APS1070

Foundations of Data Analytics and Machine Learning

Summer 2020

Wed May 27 / Week 3:

- *Decision Trees*
- *Clustering Strategies*
- *Python & Data*

Jason Riordon, PhD

Slide Attribution

These slides contain materials from various sources. Special thanks to the following authors:

- Caitlin Carnahan
- Katia Koleinik
- Ali Hadi Zadeh
- D. Hoiem

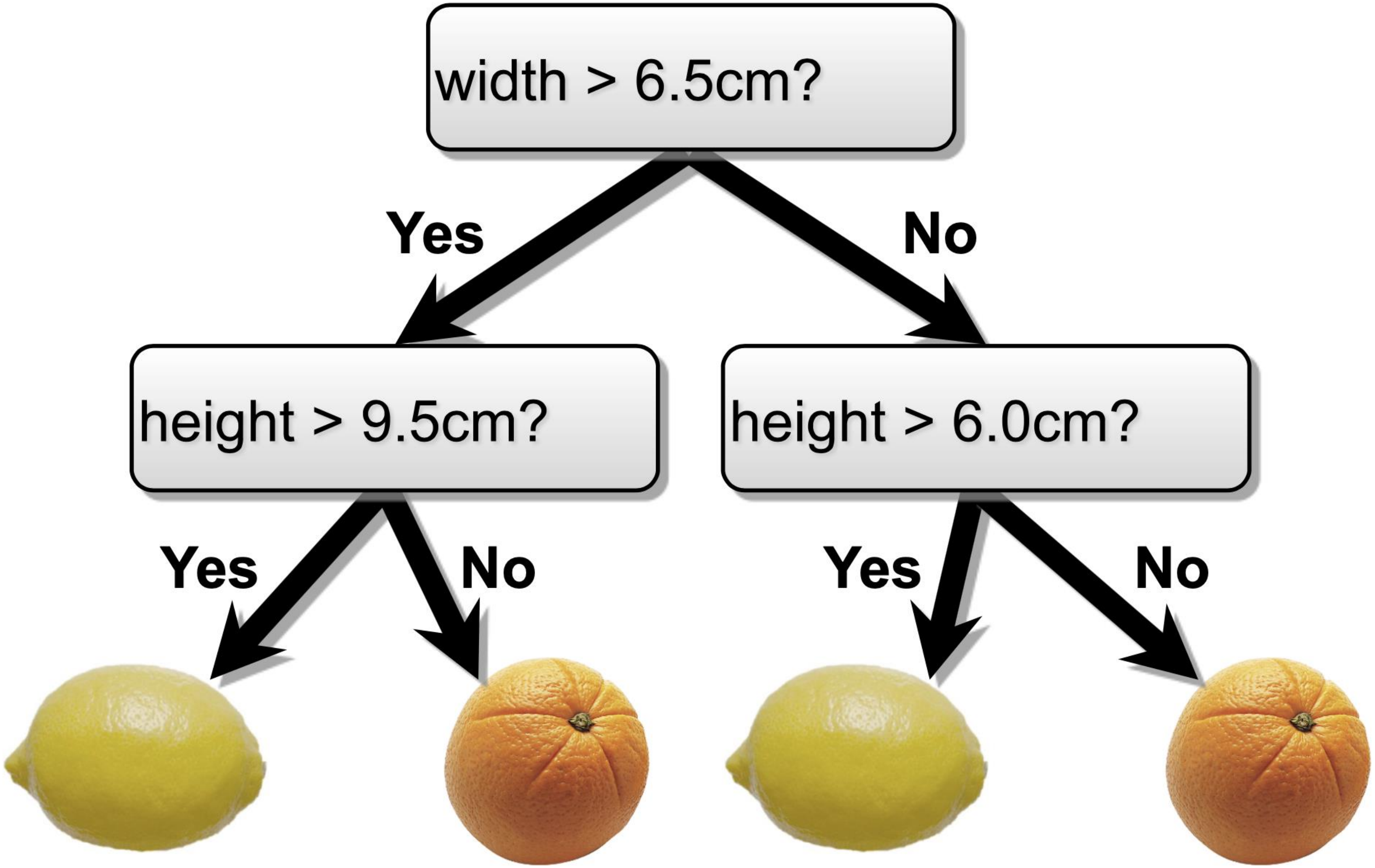
Topics (preliminary)

Wk	Date	Lecture	Chapter*
1	May 13	Lecture: Course Overview, Machine Learning – Big Picture	PFDA 1-3
	May 14	NO LAB	
2	May 20	Lecture: Nearest neighbour classifier, Cross-validation, Intro to Python Part 1: Language Basics, Ipython and Jupyter, Data Structures, Functions, NumPy, SciPy, Pandas, Project 1 Intro	PFDA 4-5
	May 21	Project 1 Tutorial - Basic Data Science	
3	May 27	Decision Trees, Clustering Strategies, Intro to Python Part 2: Data Loading/Storage/File Formats, Data Cleaning/Preparation, Data Wrangling, Plotting and Visualization	PFDA 6-9
	May 28	Project 1 Q & A - Basic Data Science **(Project 1 due at 11:00 pm)**	
4	June 3	Summary Statistics, Multivariate Gaussian Distribution, Project 2 Intro	MML 1,6,11
	June 4	Project 2 Tutorial - Anomaly Detection	
5	June 10	Precision and Recall, Linear Algebra, Analytical Geometry	MML 2-3
	June 11	Project 2 Q & A - Anomaly Detection	
6	June 17	Feedback, Matrix Decompositions, Vector Calculus	MML 4,5
	June 18	Project 2 Q & A - Anomaly Detection **(Project 2 due at 11:00 pm)**	
7	June 24	PCA, Project 3 Intro, Midterm Quercus Quiz	MML 10
	June 25	Project 3 Tutorial - PCA	
8	July 1	Canada Day – no class	
	July 2	Project 3 Q & A - PCA	
9	July 8	Continuous Optimization, Linear Regression	MML 7, ESL: 2.3, 3.1-3.2.1
	July 9	Project 3 Q & A - PCA **(Project 3 due at 11:00 pm)**	
10	July 15	Linear Classification, Project 4 Intro	ESL: 2.3, 3.1-3.2.1
	July 16	Project 4 Tutorial - Linear Regression	
11	July 22	Learning rate selection, Stochastic GD, Convexity, Naïve Bayes Classifier, Big-O Notation	MML 8,9
	July 23	Project 4 Q & A - Linear Regression	
12	July 29	An introduction to Deep Learning	
	July 30	Project 4 Q & A - Linear Regression **(Project 4 due at 11:00 pm)**	
13	Aug 5	Final Quercus Quiz	
	Aug 6	NO LAB	

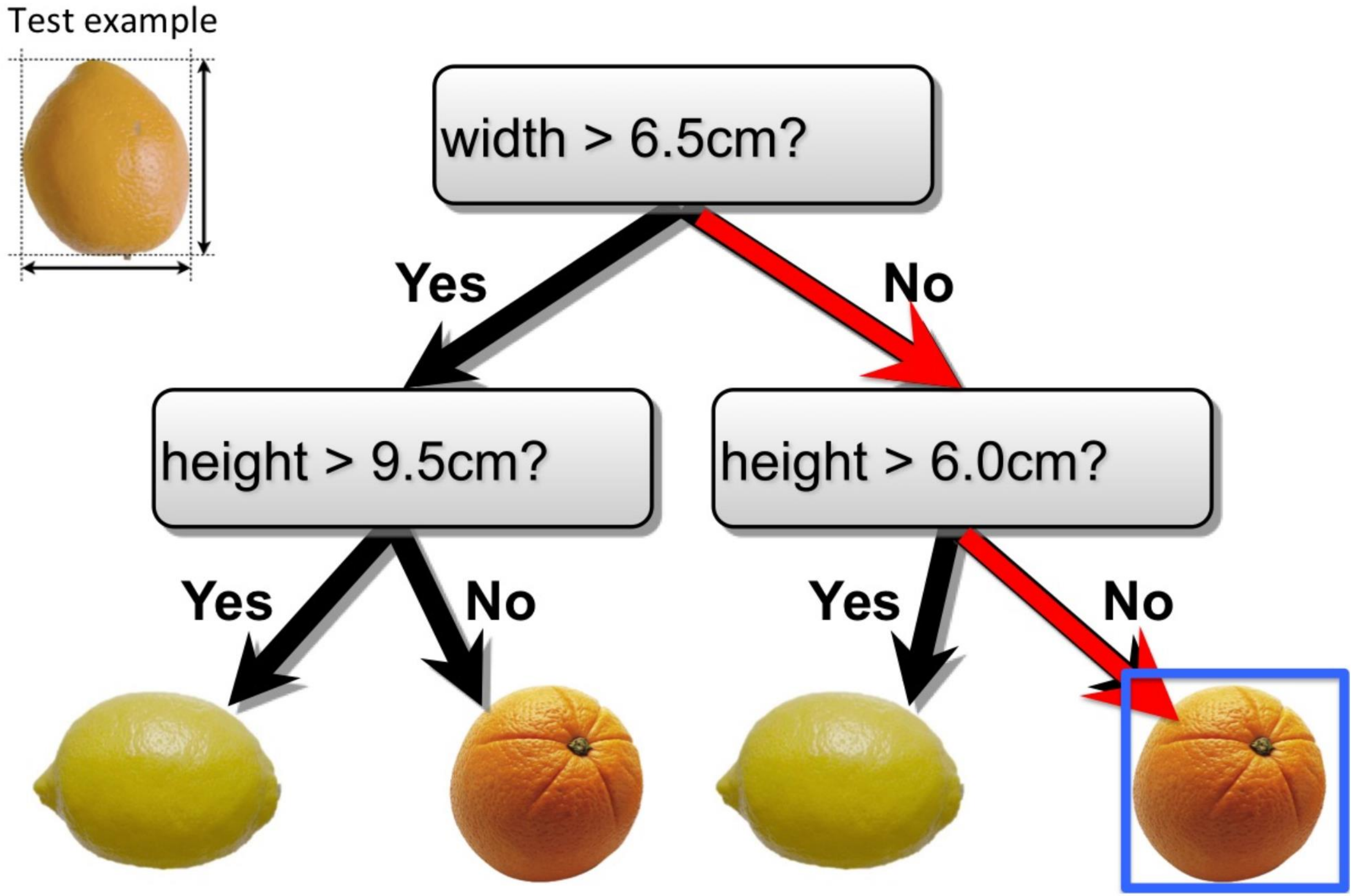
Decision Trees
(Supervised)

Lemon Vs. Orange!

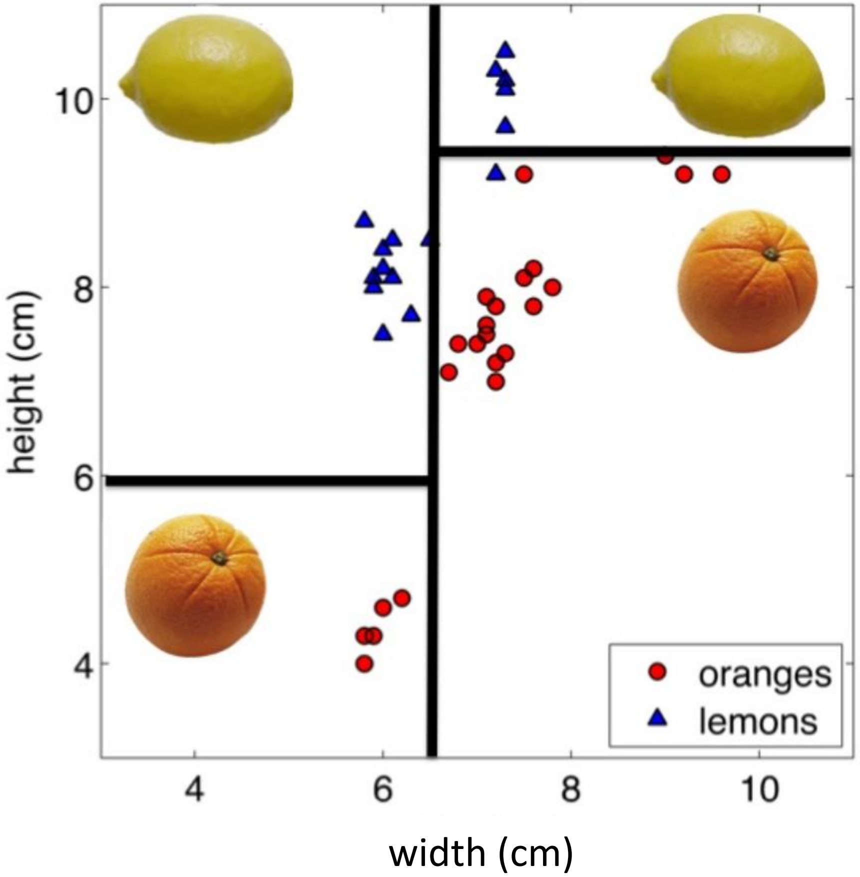
Flowchart-like structure!

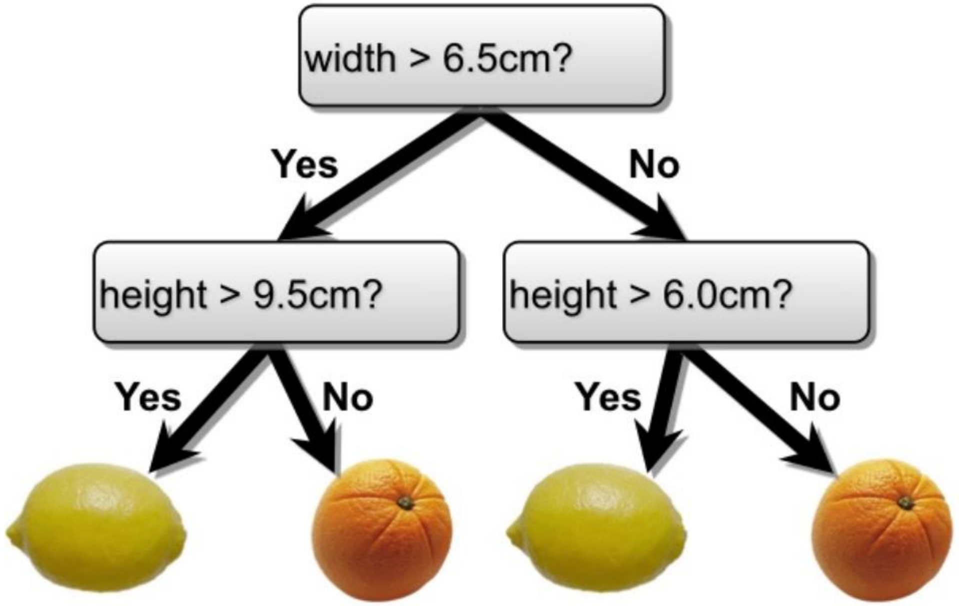


Test example



Decision trees make predictions by recursively splitting on different attributes according to a tree structure.





