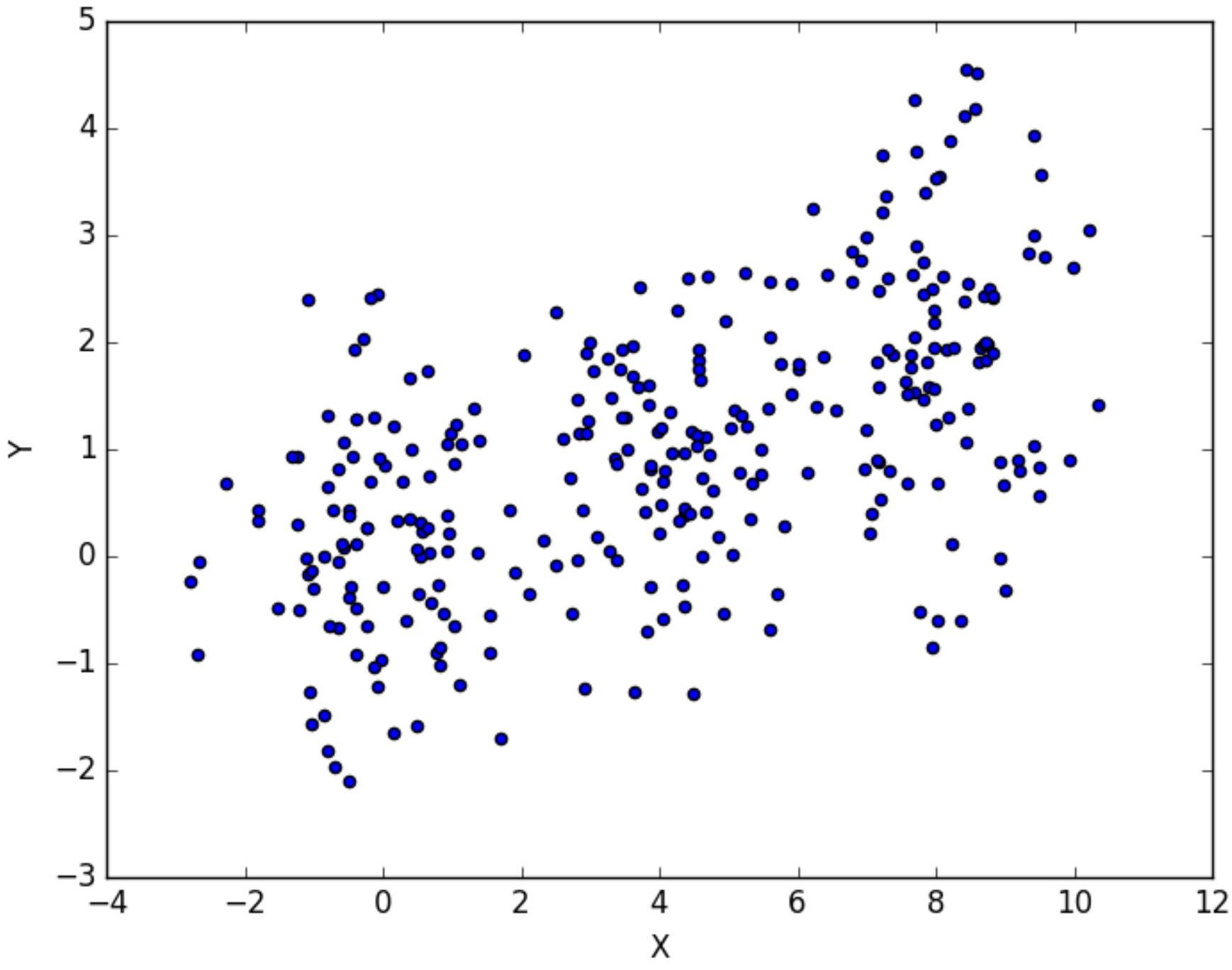
Let's visualize the algorithm:

STAGE 2: MINIMIZATION

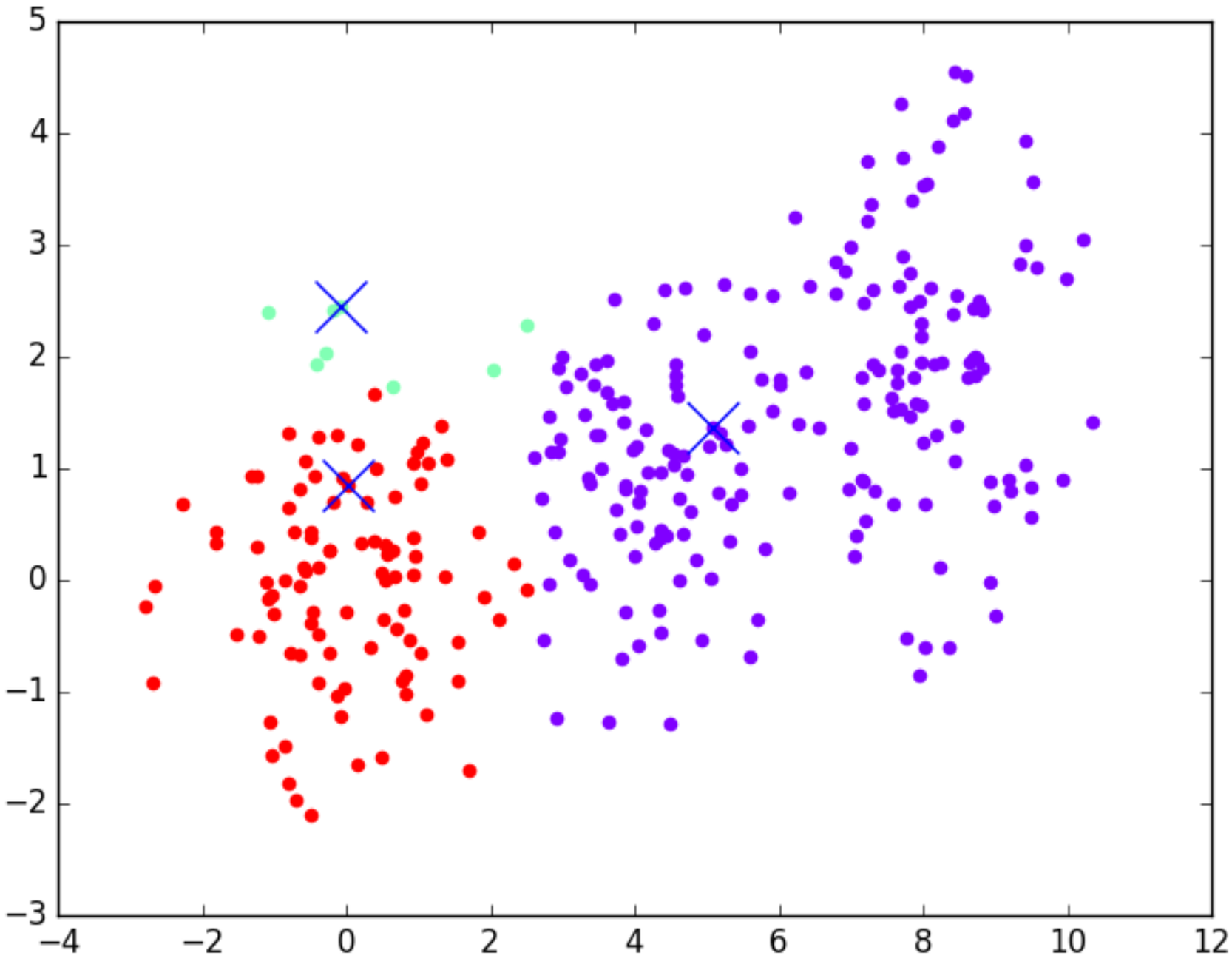


We iterate this step until we reach a stage where no members are changing between clusters.

