Clustering

PNI Summer Internship 2020
Mai Nguyen

Overview

- Supervised vs unsupervised learning
- k-means clustering
- Hierarchical clustering (agglomerative)
- When to use what?

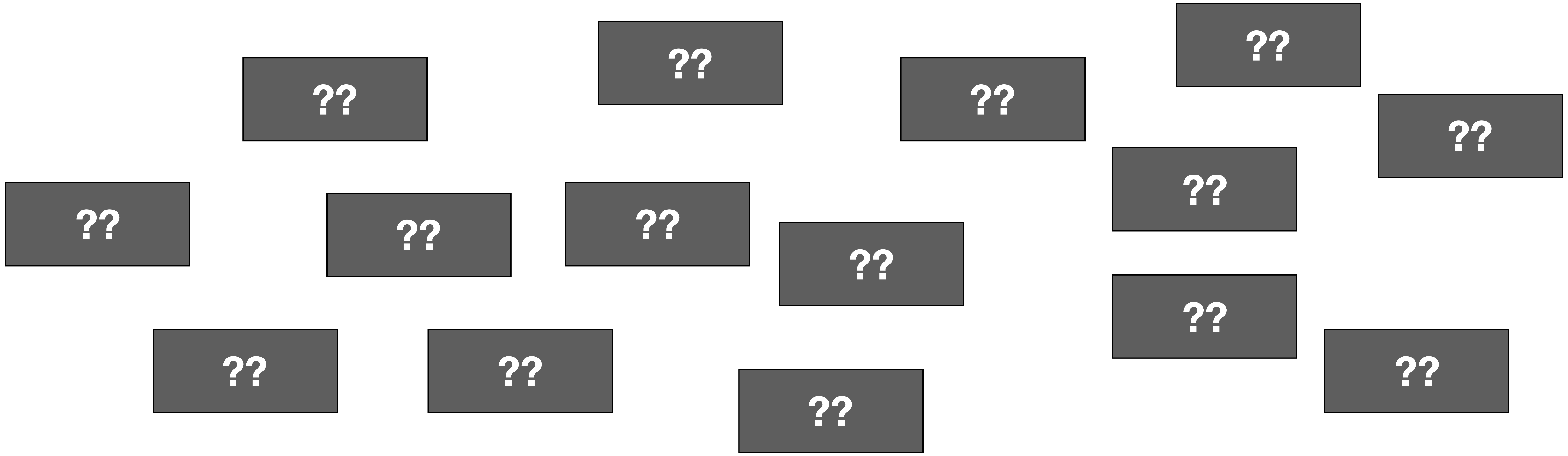
Supervised learning

- Know the “ground truth,” or the actual labels, categories, or groups of each data point

- Classifiers are canonical form of supervised learning

Classifiers are a form of supervised learning

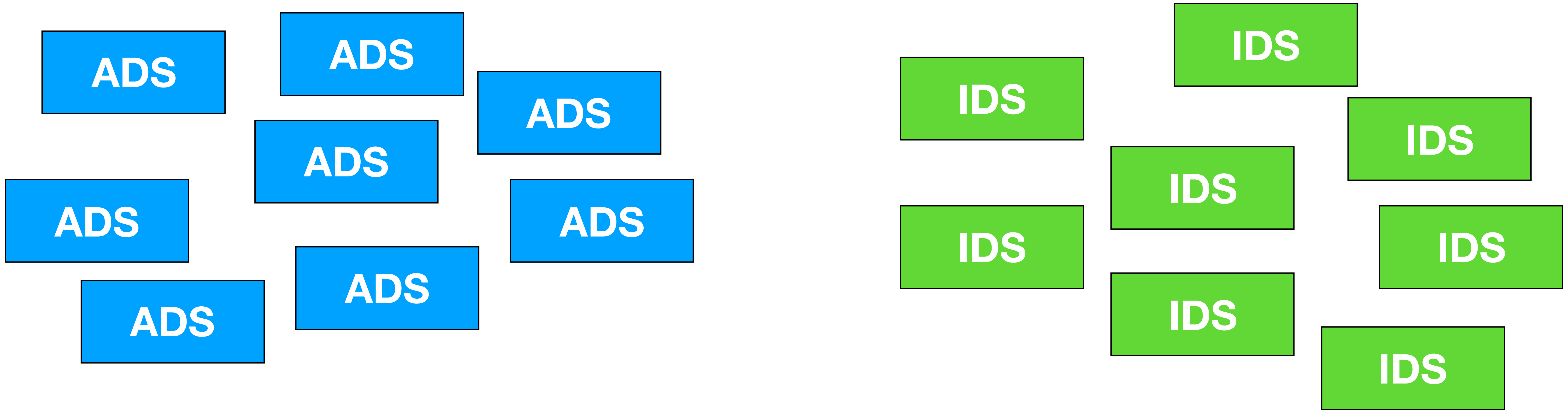
Training data



Experimenter assigns each utterance to either ADS or IDS
(based on some criteria)

Classifiers are a form of supervised learning

Training data



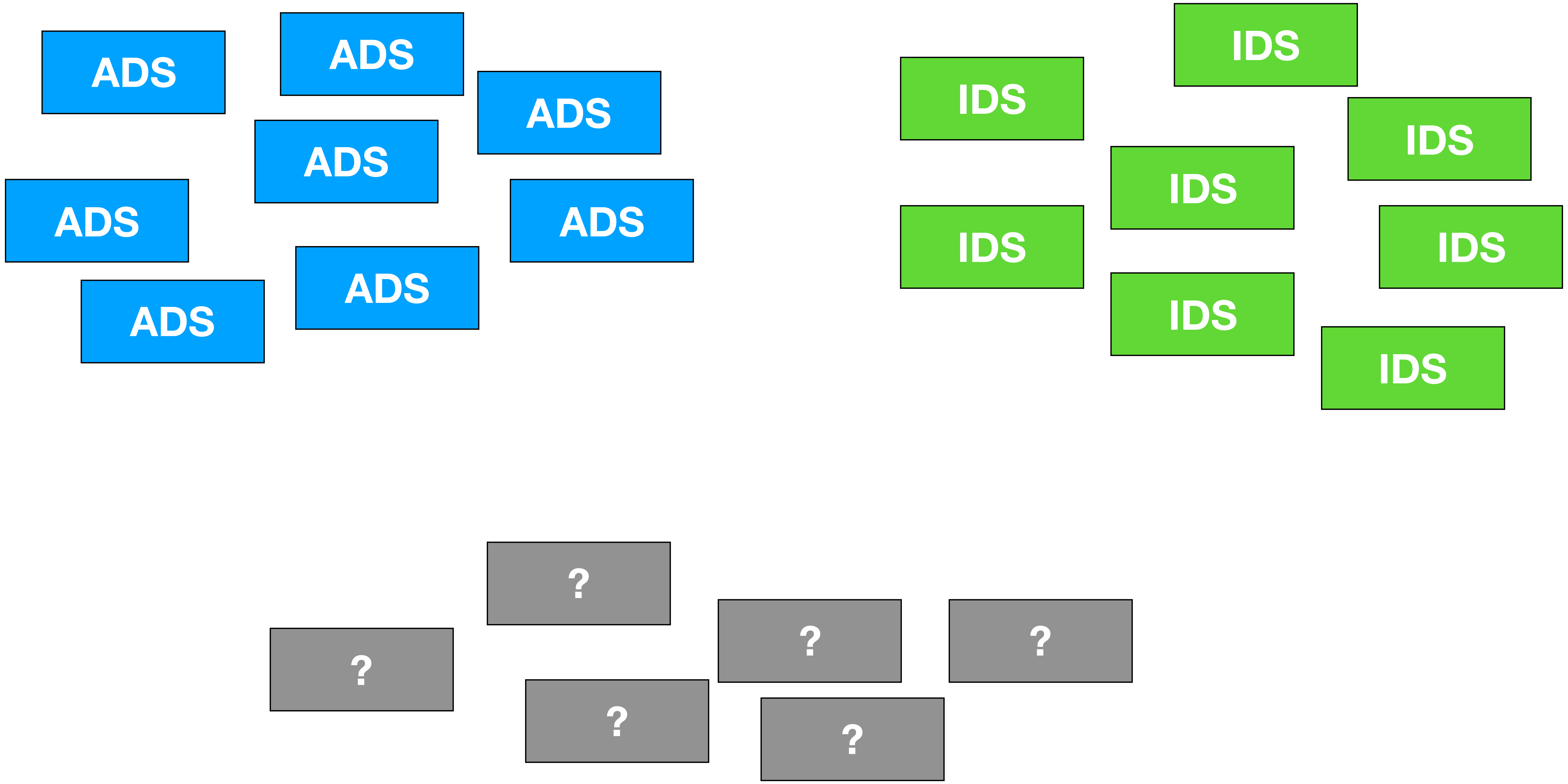
Experimenter assigns each utterance to either ADS or IDS
(based on some criteria)