- What if the attributes are discrete?

Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	Yes	No	No	Yes	Some	\$\$\$	No	Yes	French	0–10	$y_1 =$ Yes
$\mathbf{x}_2$	Yes	No	No	Yes	Full	\$	No	No	Thai	30–60	$y_2 =$ No
$\mathbf{x}_3$	No	Yes	No	No	Some	\$	No	No	Burger	0–10	$y_3 =$ Yes
$\mathbf{x}_4$	Yes	No	Yes	Yes	Full	\$	Yes	No	Thai	10–30	$y_4 =$ Yes
$\mathbf{x}_5$	Yes	No	Yes	No	Full	\$\$\$	No	Yes	French	>60	$y_5 =$ No
$\mathbf{x}_6$	No	Yes	No	Yes	Some	\$\$	Yes	Yes	Italian	0–10	$y_6 =$ Yes
$\mathbf{x}_7$	No	Yes	No	No	None	\$	Yes	No	Burger	0–10	$y_7 =$ No
$\mathbf{x}_8$	No	No	No	Yes	Some	\$\$	Yes	Yes	Thai	0–10	$y_8 =$ Yes
$\mathbf{x}_9$	No	Yes	Yes	No	Full	\$	Yes	No	Burger	>60	$y_9 =$ No
$\mathbf{x}_{10}$	Yes	Yes	Yes	Yes	Full	\$\$\$	No	Yes	Italian	10–30	$y_{10} =$ No
$\mathbf{x}_{11}$	No	No	No	No	None	\$	No	No	Thai	0–10	$y_{11} =$ No
$\mathbf{x}_{12}$	Yes	Yes	Yes	Yes	Full	\$	No	No	Burger	30–60	$y_{12} =$ Yes

Attributes:

1.	Alternate: whether there is a suitable alternative restaurant nearby.
2.	Bar: whether the restaurant has a comfortable bar area to wait in.
3.	Fri/Sat: true on Fridays and Saturdays.
4.	Hungry: whether we are hungry.
5.	Patrons: how many people are in the restaurant (values are None, Some, and Full).
6.	Price: the restaurant's price range (\$, \$\$, \$\$\$).
7.	Raining: whether it is raining outside.
8.	Reservation: whether we made a reservation.
9.	Type: the kind of restaurant (French, Italian, Thai or Burger).
10.	WaitEstimate: the wait estimated by the host (0-10 minutes, 10-30, 30-60, >60).

- What if the attributes are discrete?

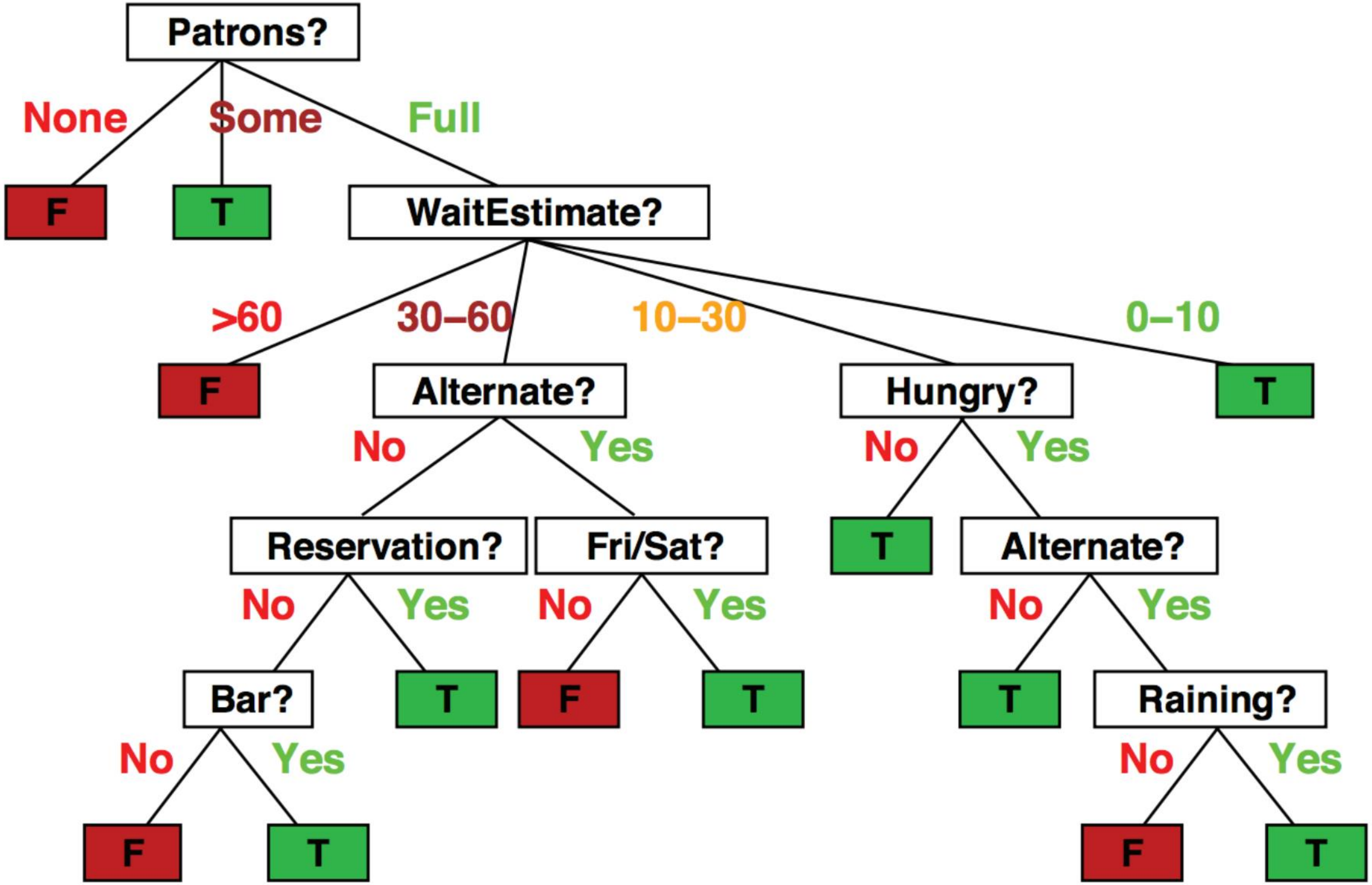
Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*\$\$\$*	*No*	*Yes*	*French*	*0–10*	$y_1 =$ *Yes*
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*\$*	*No*	*No*	*Thai*	*30–60*	$y_2 =$ *No*
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*\$*	*No*	*No*	*Burger*	*0–10*	$y_3 =$ *Yes*
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*\$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 =$ *Yes*

Attributes: Features (inputs)!
Discrete or Continuous

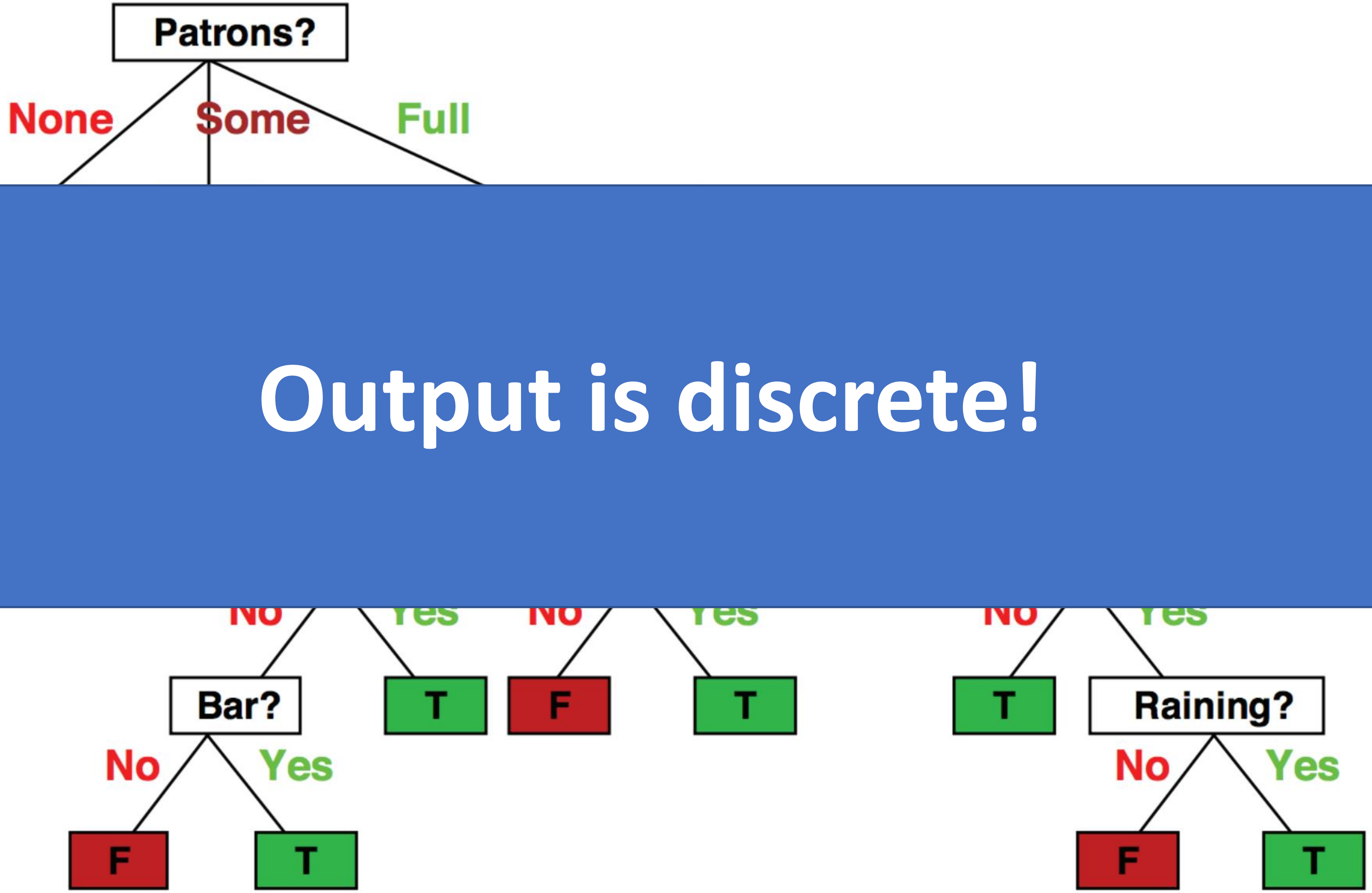
Attributes:

2.	Bar: whether the restaurant has a comfortable bar area to wait in.
3.	Fri/Sat: true on Fridays and Saturdays.
4.	Hungry: whether we are hungry.
5.	Patrons: how many people are in the restaurant (values are None, Some, and Full).
6.	Price: the restaurant's price range ($, $$, $$$).
7.	Raining: whether it is raining outside.
8.	Reservation: whether we made a reservation.
9.	Type: the kind of restaurant (French, Italian, Thai or Burger).
10.	WaitEstimate: the wait estimated by the host (0-10 minutes, 10-30, 30-60, >60).

- The tree to decide whether to wait (T) or not (F)

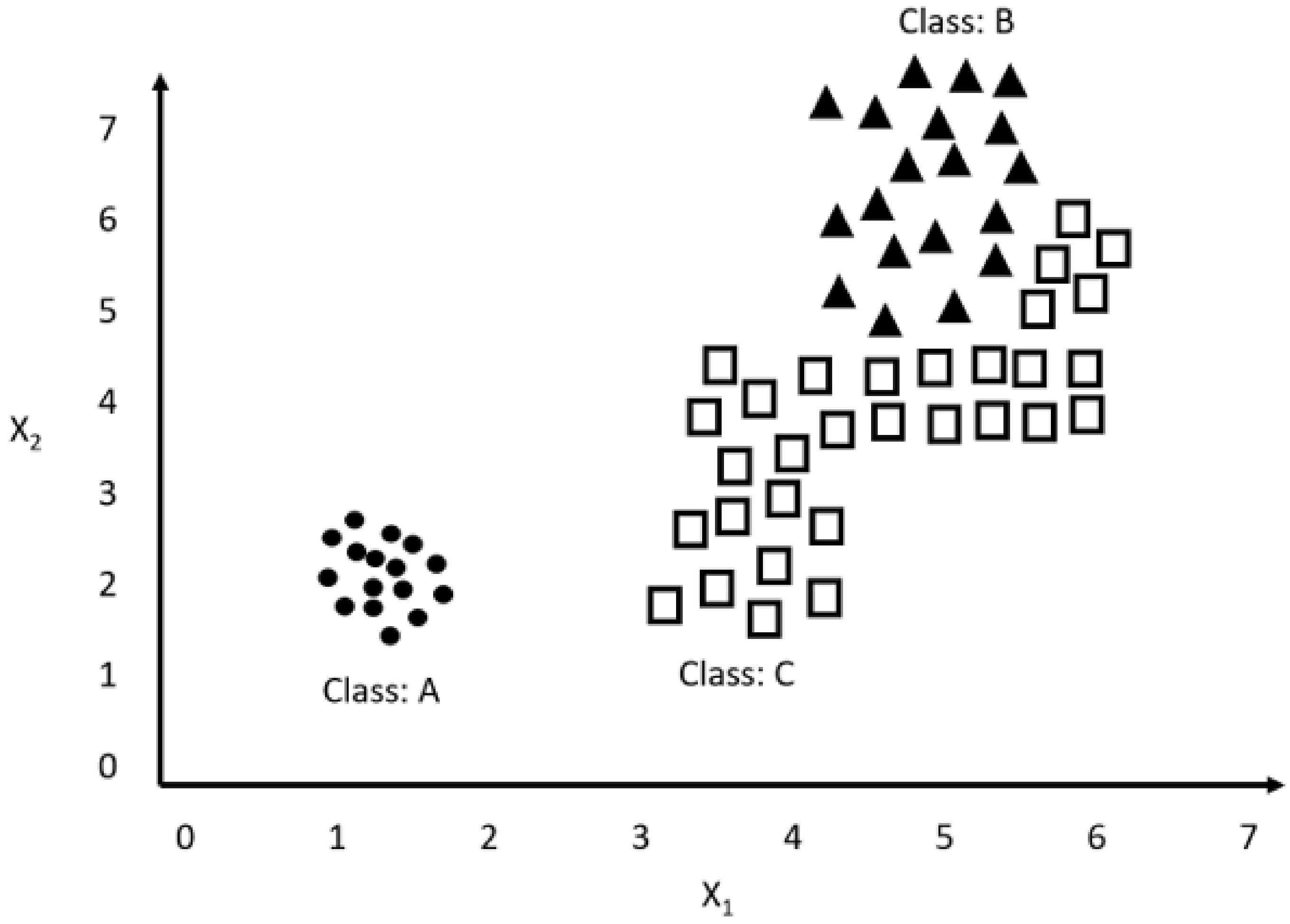


- The tree to decide whether to wait (T) or not (F)



Output is discrete!

