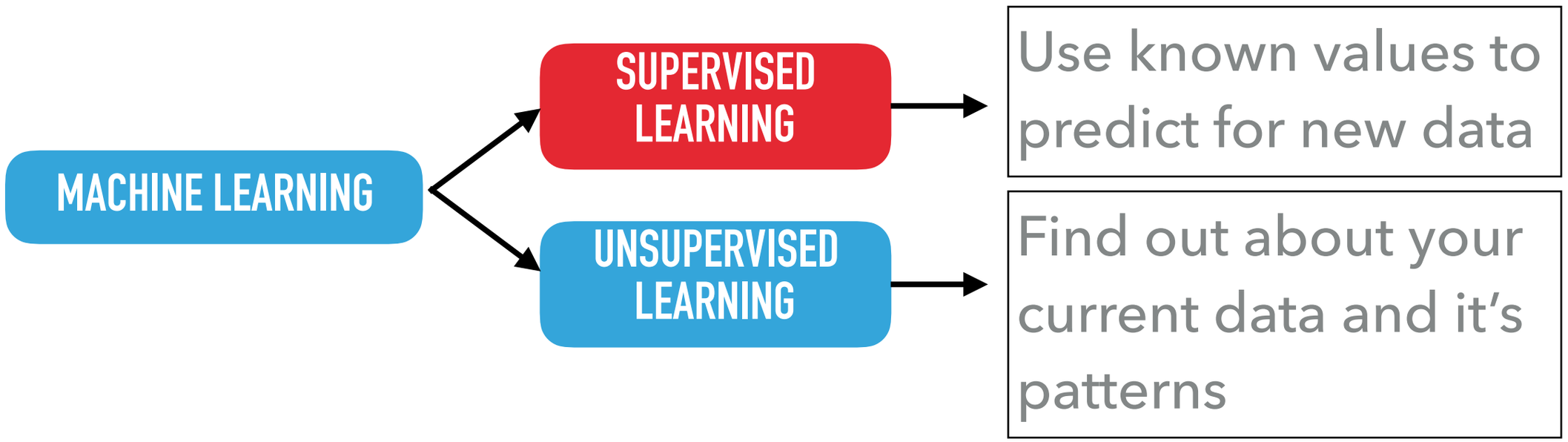
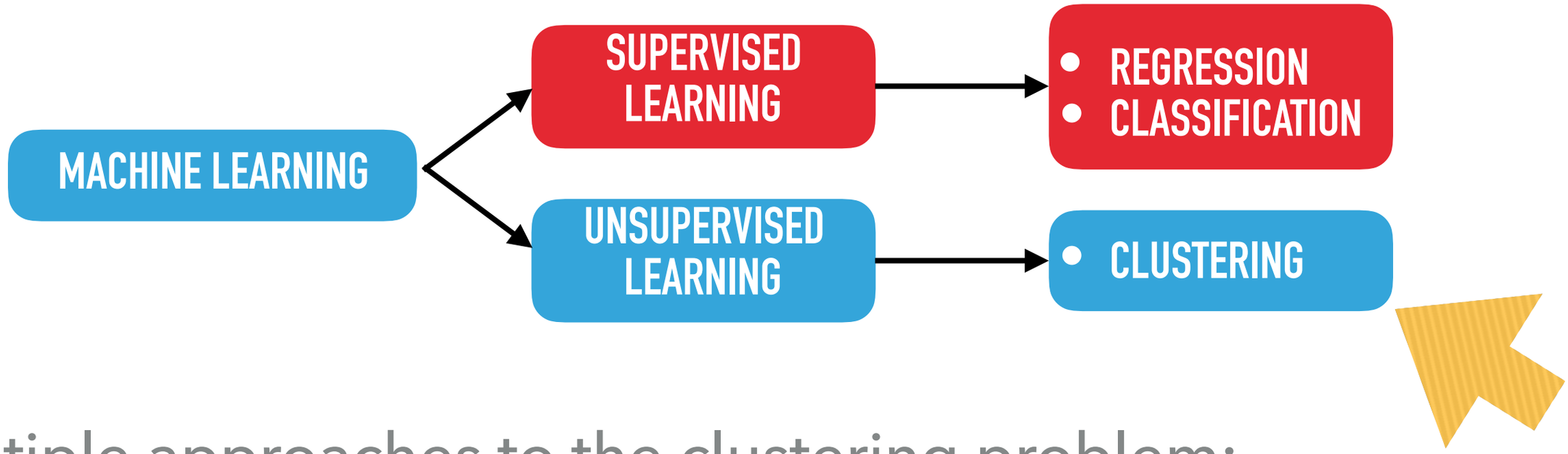
Z. W. MILLER

K-MEANS CLUSTERING

- Taking a set of objects and grouping together those that are most alike, segmenting the objects into new sub-groups.



- Taking a set of objects and grouping together those that are most alike, segmenting the objects into new sub-groups.



- Multiple approaches to the clustering problem:
  - Hierarchical Clustering
  - *Partitional Clustering (Our focus today)*

HOW MANY CLUSTERS?

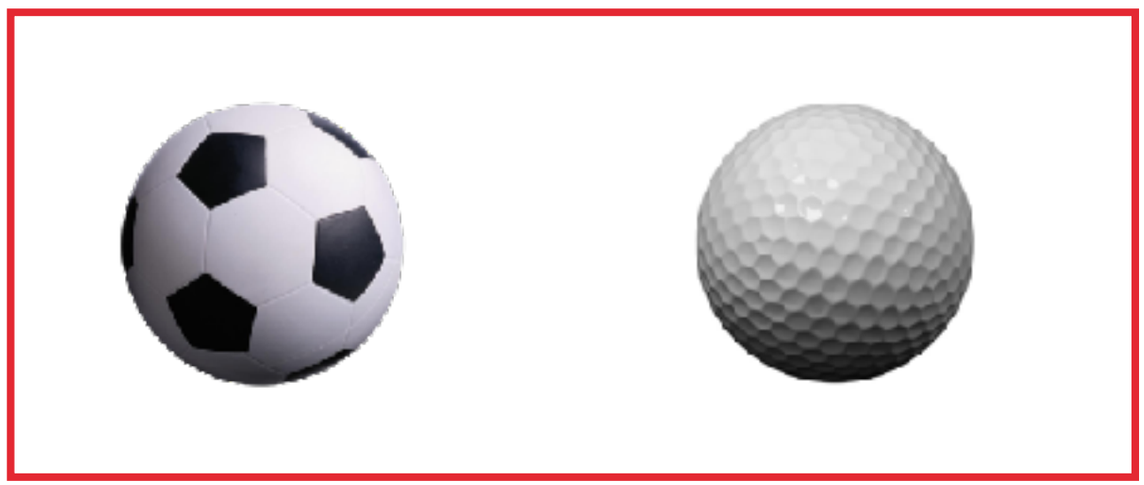
HOW MANY CLUSTERS?

By eye, we pick out two clusters fairly easily.

What if I wasn't clustering by proximity?

HOW MANY CLUSTERS?

By Color.



HOW MANY CLUSTERS?

By Shape.

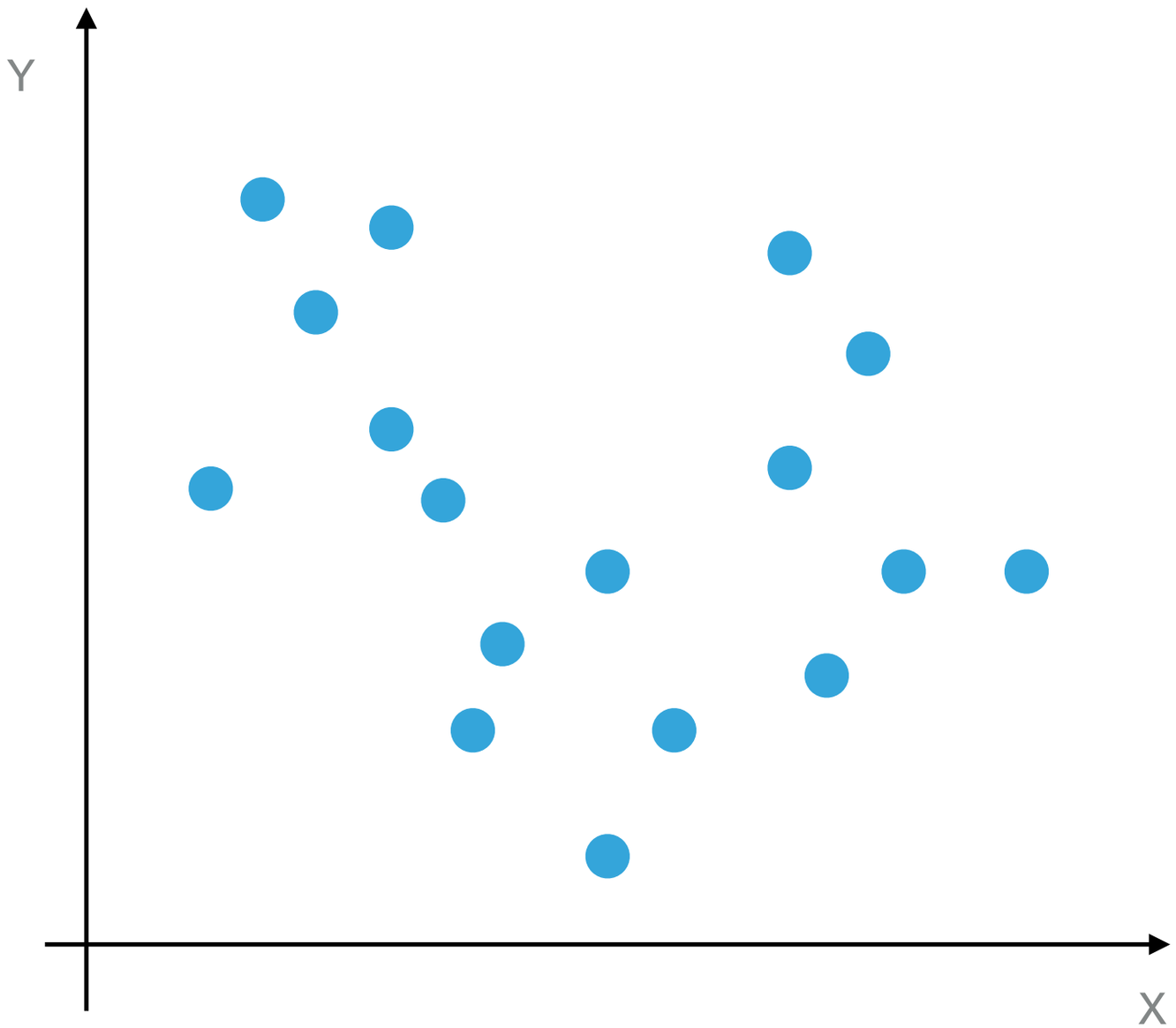
We want an algorithm that can take information about members in a set and find these clusters on it’s own. We also want to be able to classify by many variables at once.

- One of the most common clustering algorithms is known as k-means clustering.
- The idea is to partition the input set of points into k different clusters, based on some distance function. In the simplest form, we'll use Euclidean distance as our f(x); and we'll use features of the data set to compute the distance. We want to find clusters that minimize the (squared) distance from the mean value of the cluster to the members of the cluster.
- This can be done by calculating the distance in the space for each point relative to the cluster center; as shown in summation for here:

$$\sum_{i=1}^{k} \sum_{x_j \in S_i} (x_j - \mu_i)^2$$

Let's visualize the algorithm:

STAGE 1: INITIALIZATION

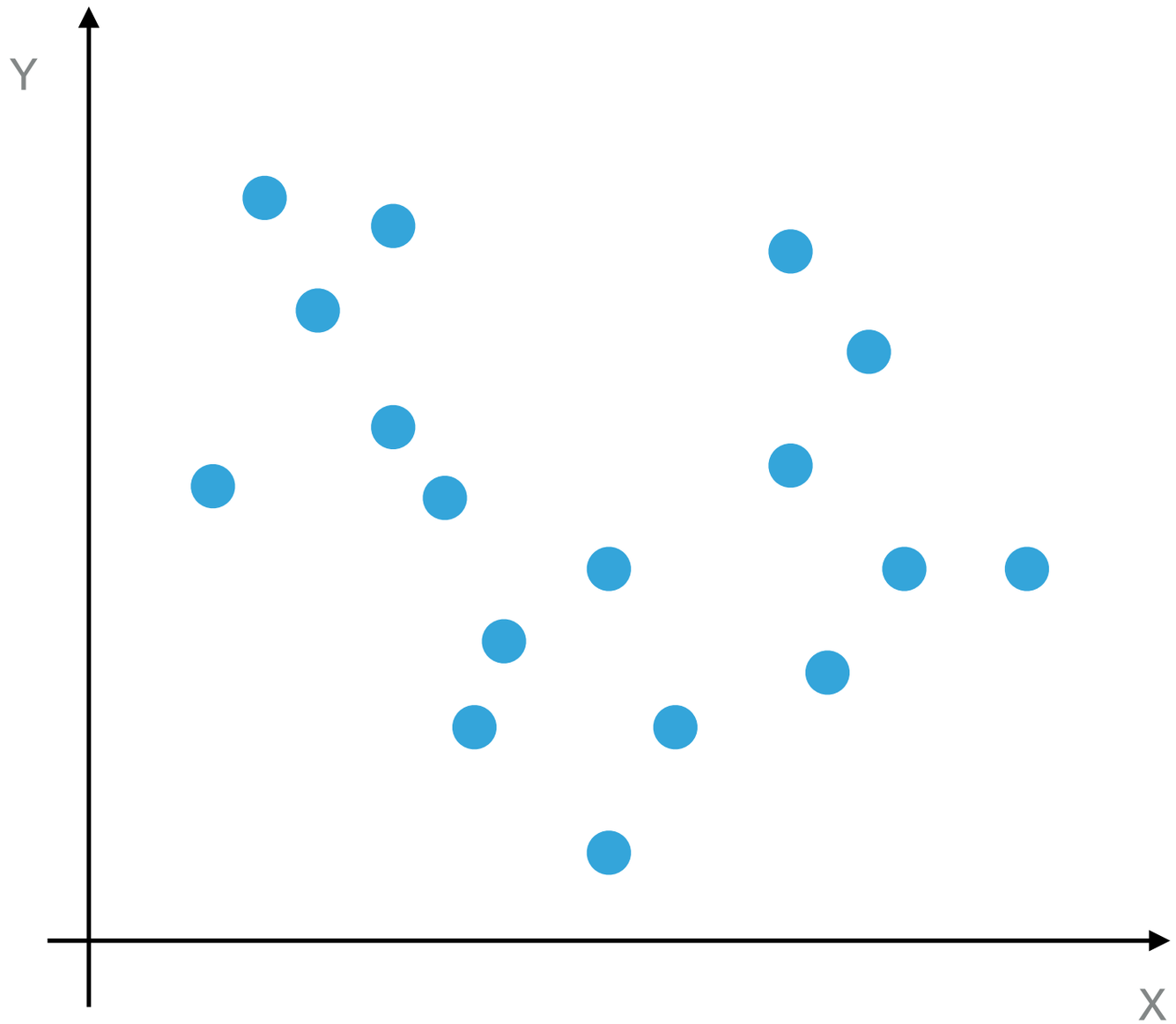


We start with some set of data distributed throughout a space. Each data point has 2 defining features, its X and Y coordinates. Now we want to group them into clusters based on this X,Y distance.

Note: it doesn't have to be just X,Y. It can be any number of dimensions!

Let's visualize the algorithm:

STAGE 1: INITIALIZATION

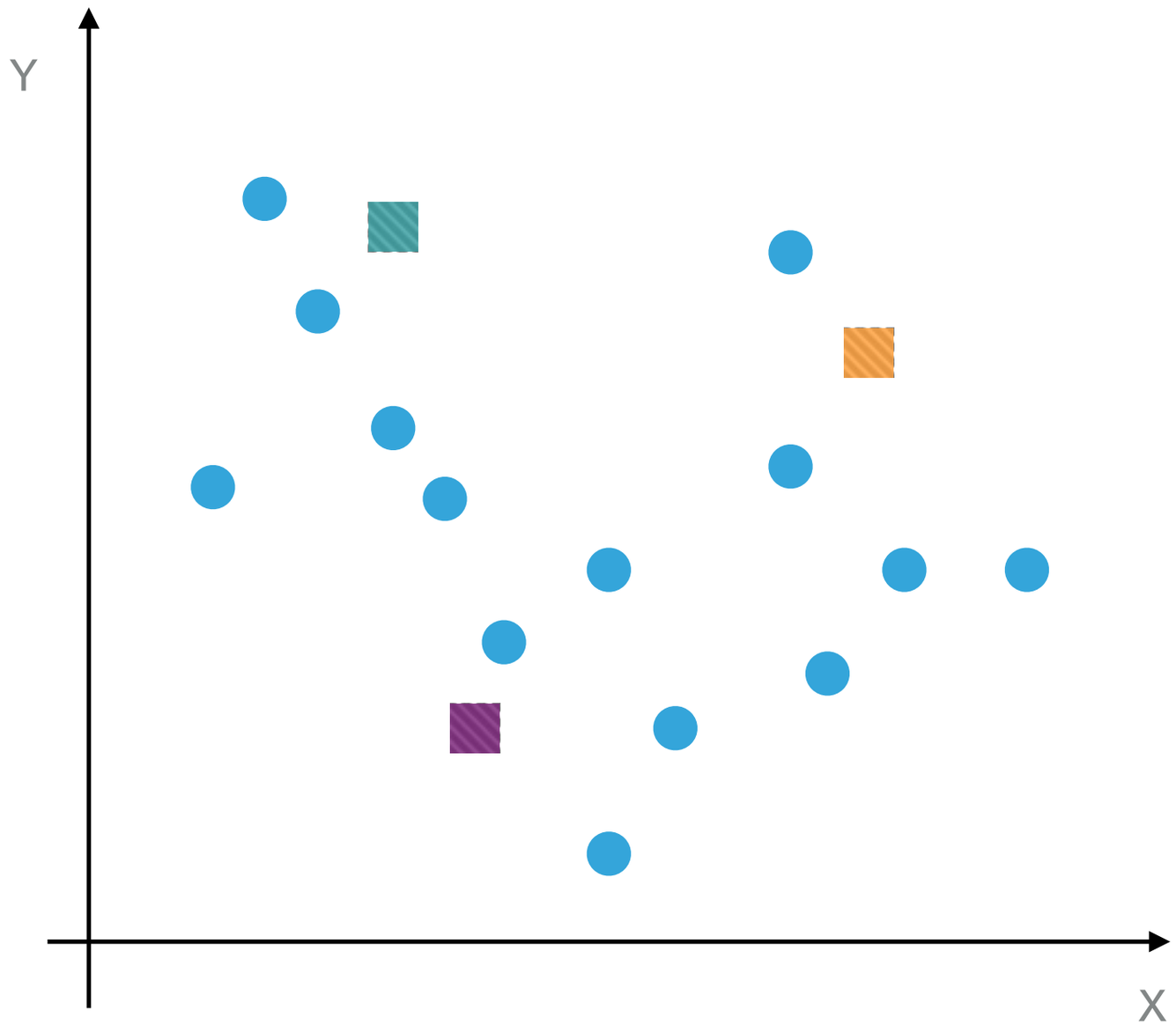


To initialize the algorithm, we know that we need to find k clusters. For this example, let's say k = 3 (more on choosing k later).

The first step is to choose k points in the space to act as the cluster seeds. There are multiple ways to do this, but two common ones are:
1) Randomly choose k points from the set to seed the clusters
2) Randomly set each point to be in a cluster, then calculate the mean value for each cluster.

Let's visualize the algorithm:

STAGE 1: INITIALIZATION



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

Notation:

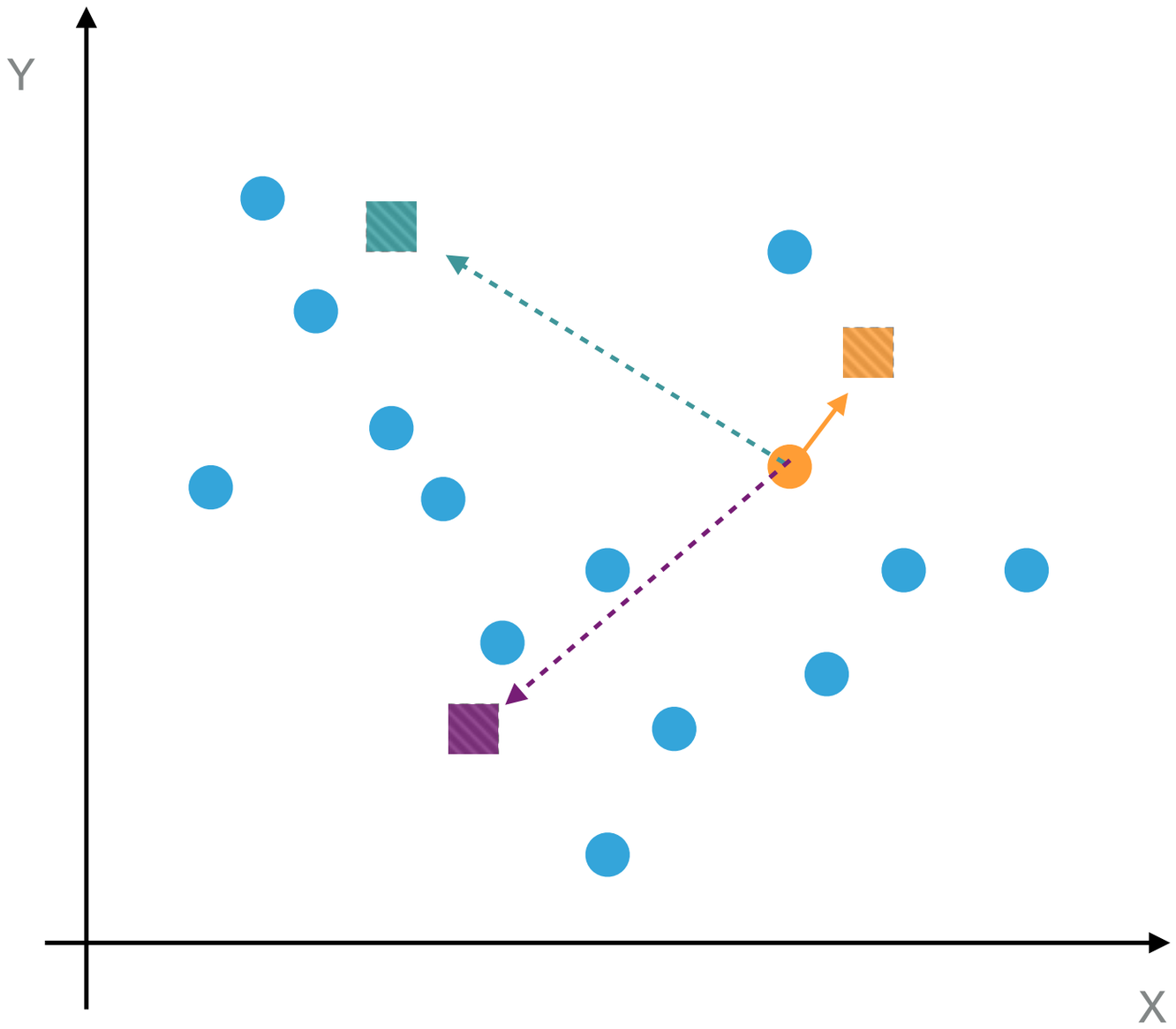
= Cluster Mean/Seed (midpoint of all the data)

= Data point in the set.

Colors represent different clusters.

Let's visualize the algorithm:

STAGE 1: INITIALIZATION

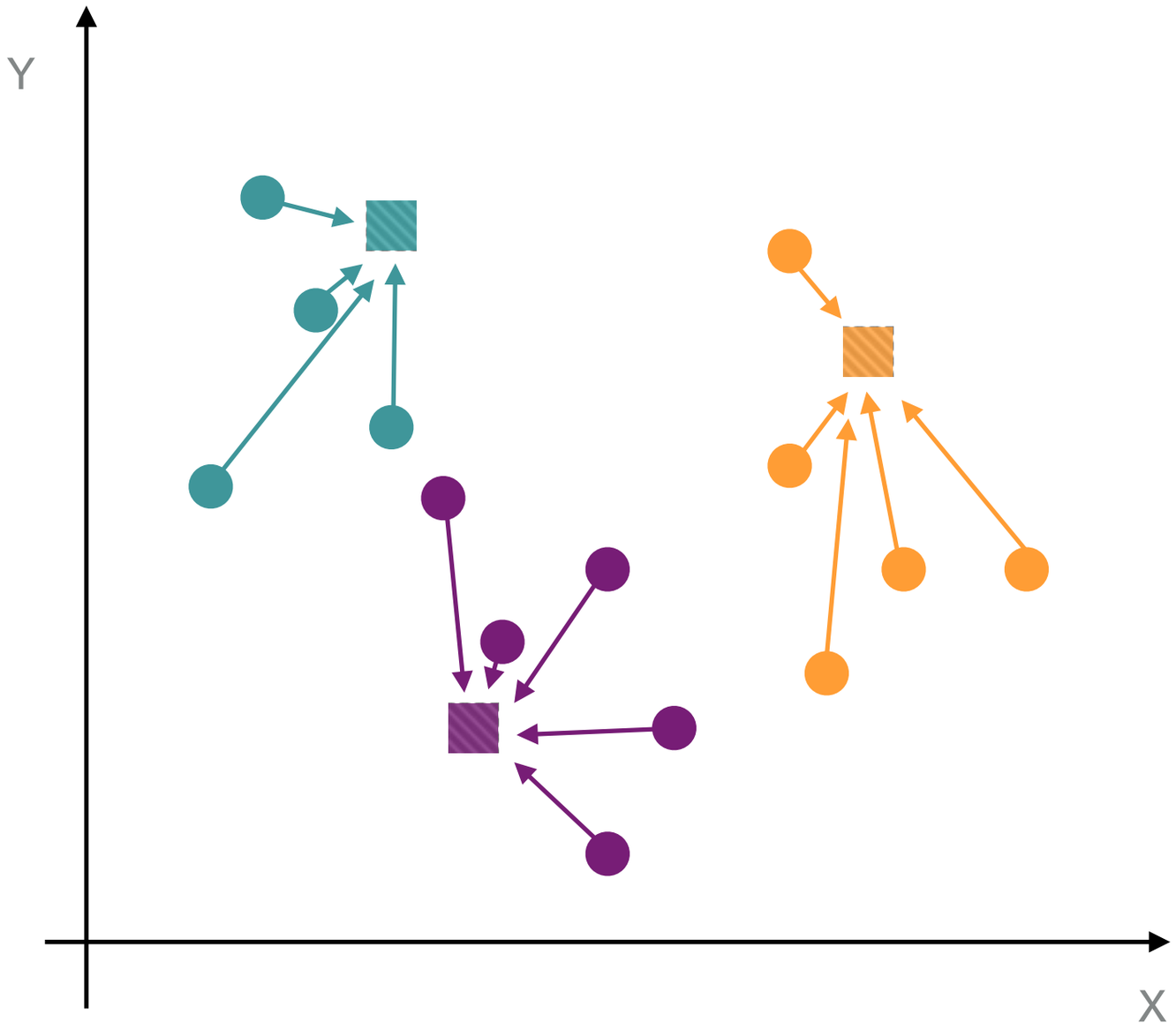


Now we compute the distance from each point to each mean and assign the point to the cluster of the closest seed.