Classifier is trained on these labels and learns to distinguish ADS from IDS

Classifiers are a form of supervised learning

Training data

Test data

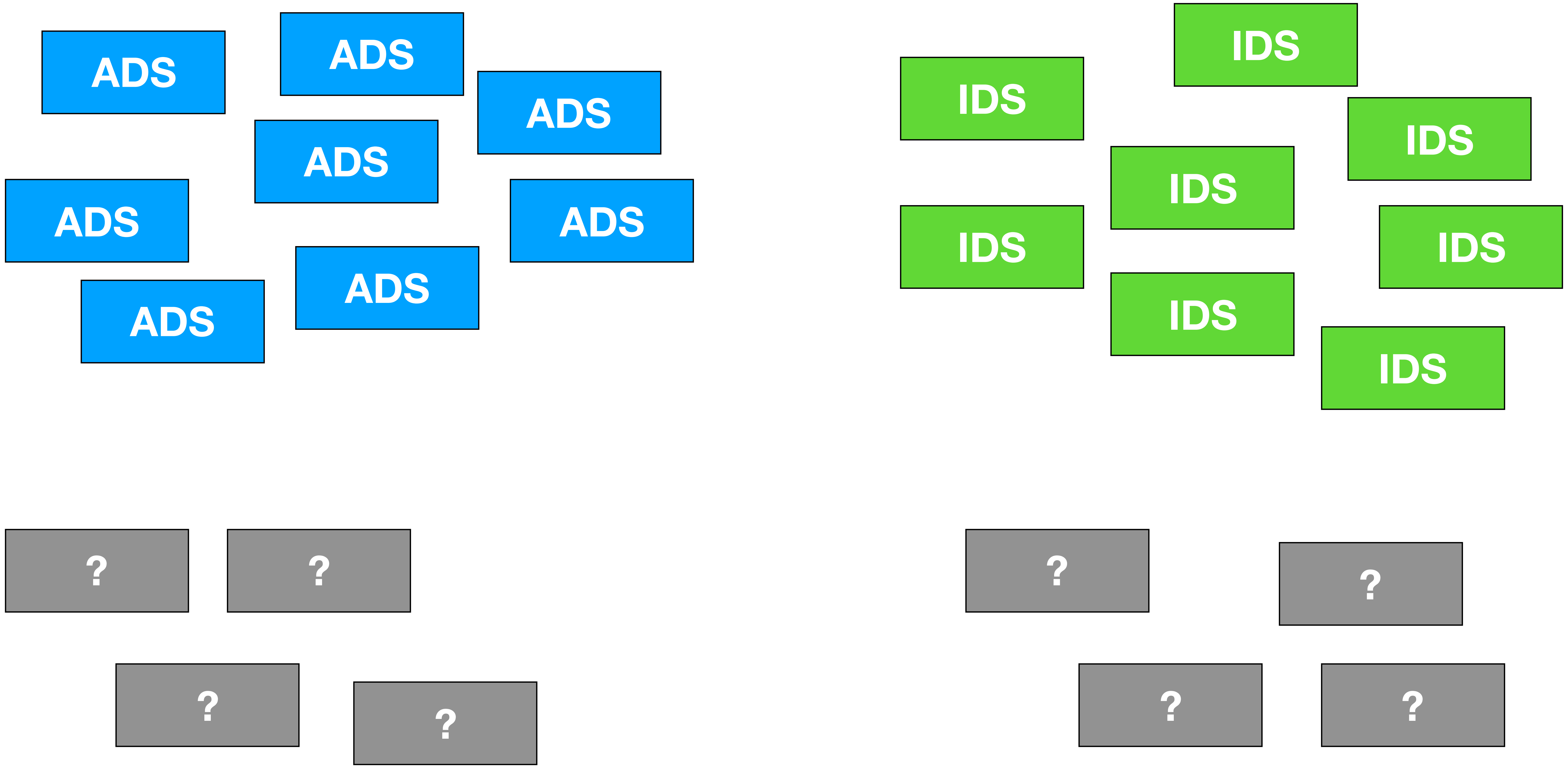


Classifier “guesses” whether each new utterance is ADS or IDS

Classifiers are a form of supervised learning

Training data

Test data

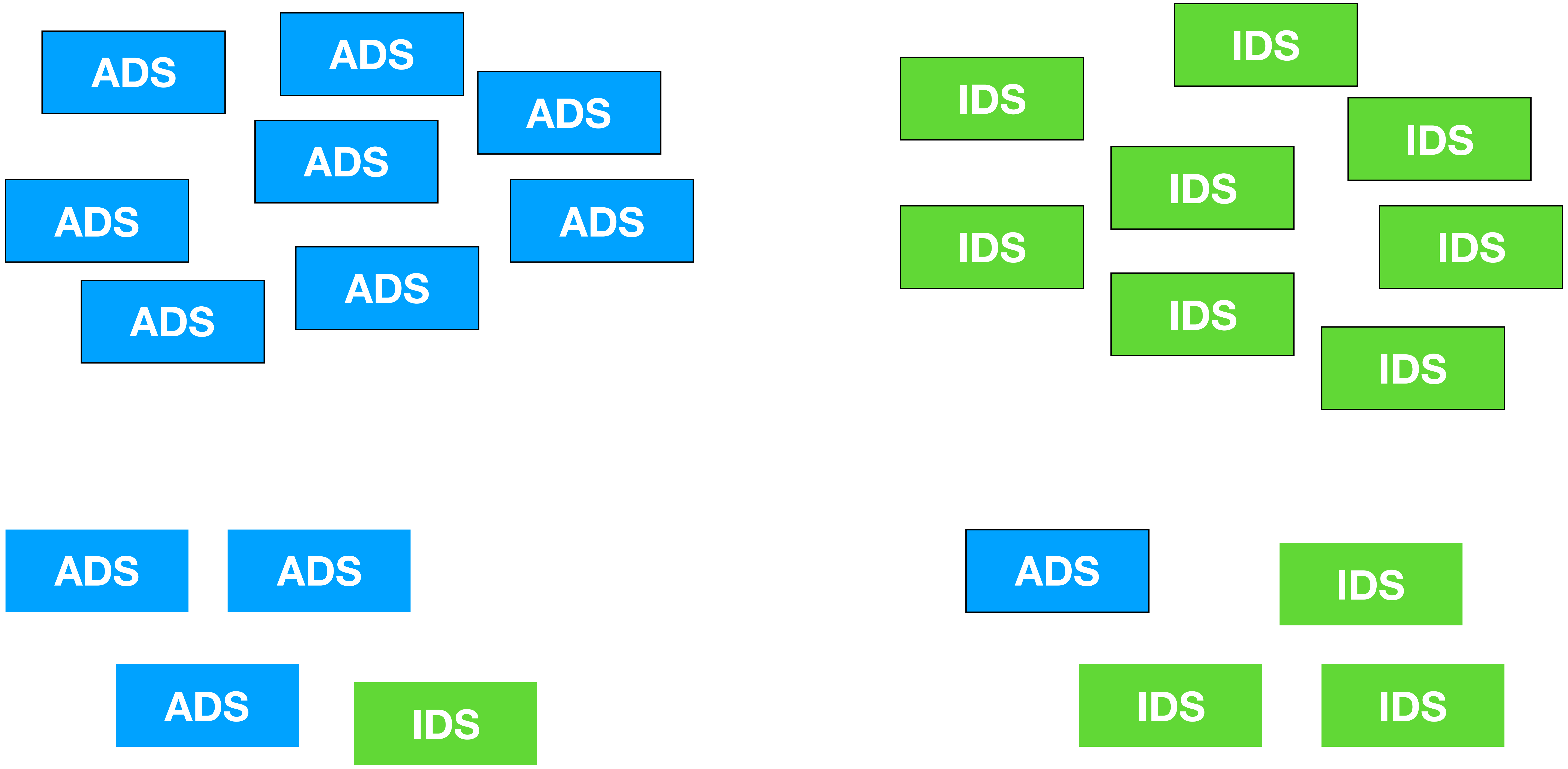


Classifier “guesses” whether each new utterance is ADS or IDS

Classifiers are a form of supervised learning

Training data

Test data

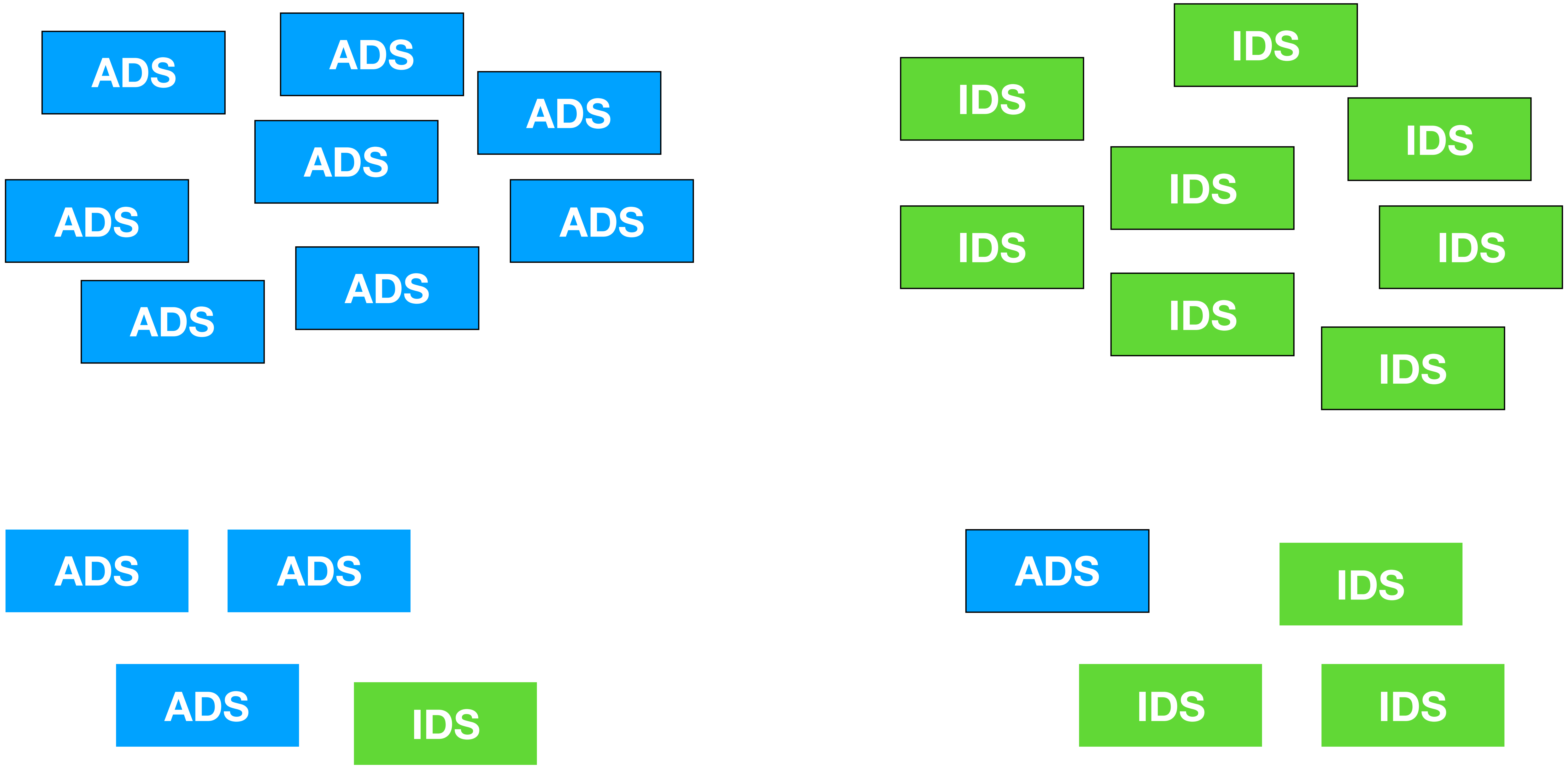


But we actually know the “ground truth,” or what type of speech each of these new utterances are

Classifiers are a form of supervised learning

Training data

Test data

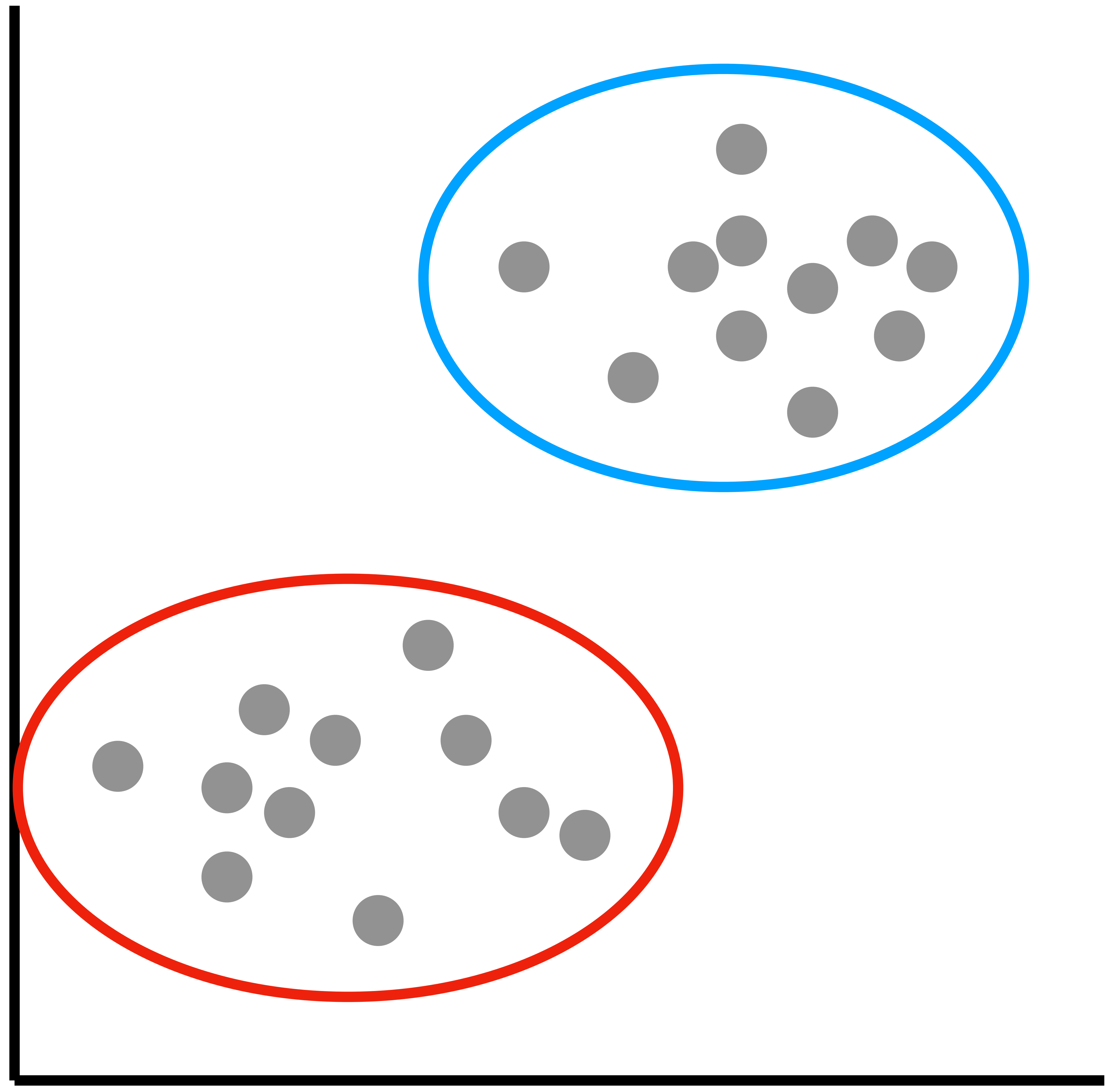


Allows for evaluating accuracy of the classifier: 75%

Unsupervised learning

- Don’t know what the groups (labels, conditions, etc) in the data are, or even if there are sensible groups
- Data-driven method
- Infer the structure of a dataset

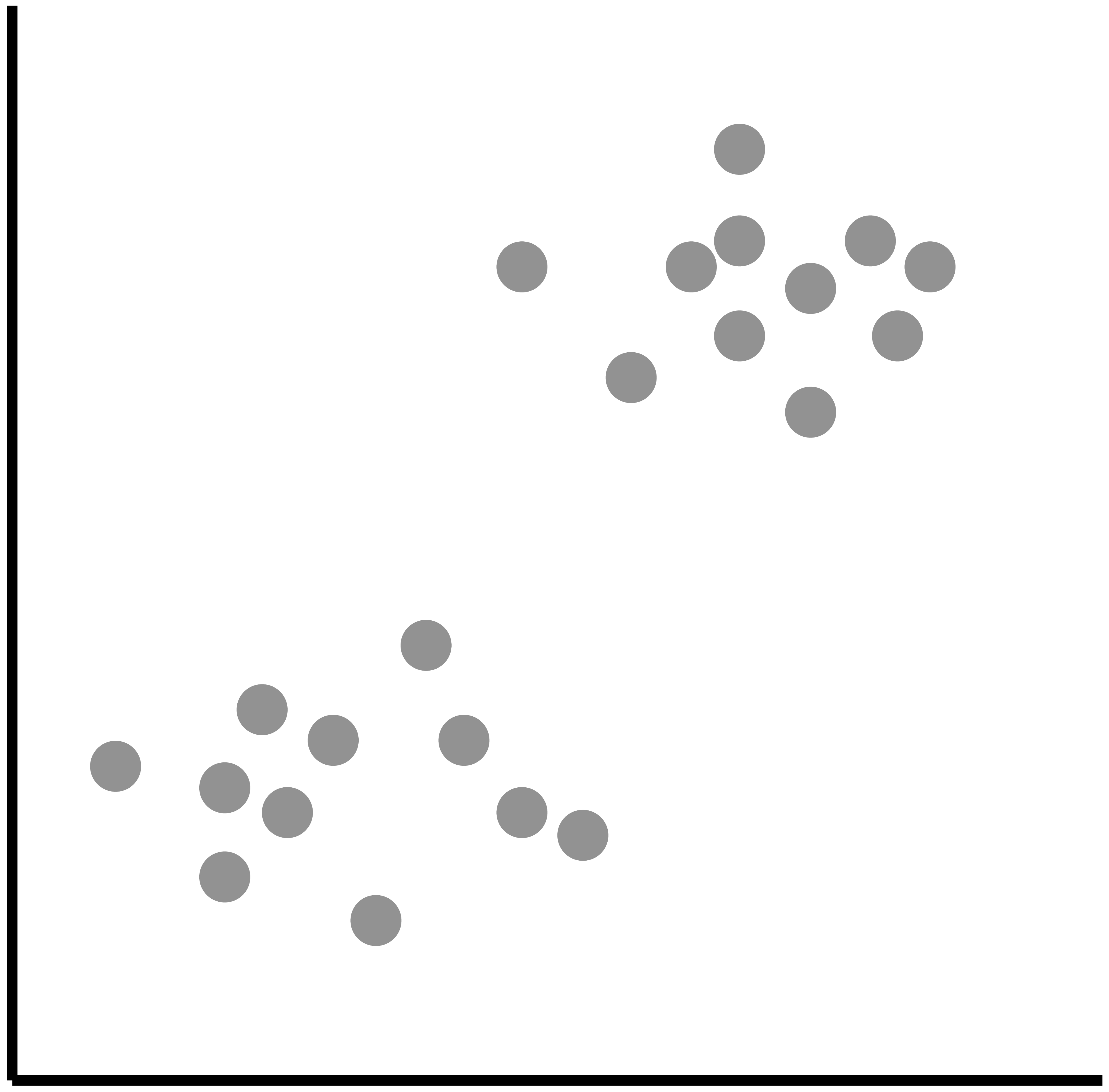
Clustering is a form of unsupervised learning



Types of clustering

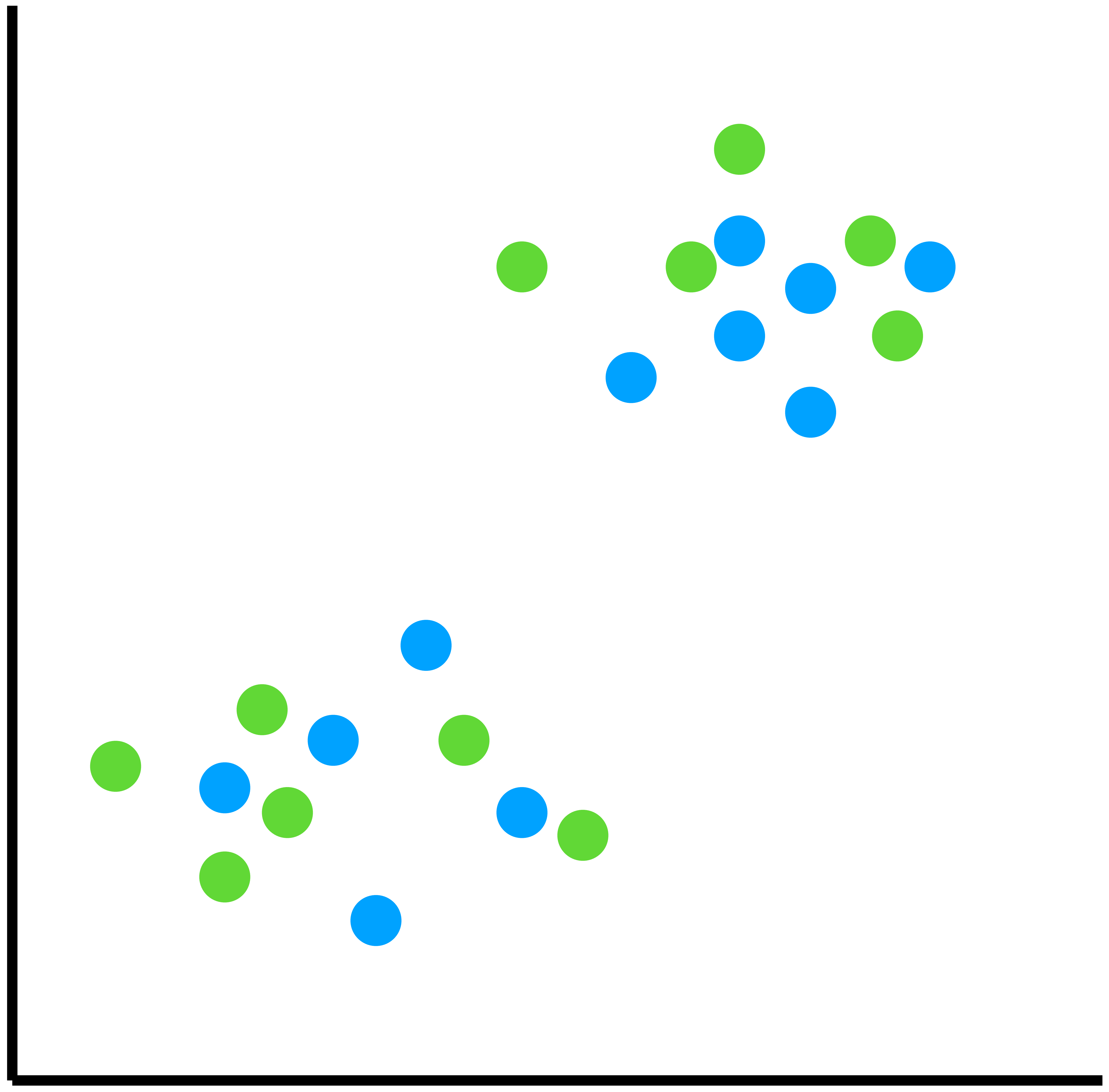
- **k means**
- **hierarchical (agglomerative)**
- Gaussian mixture models
- Density-based
- Distribution-based
- Fuzzy