4. Based on the following dataset:
   a. Draw the decision boundary of a decision tree to classify each class [2 marks].
   b. Draw the corresponding tree diagram [2 marks].



See also…

A Guide to Decision Trees for Machine Learning and Data Science

https://towardsdatascience.com/a-guide-to-decision-trees-for-machine-learning-and-data-science-fe2607241956

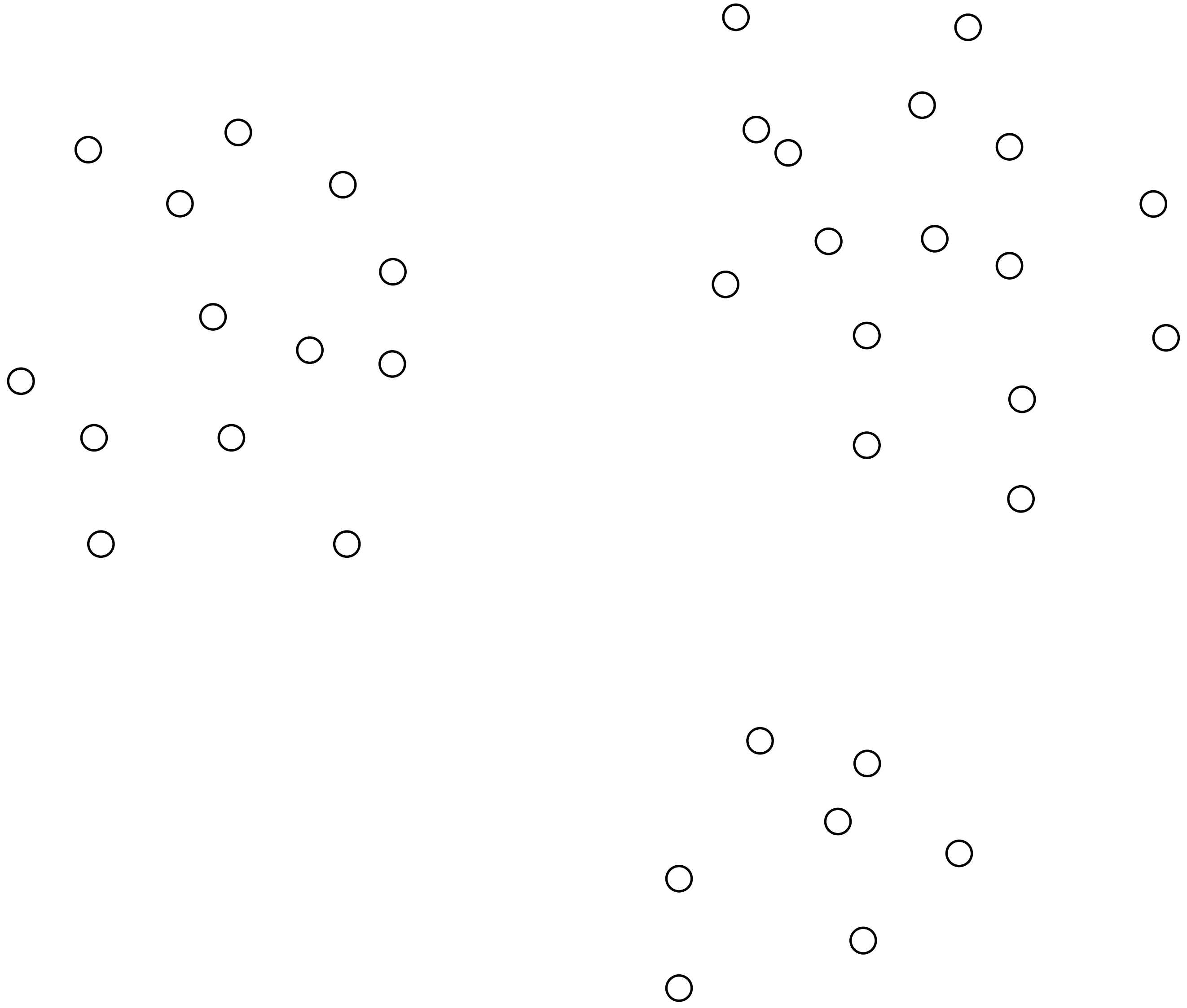
Clustering Strategies (Unsupervised)

Clustering Strategies

- K-means
  - Iteratively re-assign points to the nearest cluster center
- Agglomerative clustering
  - Start with each point as its own cluster and iteratively merge the closest clusters
- Spectral clustering
  - Split the nodes in a graph based on assigned links with similarity weights

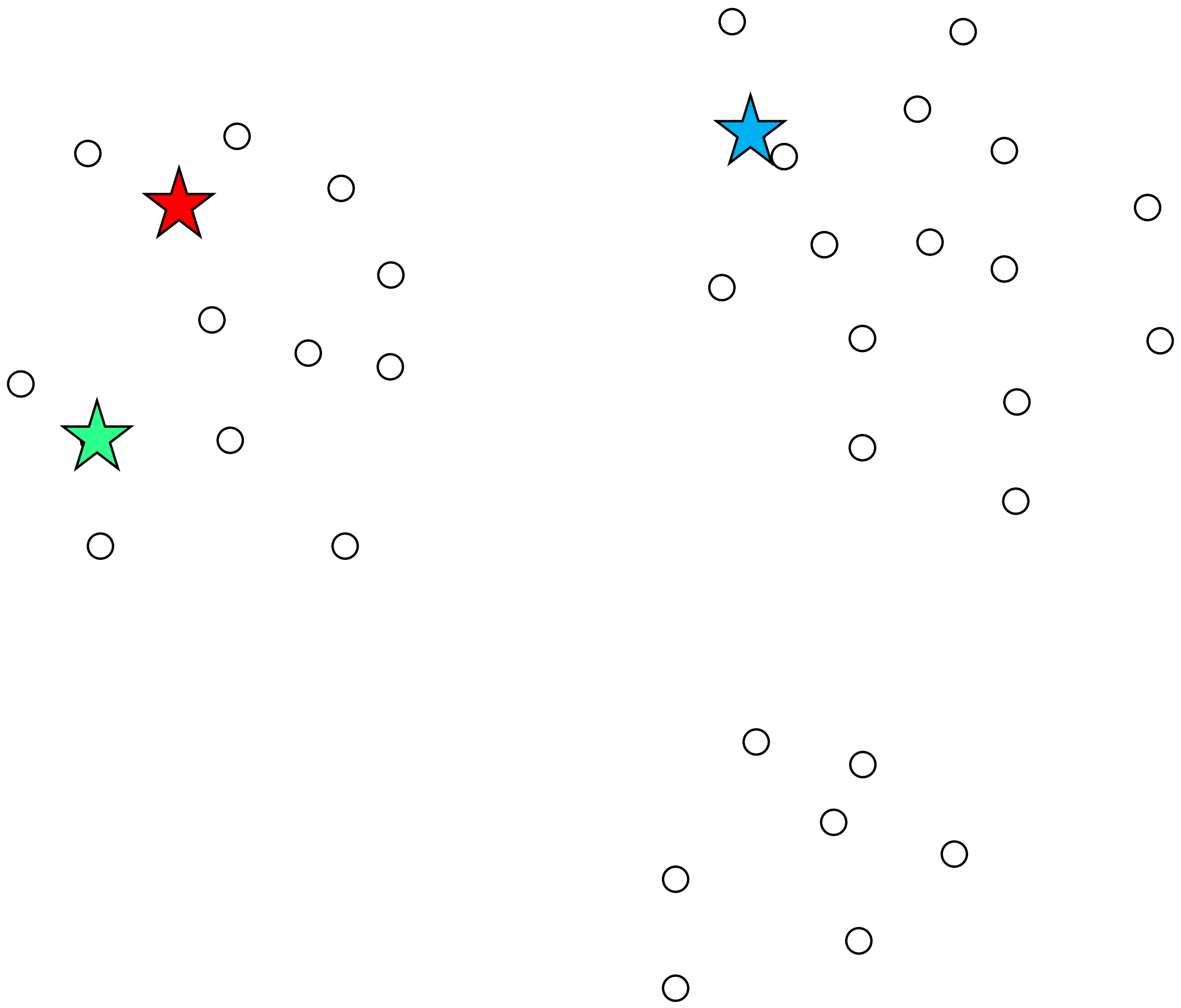
K-means

- K is number of clusters
- STEP 1: Guess center locations
- STEP 2: Map out what data point is closest to what center
- STEP 3: Center moves to the centroid of all points it "owns"
- REPEAT!



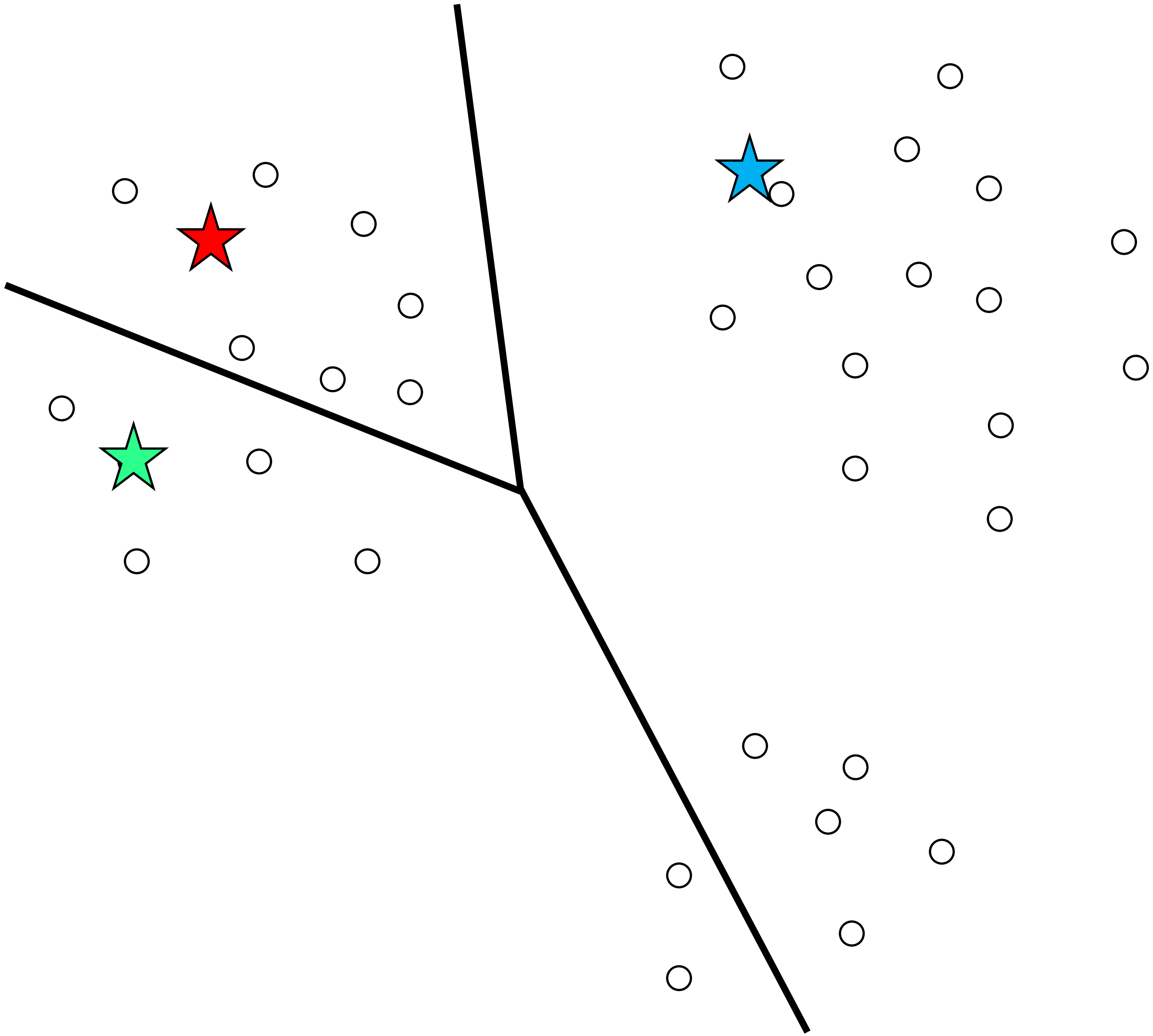
K-means

- K is number of clusters
- STEP 1: Guess center locations
- STEP 2: Map out what data point is closest to what center
- STEP 3: Center moves to the centroid of all points it "owns"
- REPEAT!