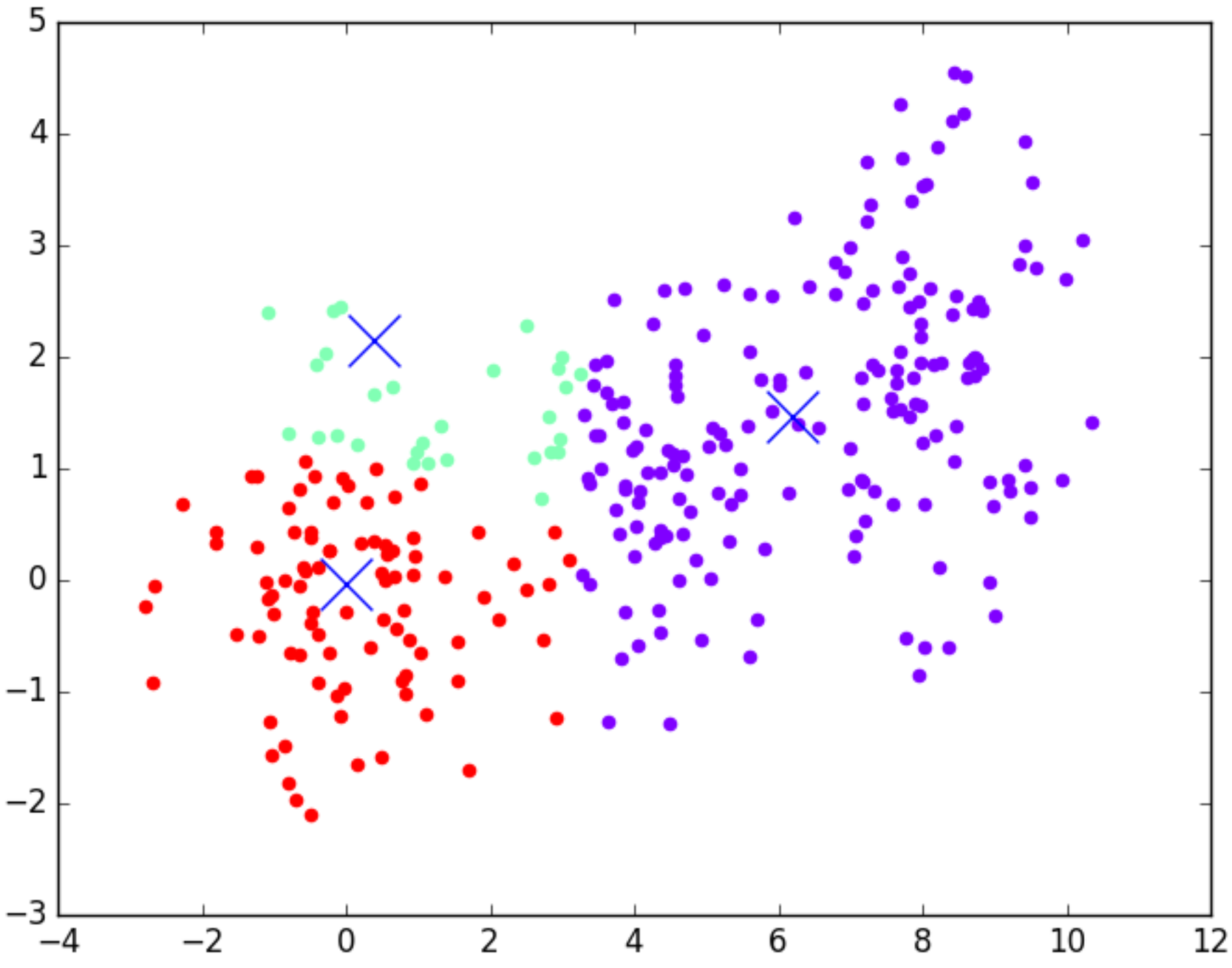
This convergence point is guaranteed to exist and k-means is generally fast at reaching it.



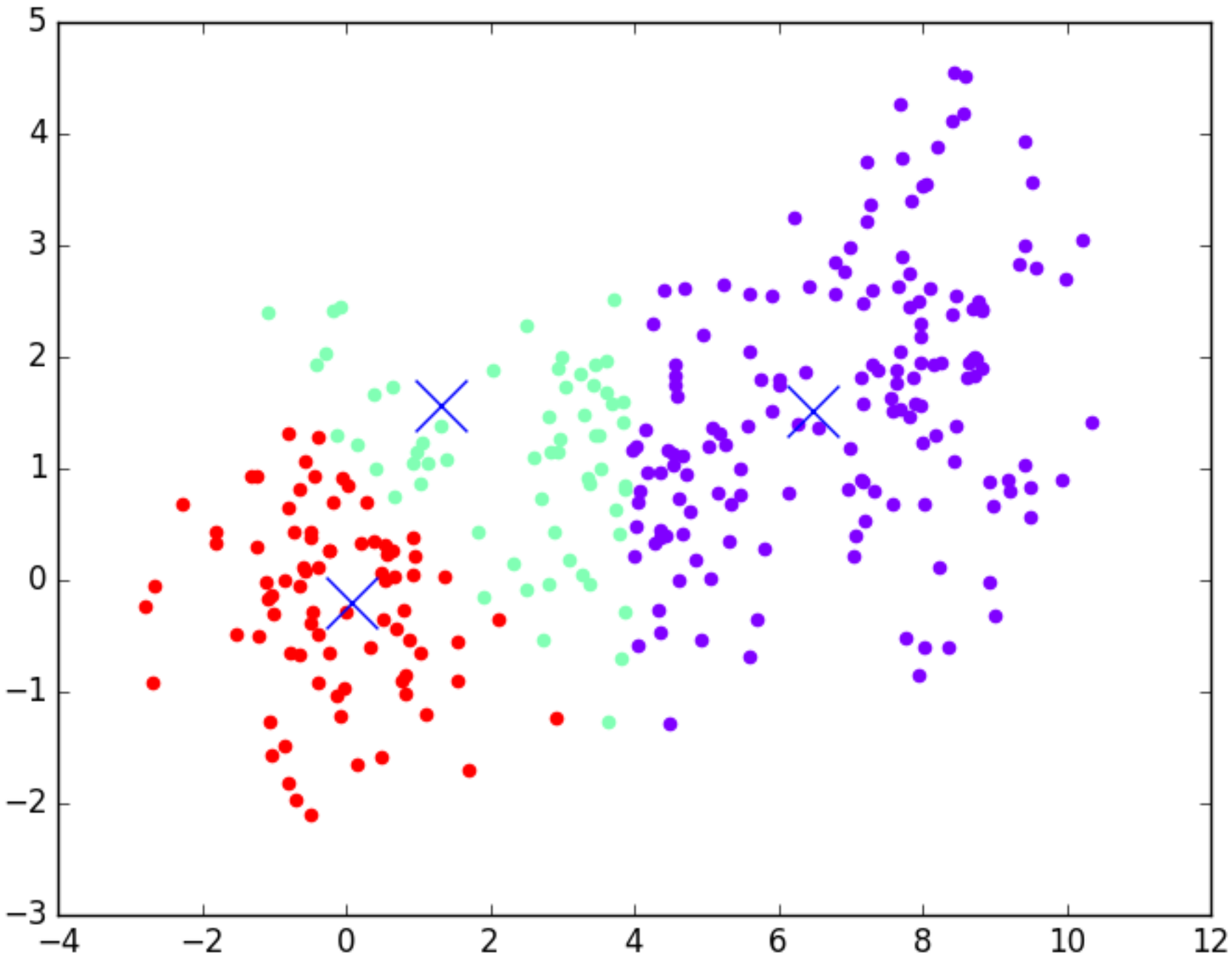
Note: You'll want to flip page to page, not scroll, to get the full effect.



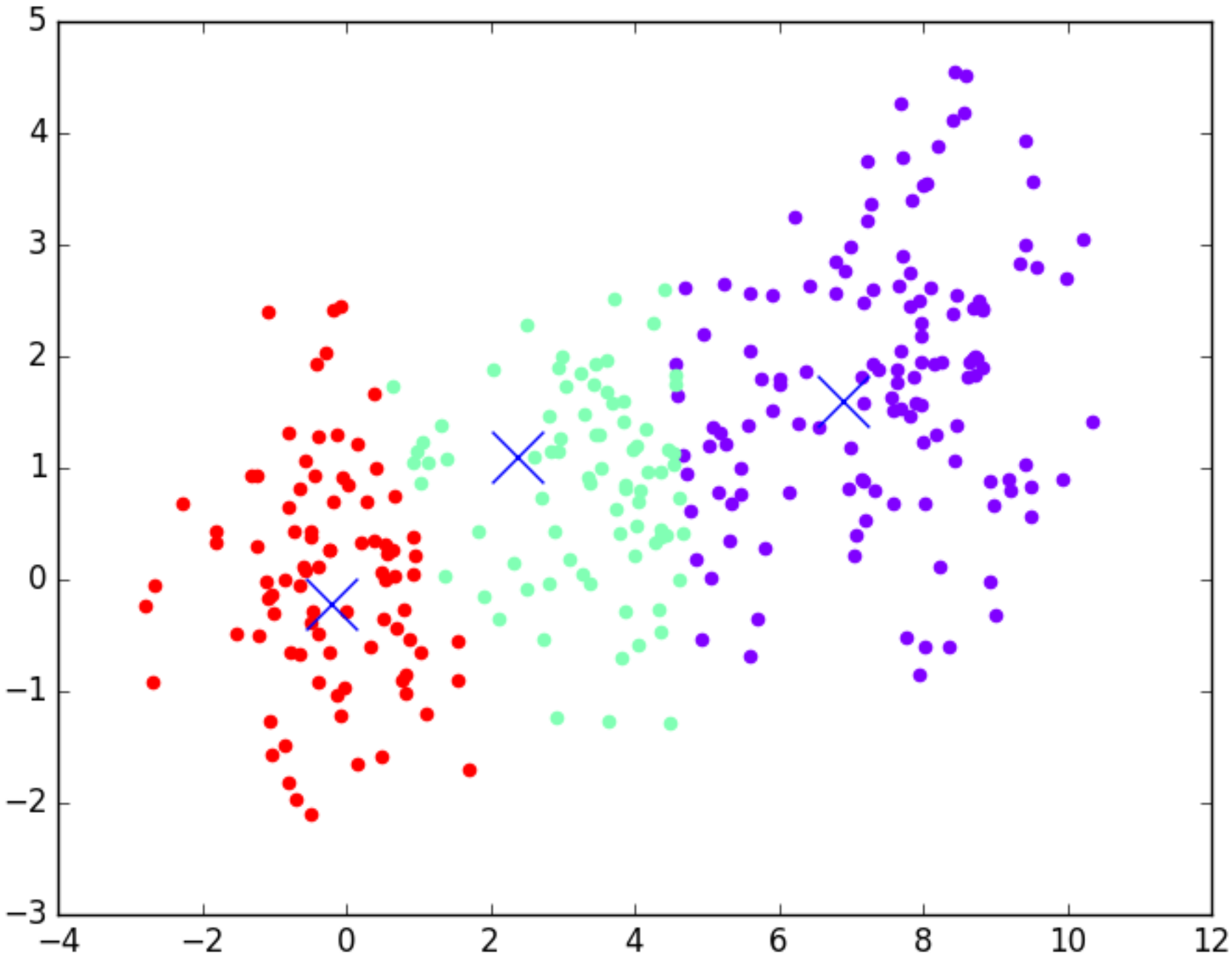
Note: You'll want to flip page to page, not scroll, to get the full effect.