k-means clustering



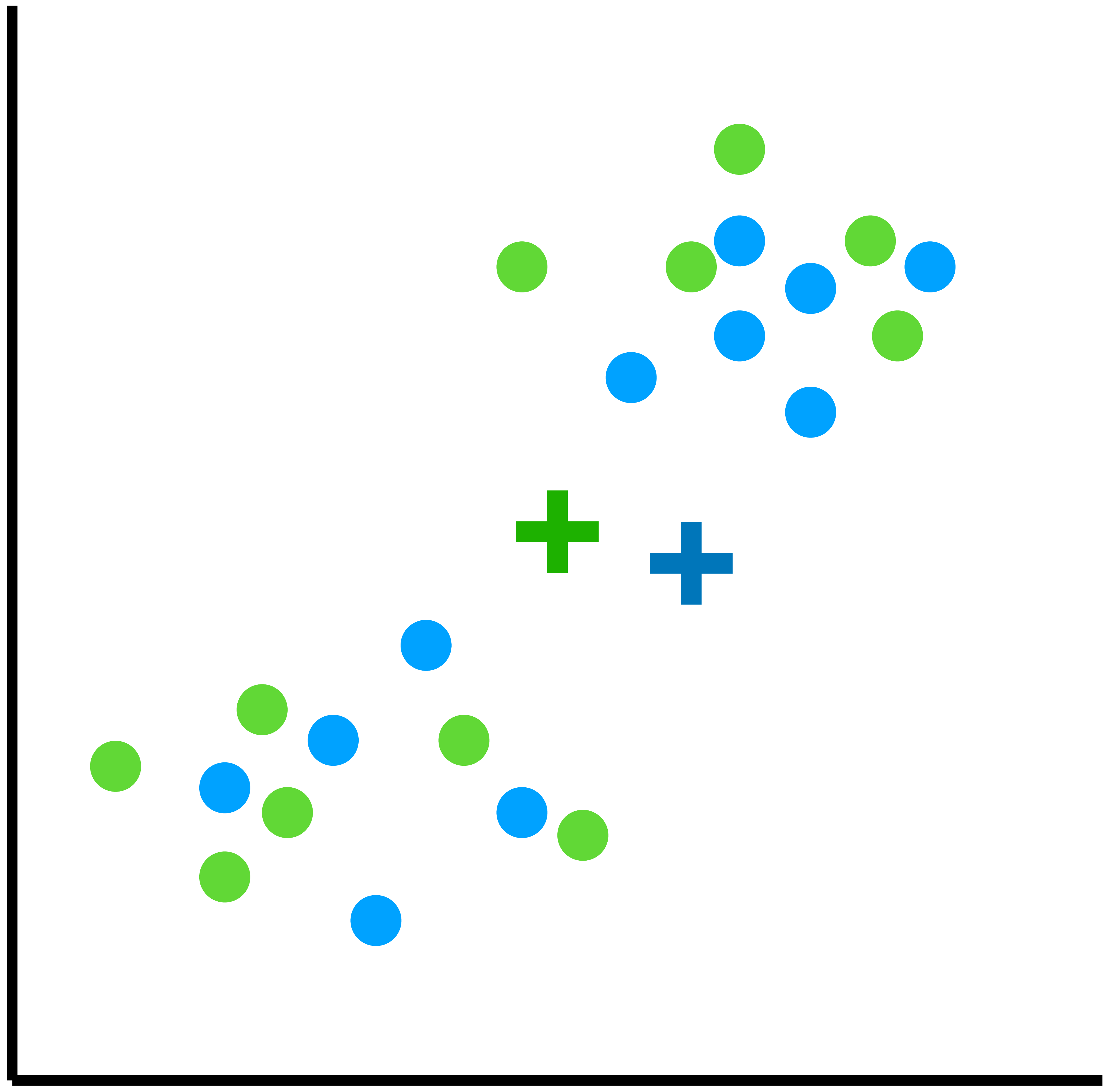
- **Step 0**: Specify # of clusters

k-means clustering



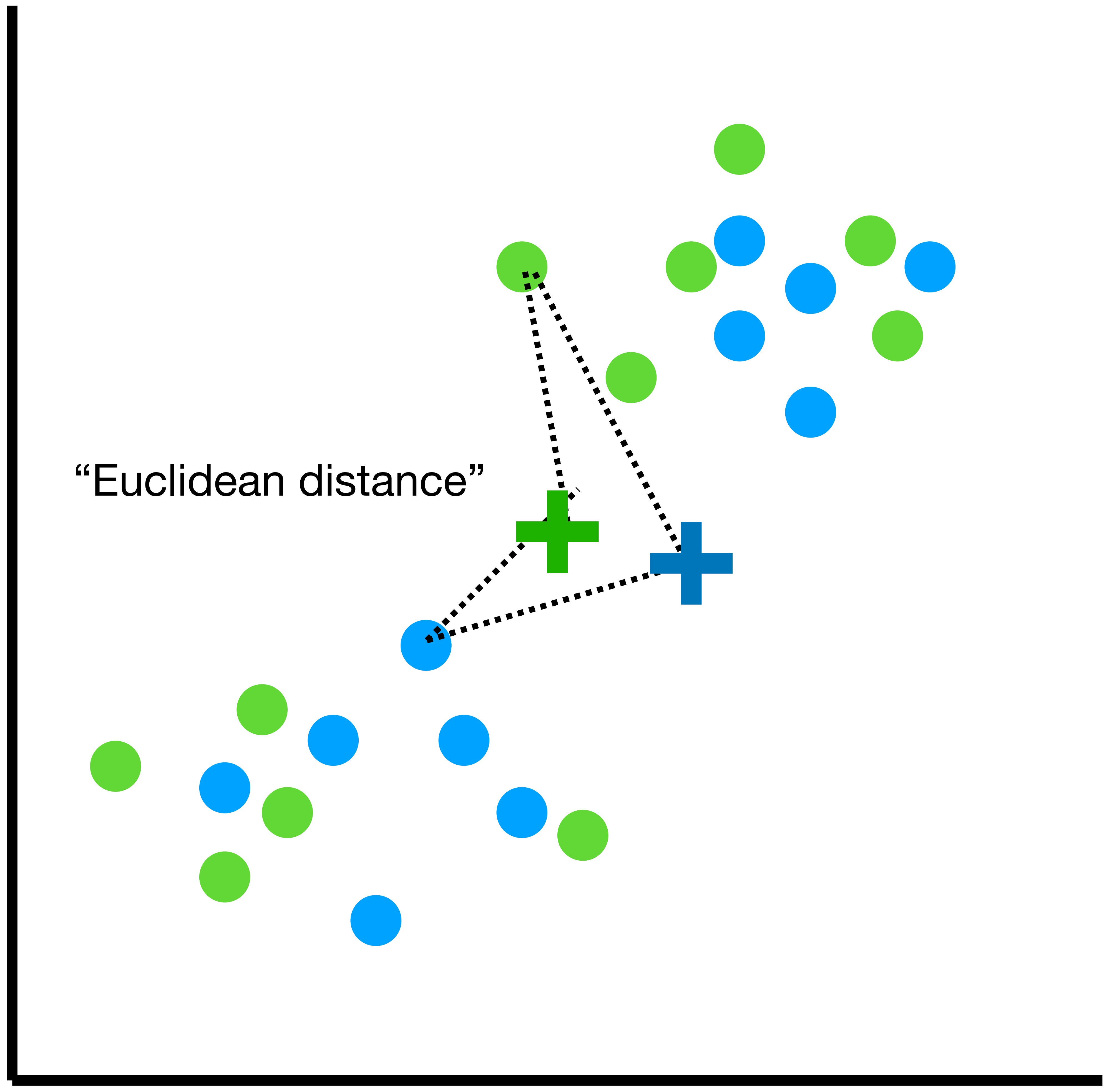
- **Step 0**: Specify # of clusters
- **Step 1**: Randomly assign each data point to a cluster

k-means clustering



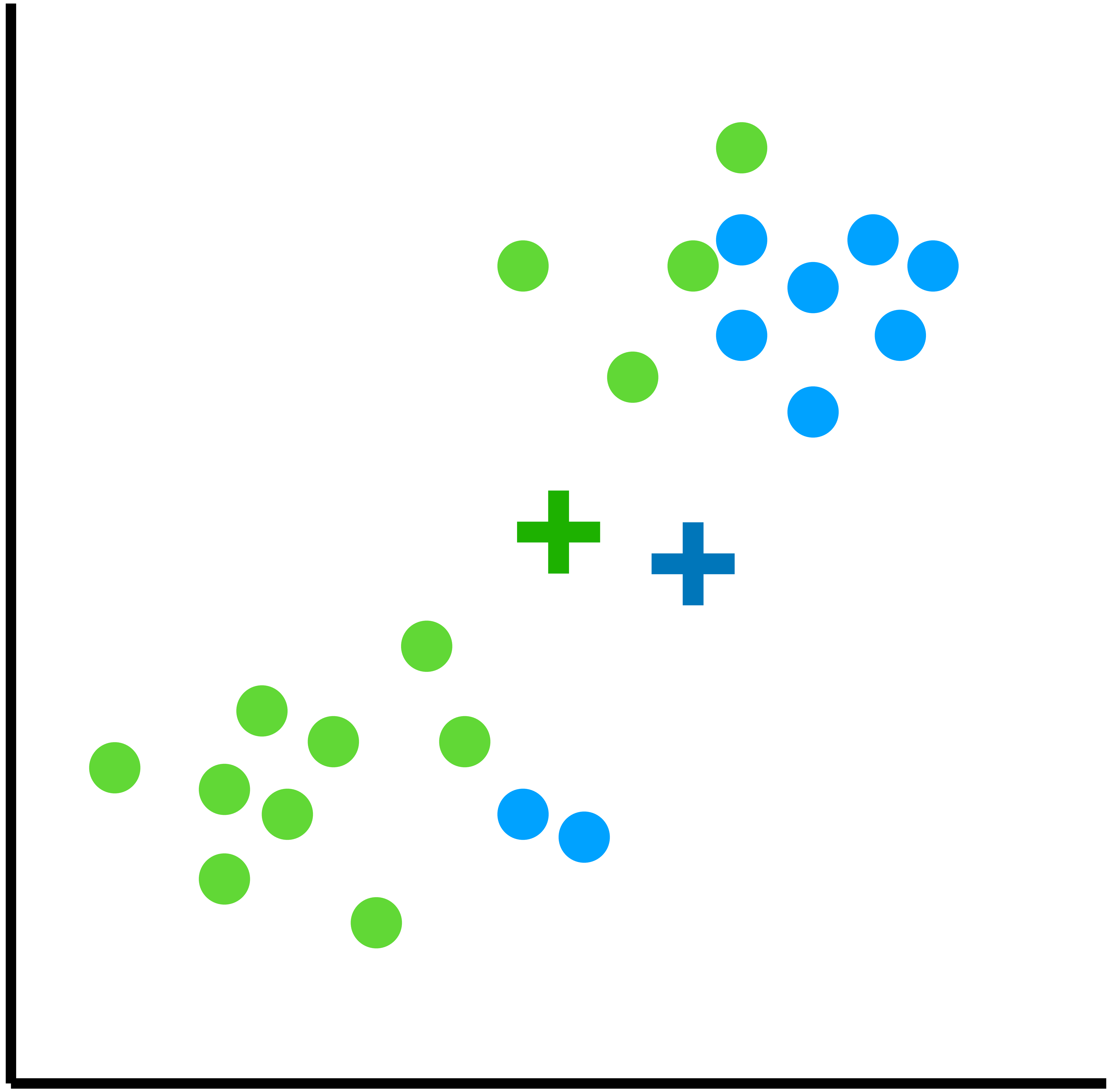
- **Step 0**: Specify # of clusters
- **Step 1**: Randomly assign each data point to a cluster
- **Step 2**: Compute centroid (middle point)

k-means clustering



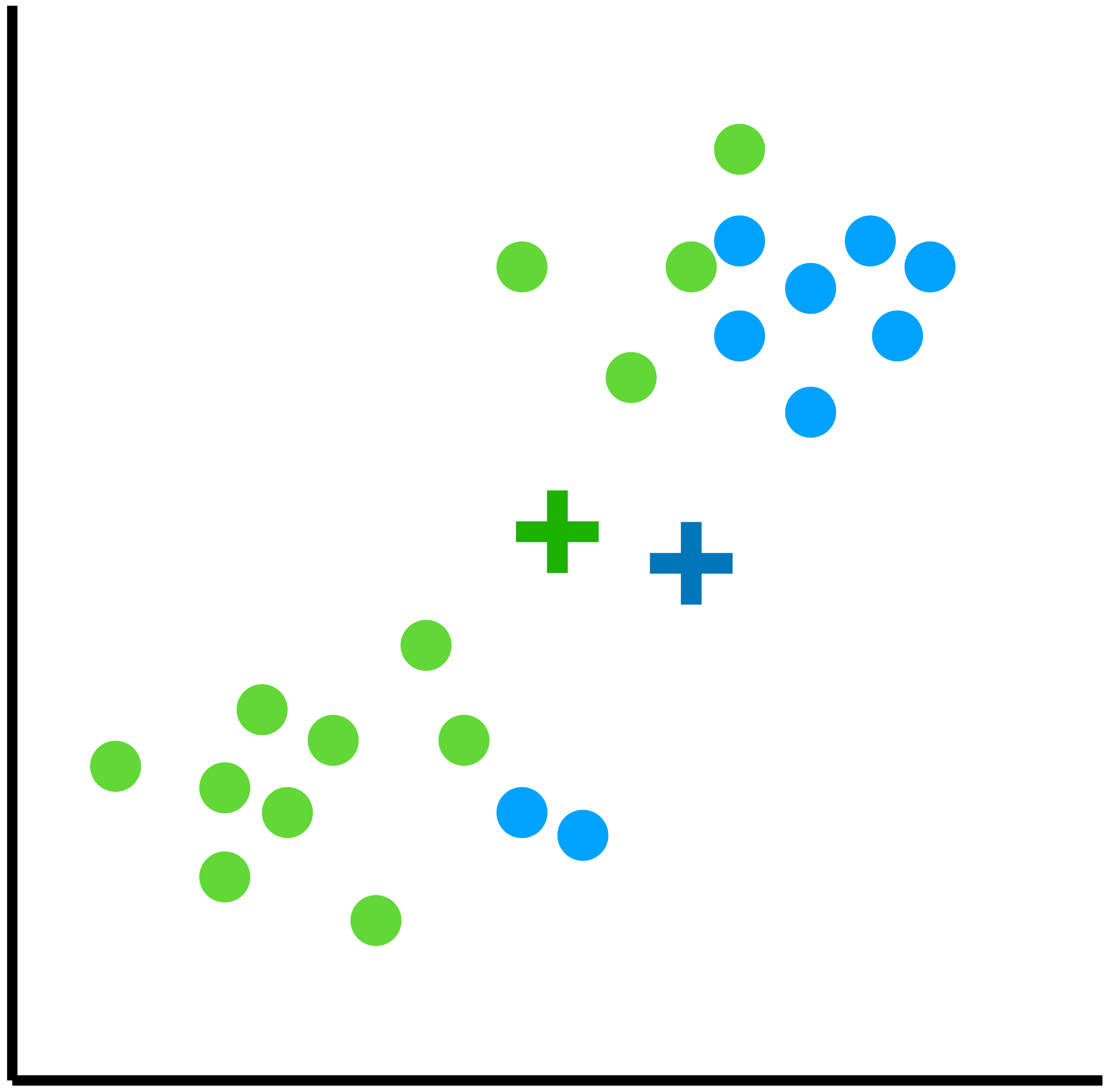
- **Step 0**: Specify # of clusters
- **Step 1**: Randomly assign each data point to a cluster
- **Step 2**: Compute centroid (middle point)
- **Step 3**: Compute sum of squared distances between each data point and each centroid

k-means clustering



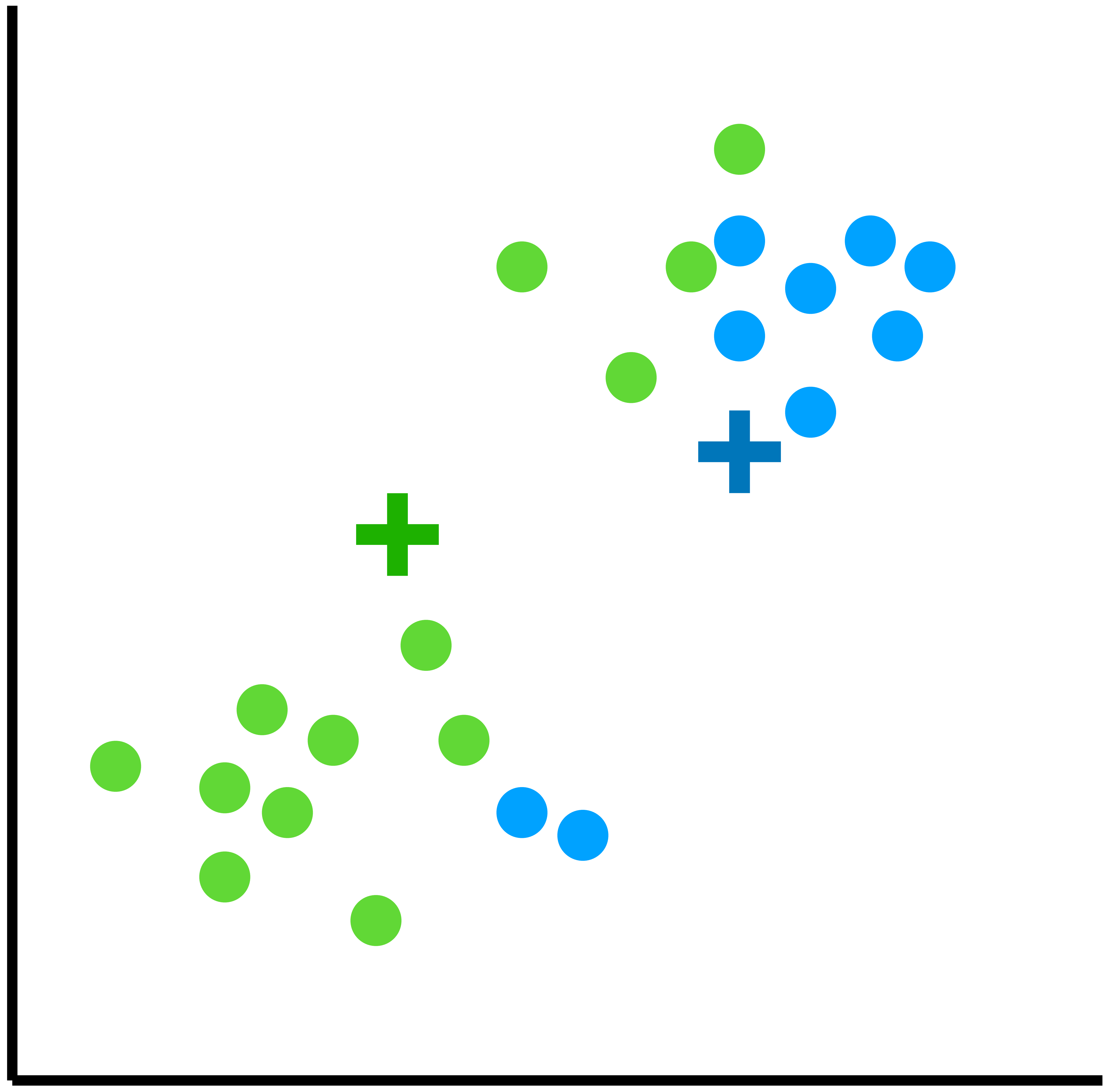
- **Step 0**: Specify # of clusters
- **Step 1**: Randomly assign each data point to a cluster
- **Step 2**: Compute centroid (middle point)
- **Step 3**: Compute sum of squared distances between each data point and each centroid
- **Step 4**: Re-assign each data point to closet centroid (cluster)