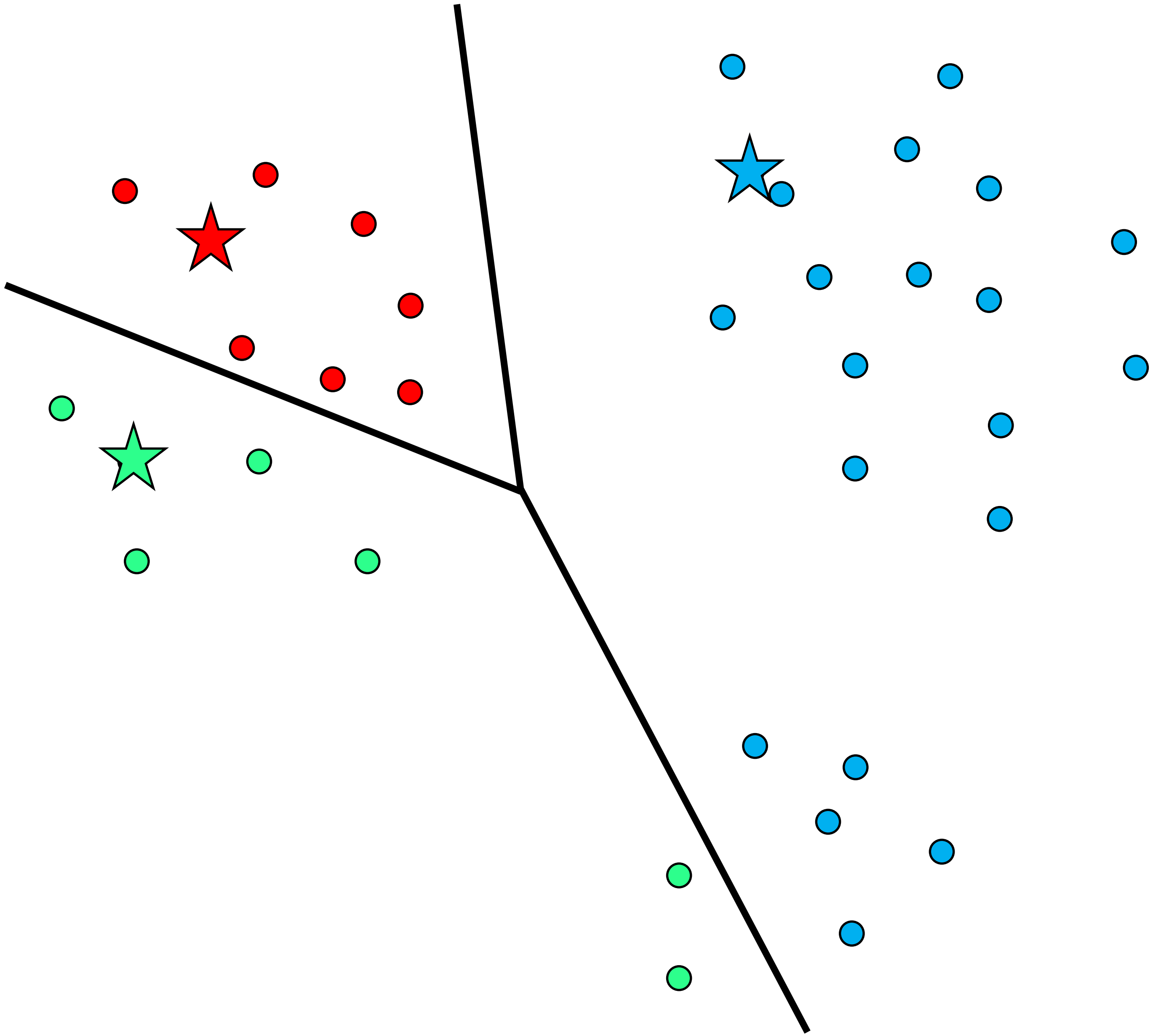
K-means

- K is number of clusters
- STEP 1: Guess center locations
- STEP 2: Map out what data point is closest to what center
- STEP 3: Center moves to the centroid of all points it "owns"
- REPEAT!



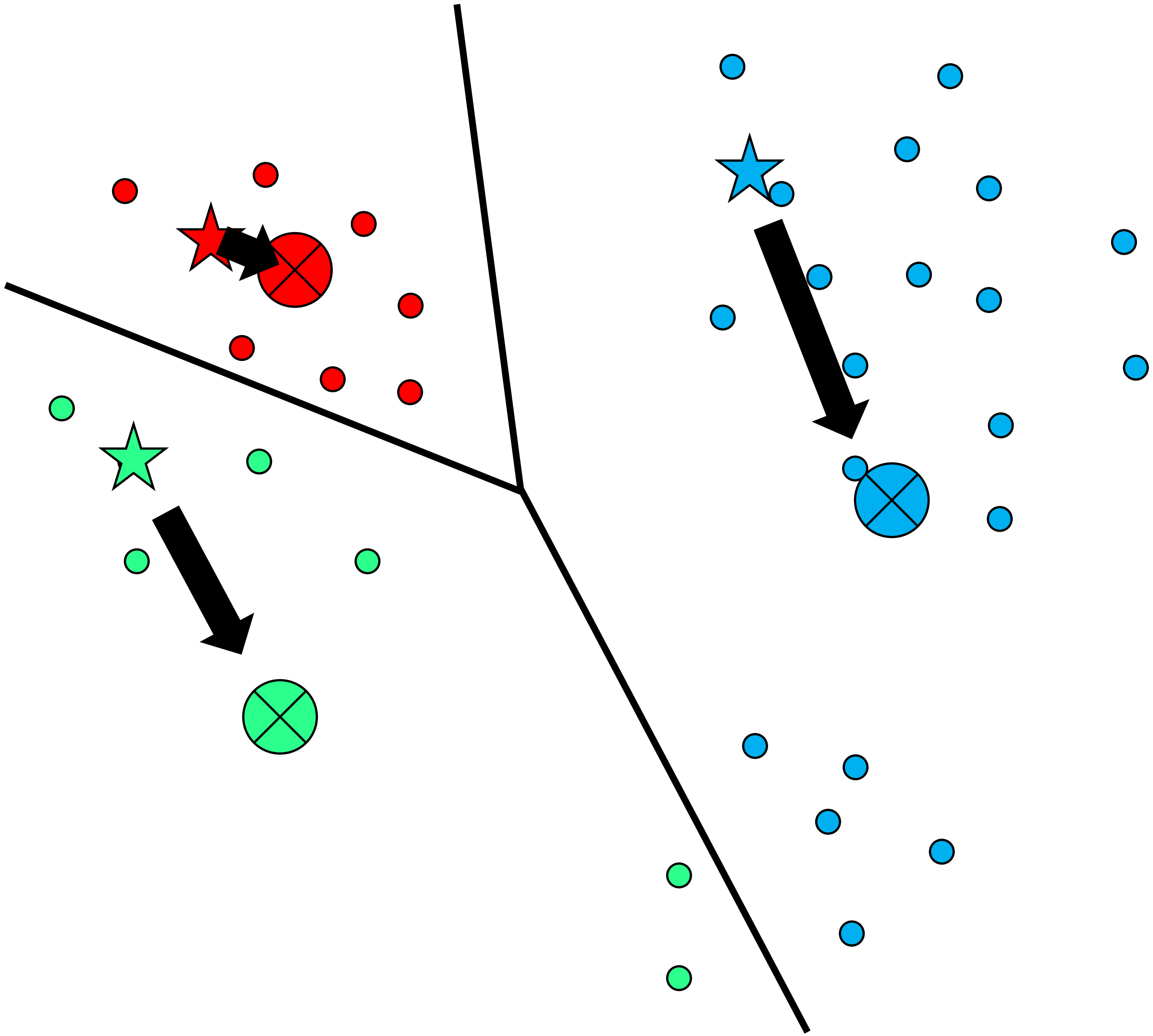
K-means

- K is number of clusters
- STEP 1: Guess center locations
- STEP 2: Map out what data point is closest to what center
- STEP 3: Center moves to the centroid of all points it "owns"
- REPEAT!



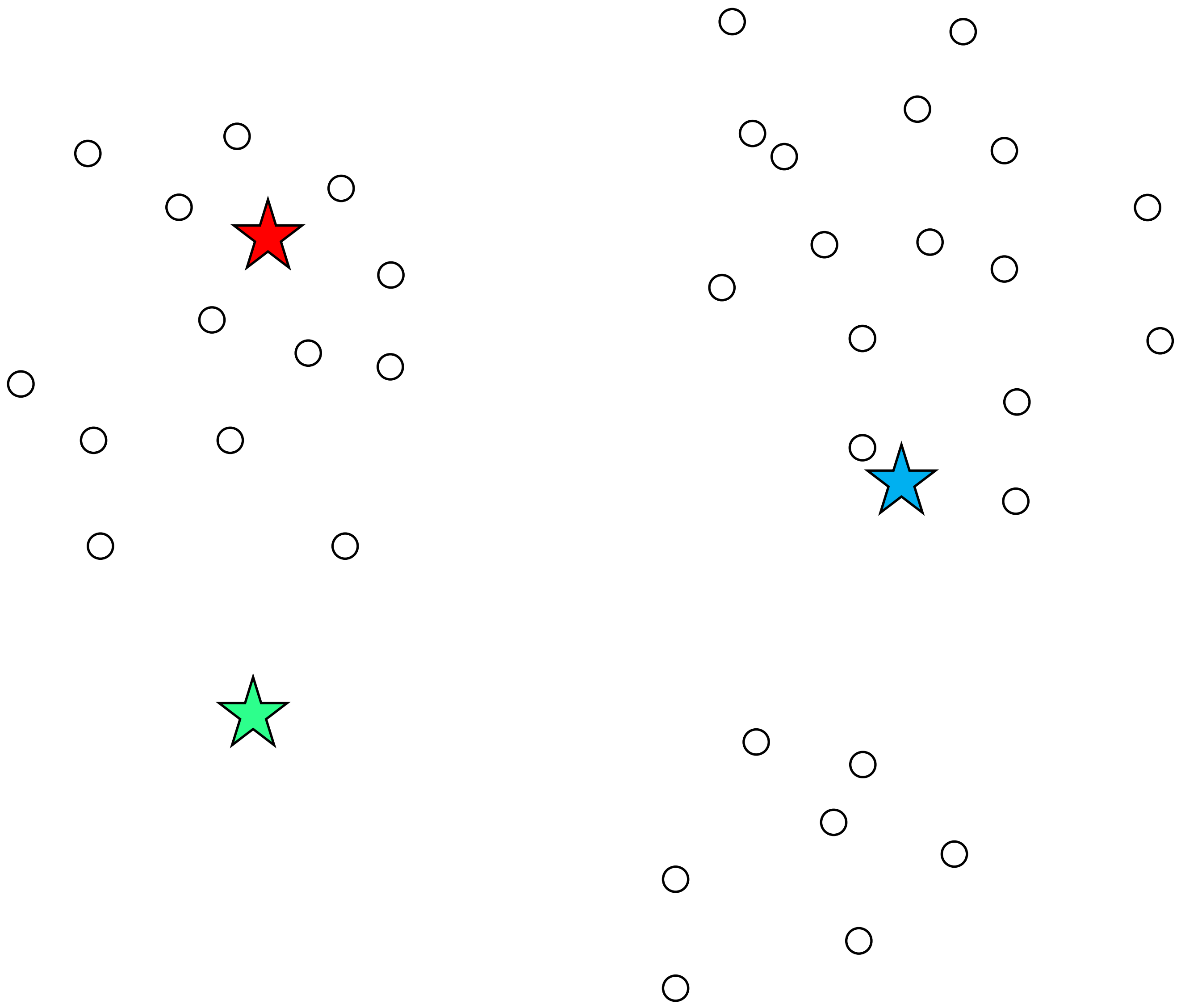
K-means

- K is number of clusters
- STEP 1: Guess center locations
- STEP 2: Map out what data point is closest to what center
- STEP 3: Center moves to the centroid of all points it "owns"
- REPEAT!



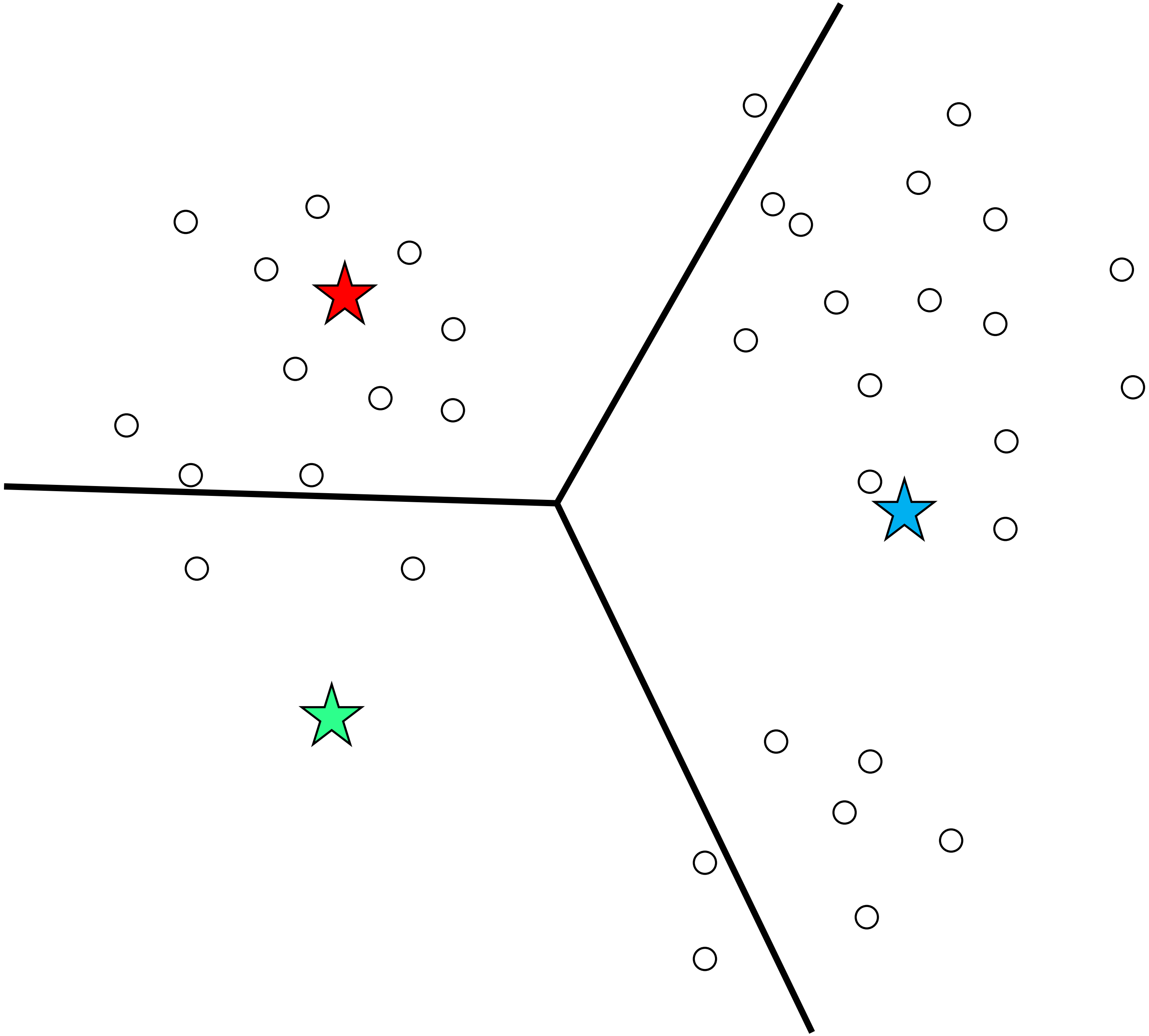
K-means

- K is number of clusters
- STEP 1: Guess center locations
- STEP 2: Map out what data point is closest to what center
- STEP 3: Center moves to the centroid of all points it “owns”
- REPEAT!