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

Let's visualize the algorithm:

STAGE 1: INITIALIZATION

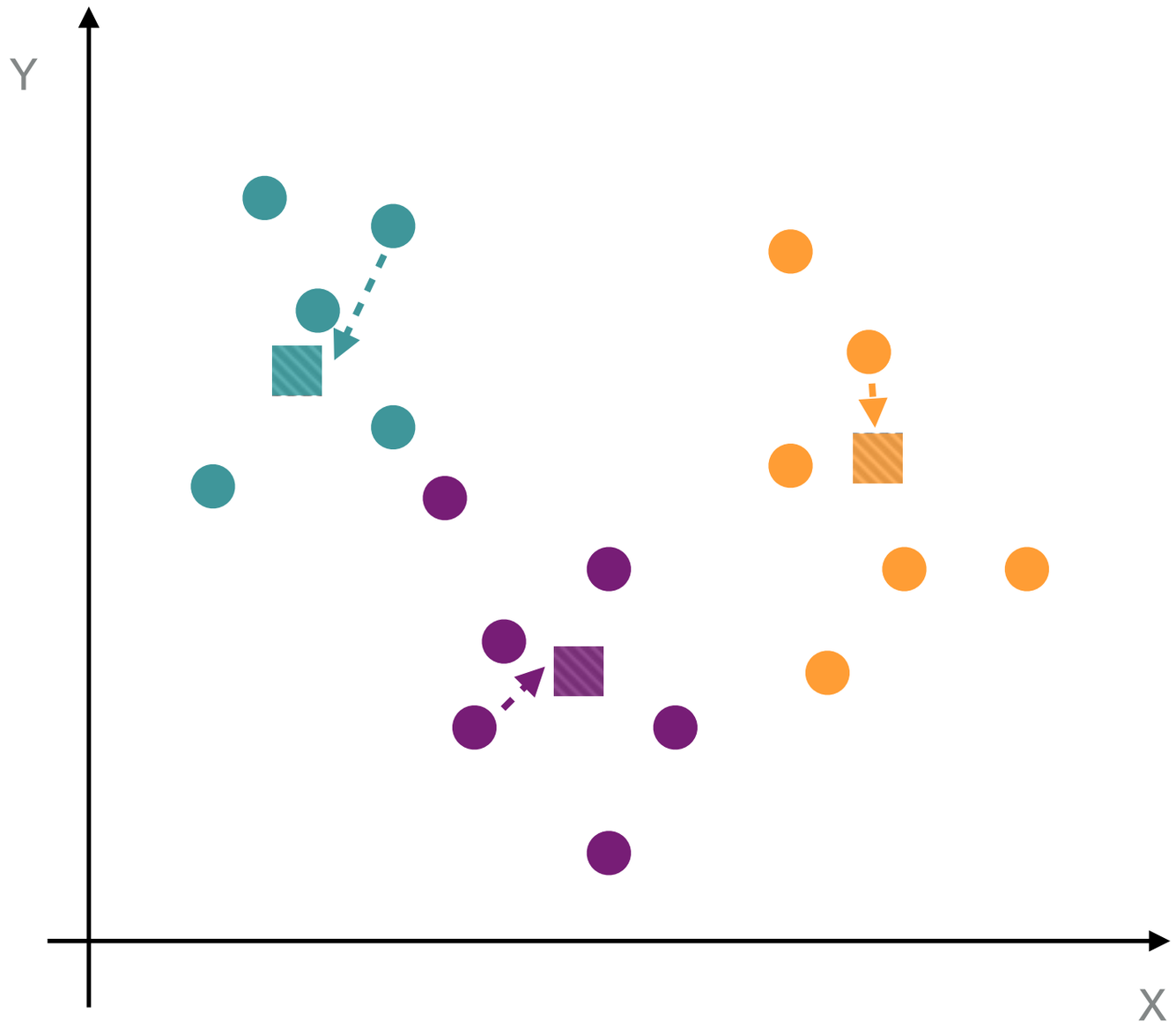


Now we compute the distance from each point to each mean and assign the point to the cluster of the closest seed.

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

Let's visualize the algorithm:

STAGE 1: INITIALIZATION

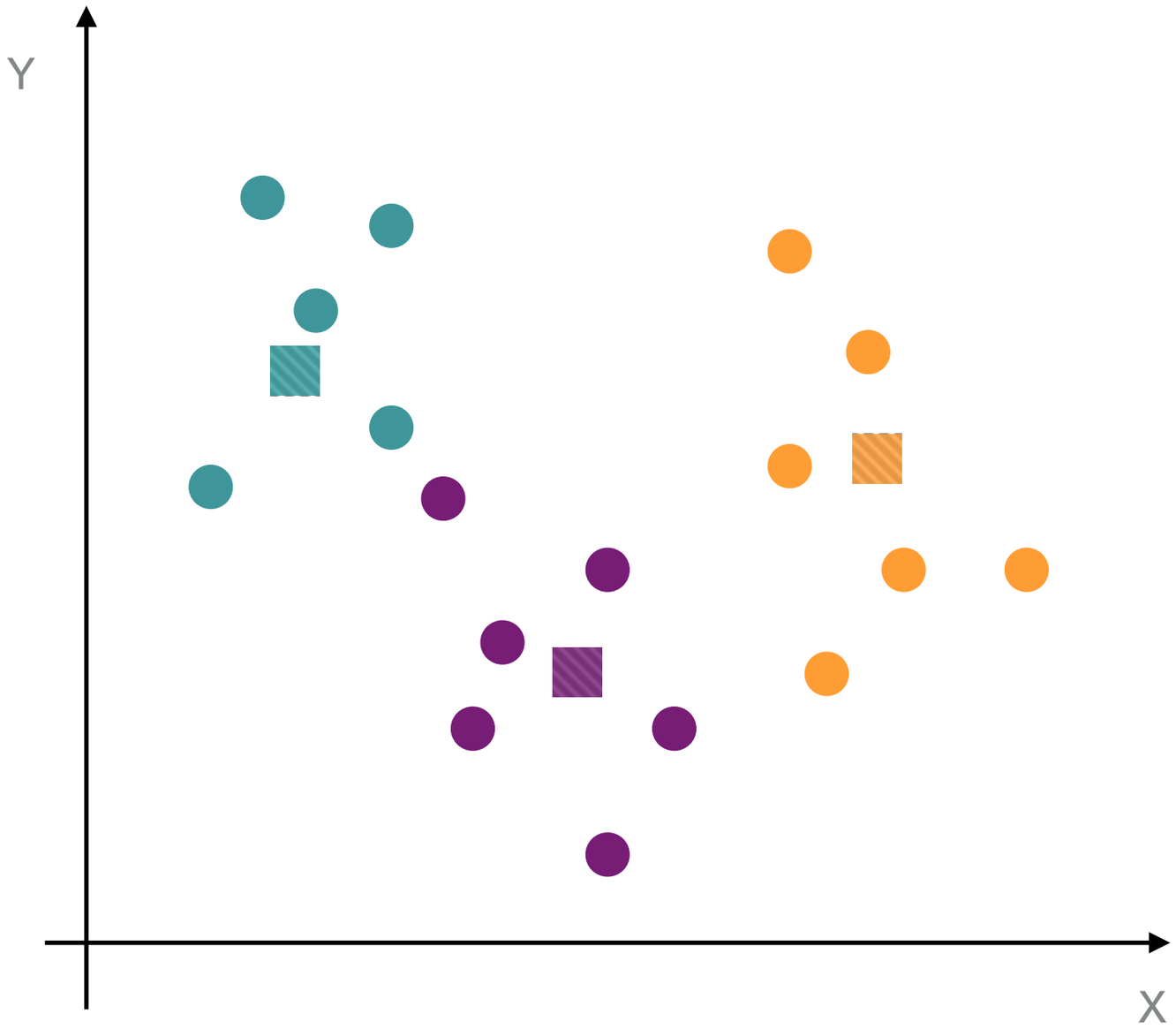


Finally, compute the mean position for each cluster and reassign the cluster seed to that point.

Now we've initialized all 3 clusters.

Let's visualize the algorithm:

STAGE 2: MINIMIZATION

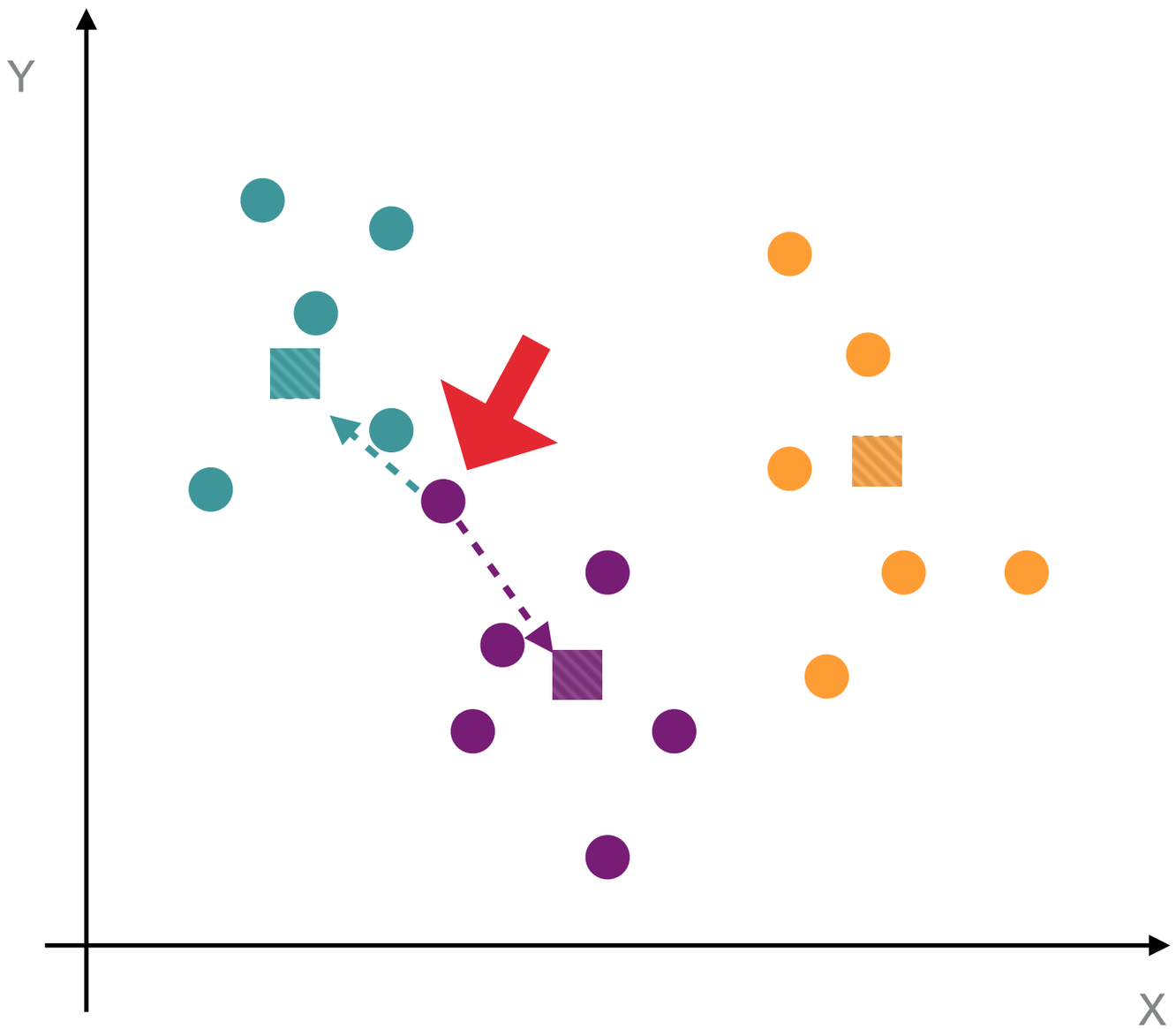


Now we enter the iterative part of the procedure, where we attempt to minimize the distance between the cluster means and all of their constituent points. To do this, we recompute the distance between each point and all the means.