k-means clustering



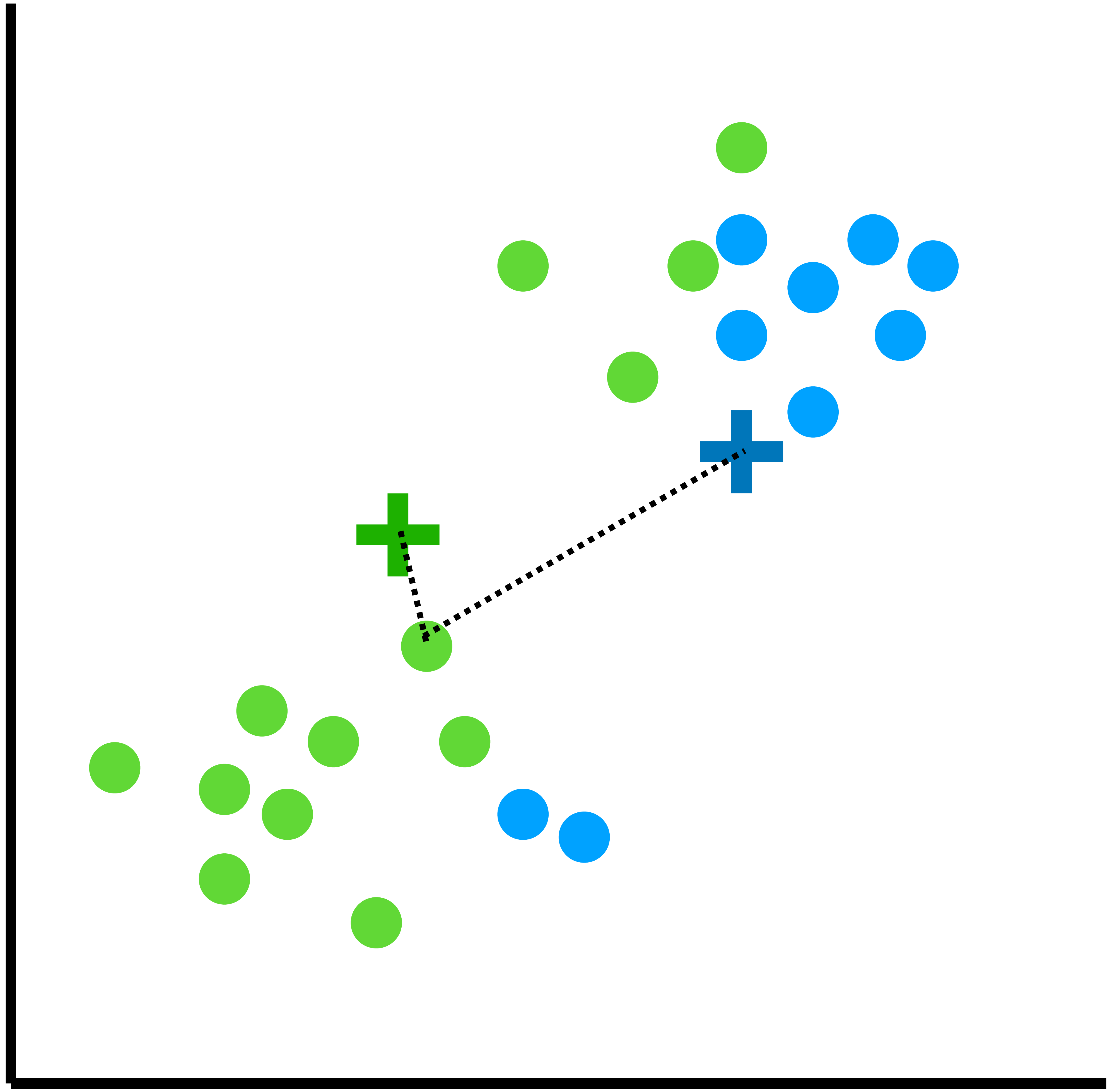
- **Step 0**: Specify # of clusters
- **Step 1**: Randomly assign each data point to a cluster
- **Step 2**: Compute centroid (middle point)
- **Step 3**: Compute sum of squared distances between each data point and each centroid
- **Step 4**: Re-assign each data point to closet centroid (cluster)
- Repeat 2-4 until centroids don't change much

k-means clustering



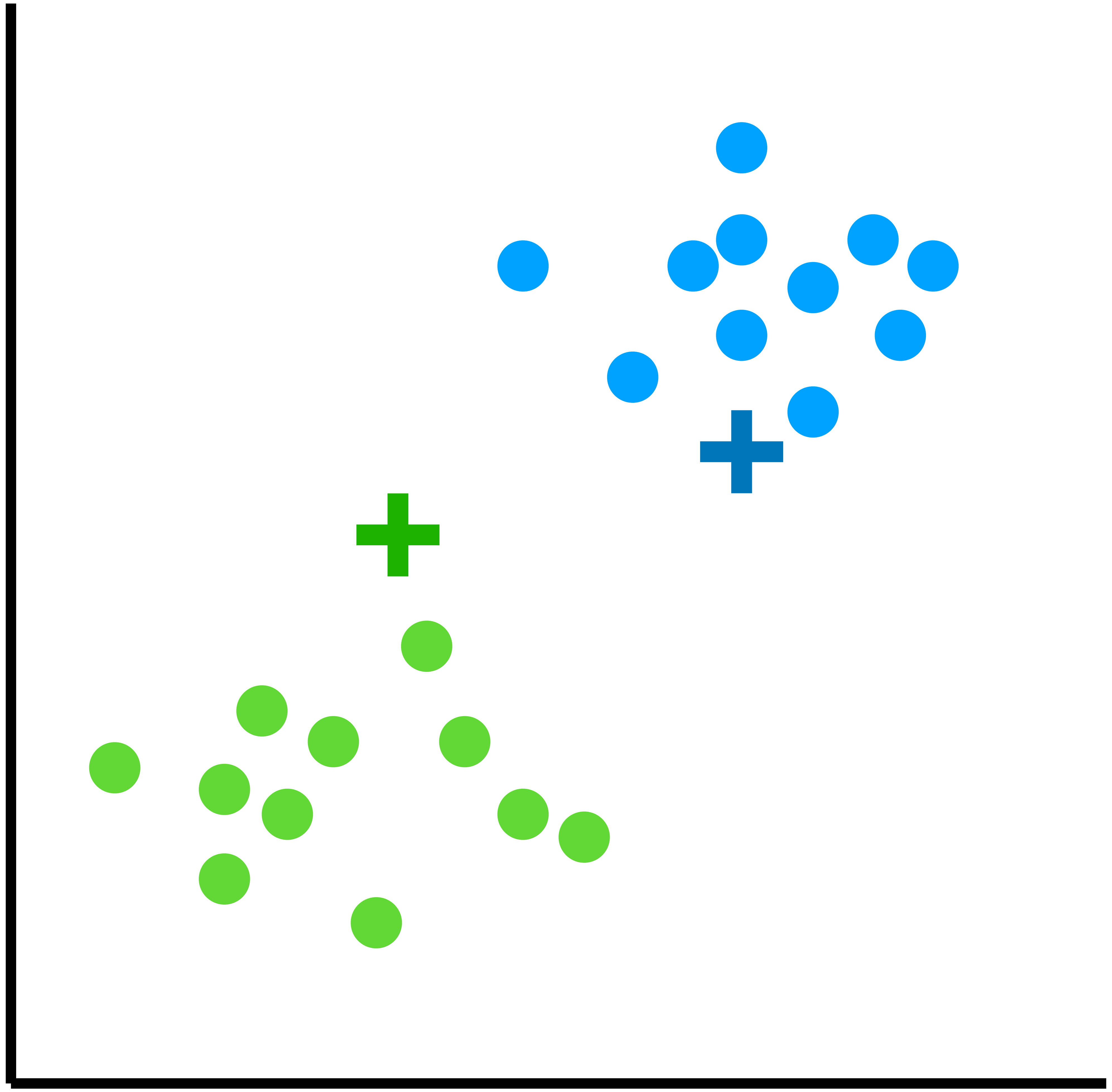
- **Step 0**: Specify # of clusters
- **Step 1**: Randomly assign each data point to a cluster
- **Step 2: Compute centroid (middle point)**
- **Step 3**: Compute sum of squared distances between each data point and each centroid
- **Step 4**: Re-assign each data point to closet centroid (cluster)
- Repeat 2-4 until centroids don't change much

k-means clustering



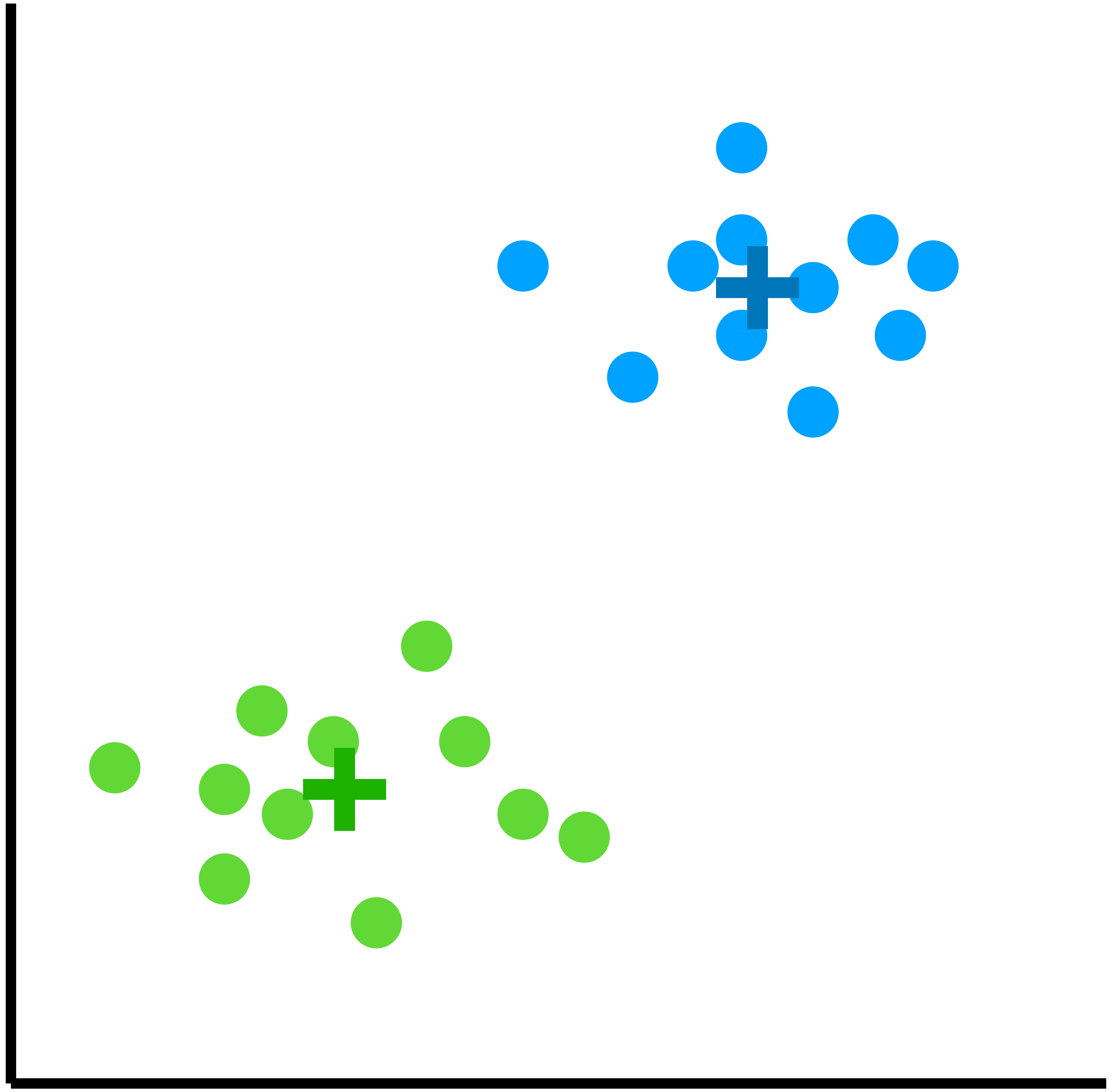
- **Step 0**: Specify # of clusters
- **Step 1**: Randomly assign each data point to a cluster
- **Step 2**: Compute centroid (middle point)
- **Step 3: Compute sum of squared distances between each data point and each centroid**
- **Step 4**: Re-assign each data point to closet centroid (cluster)
- Repeat 2-4 until centroids don’t change much

k-means clustering



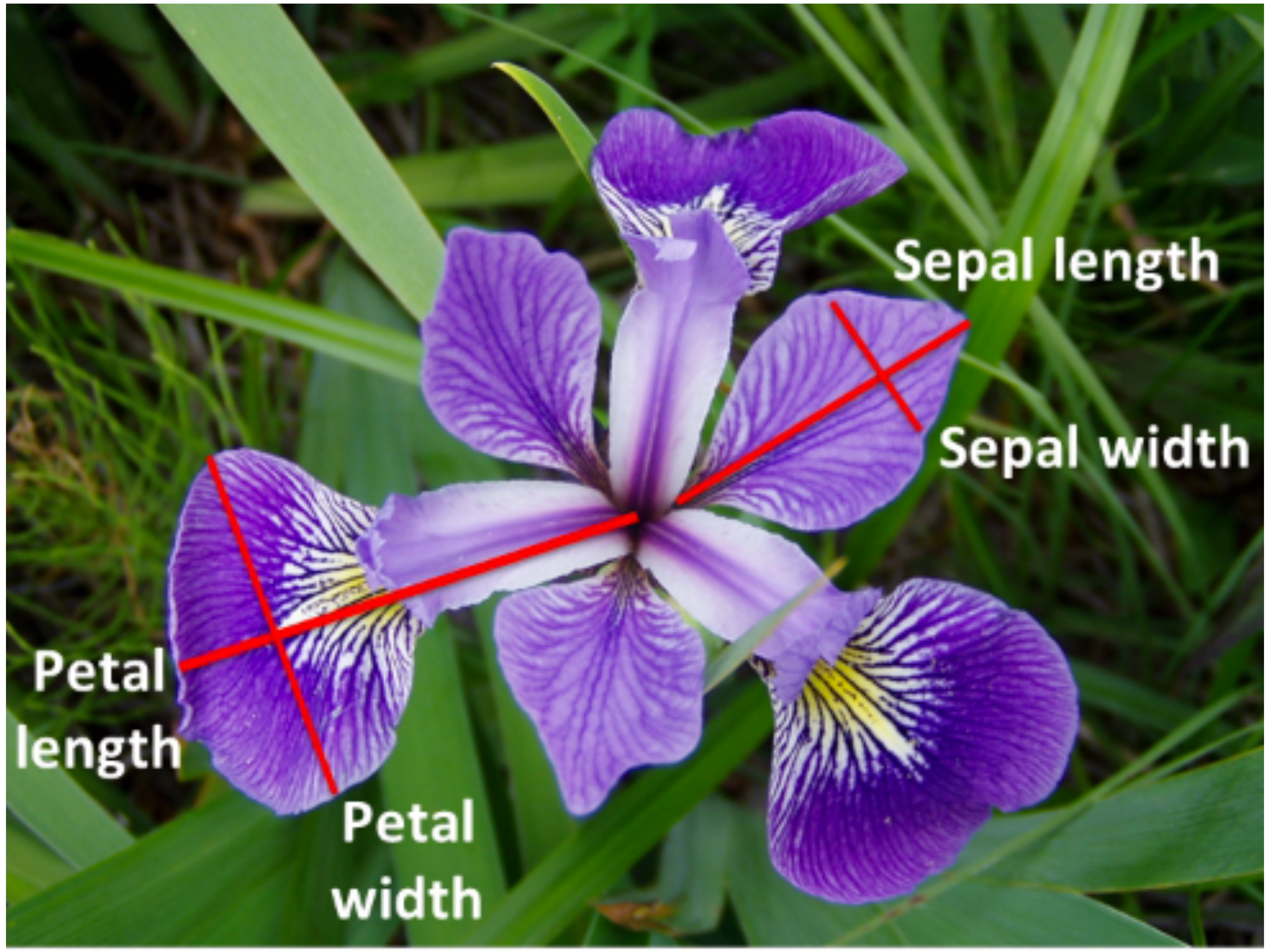
- **Step 0**: Specify # of clusters
- **Step 1**: Randomly assign each data point to a cluster
- **Step 2**: Compute centroid (middle point)
- **Step 3**: Compute sum of squared distances between each data point and each centroid
- **Step 4: Re-assign each data point to closet centroid (cluster)**
- Repeat 2-4 until centroids don’t change much

k-means clustering



- **Step 0**: Specify # of clusters
- **Step 1**: Randomly assign each data point to a cluster
- **Step 2: Compute centroid (middle point)**
- **Step 3**: Compute sum of squared distances between each data point and each centroid
- **Step 4**: Re-assign each data point to closet centroid (cluster)
- Repeat 2-4 until centroids don't change much