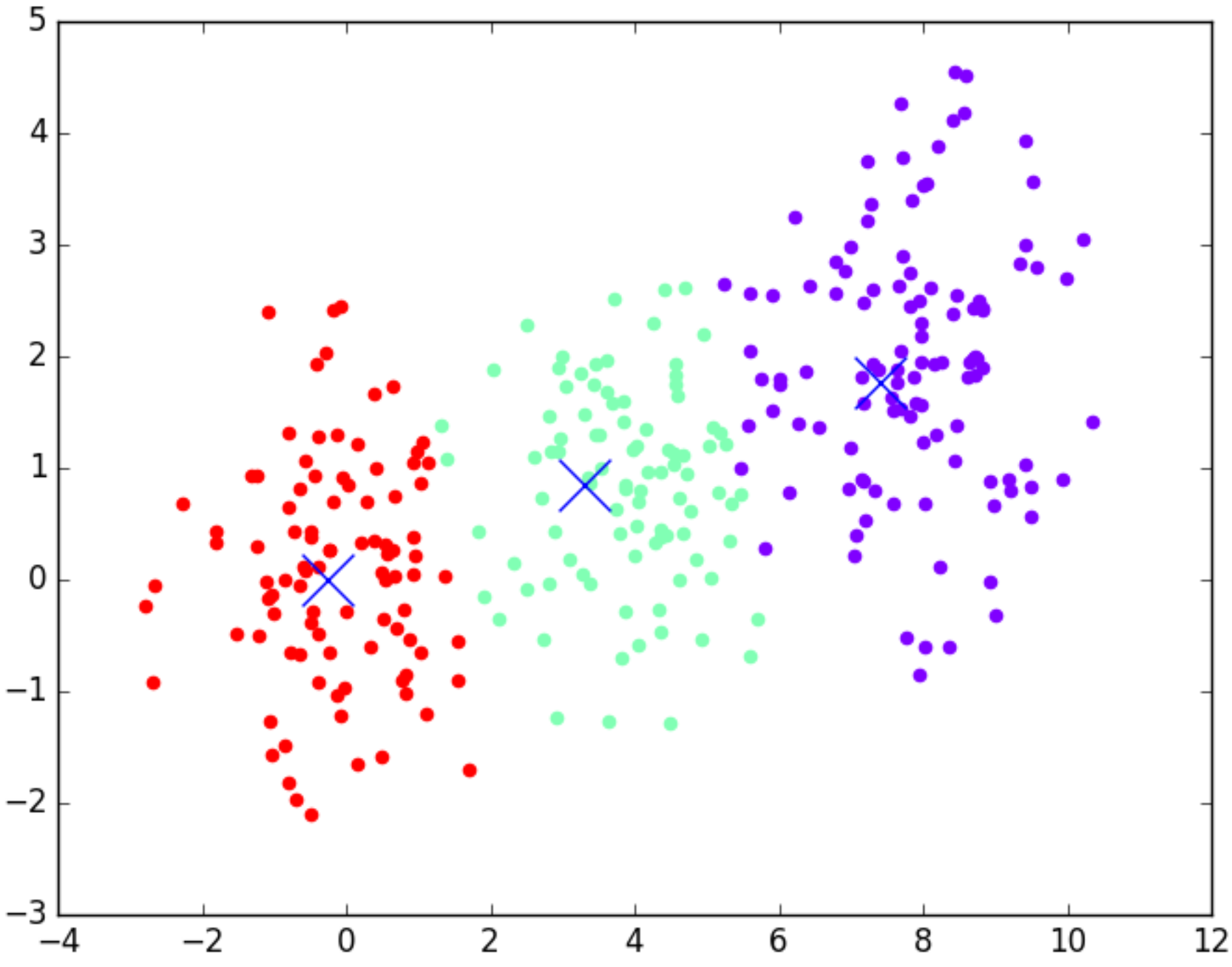
Note: You'll want to flip page to page, not scroll, to get the full effect.



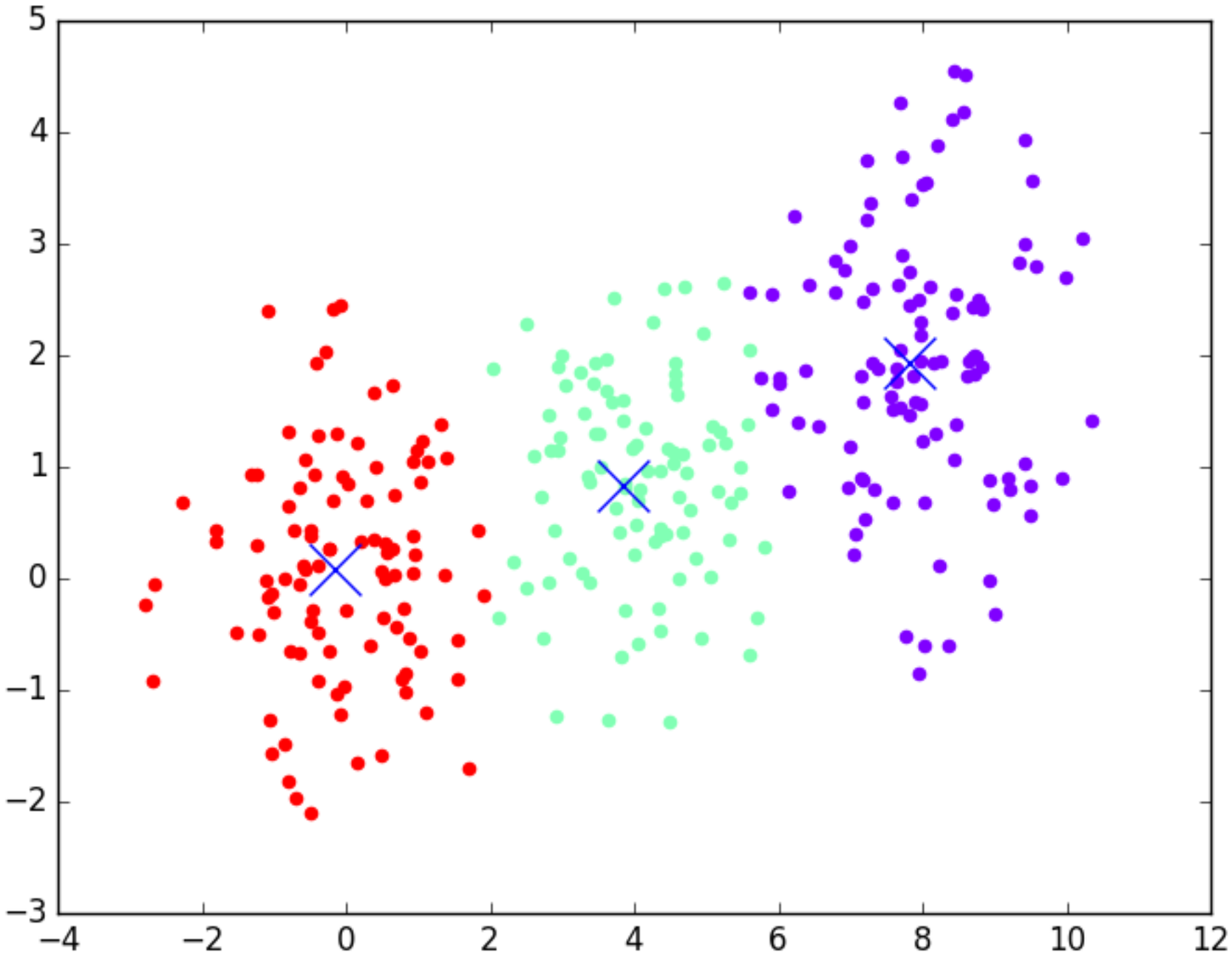
Note: You'll want to flip page to page, not scroll, to get the full effect.