Is the point closer to the mean of a different cluster than its current one? Switch clusters!

Let's visualize the algorithm:

STAGE 2: MINIMIZATION

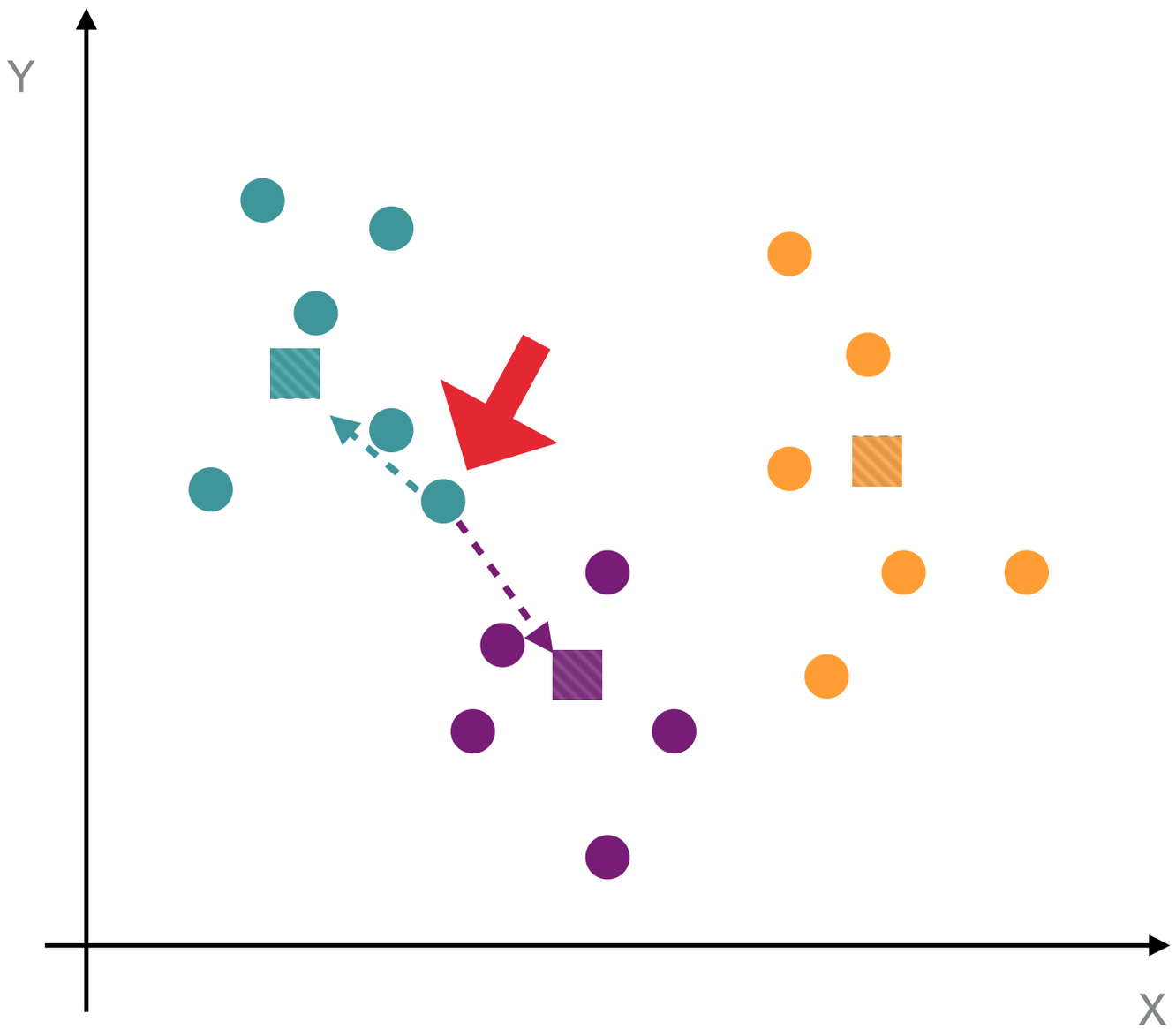


Now we enter the iterative part of the procedure, where we attempt to minimize the distance between the cluster means and all of their constituent points. To do this, we recompute the distance between each point and all the means.

Is the point closer to the mean of a different cluster than its current one? Switch clusters!

Let's visualize the algorithm:

STAGE 2: MINIMIZATION

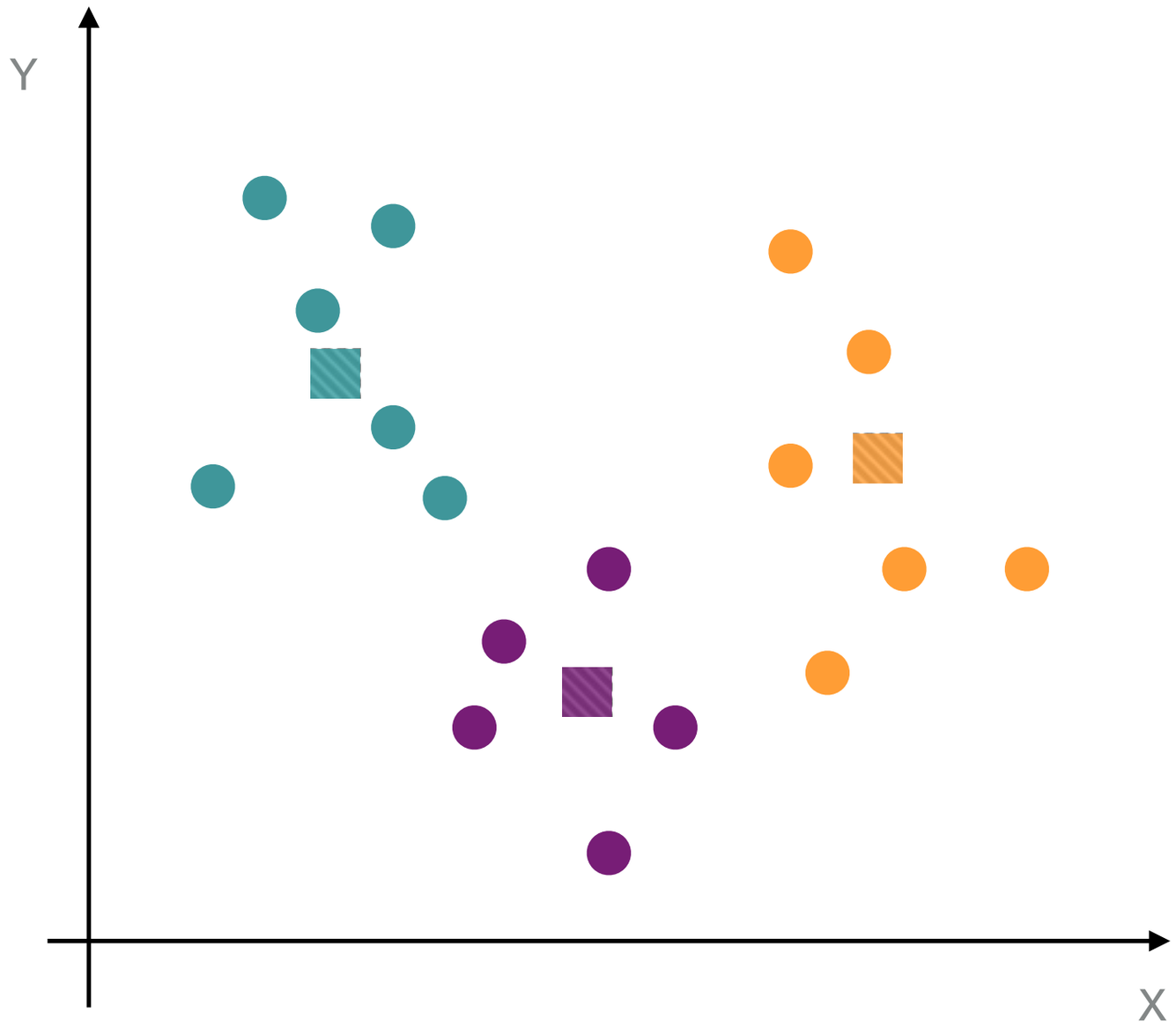


Now we enter the iterative part of the procedure, where we attempt to minimize the distance between the cluster means and all of their constituent points. To do this, we recompute the distance between each point and all the means.

Is the point closer to the mean of a different cluster than its current one? Switch clusters!

Let's visualize the algorithm:

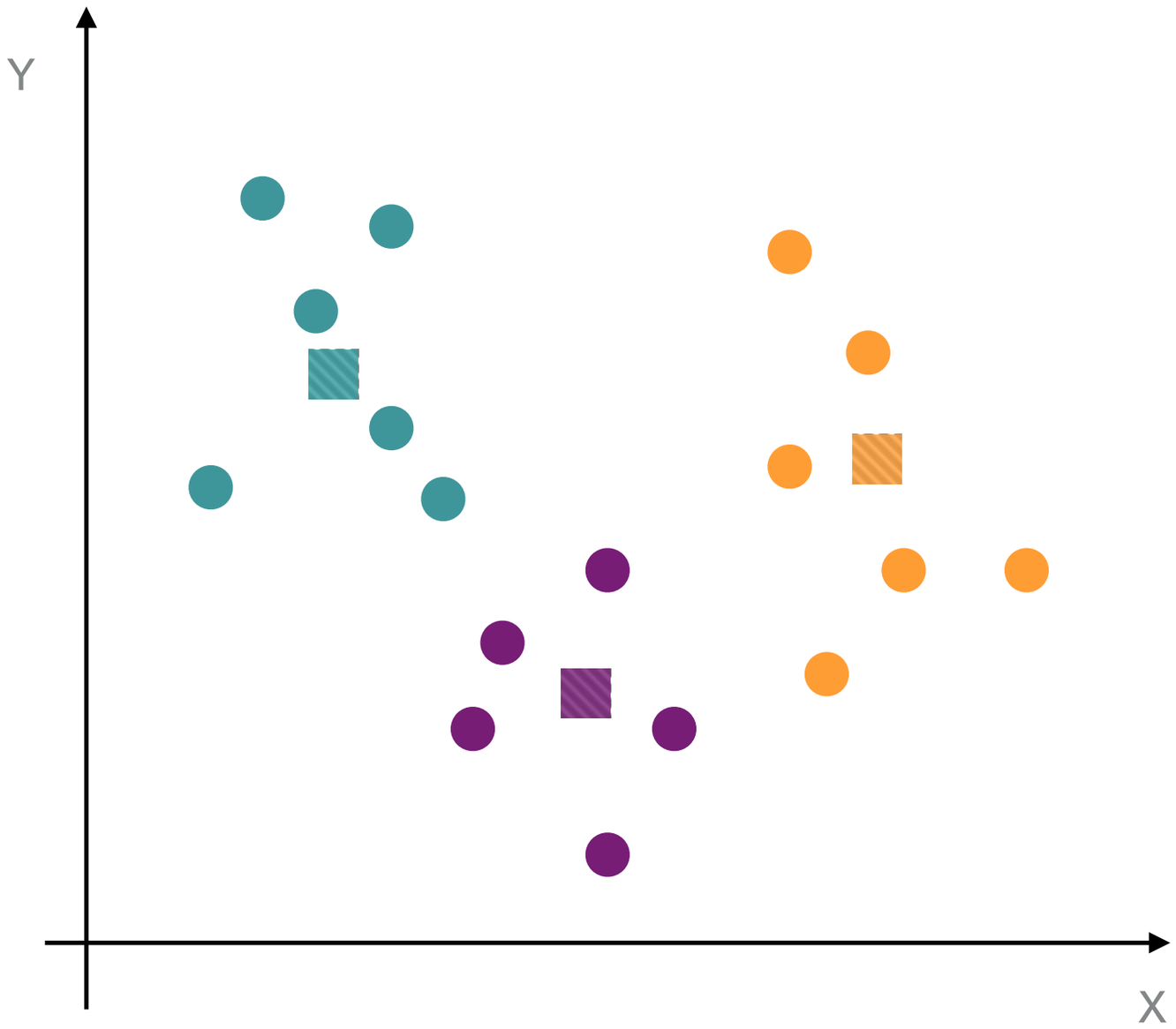
STAGE 2: MINIMIZATION



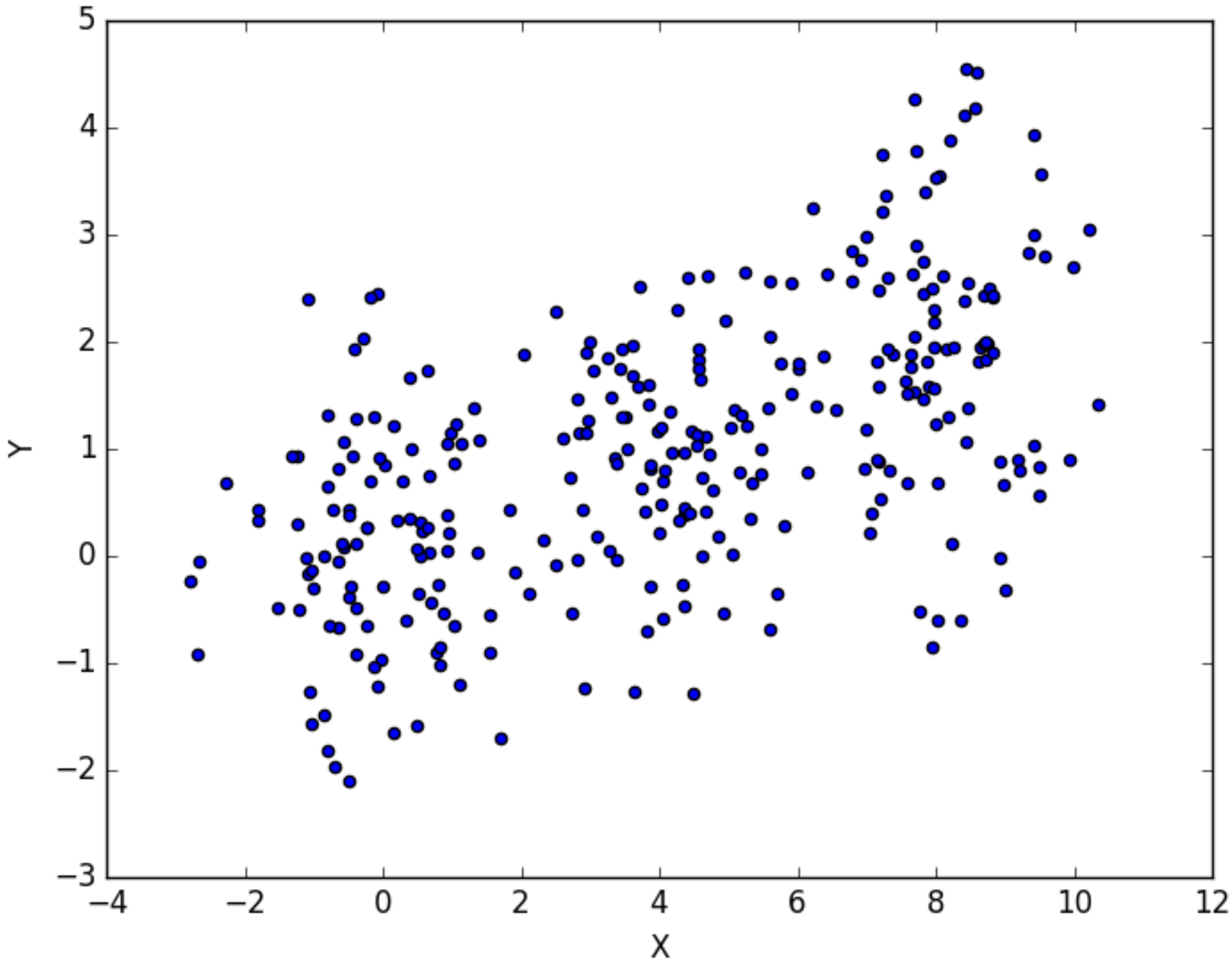
Now that a point has switched clusters, the means aren't correct any more. Update the means, and try again.

Let's visualize the algorithm:

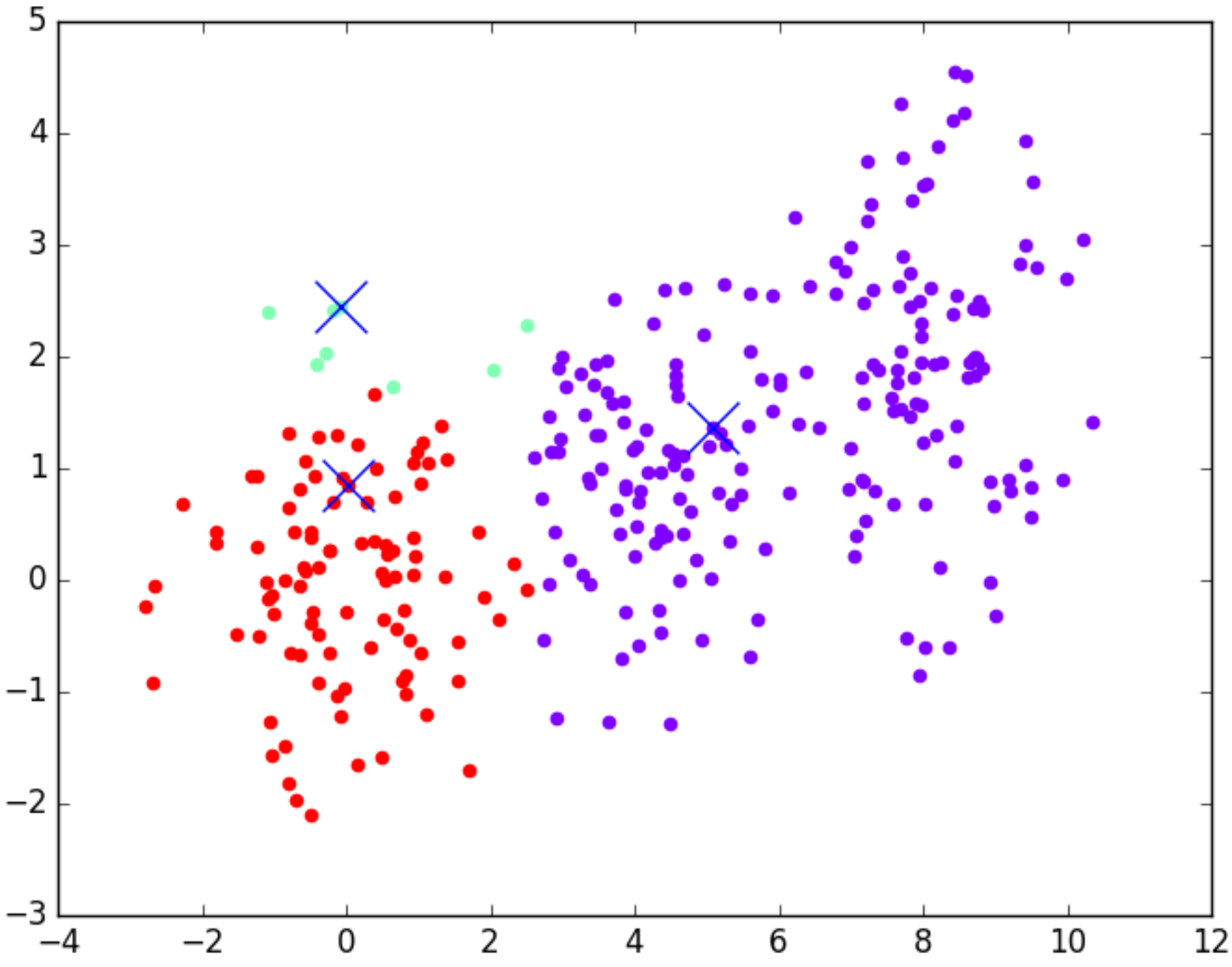
STAGE 2: MINIMIZATION



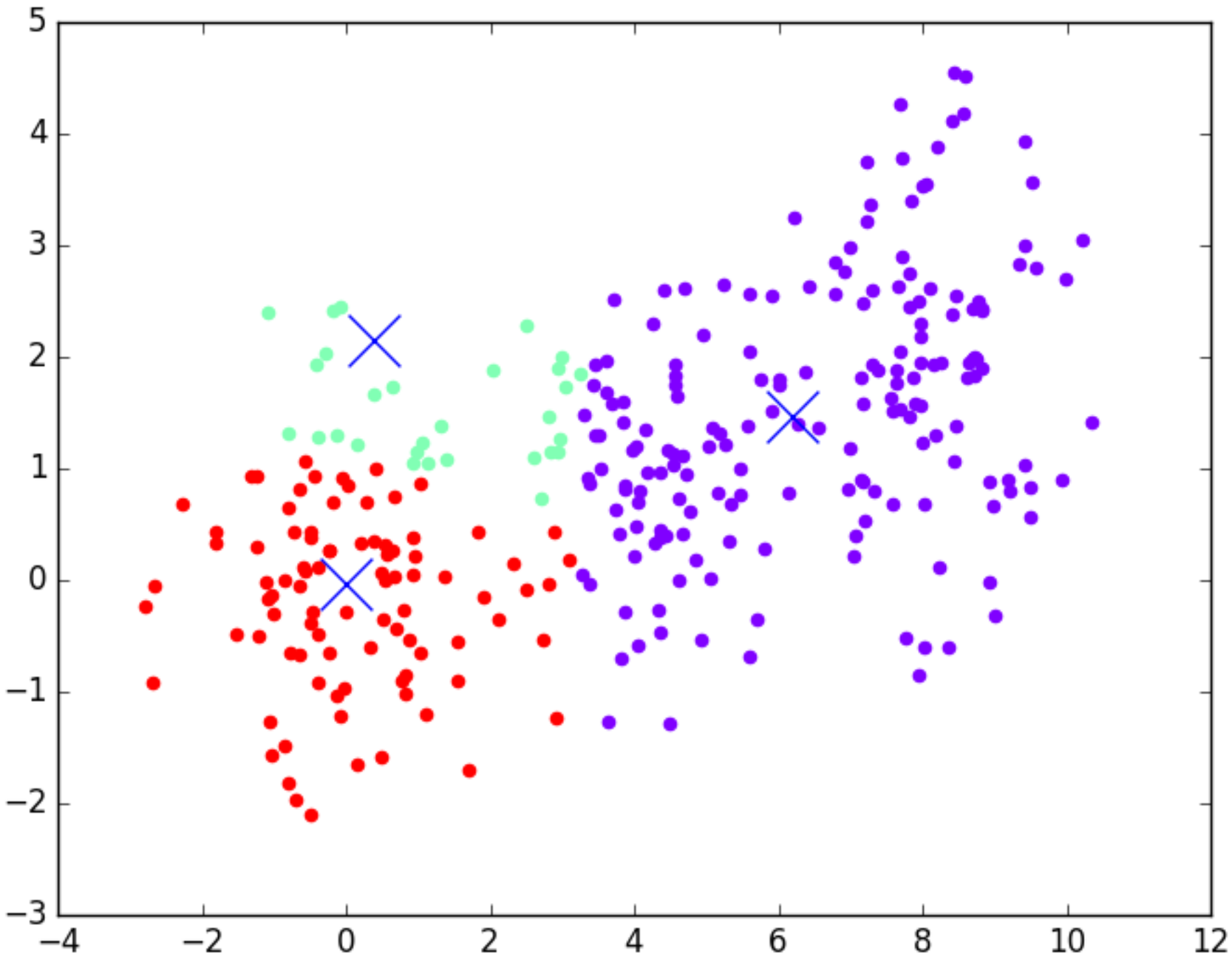
We iterate this step until we reach a stage where no members are changing between clusters. This convergence point is guaranteed to exist and k-means is generally fast at reaching it.



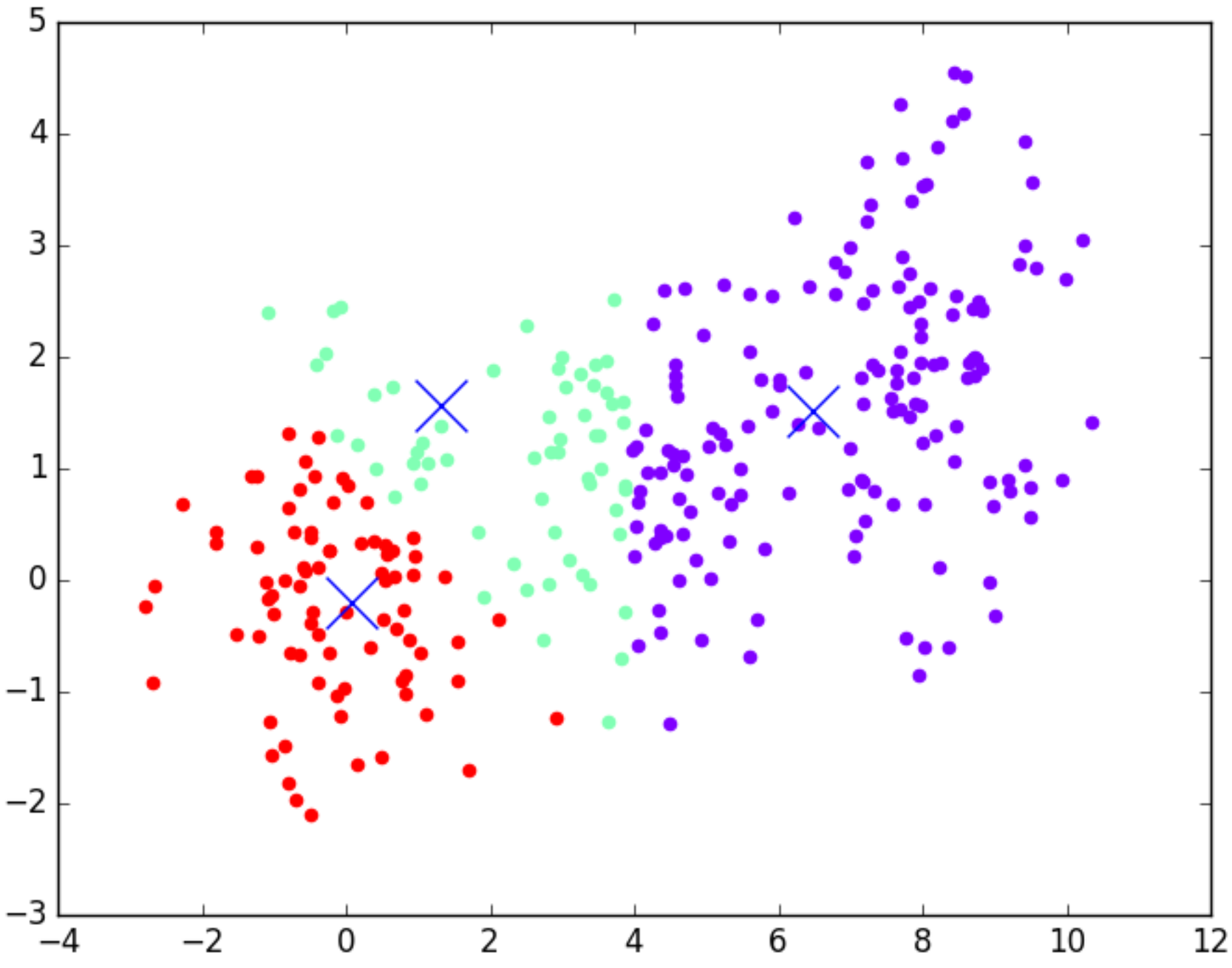
Note: You'll want to flip page to page, not scroll, to get the full effect.