k-means clustering in MATLAB



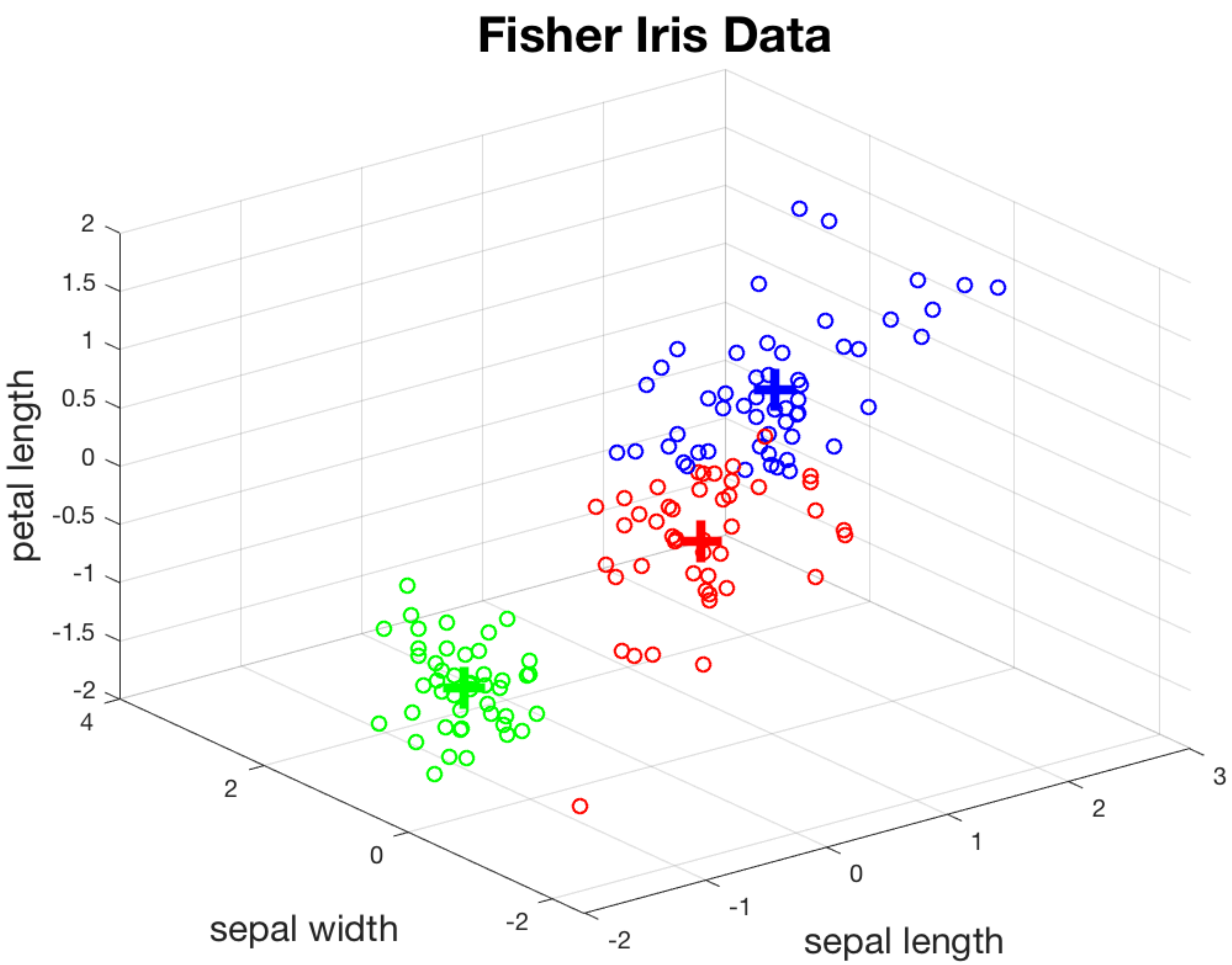
- Fisher's Iris Data: measurements of 150 irises by Ronald Fisher (1890-1962)
- Standard test data for many ML techniques

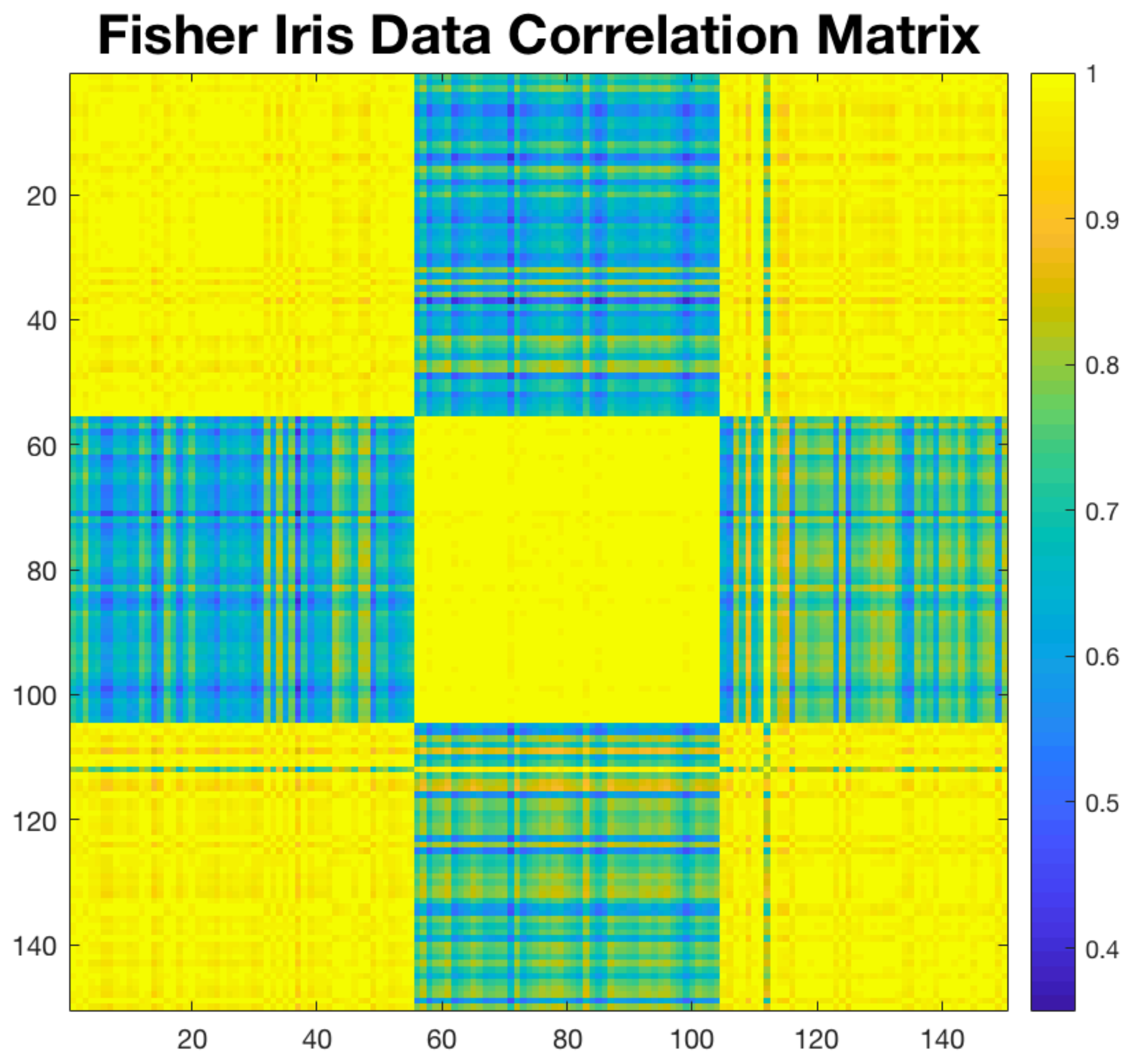
- Aside: Ronald Fisher was a raging racist, founding chair of the University of Cambridge Eugenics Society, Nazi sympathizer

k-means clustering in MATLAB

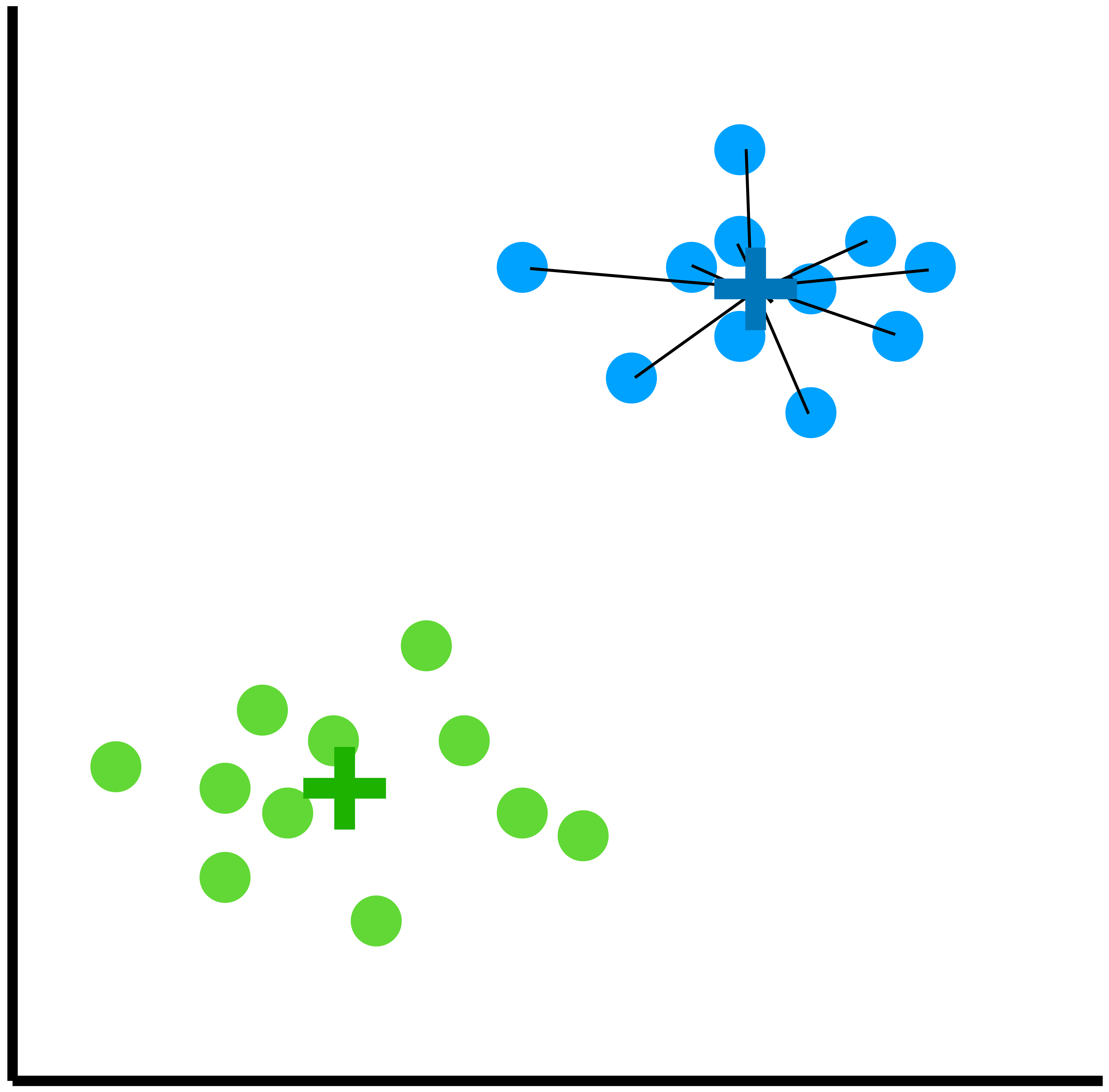
```
% normalize
data_mat_z = zscore(data_mat, [], 1);

% kmeans
k = 2;
[clust_ind, coords, sumd] = kmeans(data_mat_z, k);

% plot clusters
marker_colors = {'b', 'g'};
figure('color', 'w');
for i = 1:k
    % get data points in cluster i
    clust_i = data_mat_z(clust_ind==i,:);

    % plot data points
    plot3(clust_i(:,1), clust_i(:,2), clust_i(:,3), 'o', 'color', ...
        marker_colors{i});
    hold on;

    %plot centroid
    plot3(coords(i,1), coords(i,2), coords(i,3), '+', 'color', ...
        marker_colors{i}, 'markersize', 15, 'linewidth', 5);
end

xlabel('sepal length');
ylabel('sepal width');
zlabel('petal length');
grid on;
```

k-means clustering in MATLAB



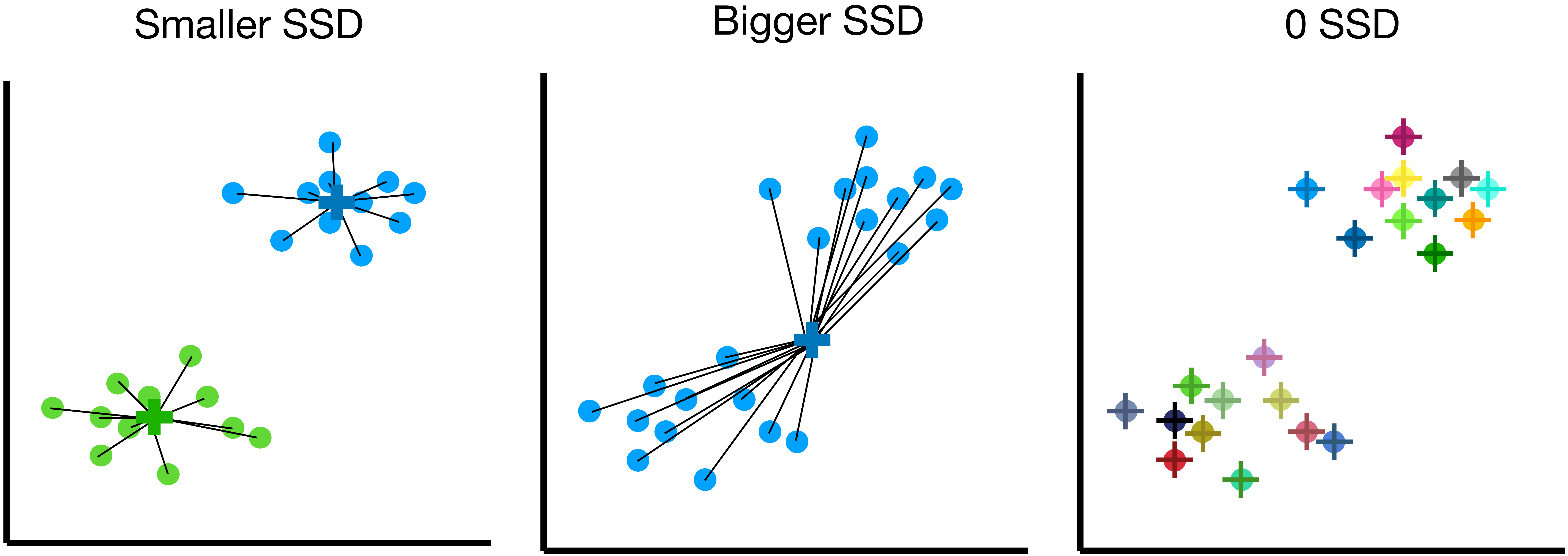


How many clusters?



- “Good” clustering means that data points are maximally close to centroid
- Sum of squared distance = measure of closeness to centroid
- Smaller sum of squared distance = better clustering solution

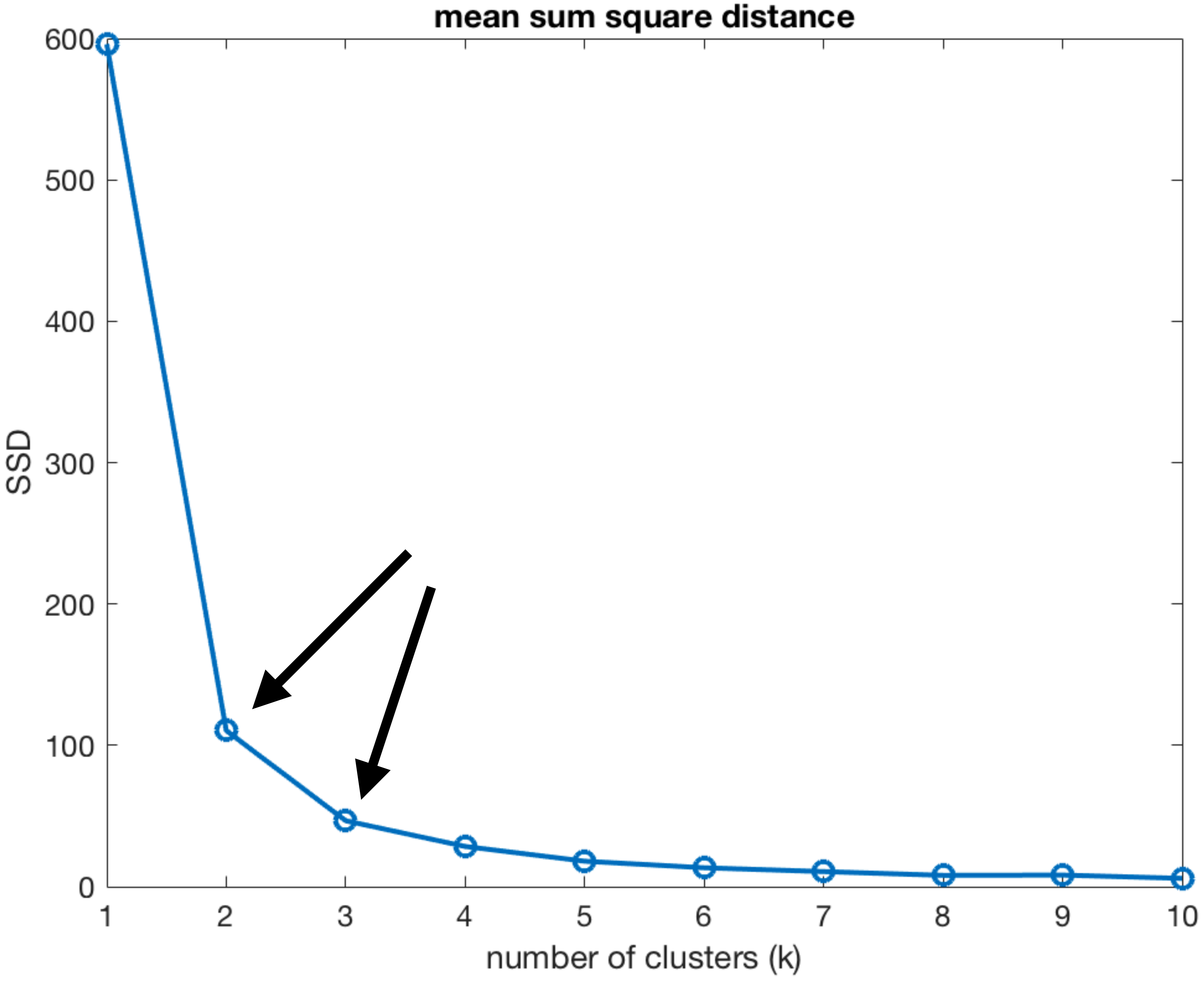
How many clusters?