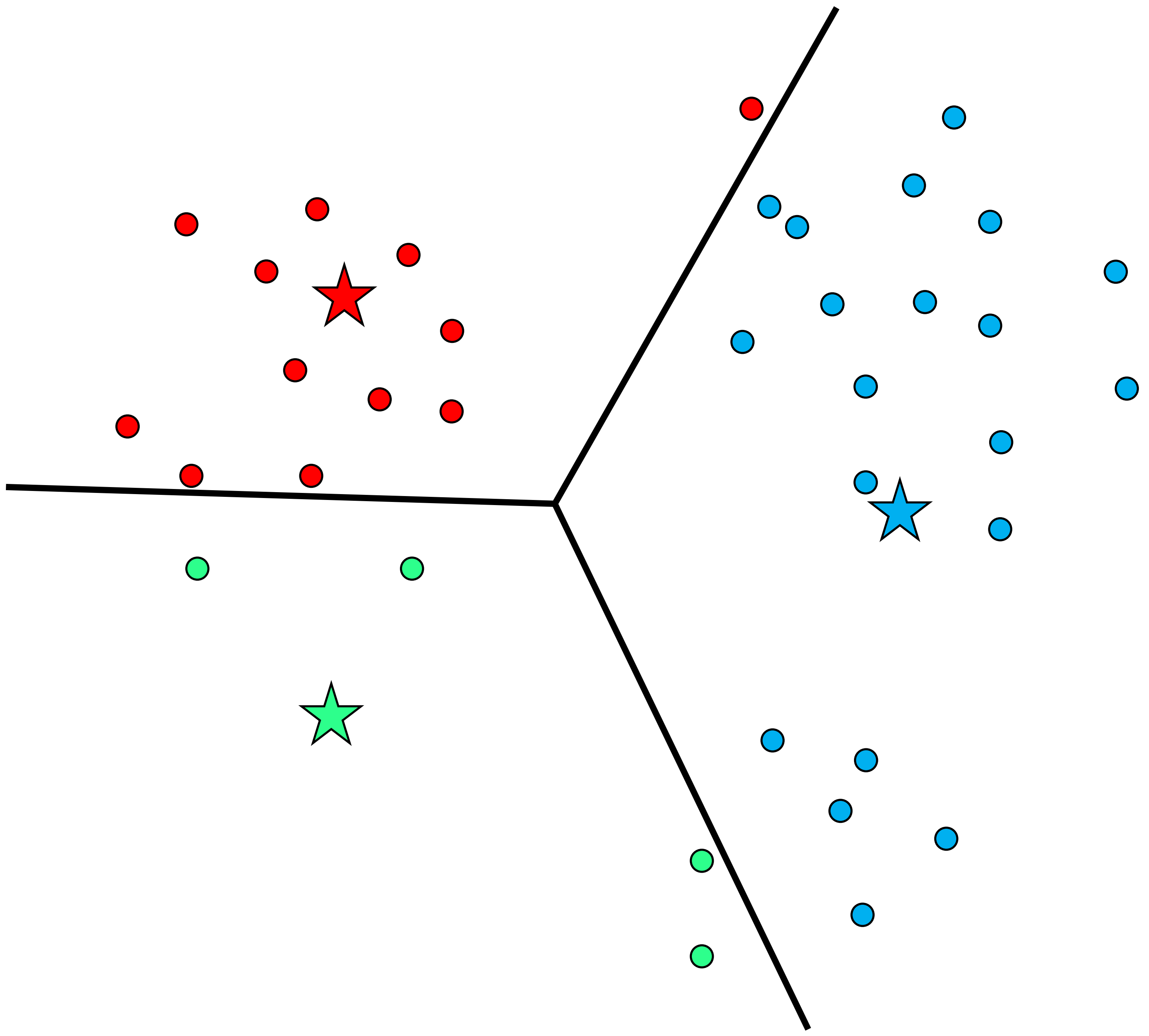
K-means

- K is number of clusters
- STEP 1: Guess center locations
- STEP 2: Map out what data point is closest to what center
- STEP 3: Center moves to the centroid of all points it "owns"
- REPEAT!



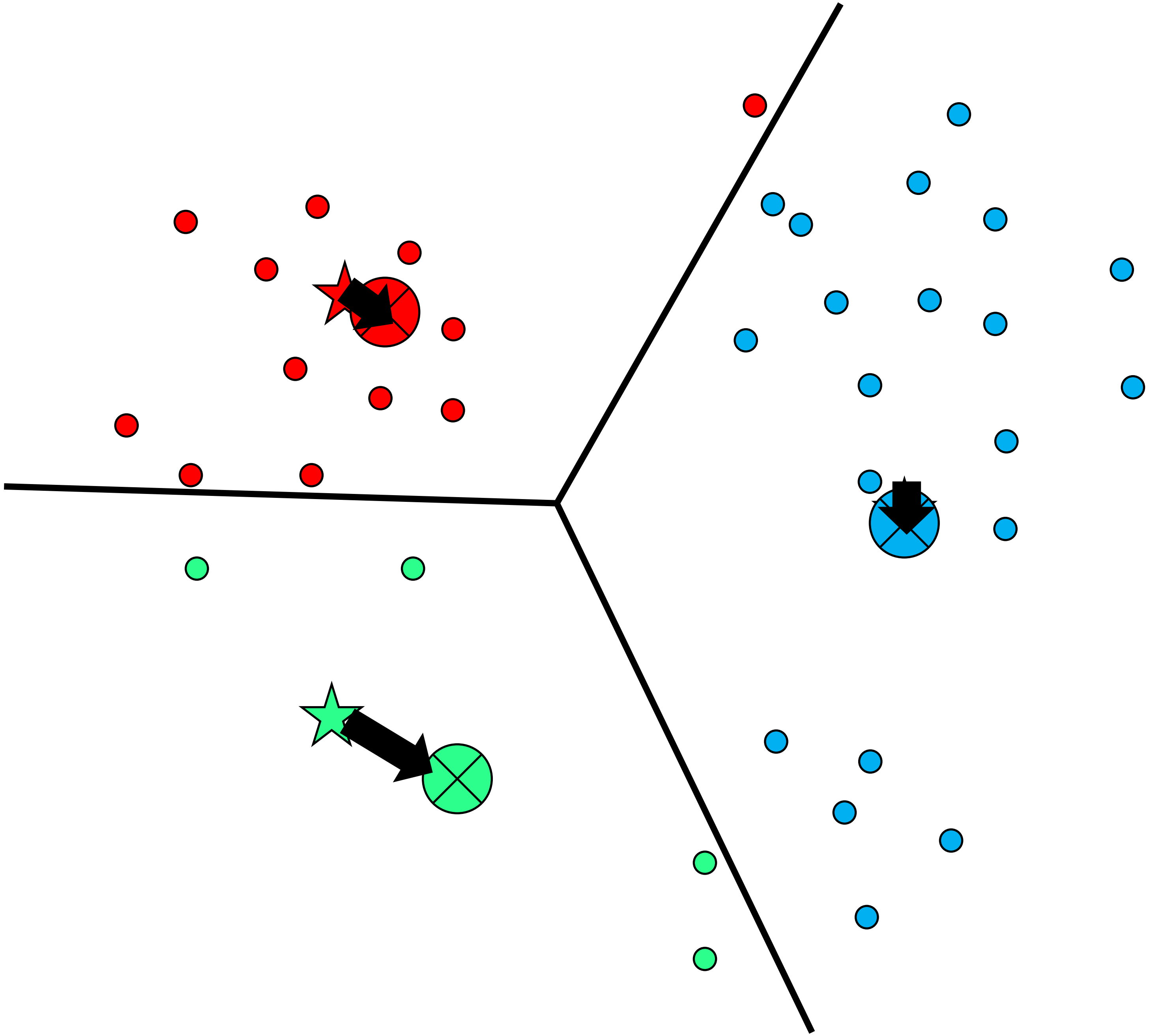
K-means

- K is number of clusters
- STEP 1: Guess center locations
- STEP 2: Map out what data point is closest to what center
- STEP 3: Center moves to the centroid of all points it “owns”
- REPEAT!



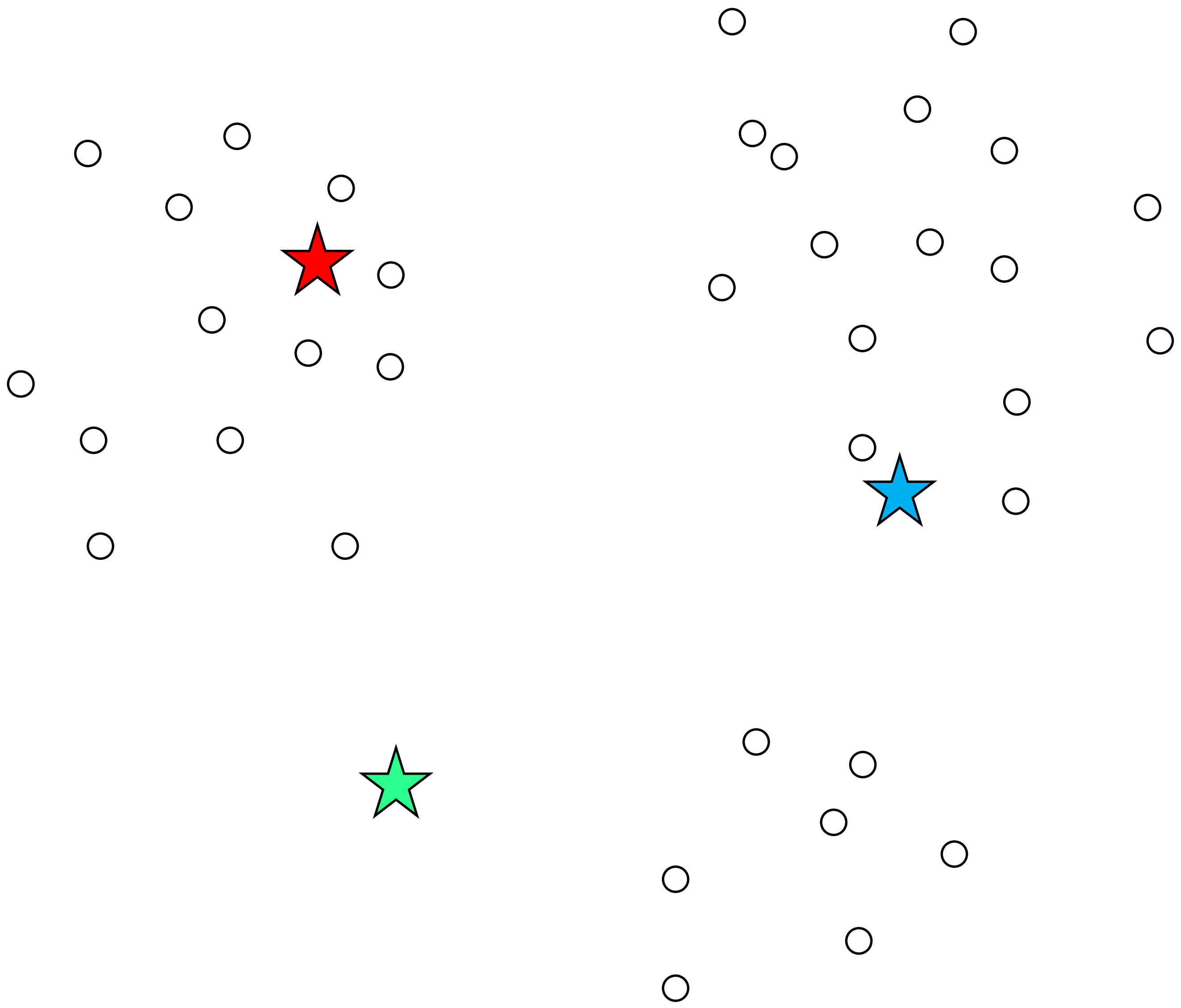
K-means

- K is number of clusters
- STEP 1: Guess center locations
- STEP 2: Map out what data point is closest to what center
- STEP 3: Center moves to the centroid of all points it “owns”
- REPEAT!