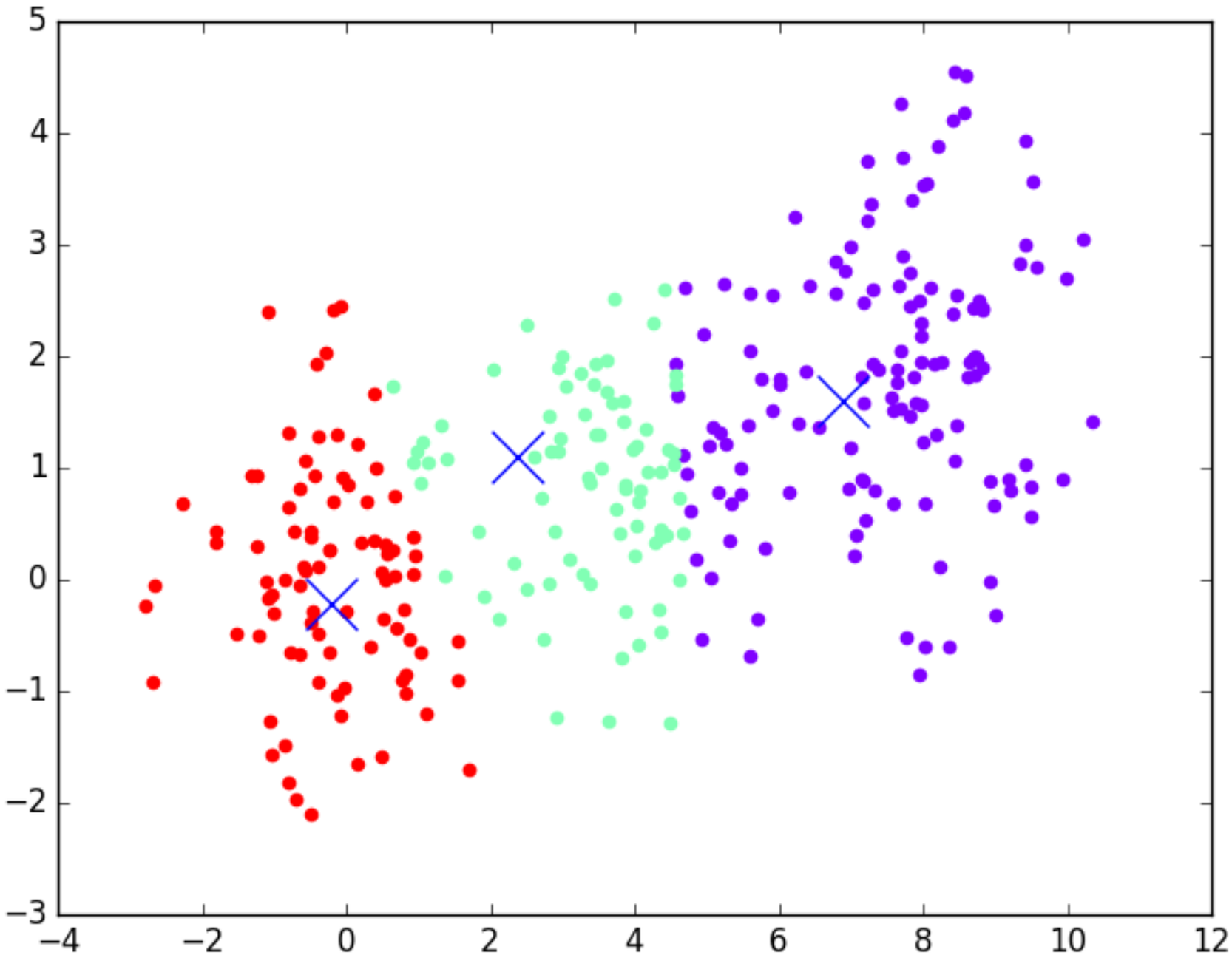
Note: You'll want to flip page to page, not scroll, to get the full effect.



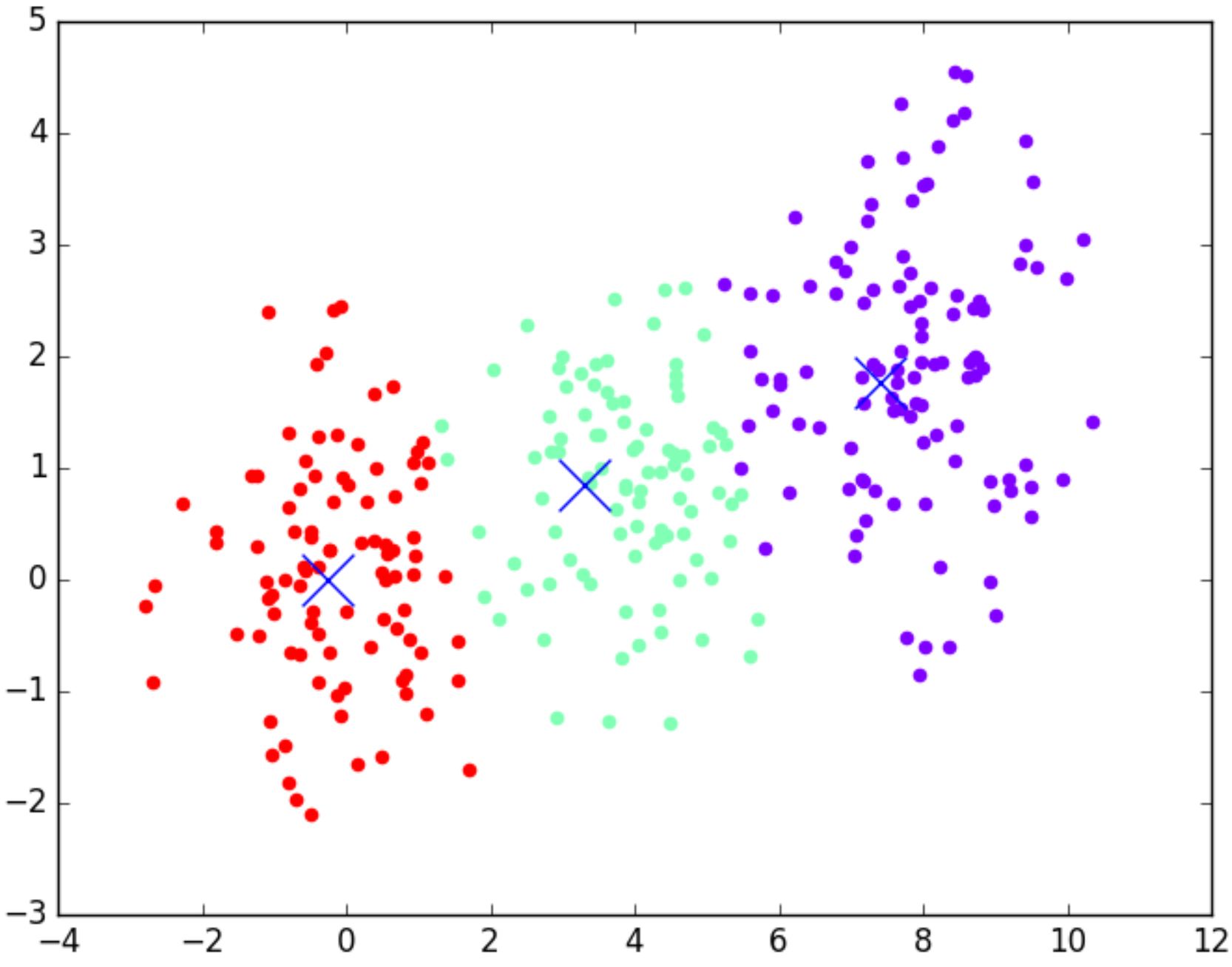
Note: You'll want to flip page to page, not scroll, to get the full effect.



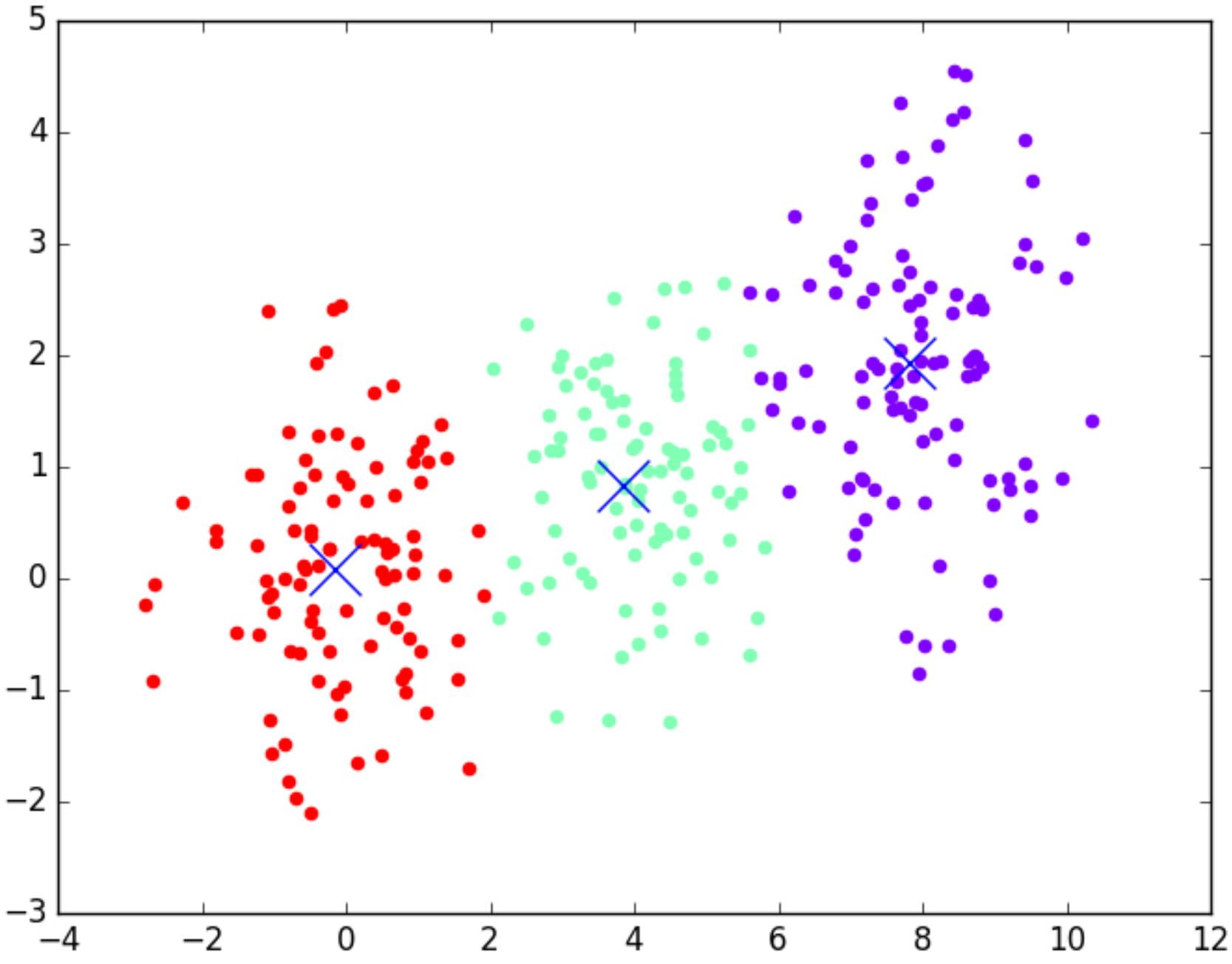
Note: You'll want to flip page to page, not scroll, to get the full effect.