Balance adding more clusters with how much gaining in explaining the data (reducing SSD)

How many clusters?

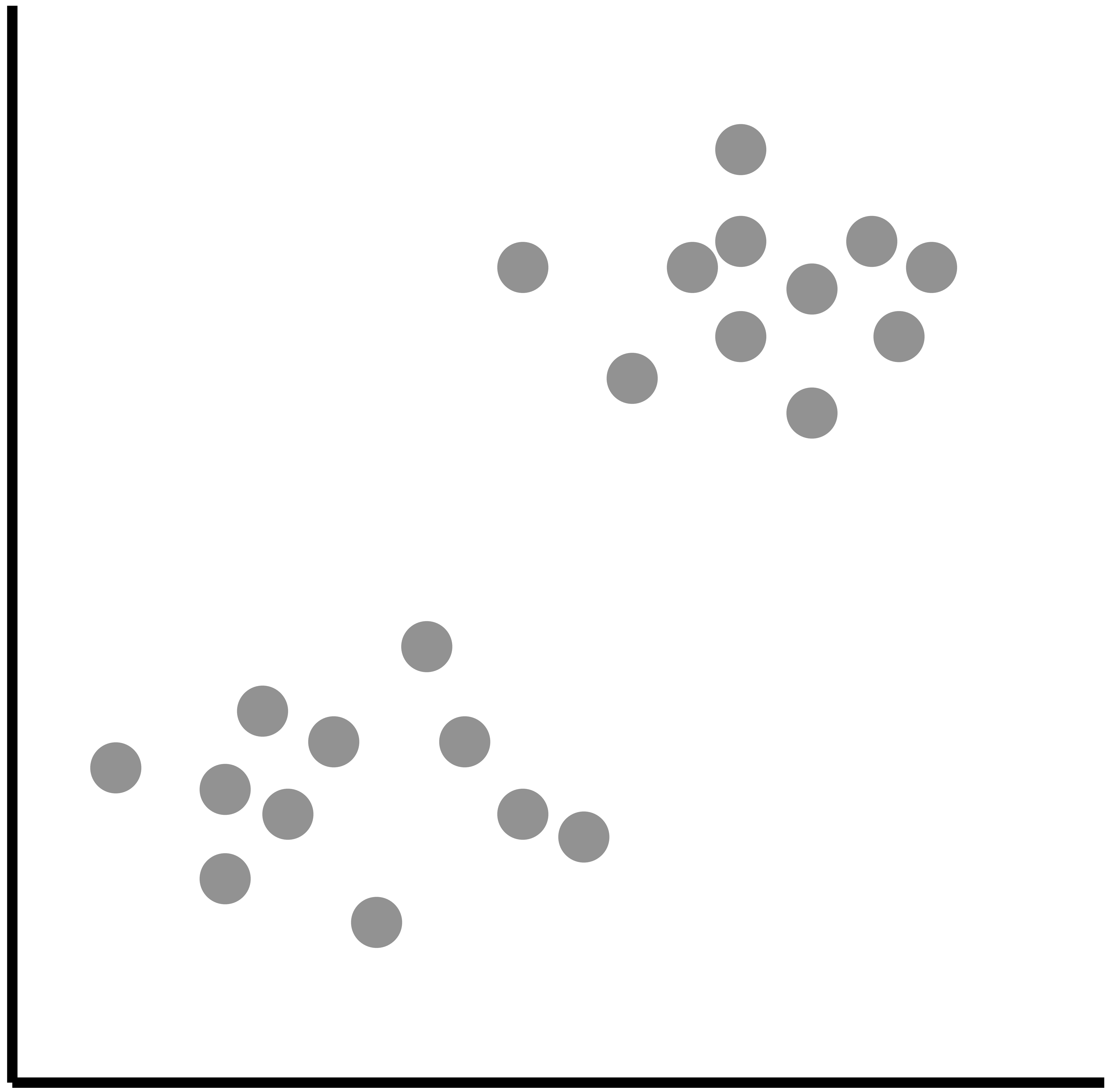
- Step 1: Select a range of k's (number of clusters)
- Step 2: apply kmeans clustering for each k and get SSD
- Step 3: make elbow plot
- Step 4: number of clusters = "elbow"



Make elbow plot in MATLAB

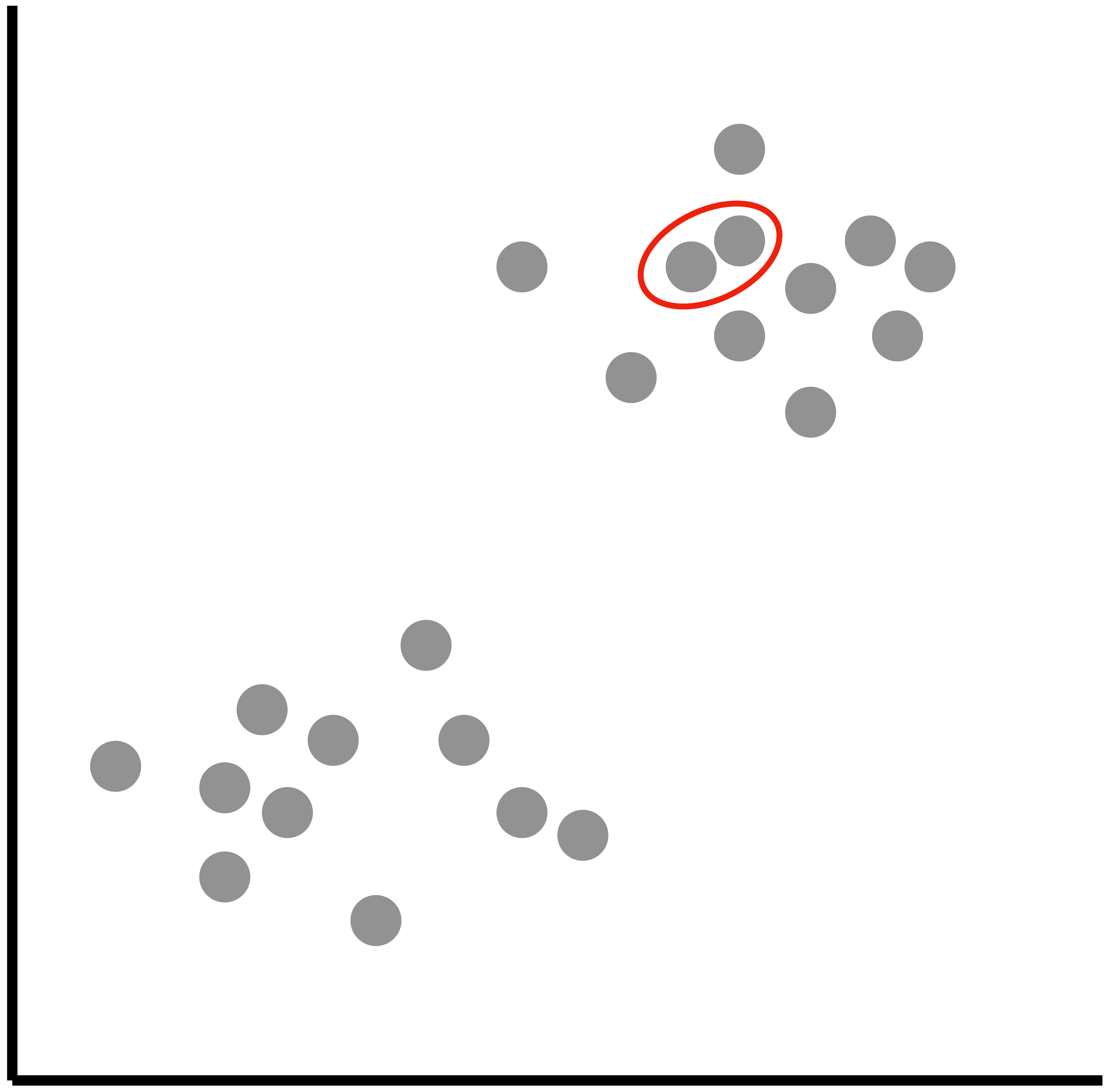
```
% elbow method
k_s = [1:10];

clust_ind = cell(size(k_s));
coords = cell(size(k_s));
sumd = cell(size(k_s));
sumd_mean = NaN(size(k_s));
for i = k_s
    [clust_ind{i}, coords{i}, sumd{i}] = kmeans(data_mat_z, k_s(i));
    sumd_mean(i) = mean(sumd{i});
end

% plot mean sumd
figure('color', 'w');
plot(k_s, sumd_mean, 'o-', 'linewidth', 2, 'markersize', 8);
title('mean sum square distance', 'fontsize', 18);
xlabel('number of clusters (k)', 'fontsize', 12);
ylabel('SSD', 'fontsize', 12);
set(gca, 'fontsize', 12)
```

Agglomerative hierarchical clustering



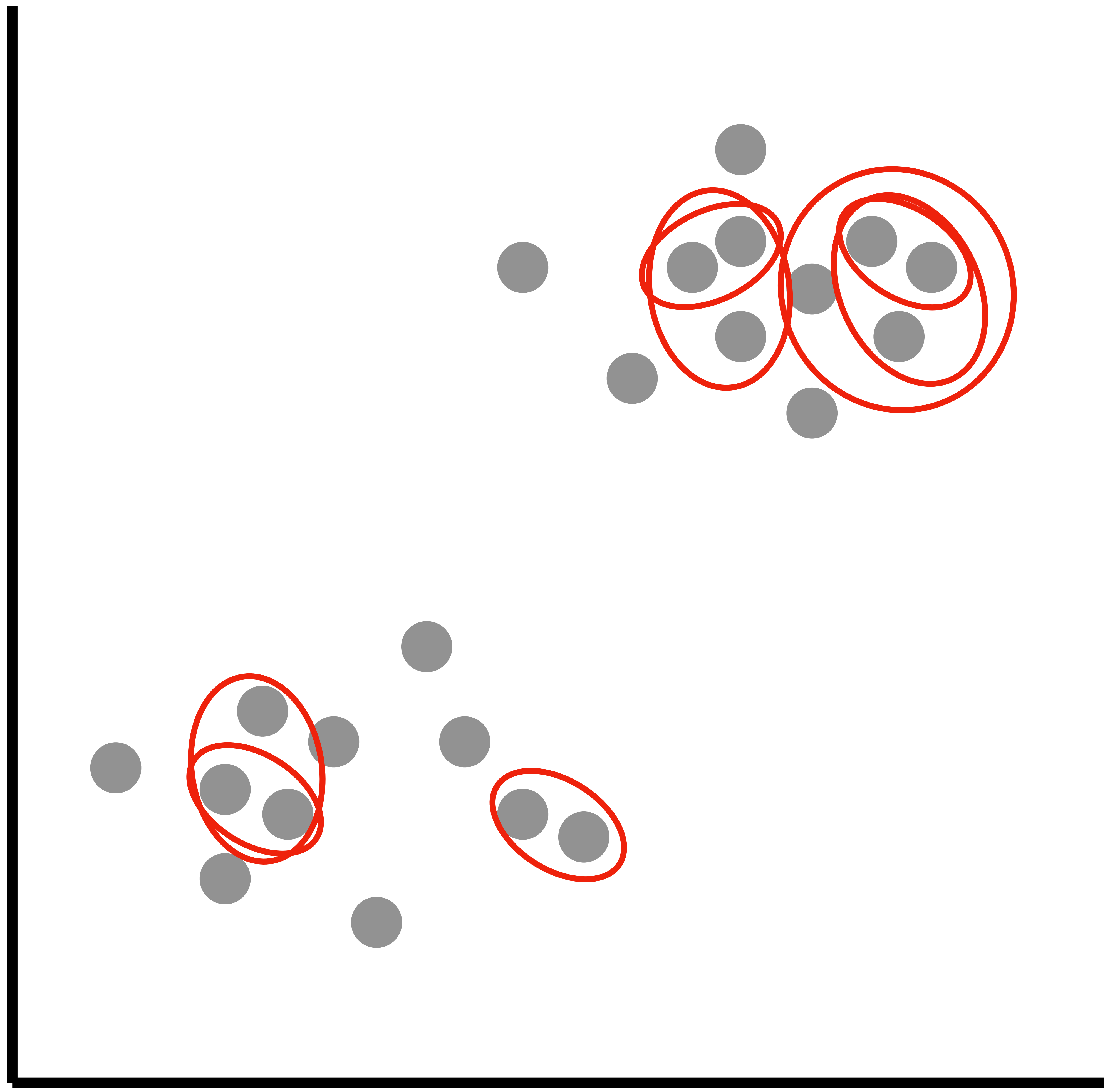
- **Step 0**: Every data point is its own cluster

Agglomerative hierarchical clustering



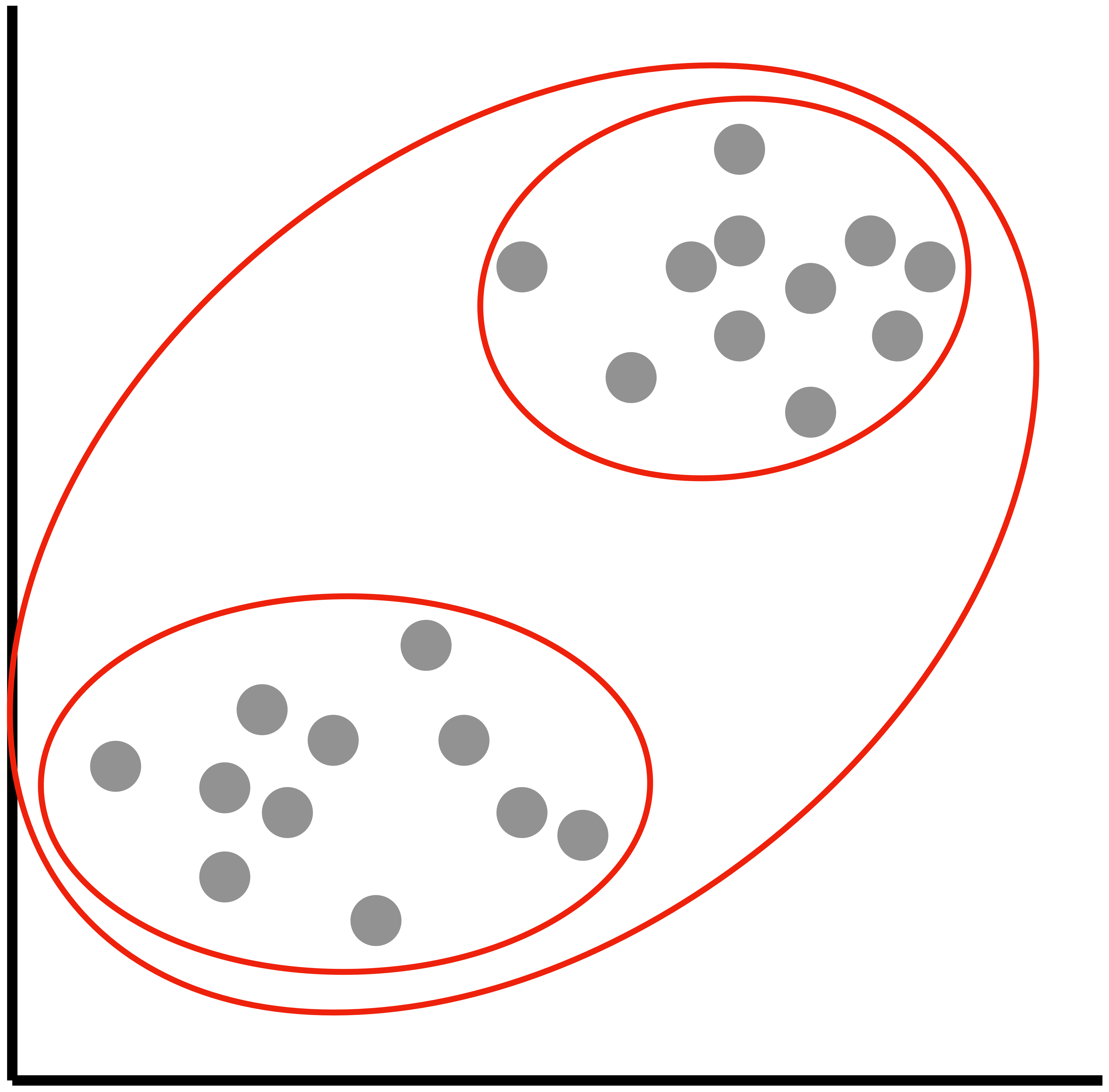
- **Step 0**: Every data point is its own cluster
- **Step 1**: Calc distance between clusters (many ways of doing this)
- **Step 2**: Merge the two clusters that are closest to each other