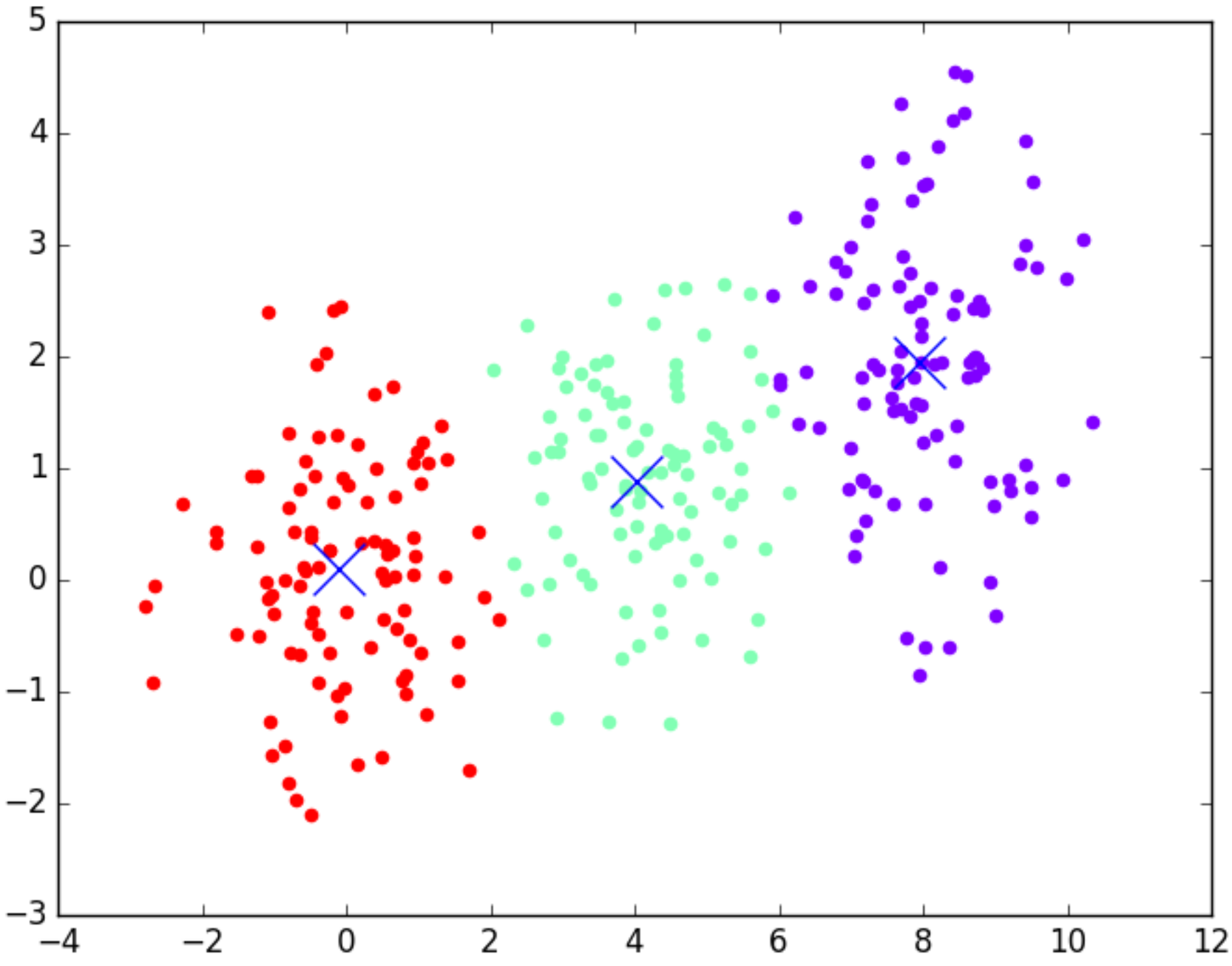
Note: You'll want to flip page to page, not scroll, to get the full effect.



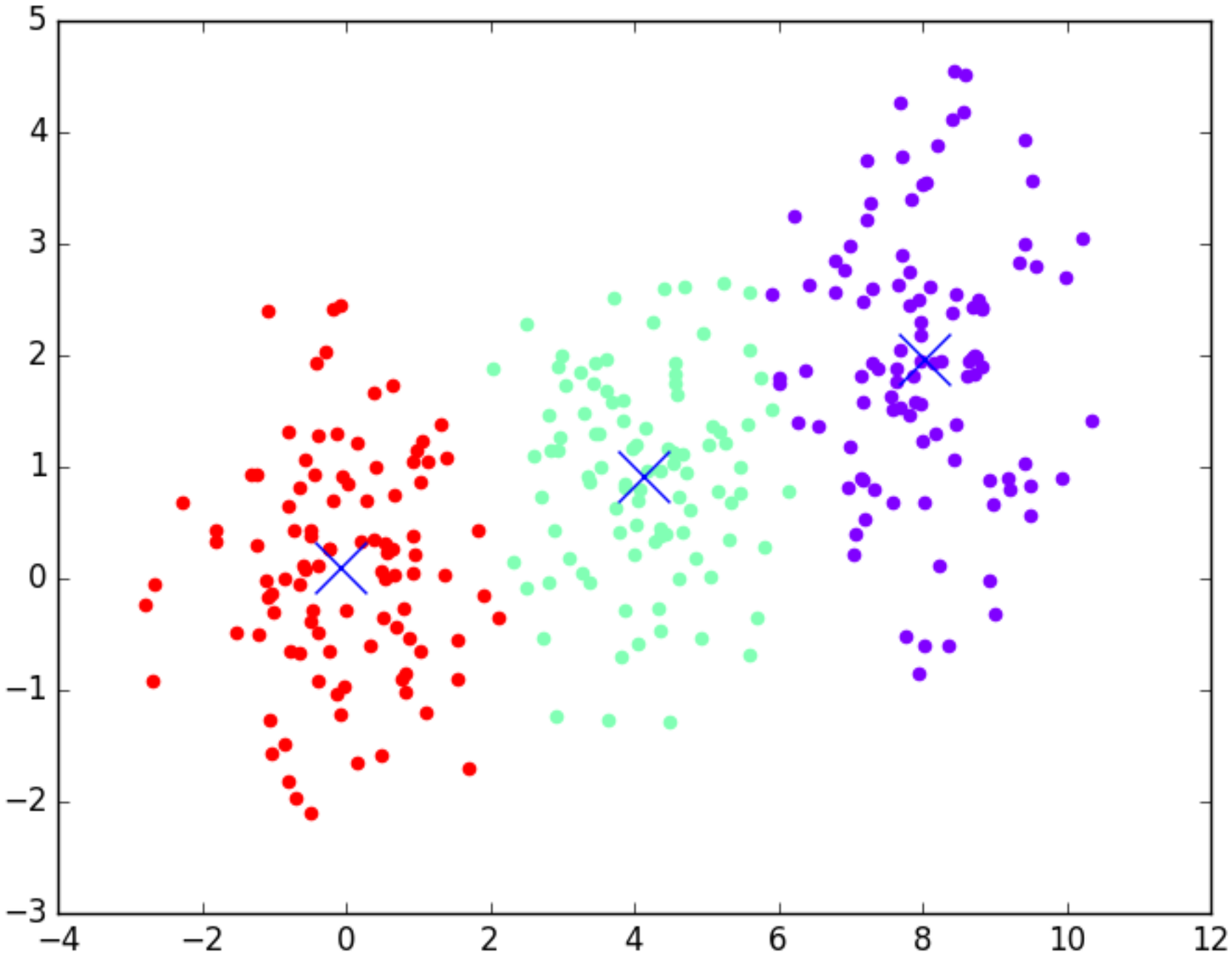
Note: You'll want to flip page to page, not scroll, to get the full effect.



Note: You'll want to flip page to page, not scroll, to get the full effect.



Note: You'll want to flip page to page, not scroll, to get the full effect.



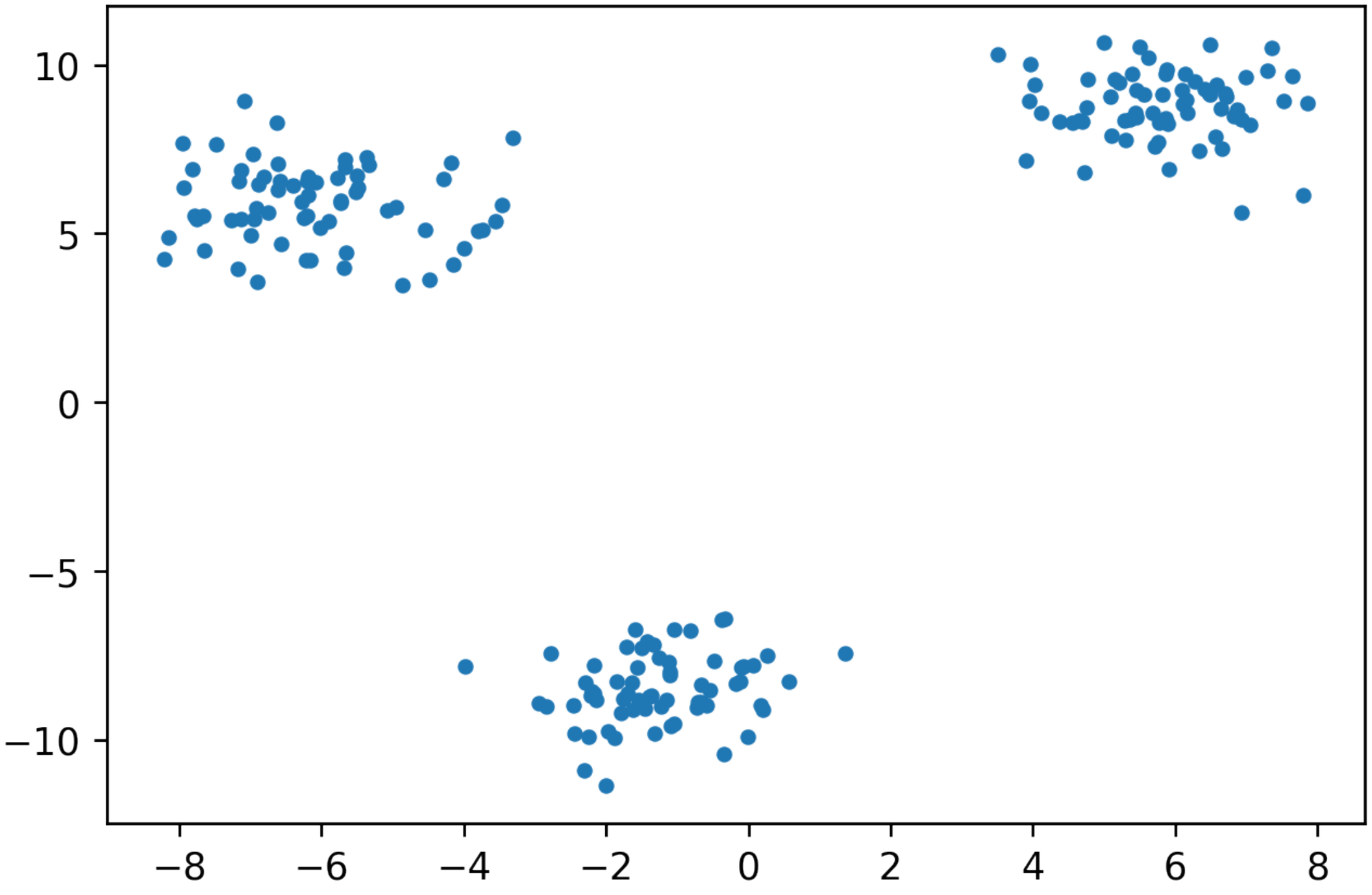
Note: You'll want to flip page to page, not scroll, to get the full effect.