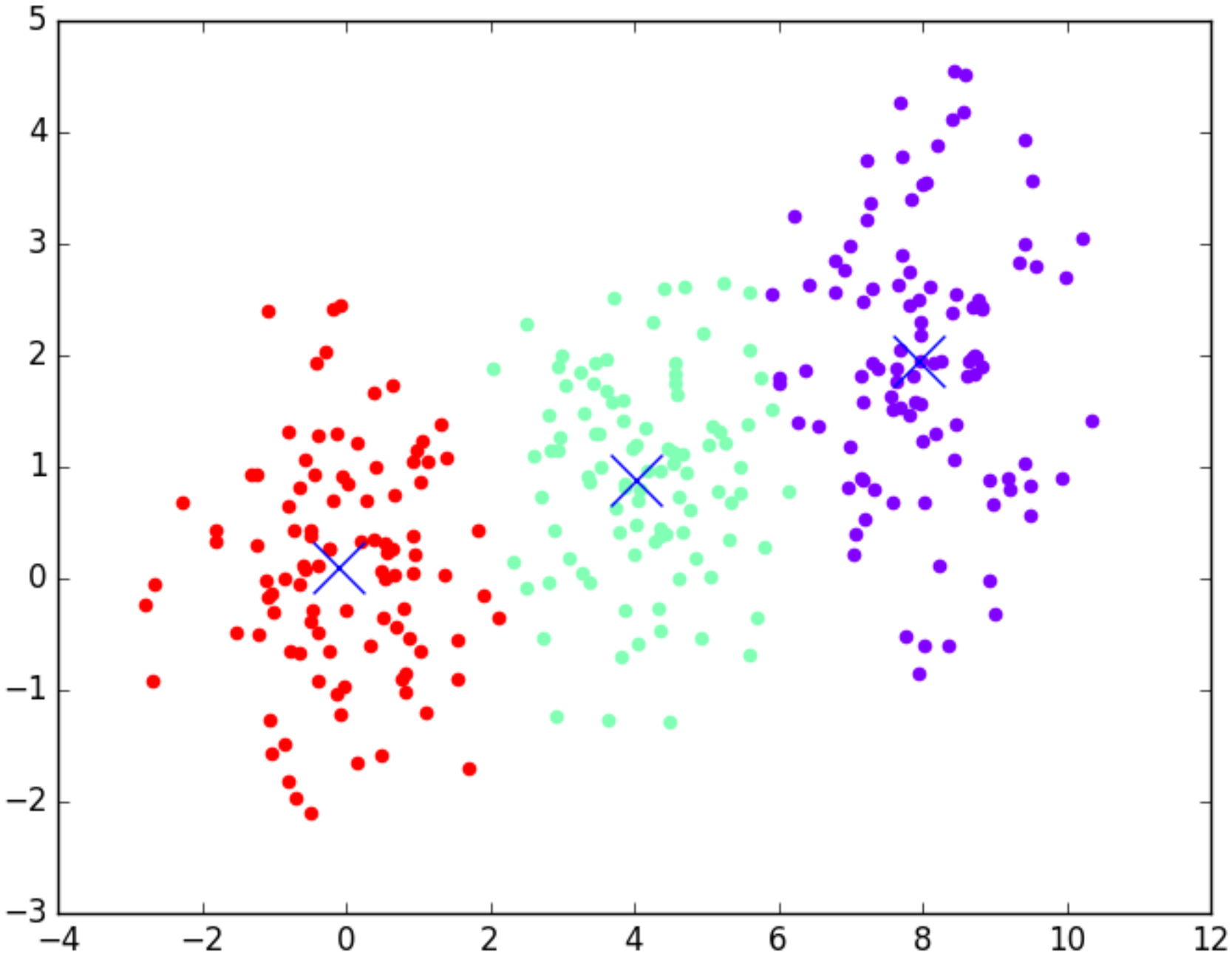
Note: You'll want to flip page to page, not scroll, to get the full effect.



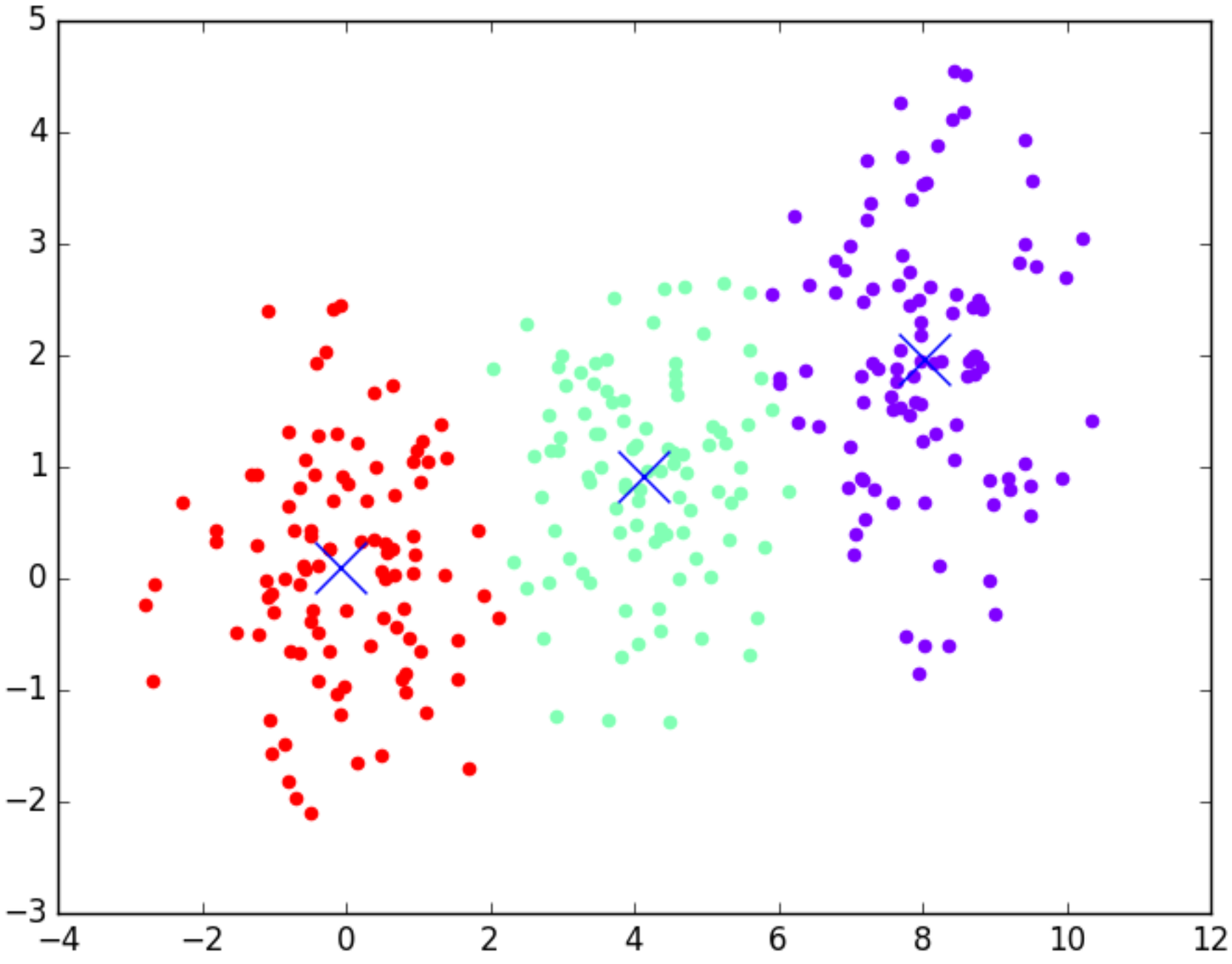
Note: You'll want to flip page to page, not scroll, to get the full effect.