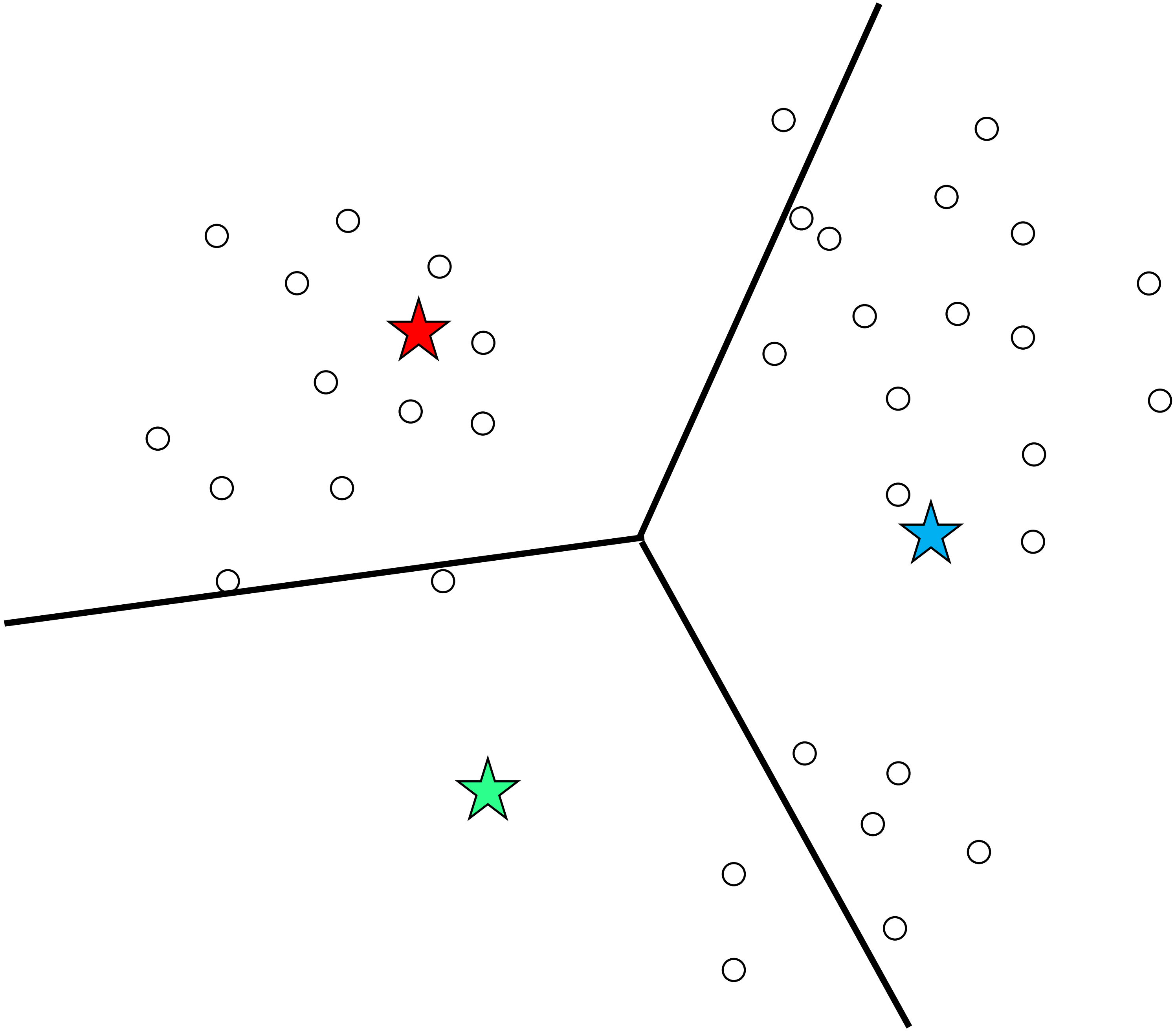
K-means

- K is number of clusters
- STEP 1: Guess center locations
- STEP 2: Map out what data point is closest to what center
- STEP 3: Center moves to the centroid of all points it “owns”
- REPEAT!



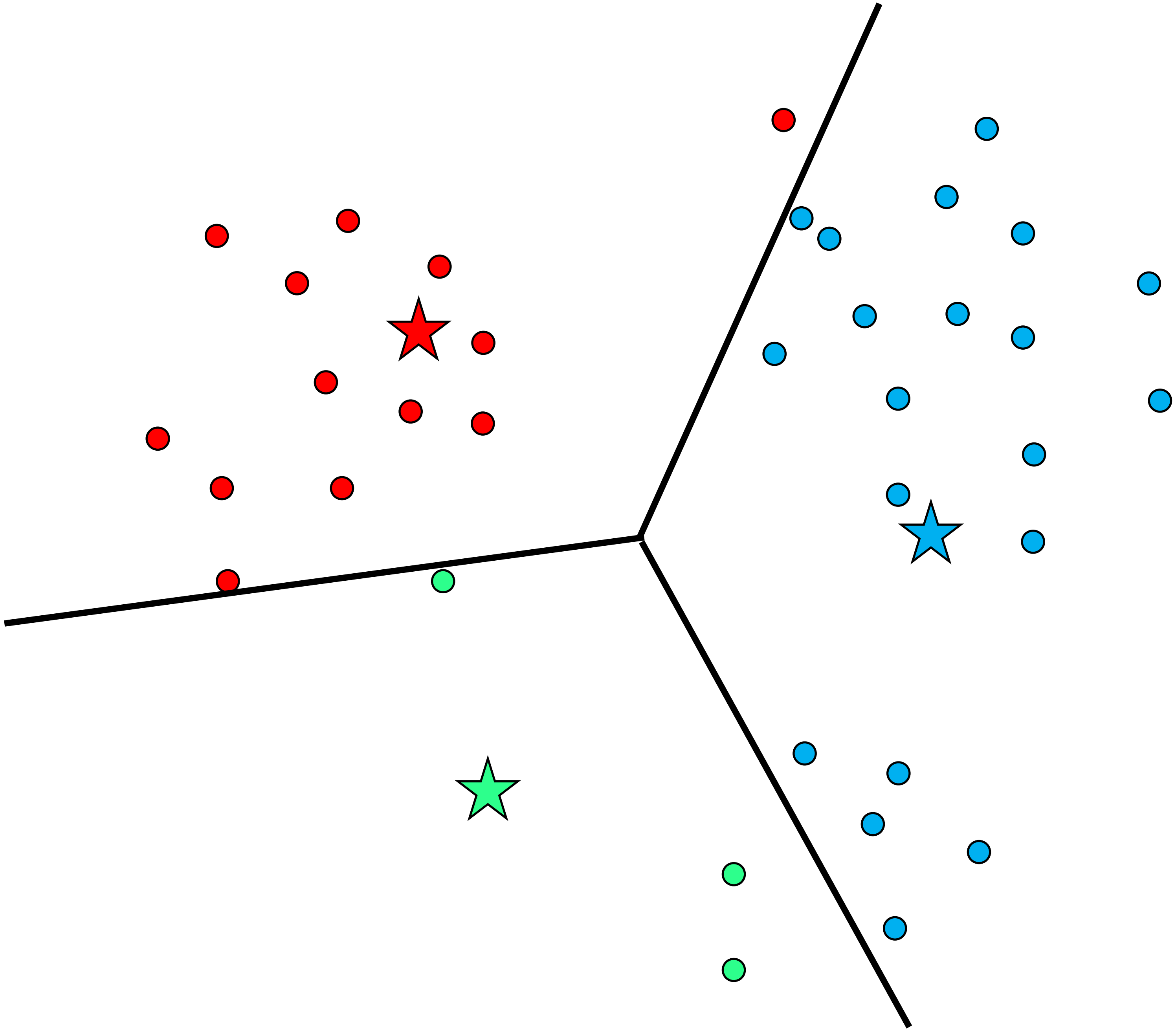
K-means

- K is number of clusters
- STEP 1: Guess center locations
- STEP 2: Map out what data point is closest to what center
- STEP 3: Center moves to the centroid of all points it “owns”
- REPEAT!



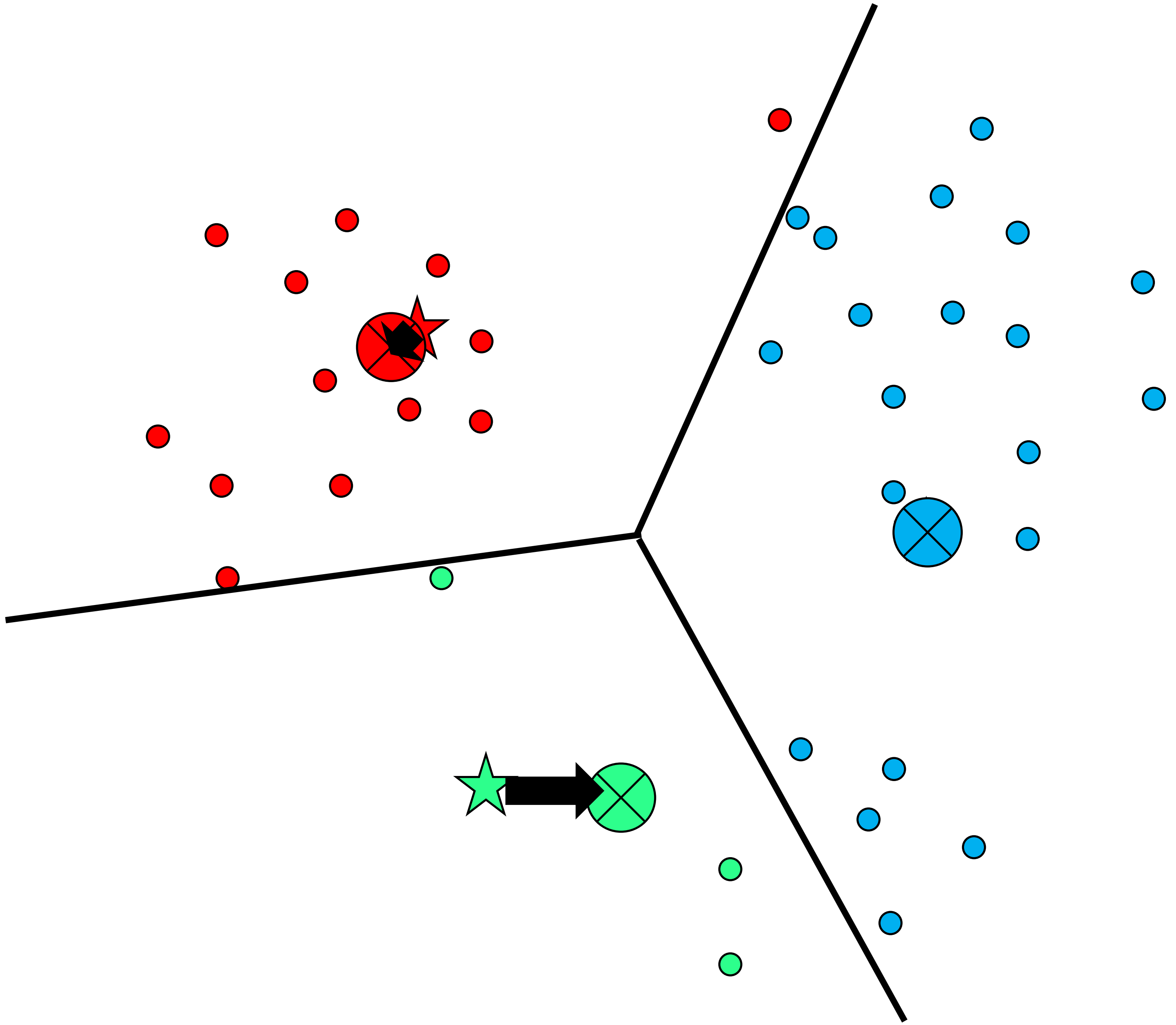
K-means

- K is number of clusters
- STEP 1: Guess center locations
- STEP 2: Map out what data point is closest to what center
- STEP 3: Center moves to the centroid of all points it "owns"
- REPEAT!



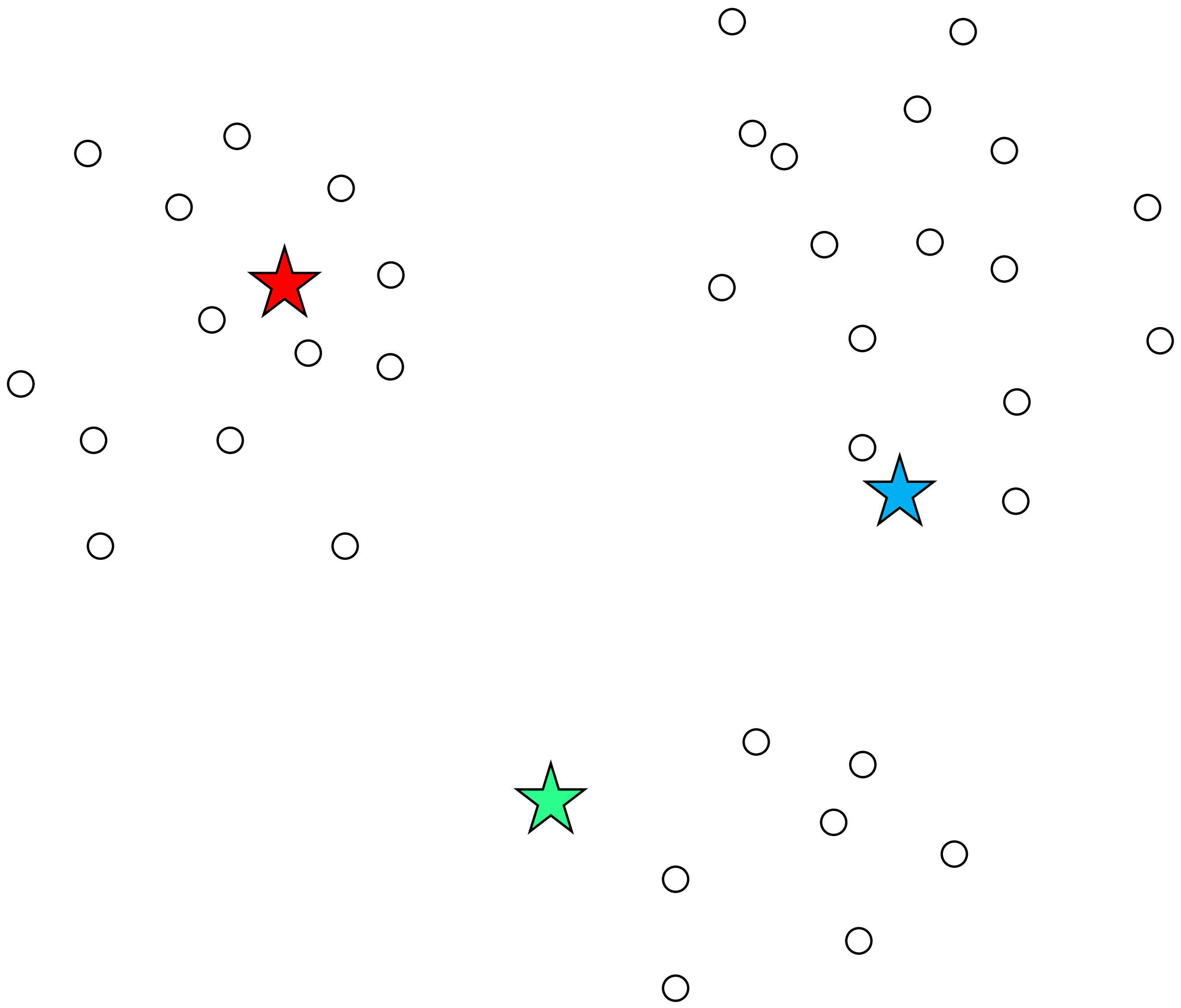
K-means

- K is number of clusters
- STEP 1: Guess center locations
- STEP 2: Map out what data point is closest to what center
- STEP 3: Center moves to the centroid of all points it "owns"
- REPEAT!



