# $k$-means clustering

Computing Workshop: Machine Learning

$k$-means clustering discovers $k$ clusters of points in a plot. These clusters correspond to the *labels* that the clustering algorithm discovers.

1. $k$ points are randomly placed in the plot. These points are called the *centroids*.
2. Assign labels to data points in the plot according to what centroid is nearest.
3. The centroids' positions are recalculated by taking the *mean* (center) of the points assigned to each cluster.
4. Repeat steps 2 and 3 until the positions of the centroids are unchanging. We say that the clusters have *converged*.

## Questions

Visit `https://computing-workshop.com/lab.html` and follow the link to Naftali Harris's "Visualizing K-Means Clustering" blog post. Using the visualization, answer the following questions.

1. Choose initial centroids randomly, and choose the Gaussian Mixture dataset. Add four centroids by repeatedly clicking "Add Centroid". Then click "Go" and "Update Centroids" until the algorithm converges. Repeat this a few times.

   (a) Compare and contrast the results of the simulations: what was similar and what was different?

   (b) What is the significance of the initial placement of the centroids? How do you think this impacts the performance of the algorithm?

2. Choosing initial centroids randomly, run the algorithm on the "Uniform Points", "Gaussian Mixture", "Smiley Face", and "Packed Circles" data sets. Feel free to add centroids as the algorithm is running to find the optimal number $k$.

   Are some of these more amenable to $k$-means clustering than others? Why?

3. Based on your experience, what are some disadvantages of $k$-means clustering? How could you improve $k$-means clustering?

## The DBSCAN Algorithm

Return to `https://computing-workshop.com/lab.html` to visit Naftali Harris's blog post "Visualizing DBSCAN Clustering". This is a different clustering algorithm, based on the idea that the points in a cluster should be densely packed.

In the visualizer, choose the "DBSCAN Rings" dataset.

4. Move the right slider all the way to the right. This affects the `minPoints` value in the bottom left; now the value should be 6. Press "Go".

   In the resulting clusters, how many points are in each cluster?

   Restart the simulation and choose "DBSCAN Rings" again. Choose `minPoints = 2` this time. Press "Go".

   What do you think the effect of `minPoints` is?

5. Restart the simulation on the "DBSCAN Rings" dataset again. Play with the left slider, which affects the `epsilon` value. What is the effect of this parameter on the performance of the algorithm?

6. Try DBSCAN on the "Smiley Face" dataset. Compare its performance against the $k$-means clustering you did before.

7. In general, when do you think it's more appropriate to use $k$-means clustering or DBSCAN?