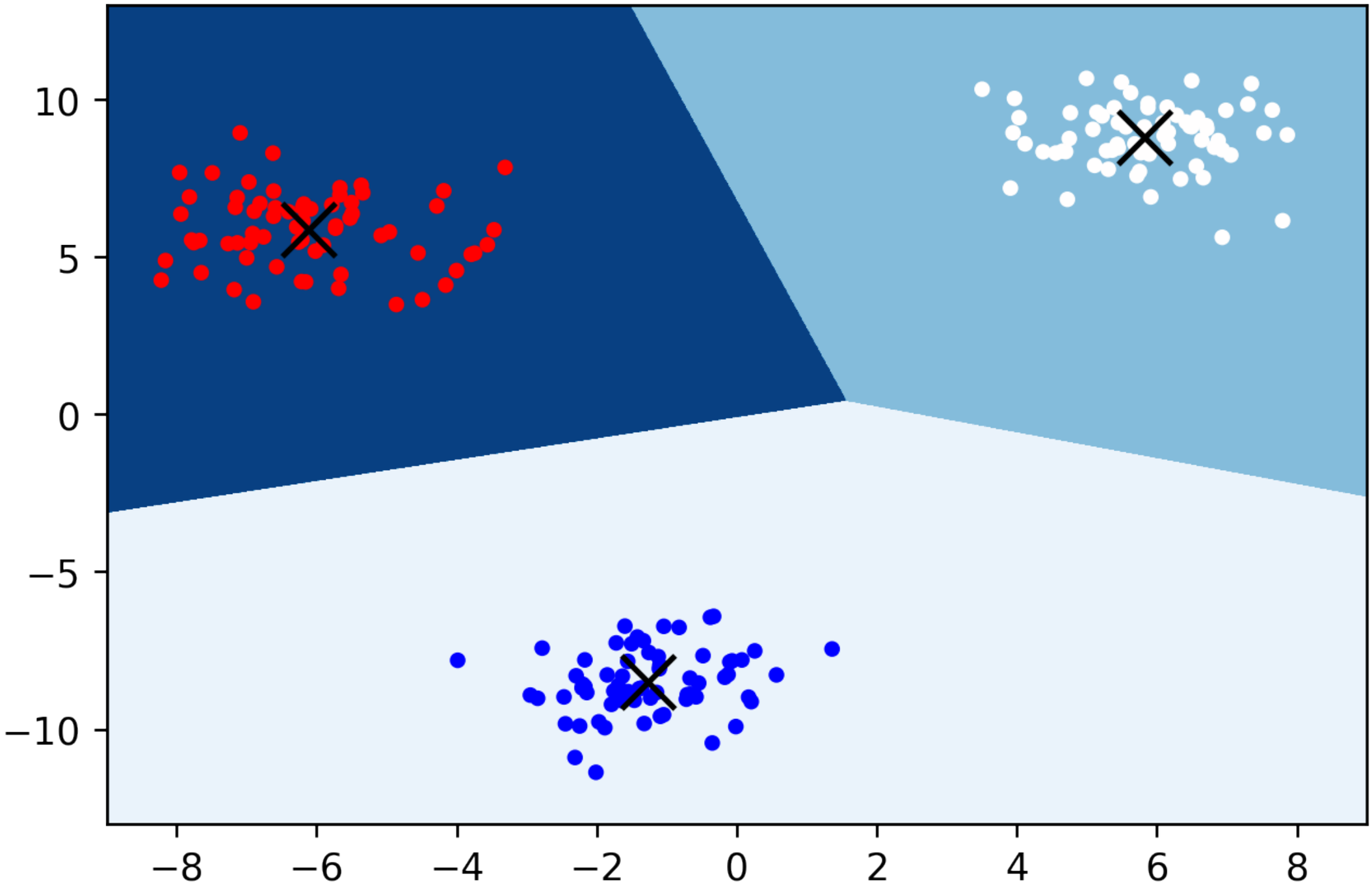
STAGE 1: INITIALIZATION

```
data = get_data("theseAreDataPoints.csv")
for i in range(0,k):
  clusters.append(new cluster)
  clusters[i].set_mean(random.choice(data))
for point in data:
  n = find_cluster(point, clusters)
  clusters[n].add_point(point)
  current_clustering.append(n)
for c in cluster:
  c.mean = compute_mean_position(c)
```