Agglomerative hierarchical clustering



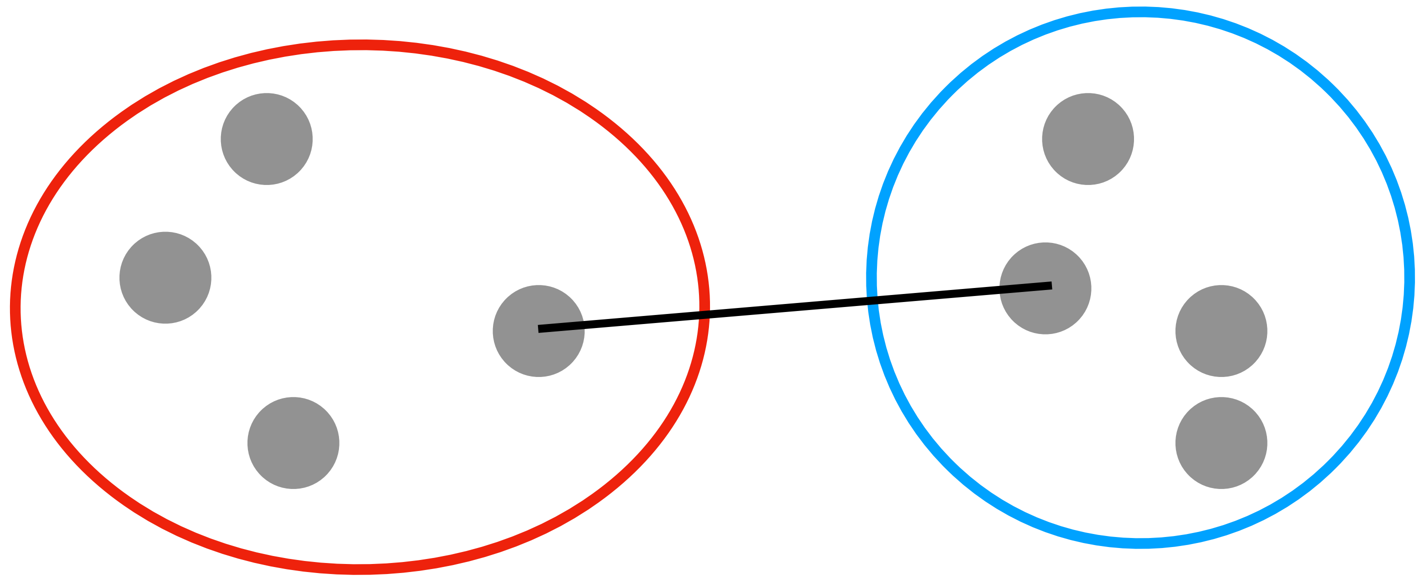
- **Step 0**: Every data point is its own cluster
- **Step 1**: Calc distance between clusters (many ways of doing this)
- **Step 2**: Merge the two clusters that are closest to each other
- **Step 3:** Repeat Steps 1 and 2 until have a single cluster

Agglomerative hierarchical clustering

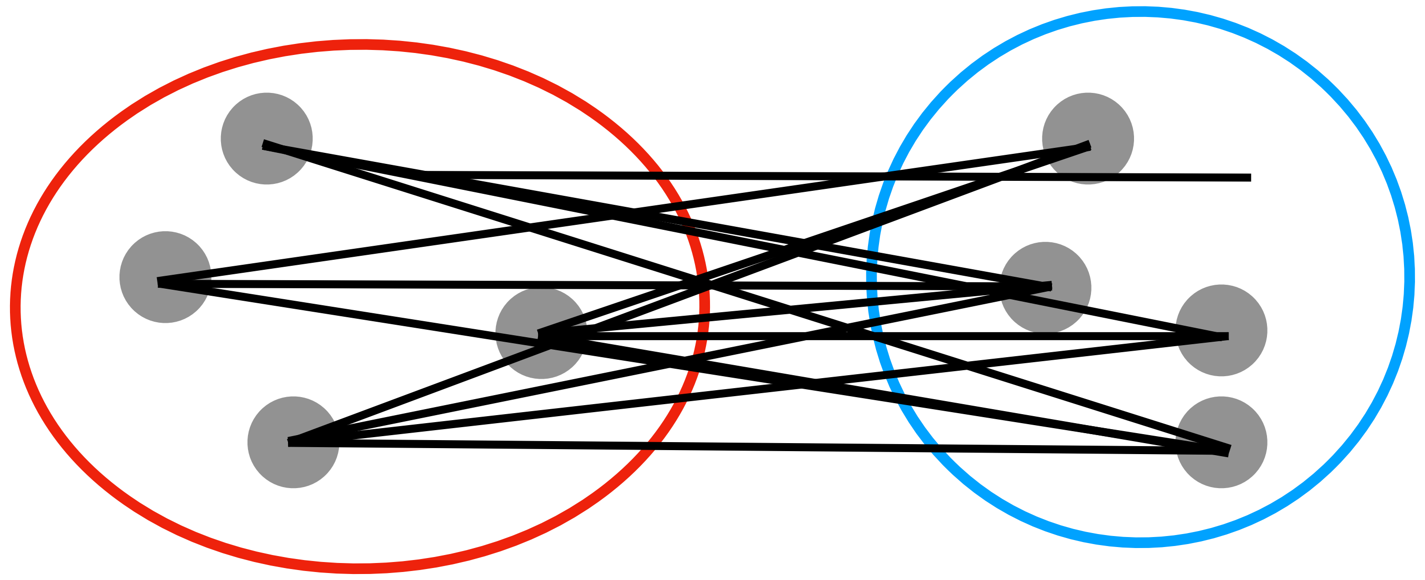


- **Step 0**: Every data point is its own cluster
- **Step 1**: Calc distance between clusters (many ways of doing this)
- **Step 2**: Merge the two clusters that are closest to each other
- **Step 3:** Repeat Steps 1 and 2 until have a single cluster

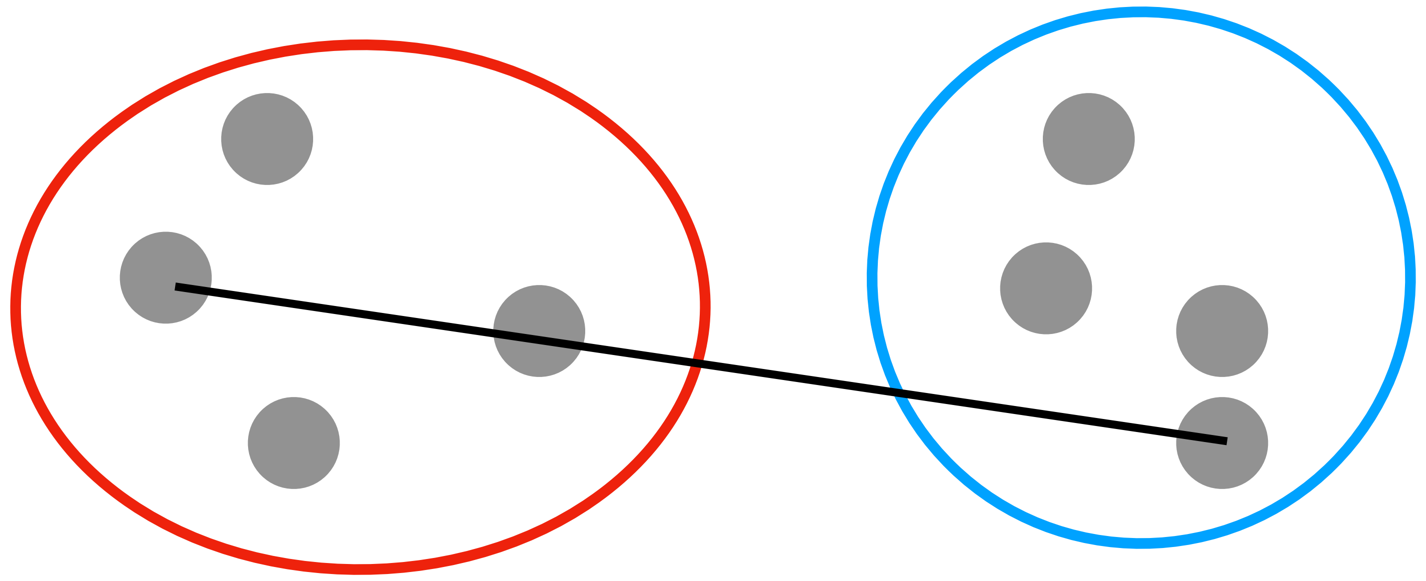
How do you measure “distance” between clusters?



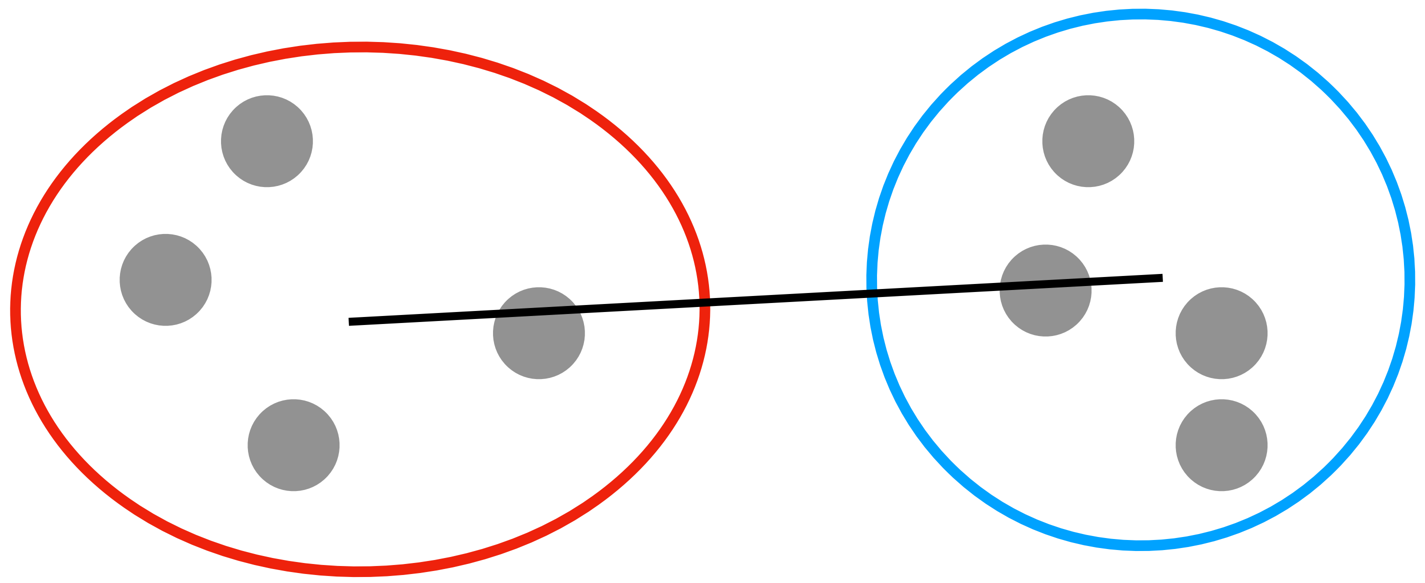
Single linkage
Distance between two closet points