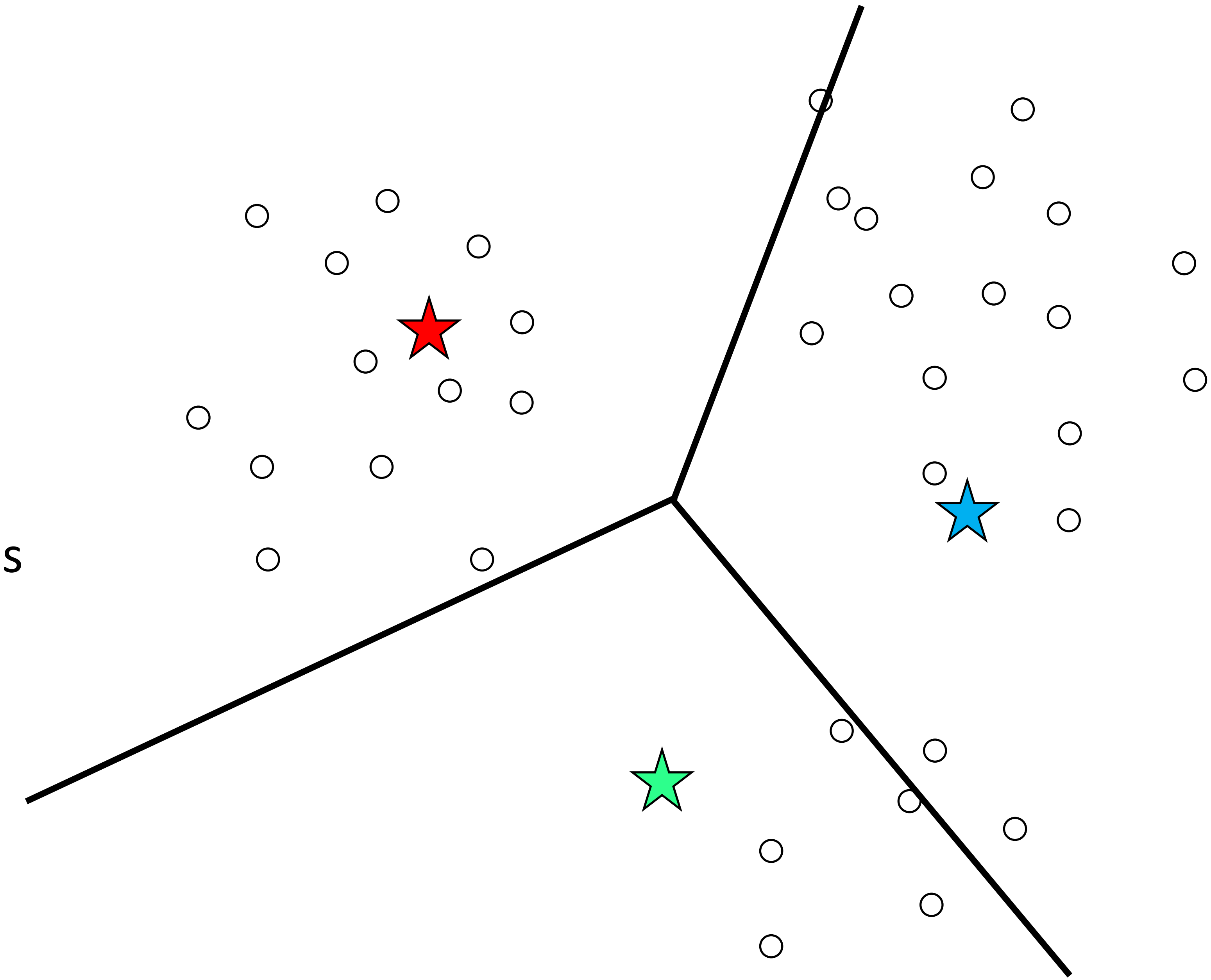
K-means

- K is number of clusters
- STEP 1: Guess center locations
- STEP 2: Map out what data point is closest to what center
- STEP 3: Center moves to the centroid of all points it "owns"
- REPEAT!



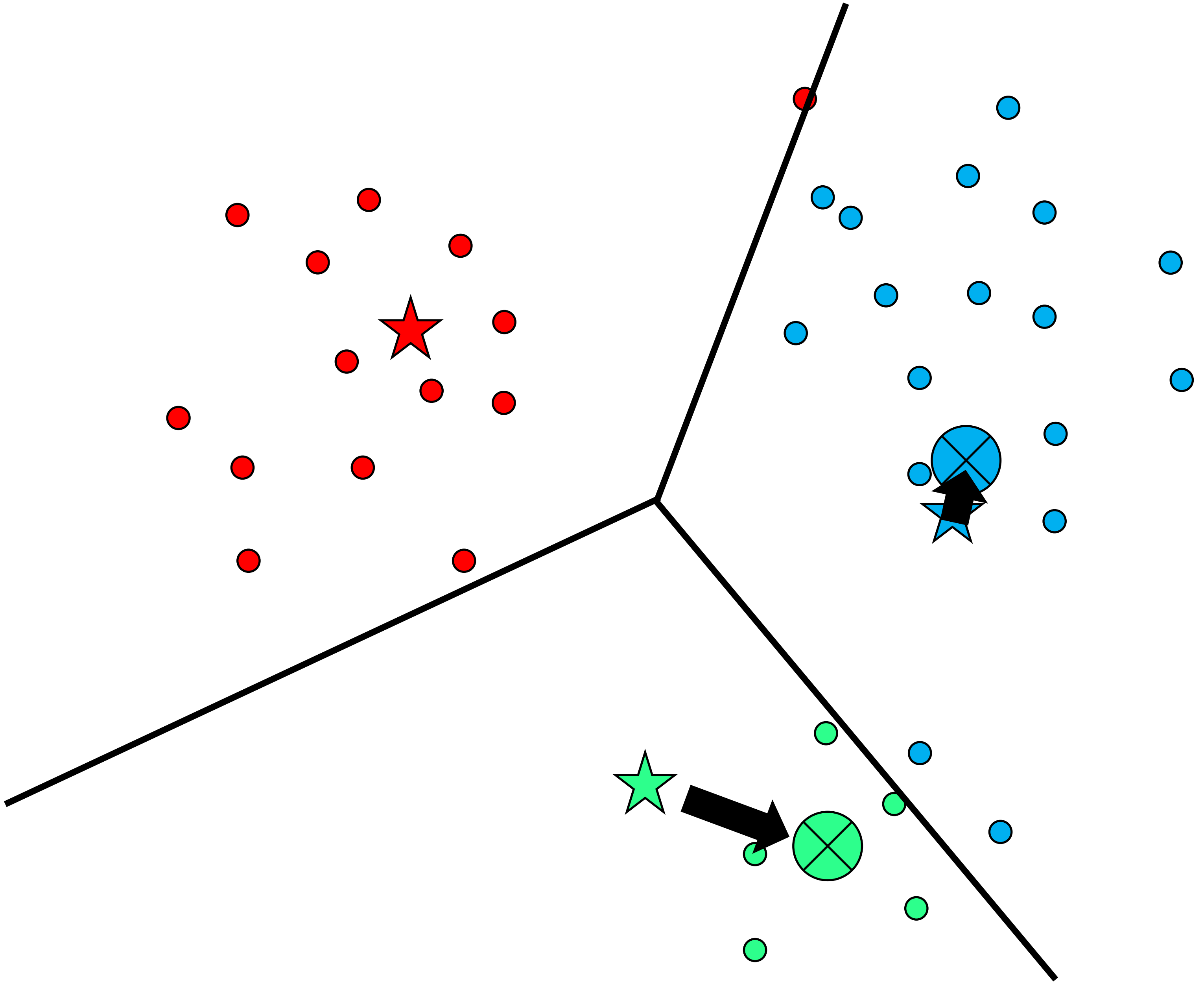
K-means

- K is number of clusters
- STEP 1: Guess center locations
- STEP 2: Map out what data point is closest to what center
- STEP 3: Center moves to the centroid of all points it “owns”
- REPEAT!



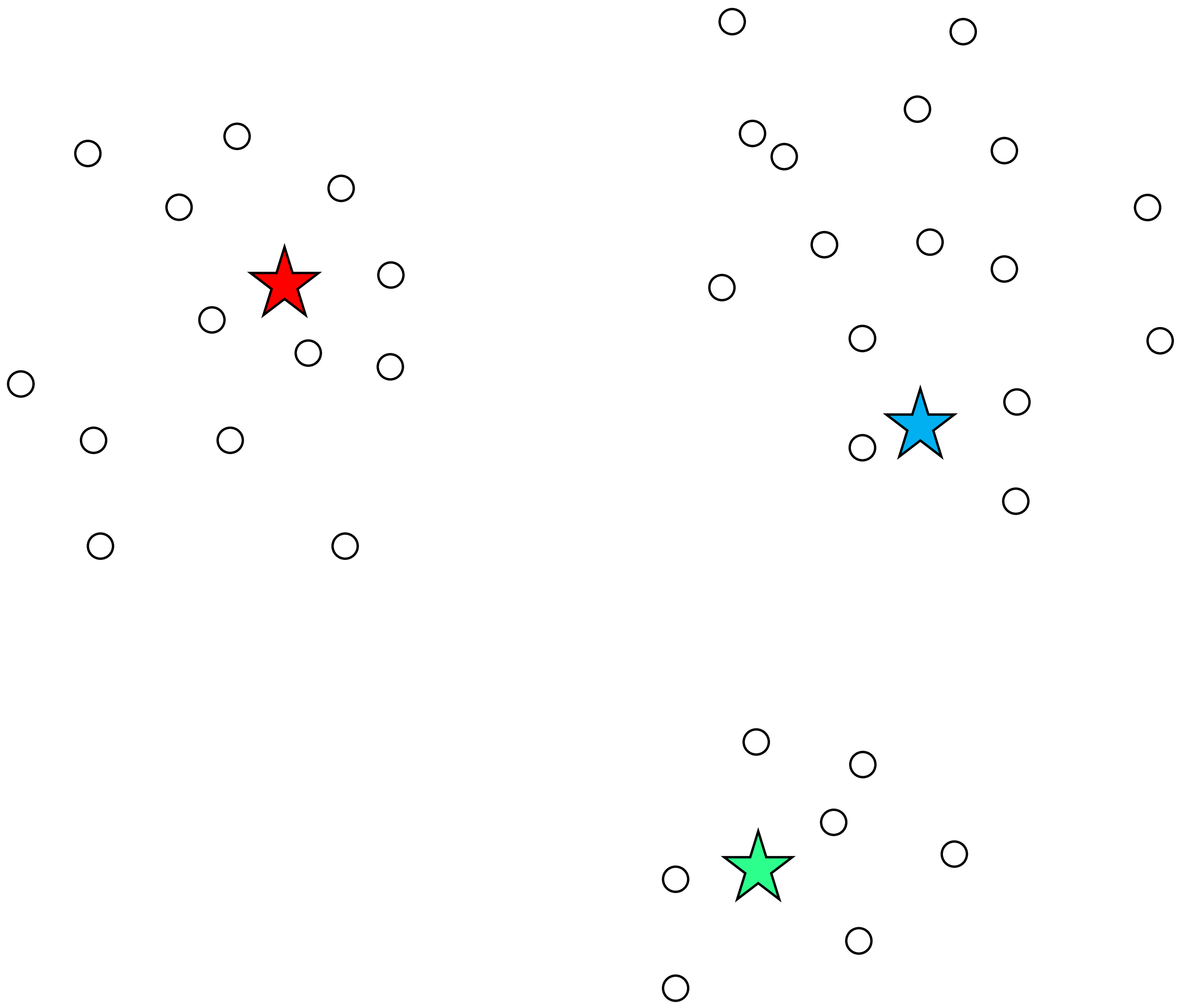
K-means

- K is number of clusters
- STEP 1: Guess center locations
- STEP 2: Map out what data point is closest to what center
- STEP 3: Center moves to the centroid of all points it "owns"
- REPEAT!



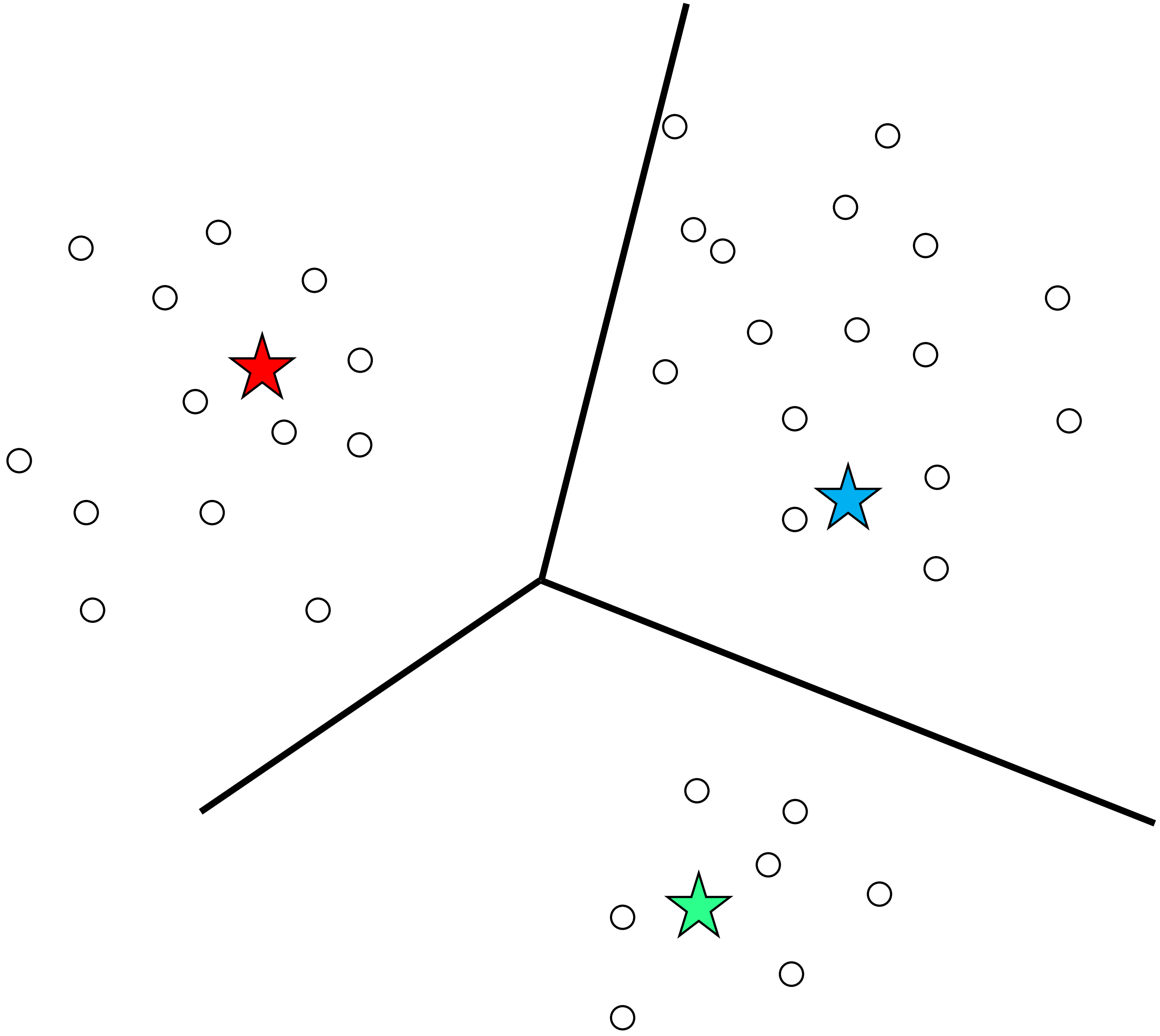
K-means

- K is number of clusters
- STEP 1: Guess center locations
- STEP 2: Map out what data point is closest to what center
- STEP 3: Center moves to the centroid of all points it "owns"
- REPEAT!