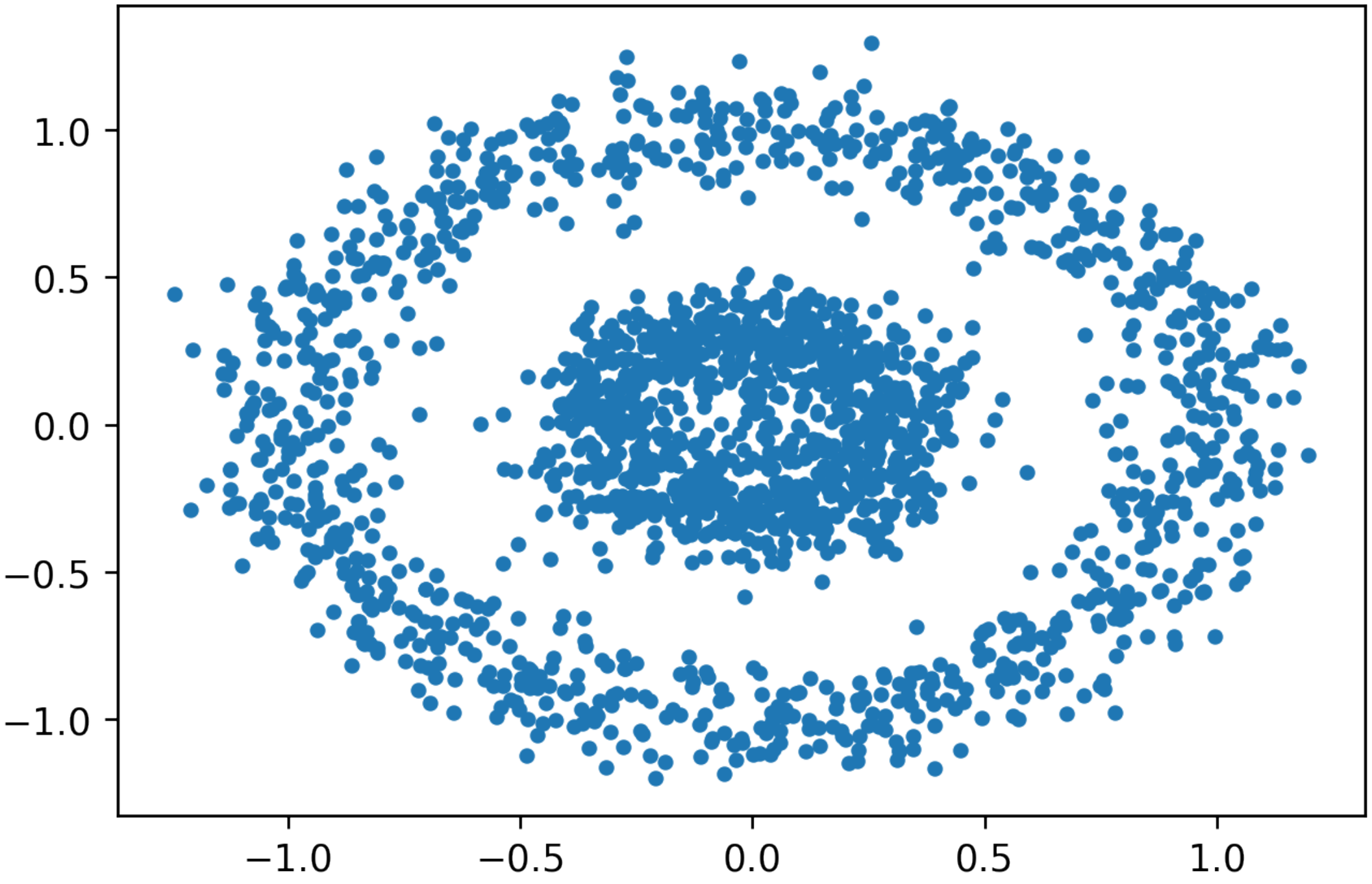
STAGE 2: MINIMIZATION

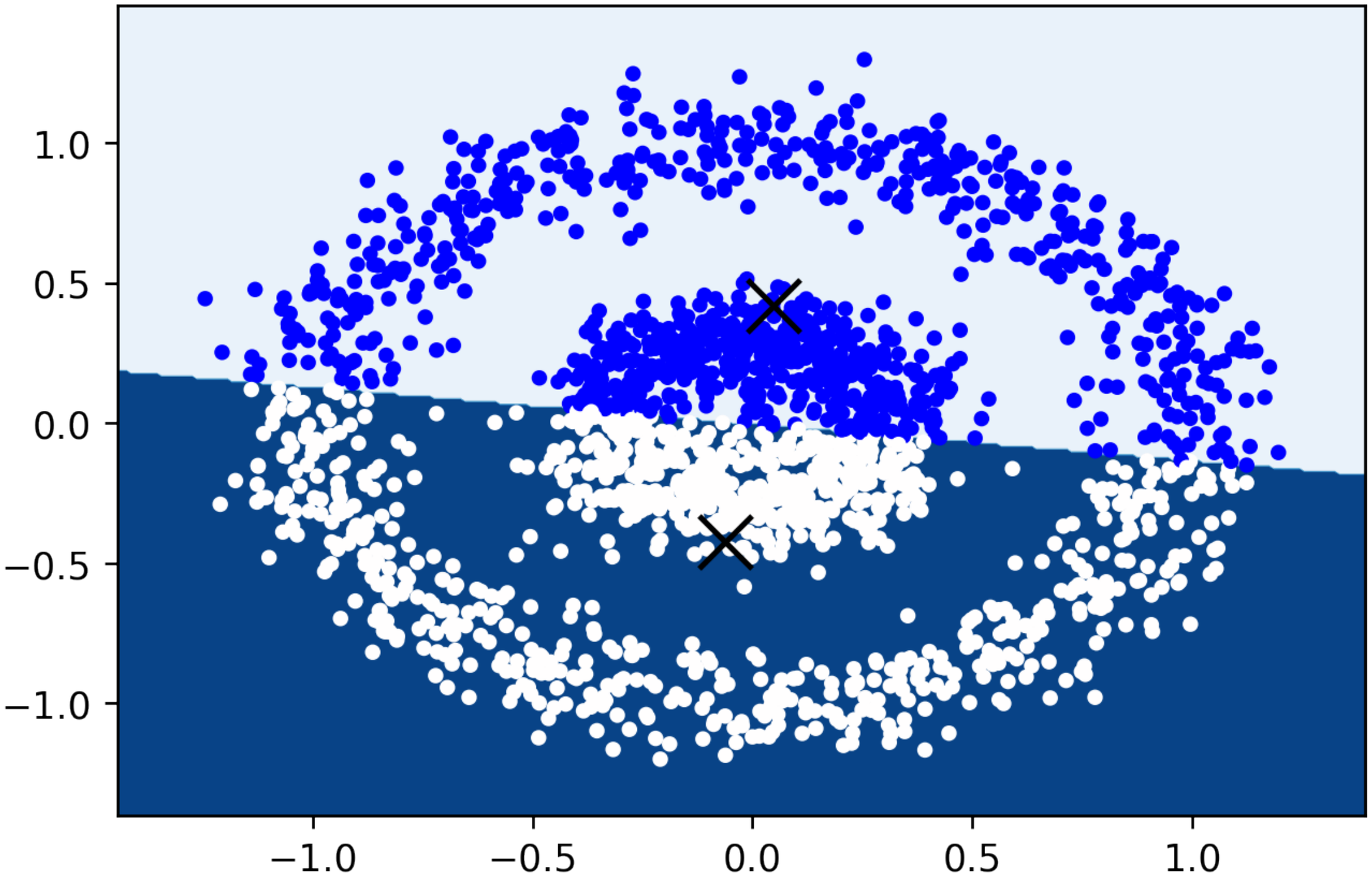
```
while previous_clustering != current_clustering:
  previous_clustering = current_clustering
  current_clustering = []
for c in cluster:
  c.mean = compute_mean_position(c)
for point in data:
  n = find_cluster(point,clusters)
  cluster[n].add_point(point)
  current_clustering.append(n)
```

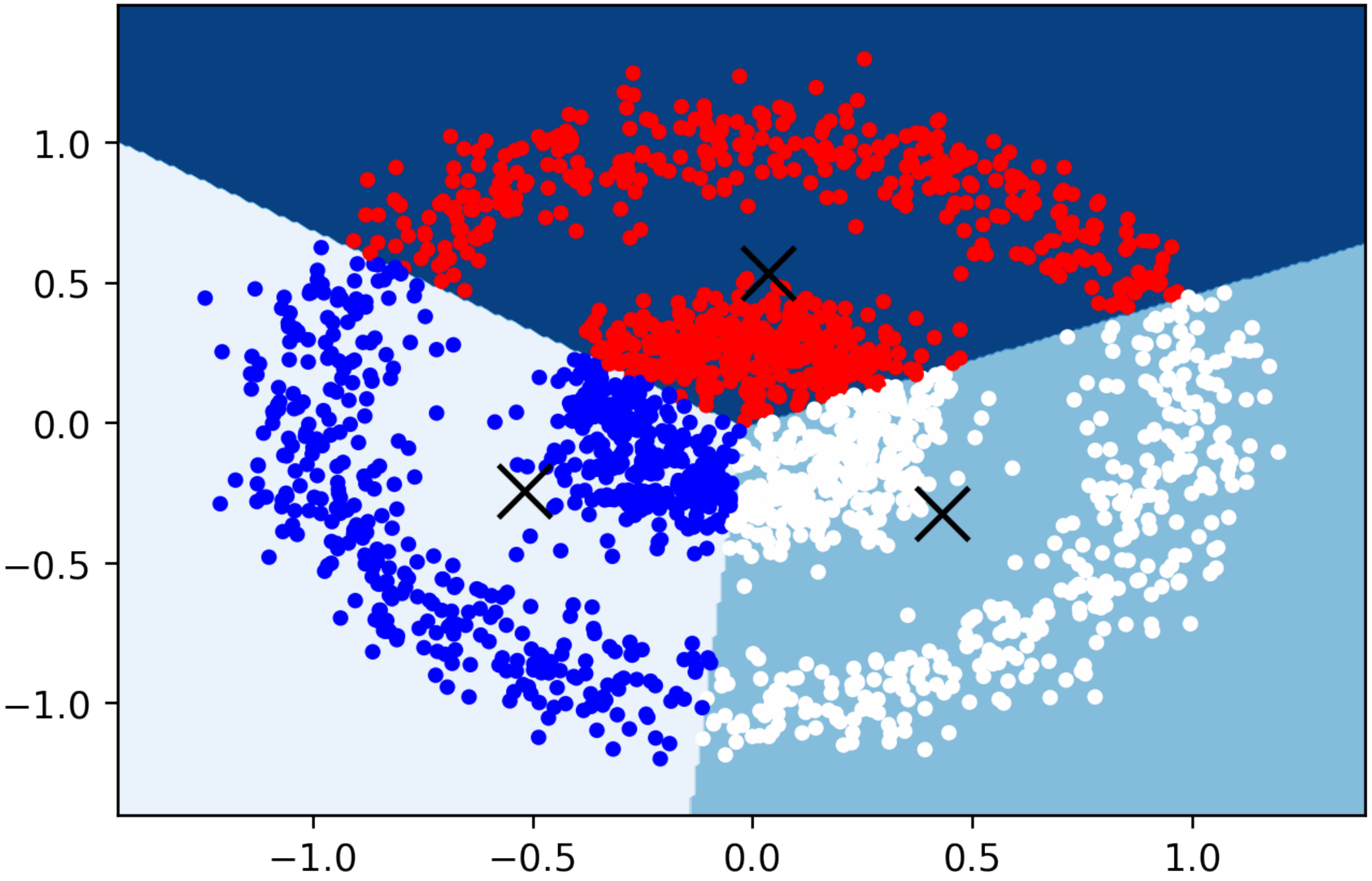

10
5
0
−5
−10
−8
−6
−4
−2
0
2
4
6
8


10
5
0
−5
−10
−8
−6
−4
−2
0
2
4
6
8

- Naturally finds (hyper)spherical clusters of data.
- Can’t cluster by density, since it’s always minimizing a distance-based metric to the cluster means.


1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0


1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0


1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0

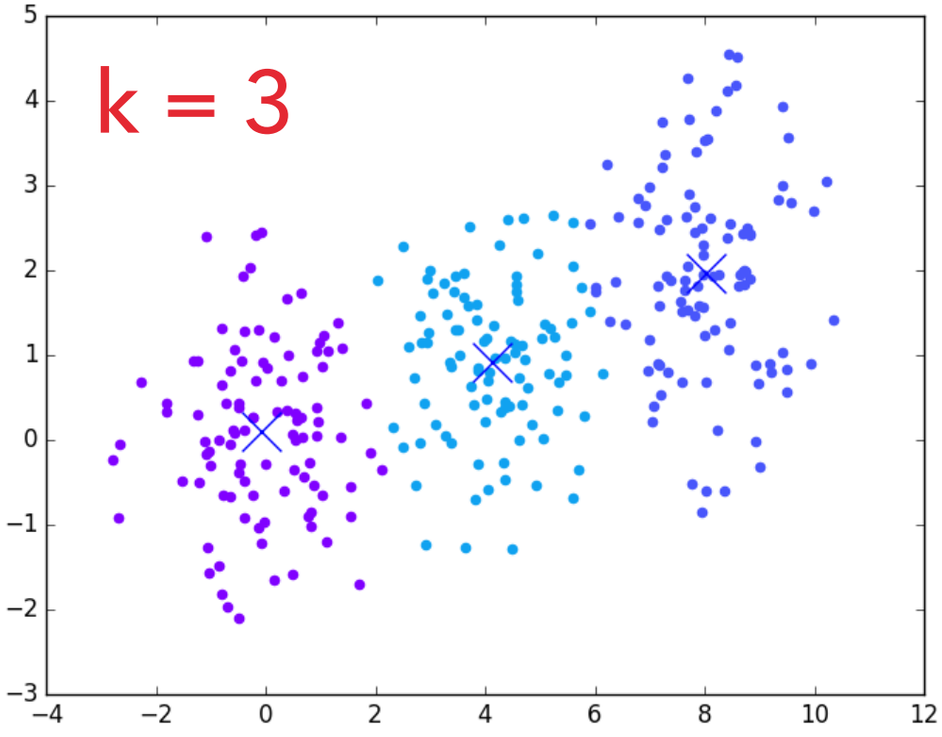
- Initialization matters. The algorithm is guaranteed to converge, but it's not guaranteed to find a global optimum.
  - What if we choose our starting points more optimally?
    - kMeans++
  - How do we test if we are at the optimal solution?