Average linkage
Average of every point to every other point

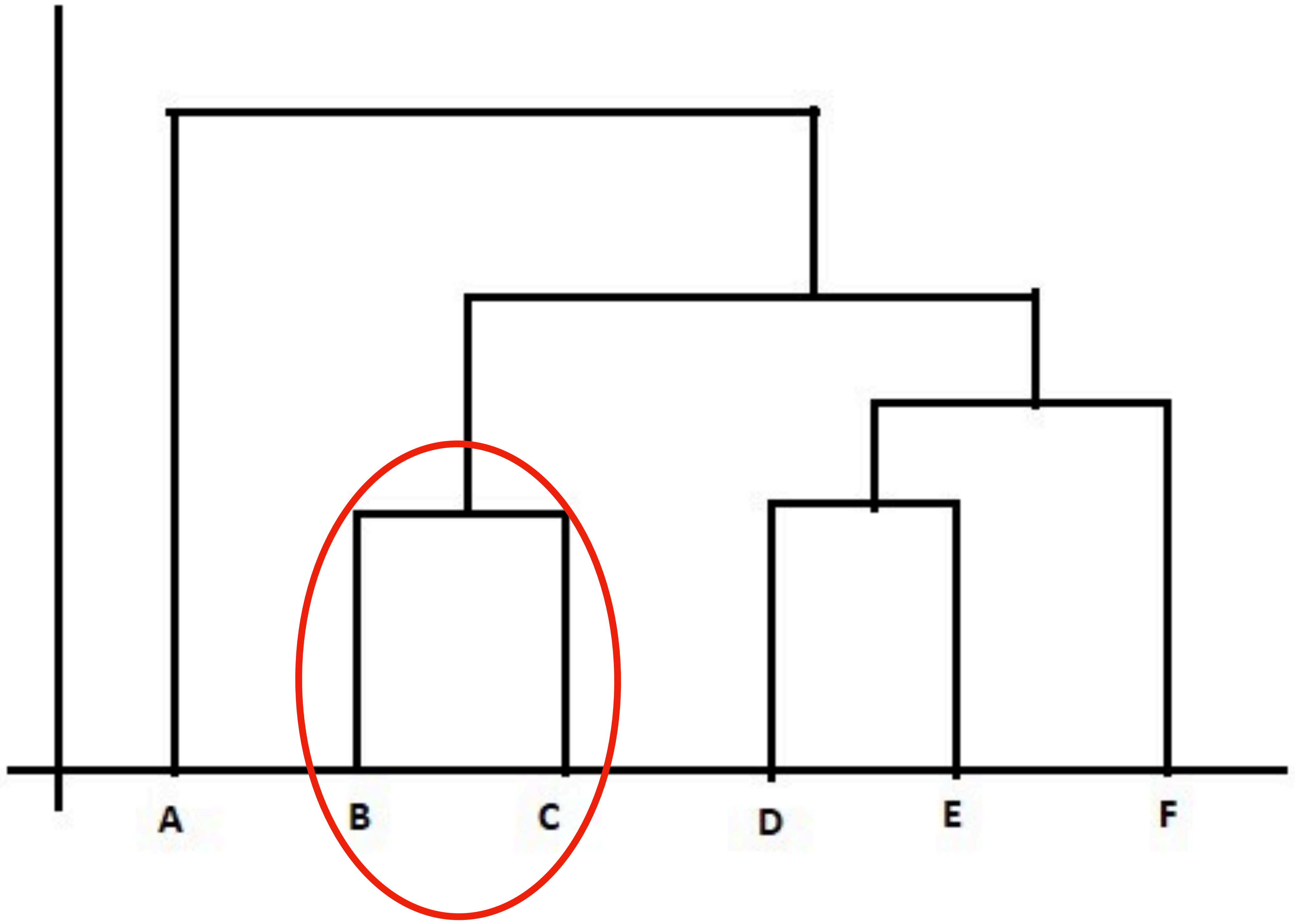


Complete linkage
Distance between two furthest points

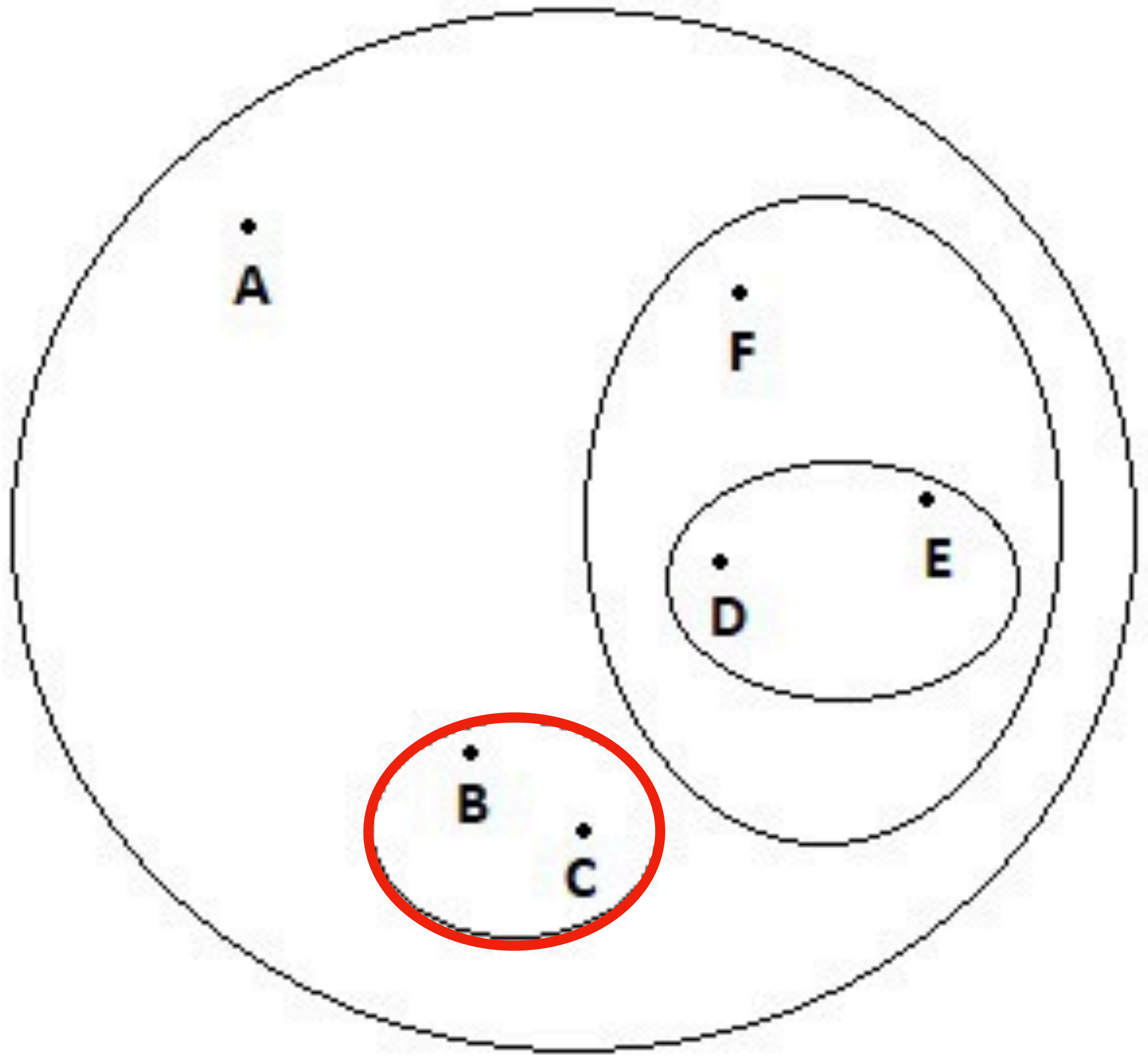


Centroid linkage
Distance between two closet points

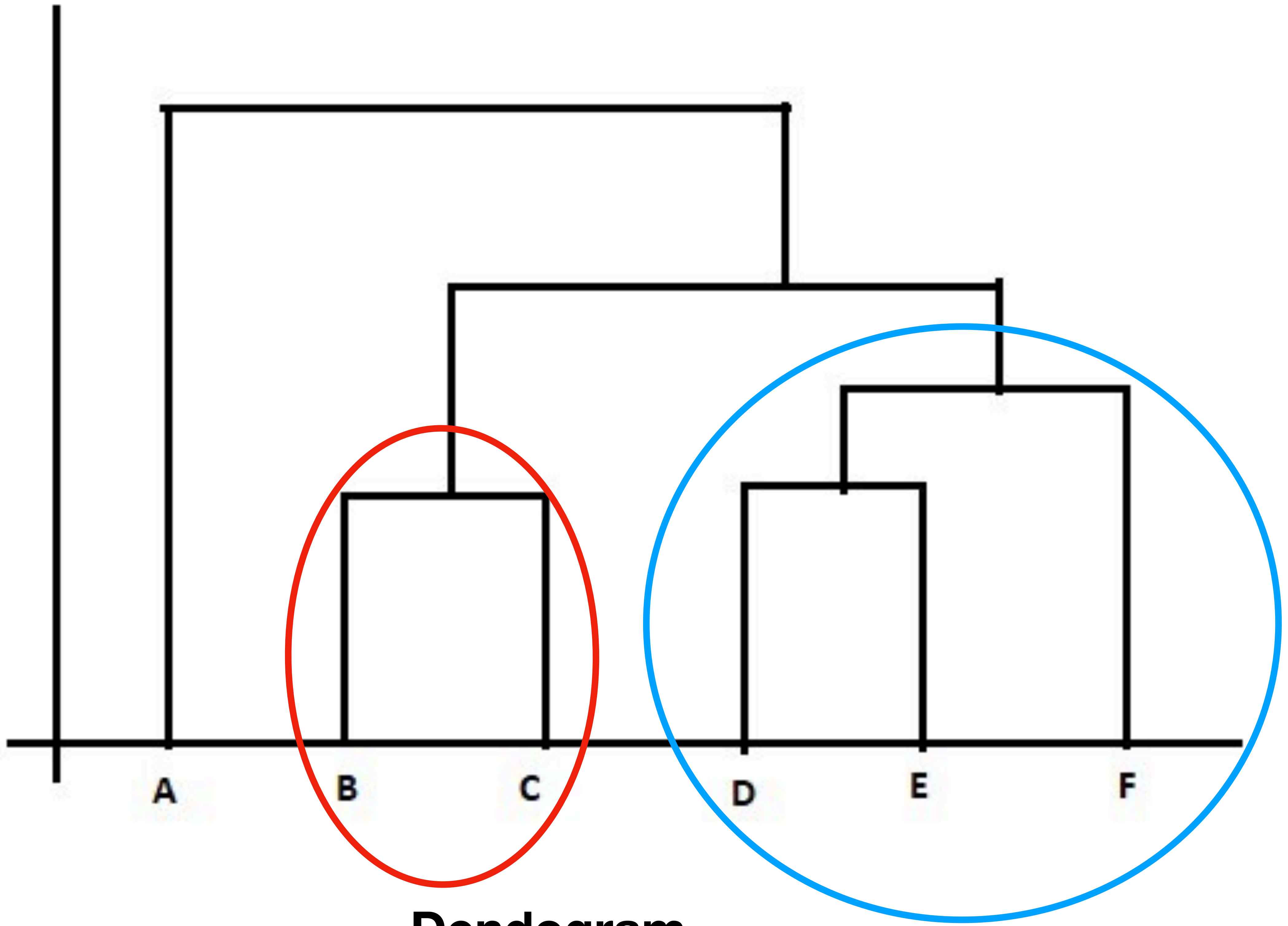
Dendrogram: represents sequence of merges



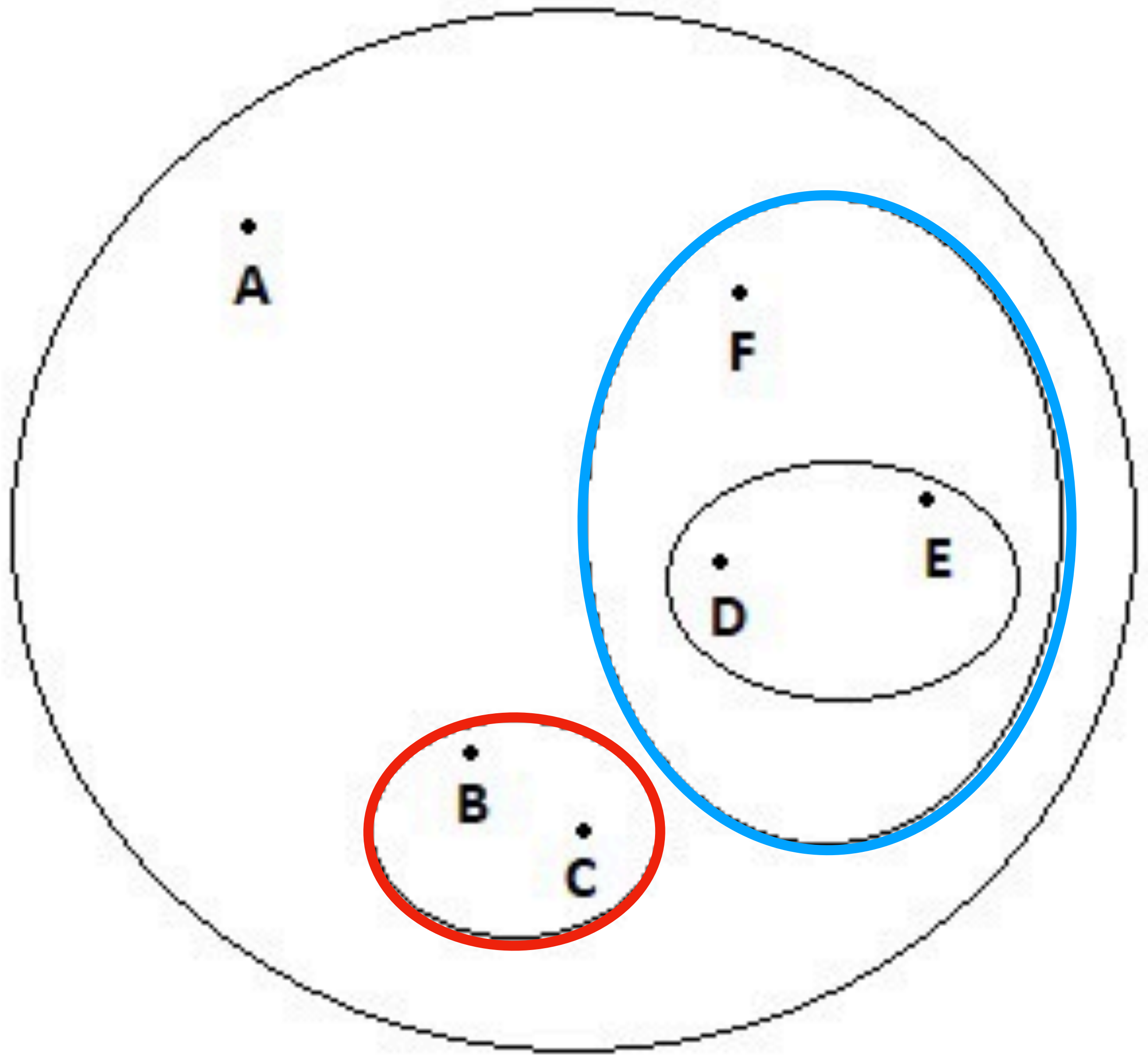
Dendogram



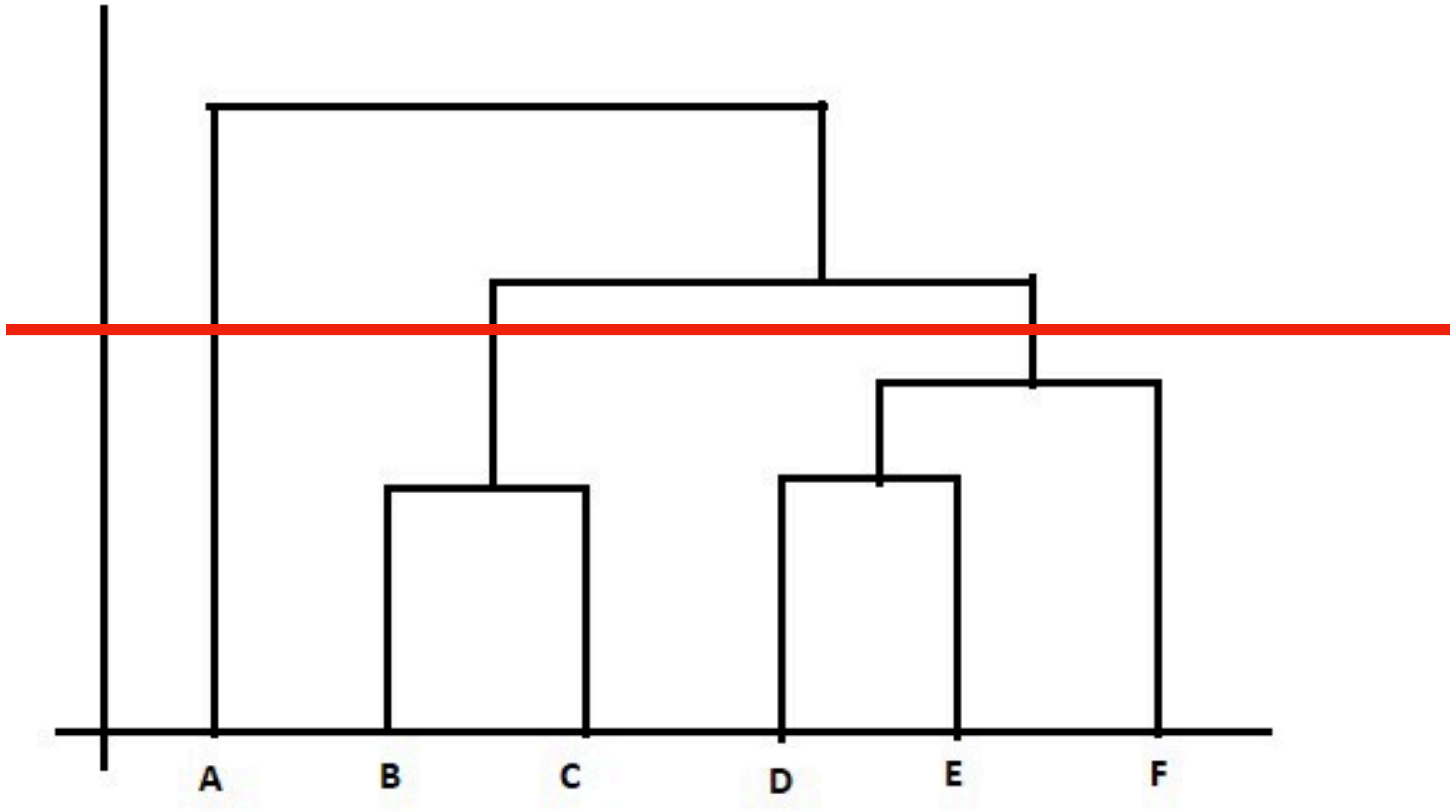
Dendrogram: represents sequence of merges



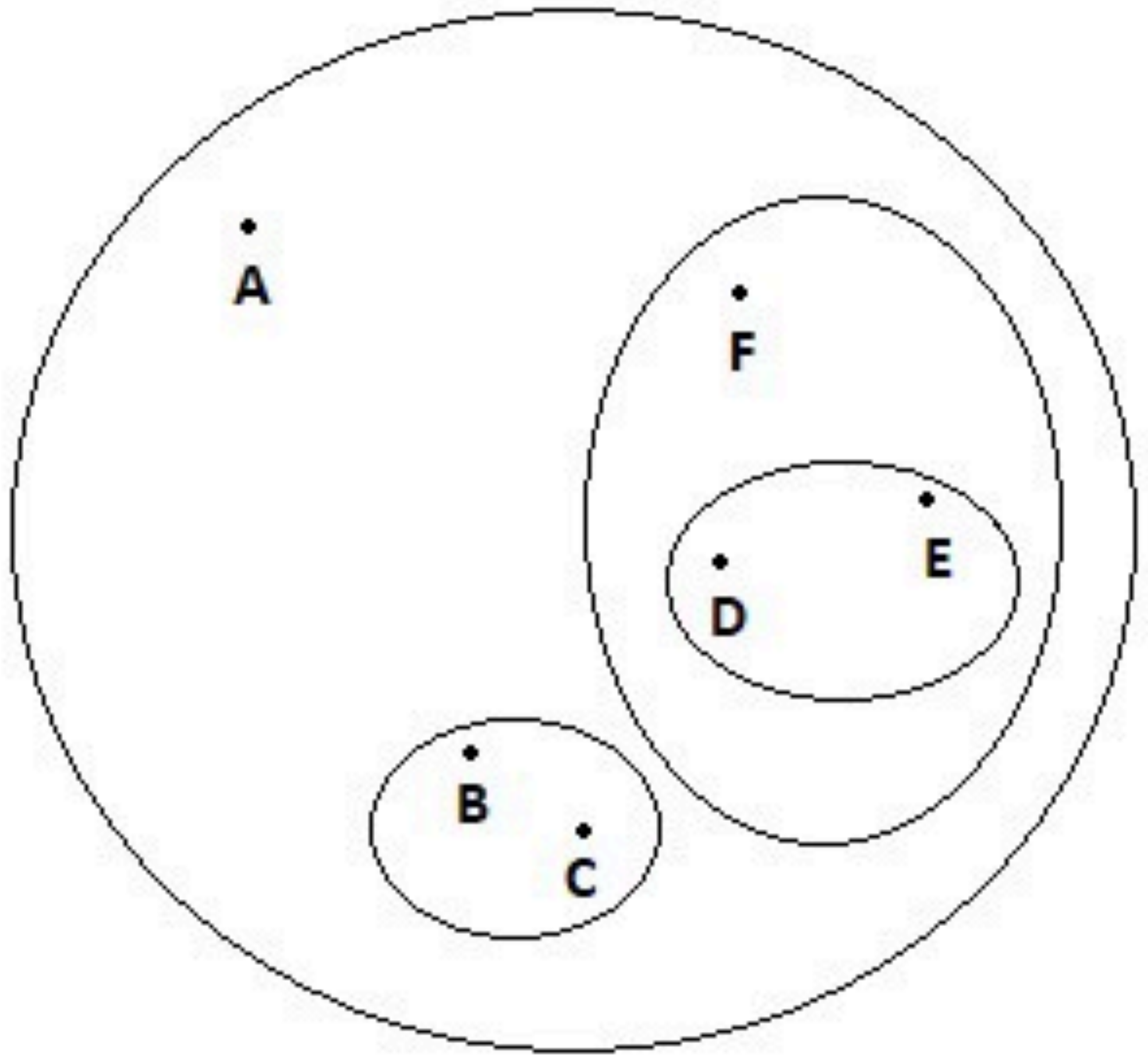
Dendogram



Slice dendrogram to define clusters



Dendogram



Hierarchical clustering MATLAB

```
% calculate distance between every point
dist = pdist(data_mat, 'euclidean');

% hierarchical cluster
clust_hier = linkage(dist, 'average');

% visualize
figure('color', 'w');
[h, nodes] = dendrogram(clust_hier,0);

% "cut" dendrogram to form clusters
clust_inds = cluster(clust_hier,'criterion','distance','cutoff',1.5);
```

k-means vs hierarchical clustering

- Method for clustering, defining clusters, defining parameters are very subjective - follow practices of your field
- **k-means clustering**
    - Simple, widely used, computationally fast. Can be used if have when have a priori idea of number of clusters and when don’t
    - But: non-deterministic, assumes clusters are spheroid
- **Hierarchical clustering**
    - Allows for visualizing structure of data in informative and simple way, doesn’t require imposing specific number of clusters, deterministic
    - But: computationally difficult, high experimenter degree-of-freedom with potential significant impact on outcomes

Next week

- Wed, July 22 at 2pm: Natural Language Processing with Zaid Zada
- Fri, July 24 at 2pm: Open science, reproducibility, and GitHub with Sam Nastase