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 = getData("theseAreDataPoints.csv")
for i in range(0,k):
  clusters.append[new cluster]
  clusters[i].setMean(random.choice(data))
for point in data:
  findCluster(point, clusters)
for c in cluster:
  c.mean = computeMeanPosition(c)
```

STAGE 2: MINIMIZATION

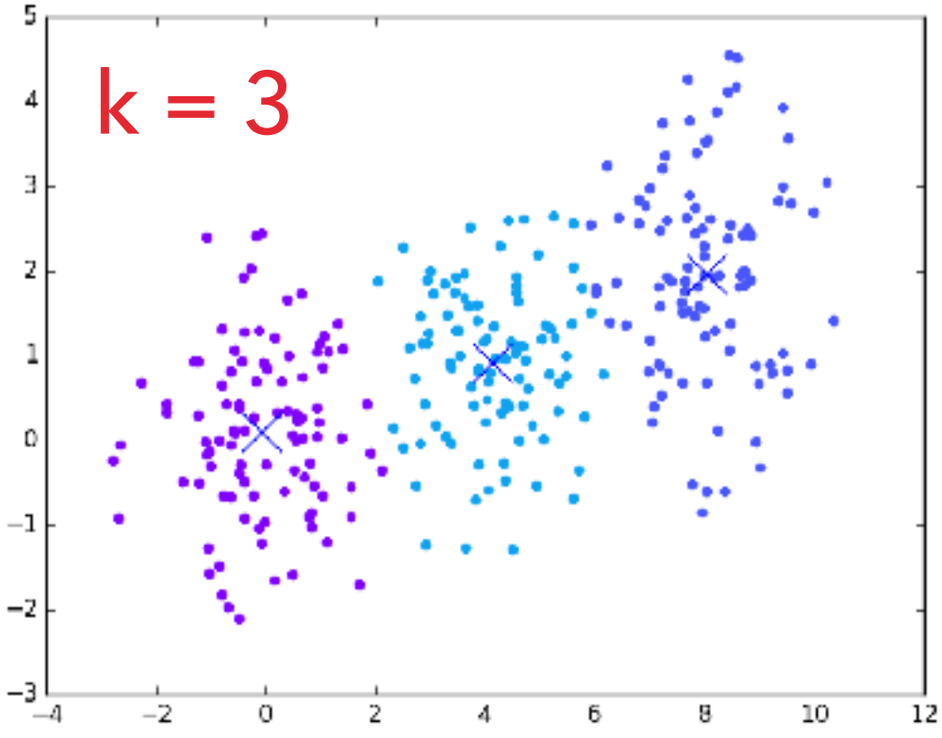
```
while previousClustering != currentClustering:
  c.previousClustering = c.currentClustering
  c.currentClustering = []
for c in cluster:
  c.mean = computeMeanPosition(c)
for point in data:
  n = findCluster(point,clusters)
  cluster[n].addPoint(point)
```

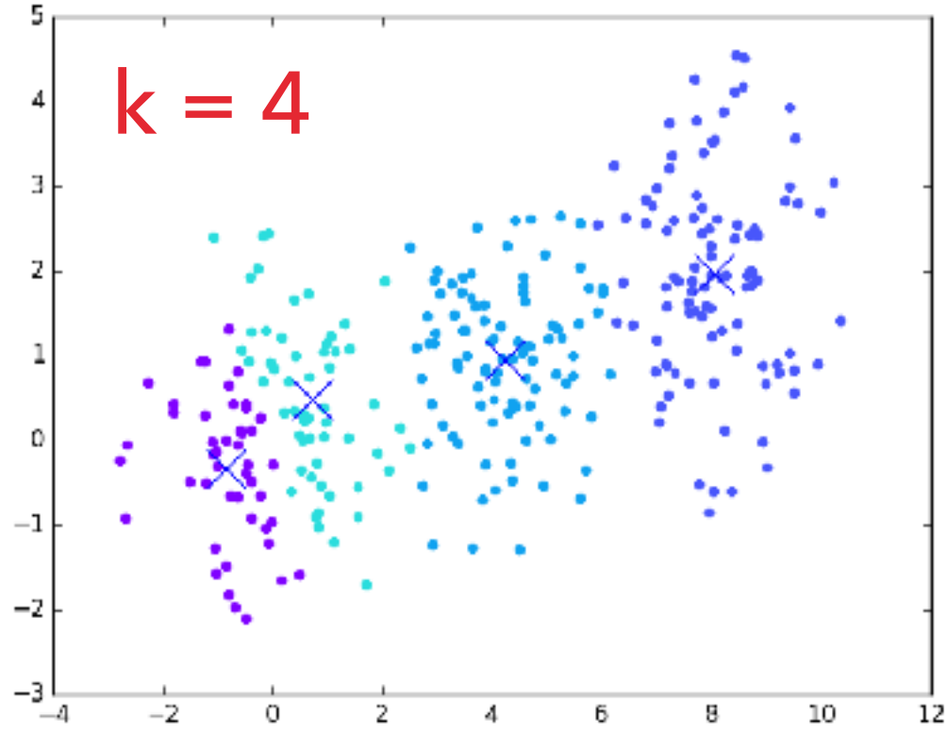
- Initialization matters. The algorithm is guaranteed to converge, but it's not guaranteed to find a global minimum.
  - What if we choose our starting points more optimally?
    - kMeans++
  - How do we test if we are at the optimal solution?
    - Advanced topics: Intertia, Silhouette Scores
- Curse of dimensionality
  - If we go from 2D to 27D, clustering becomes a really hard problem. Everything is far from everything in 27D.
  - Possible solution: Dimensionality reduction (PCA, etc)

- Naturally finds spherical clusters of data.
- Can't cluster by density, since it's always minimizing a distance function to the mean.
- How do we choose the number of clusters if we don't know a priori or via domain knowledge?
  - Finding k (number of clusters) is non-trivial, but there are some good rules of thumb (next pages)

k = 1

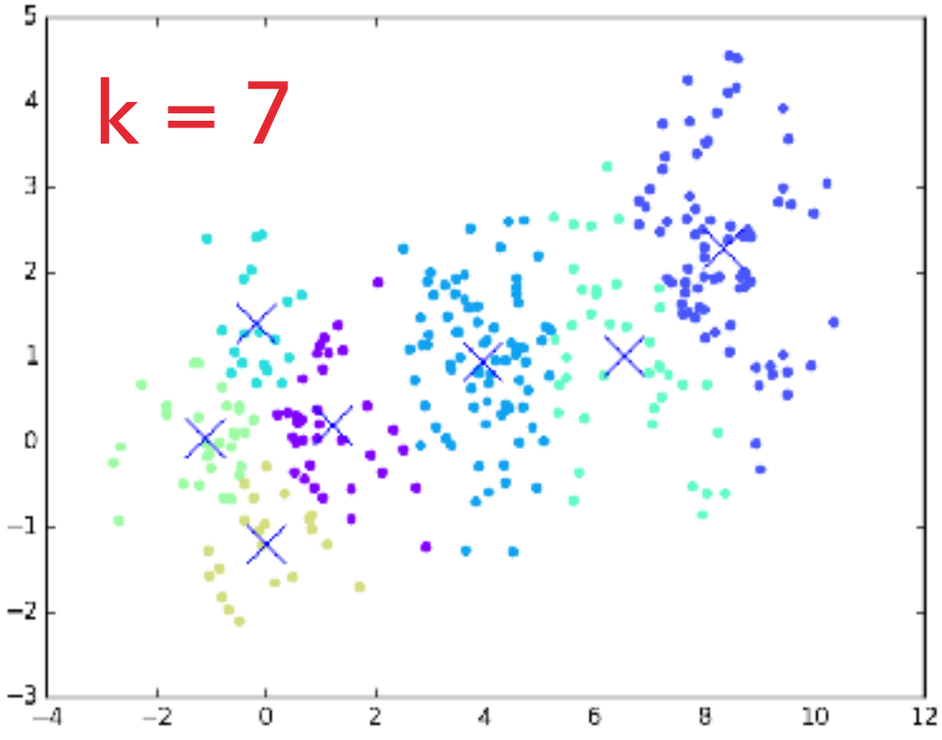
k = 2

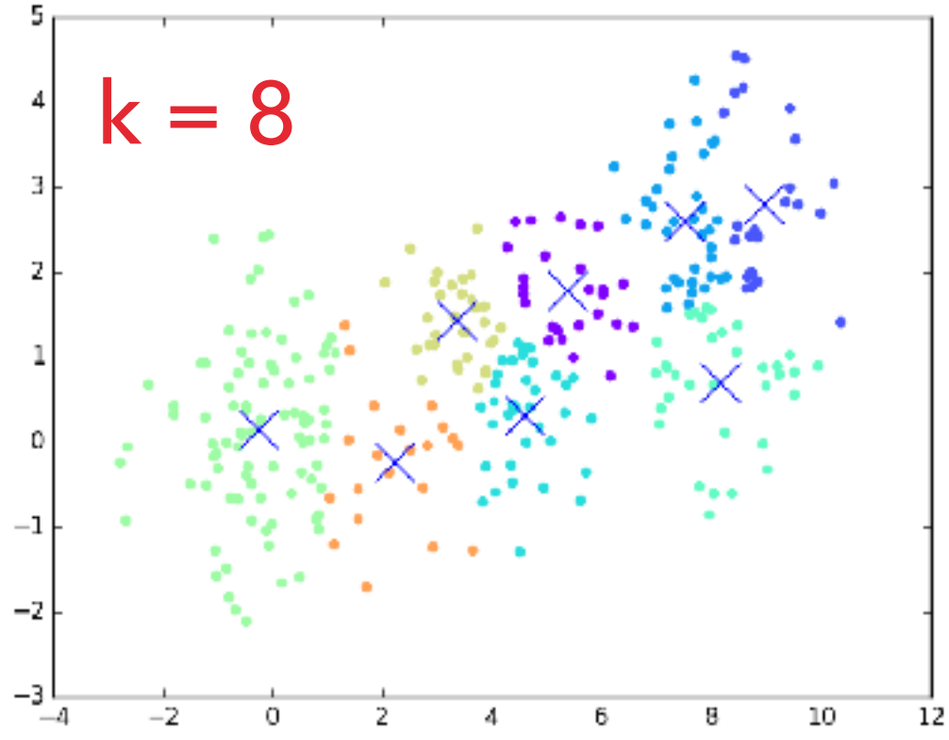

k = 3


k = 4

k = 5

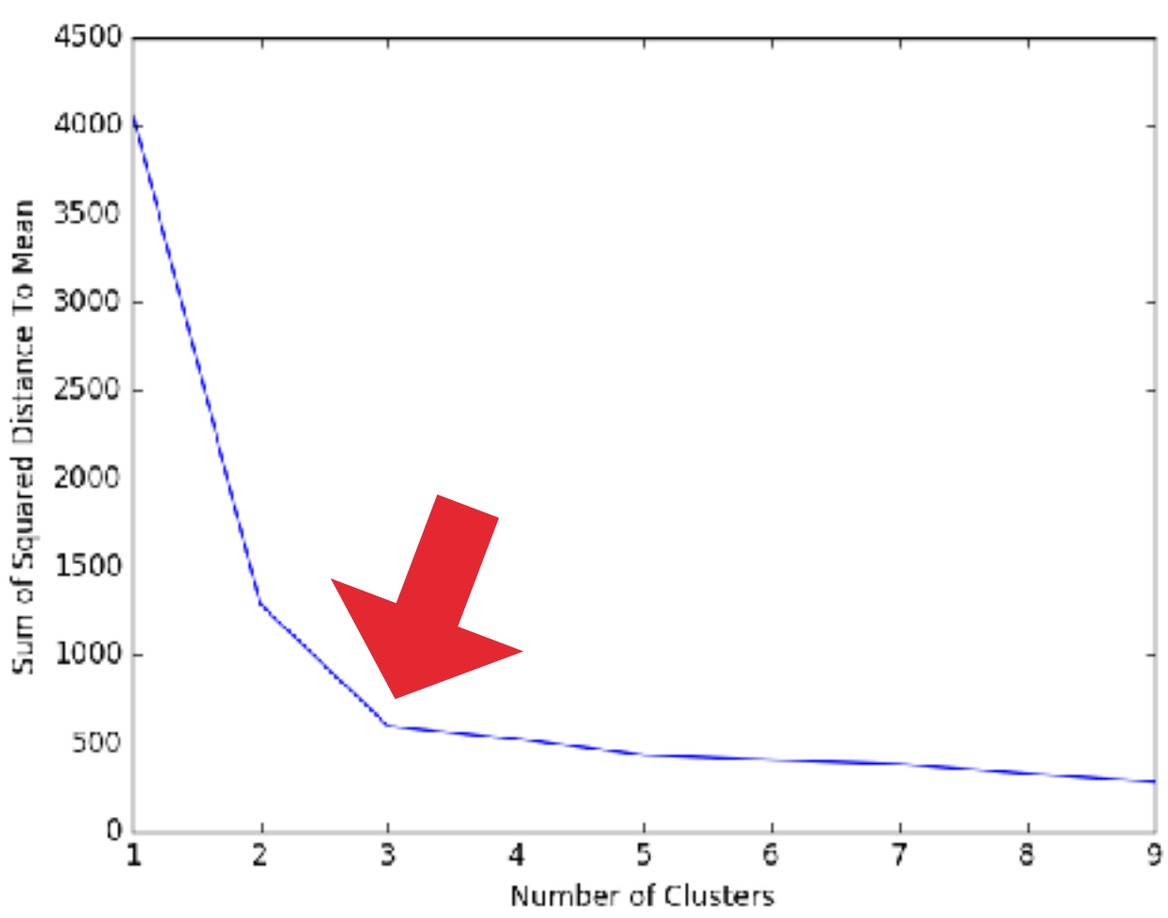
k = 6


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.

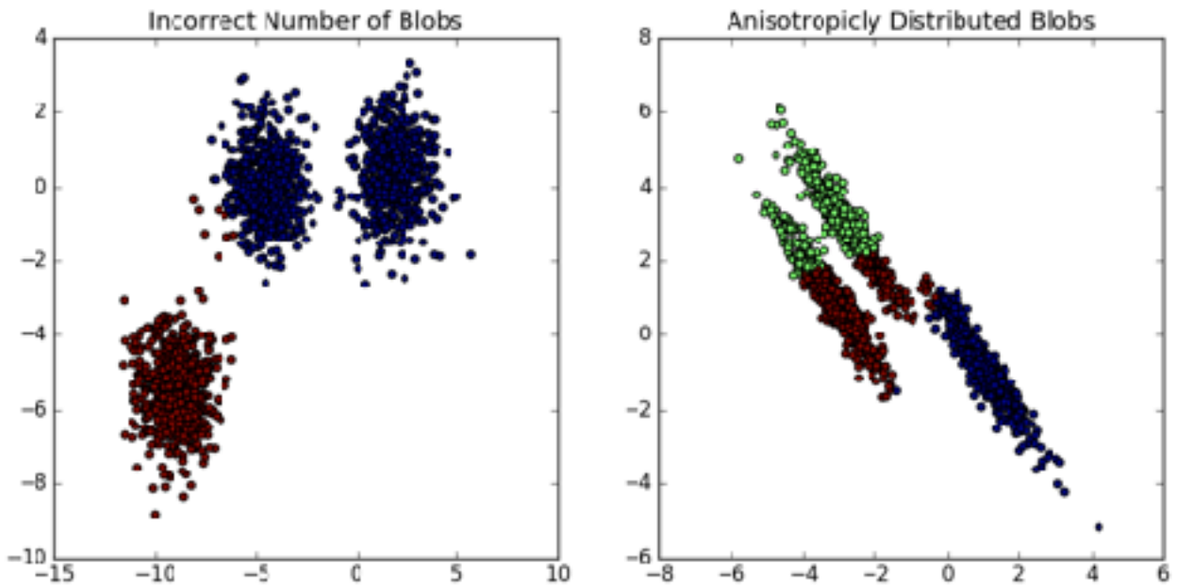
NOTE: This is just a guide. The elbow method is totally heuristic and you can use it as a starting point for your data set and evaluate from there.



- 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:
  - Finding optimal middle points in geospatial data
  - Dimensionality Reduction
  - 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_
```

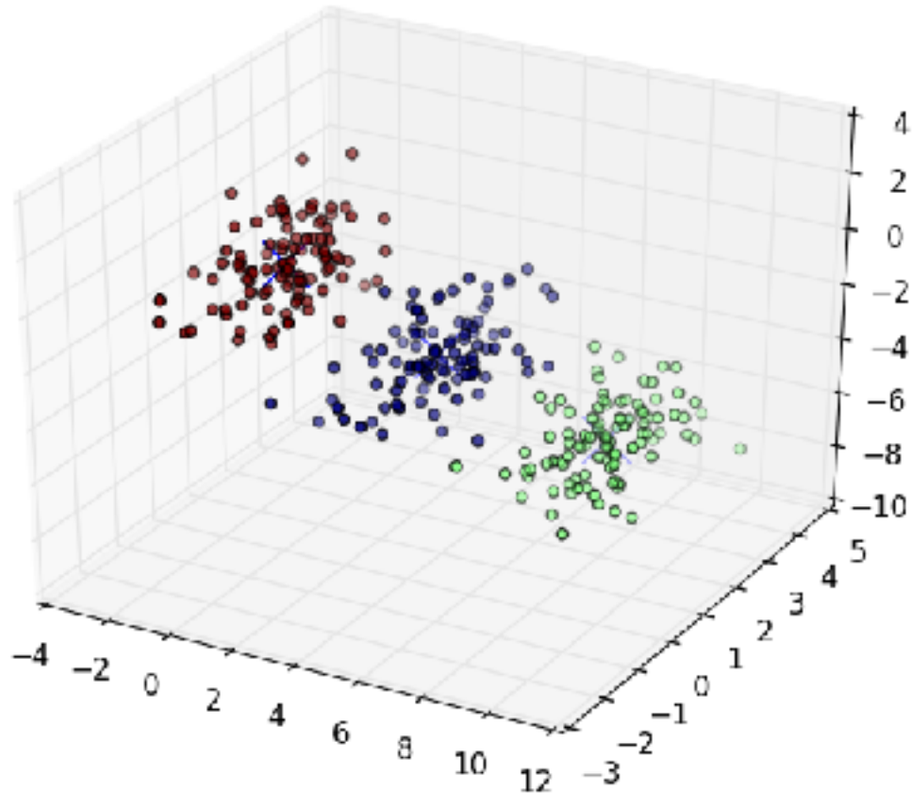
Details:

- 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

CHALLENGE TIME

- **Generate** some 3D data using numpy's random.normal function. Spread the data out into some set of N clusters. Now use sci-kit to run the k-means function. Plot your data using matplotlib's 3D plotting, and make sure to color each point as part of a cluster.
- Try playing with different initializations. What happens if n_init = 1 for close clusters? Try running again with a different random number seed.
- Try changing how you generate the data such that all the clusters are overlapping. Does k-means still work?
- Can you smear the data in a way that makes k-means return ugly or non-intuitive clusters?
- What happens if you make one of the generated clusters only have 1 or 2 points and the rest have 100s?



- 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