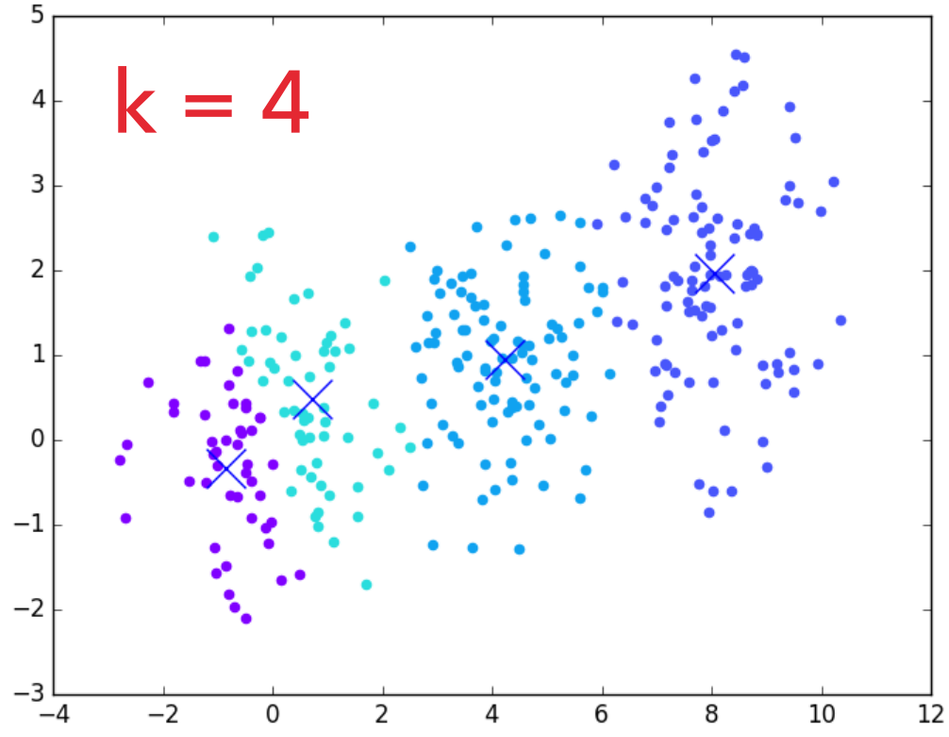
- Initialization matters. The algorithm is guaranteed to converge, but it's not guaranteed to find a global optimum.
  - What if we choose our starting points more optimally?
    - kMeans++
  - How do we test if we are at the optimal solution?
- Curse of dimensionality
  - If we go from 2D to 207D, clustering becomes a really hard problem. Everything is far from everything in 207D.
  - Possible solution: Dimensionality reduction (PCA, etc)

HOW DO YOU CHOOSE K?

k = 1

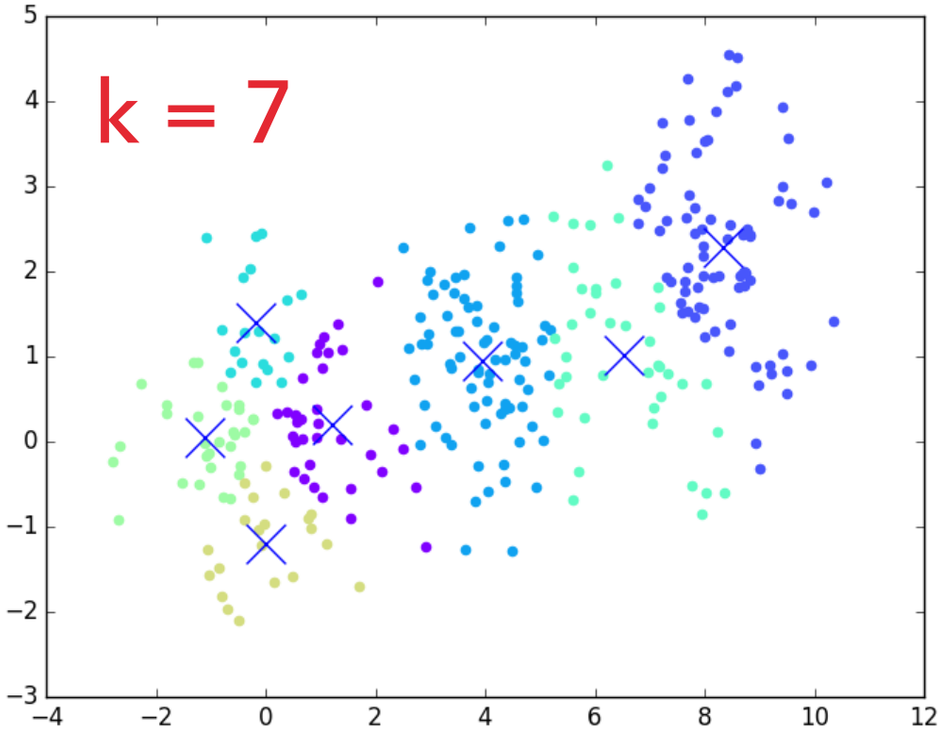
k = 2

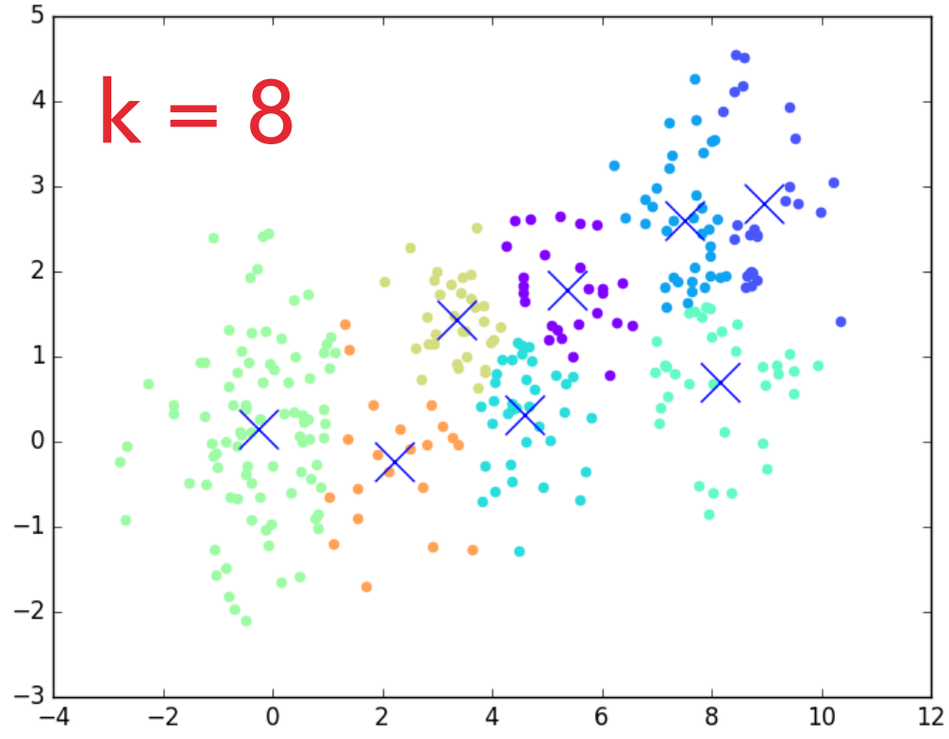

k = 3


k = 4

k = 5

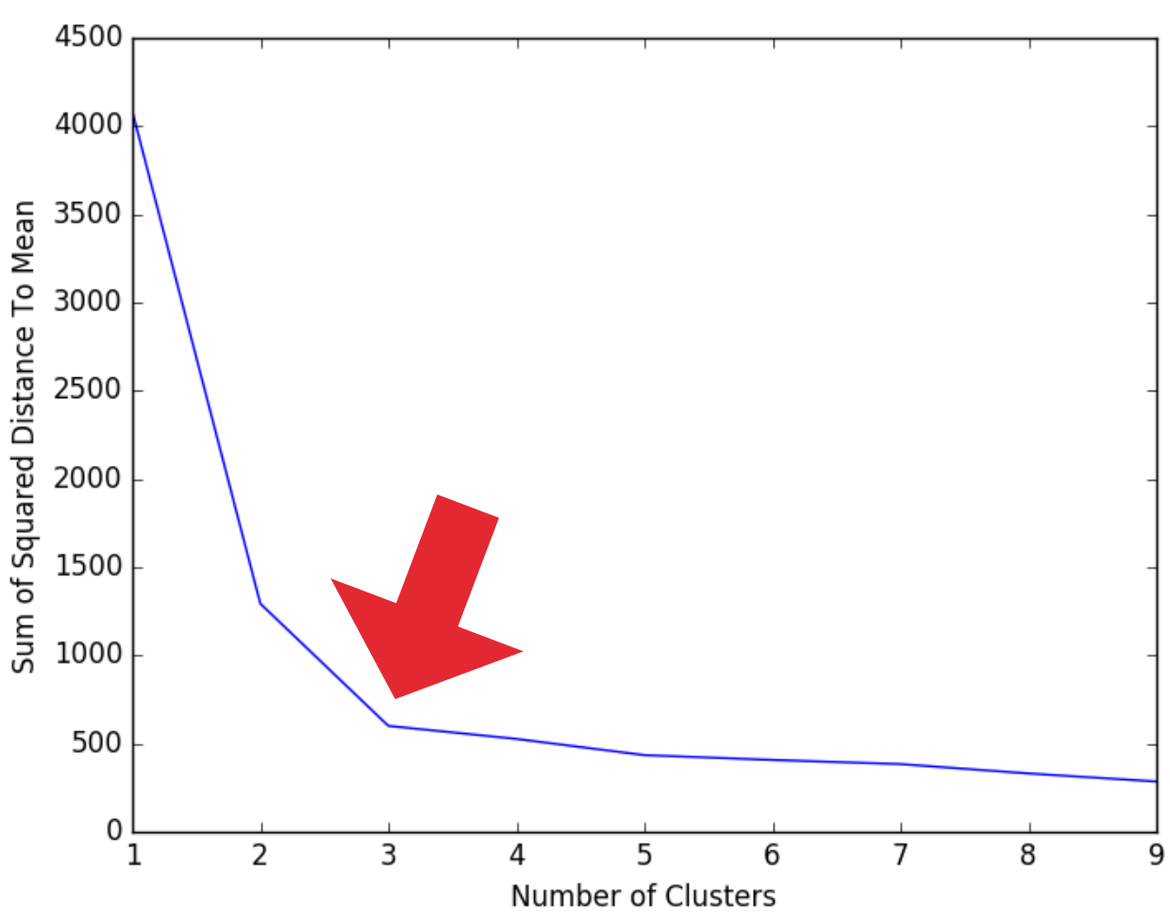
k = 6


k = 7


k = 8

There are many ways to choose k, but one of the easiest is to plot the total squared distance between each point and the mean of it's cluster. You can look for an elbow in the distribution as a function of number of clusters.

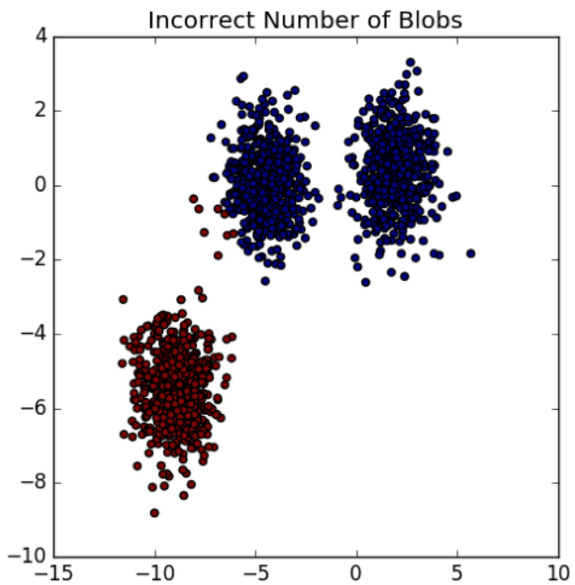
NOTE: This is just a guide. The elbow method is totally heuristic and you can use it as a starting point for your data set and evaluate from there.

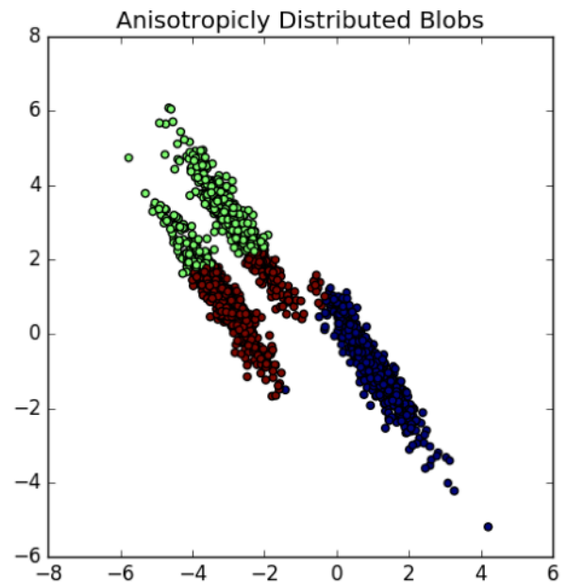


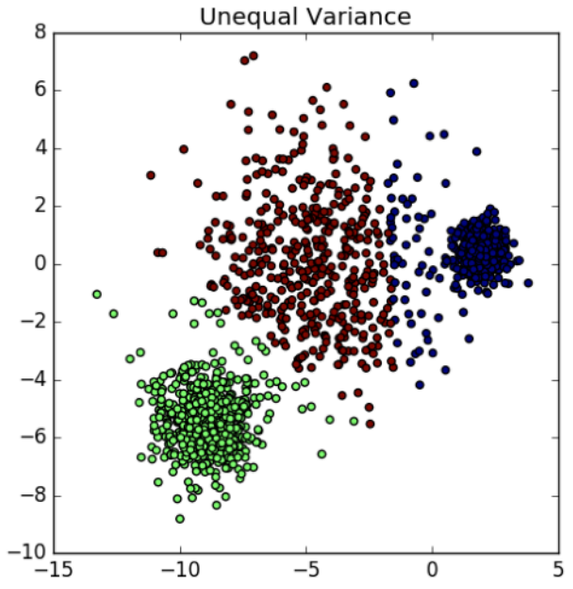
- K-means is an unsupervised learning algorithm that does clustering.
- The general rules: make clusters, assign points to those clusters by distance, (recalculate the centers with the new points, and reassign points)*∞ until convergence.
- Sample Uses:
  - Population Segmentation
  - Computer Vision
  - Improving supervised models
  - Finding Cancer!

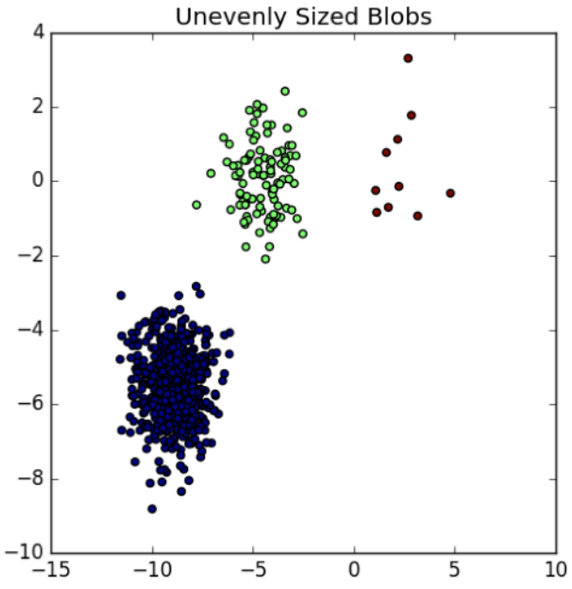
Dubey AK, Gupta U, Jain S"Analysis of k-means clustering approach on the breast cancer Wisconsin dataset." Int J Comput Assist Radiol Surg. 2016 Nov;11(11):2033-2047. Epub 2016 Jun 16.

- There are some challenges with k-means and some built in assumptions.
  - Do you know k?
  - Global vs local minimum?
  - Are your clusters hyper-spherical?
  - Are some of your clusters really loosely grouped while others are not?
  - Dimensionality, a blessing and a curse.









But it's still a fast and powerful tool for drawing conclusions and exploring data; assuming you understand what it assumes you understand.

```
from sklearn.cluster import KMeans

estimator = KMeans(n_clusters=k, n_init=10,
init='k-means++')

clusters = estimator.fit_predict(data)

means = estimator.cluster_centers_
```

Details:

- Lots of initialization options, but most important ones are number of clusters, the initialization scheme, and n_init.
  - Initialization schemes are smarter ways to pick your first points. k-means++ is a way of spreading out the initial cluster seeds, which results in more reliability in finding the optimal solution
  - n_init is the number of times the algorithm is re-initialized and run. The returned result is the most stable solution out of the set of solutions.

http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

EXTRA RESOURCES

- Using KMeans to posterize images: https://github.com/ZWMiller/PythonProjects/blob/master/imageFlattener/kmeans_image_posterizer_demo.ipynb
- Visualizing KMeans: https://www.naftaliharris.com/blog/visualizing-k-means-clustering/
- Using KMeans to find patterns in hand-writing: http://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_digits.html
- KMeans from Scratch: https://github.com/ZWMiller/machine_learning_from_scratch/blob/master/notebooks/clustering/kmeans.ipynb
- Silhouette Score: https://en.wikipedia.org/wiki/Silhouette_(clustering)

- Island Image (pg 1): http://www.simflight.com/wp-content/uploads/2012/11/World-islands.jpg
- K-means color reduction: http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_ml/py_kmeans/py_kmeans_opencv/py_kmeans_opencv.html
- K-means assumption plots: http://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_assumptions.html#sphx-glr-auto-examples-cluster-plot-kmeans-assumptions-py