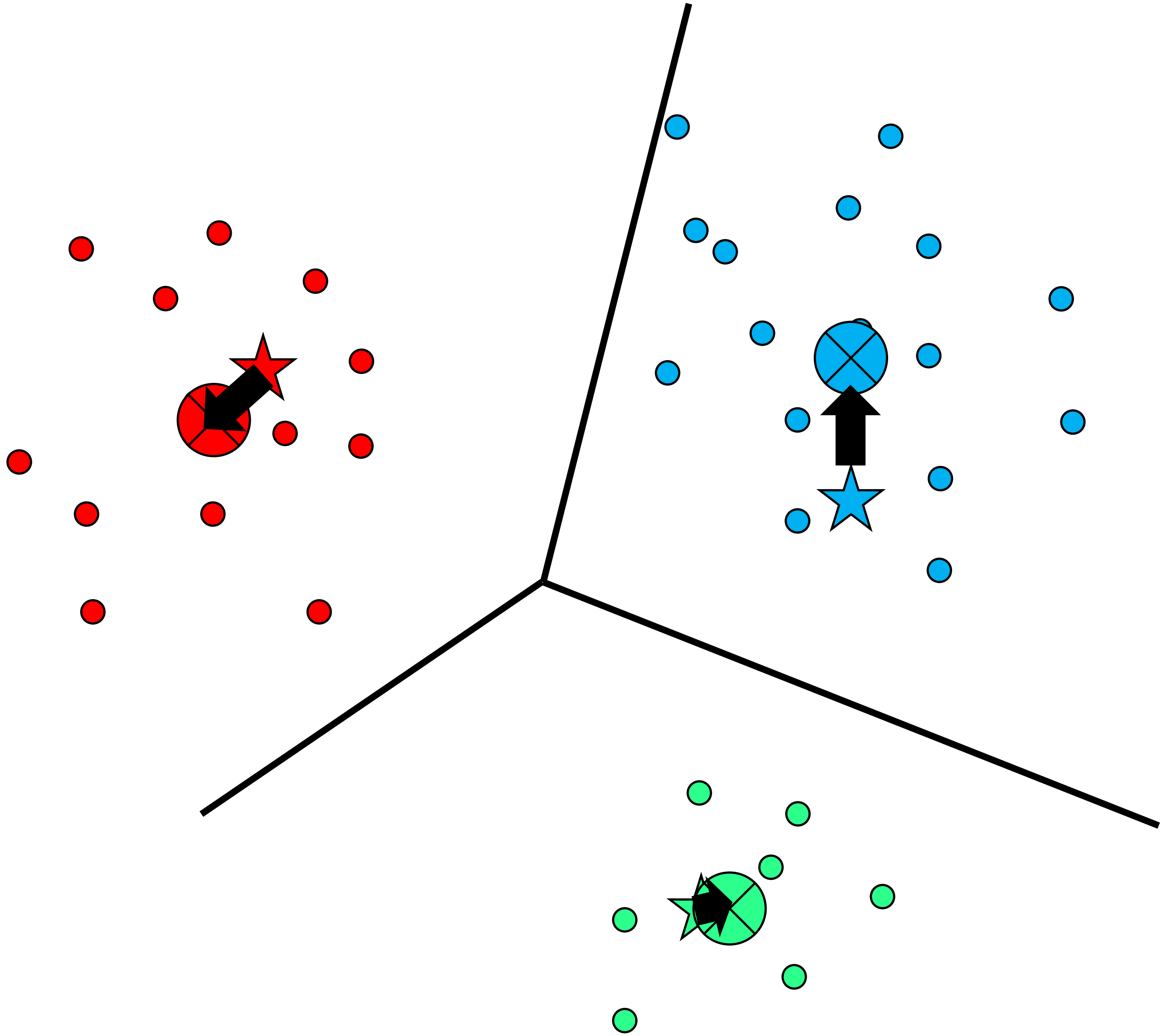
K-means

- K is number of clusters
- STEP 1: Guess center locations
- STEP 2: Map out what data point is closest to what center
- STEP 3: Center moves to the centroid of all points it "owns"
- REPEAT!



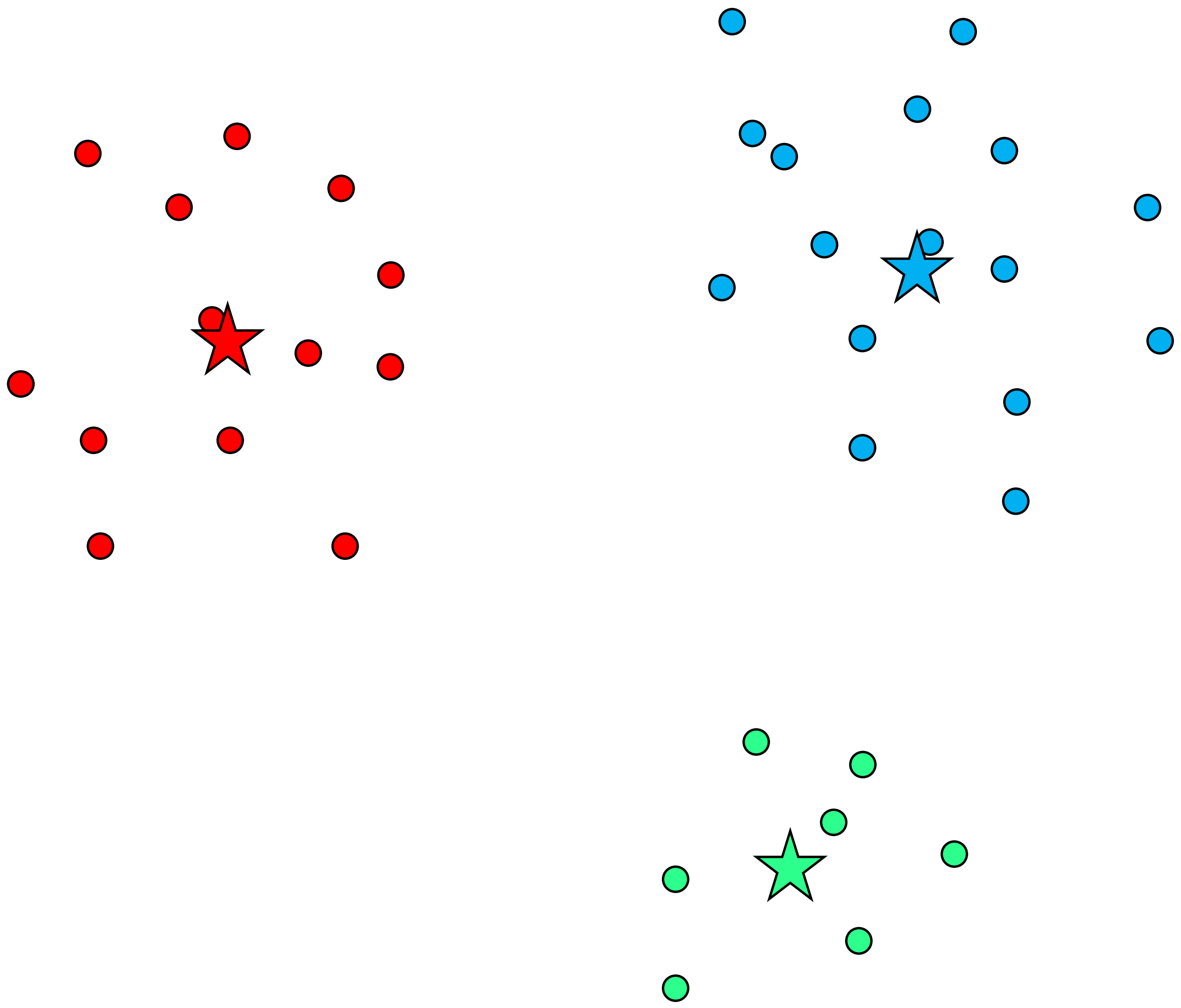
K-means

- K is number of clusters
- STEP 1: Guess center locations
- STEP 2: Map out what data point is closest to what center
- STEP 3: Center moves to the centroid of all points it "owns"
- REPEAT!



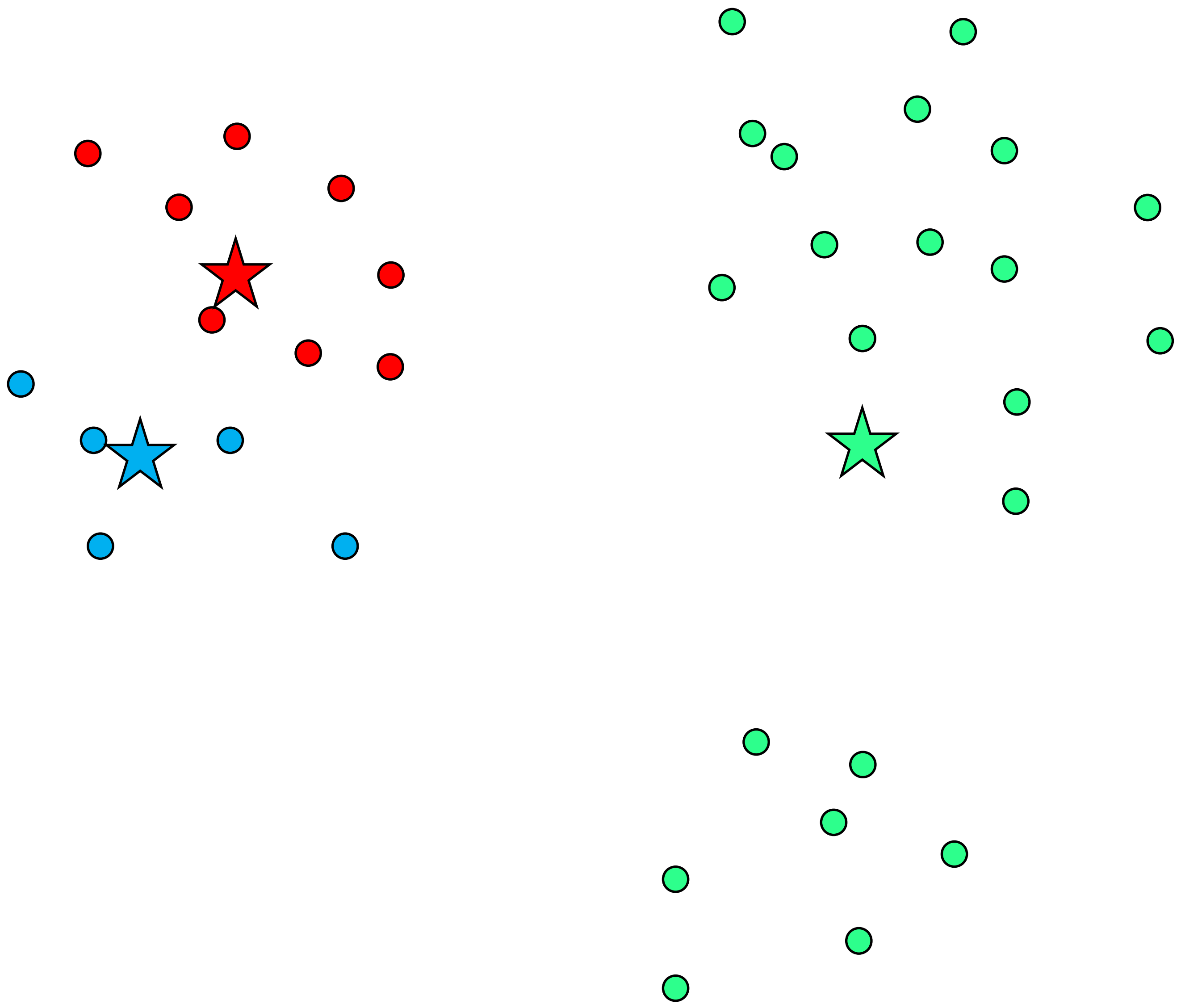
K-means

- K is number of clusters
- STEP 1: Guess center locations
- STEP 2: Map out what data point is closest to what center
- STEP 3: Center moves to the centroid of all points it "owns"
- REPEAT!



K-means

- Doesn’t always work… can get stuck



What is actually being optimized?

K-means Objective:

Find cluster centers **m** and assignments **r** to minimize the sum of squared distances of data points $\{\mathbf{x}^{(n)}\}$ to their assigned cluster centers

$$\min_{\{\mathbf{m}\},\{\mathbf{r}\}} J(\{\mathbf{m}\},\{\mathbf{r}\}) = \min_{\{\mathbf{m}\},\{\mathbf{r}\}} \sum_{n=1}^{N}\sum_{k=1}^{K} r_k^{(n)} ||\mathbf{m}_k - \mathbf{x}^{(n)}||^2$$

$$\text{s.t.} \sum_k r_k^{(n)} = 1, \forall n, \quad \text{where} \quad r_k^{(n)} \in \{0,1\}, \forall k, n$$

where $r_k^{(n)} = 1$ means that $\mathbf{x}^{(n)}$ is assigned to cluster k (with center $\mathbf{m}_k$)

- Optimization method is a form of coordinate descent ("block coordinate descent")
  - Fix centers, optimize assignments (choose cluster whose mean is closest)
  - Fix assignments, optimize means (average of assigned datapoints)

Slide credit: Ethan Fetaya, James Lucas, Emad Andrews

- Initialization: Set K cluster means $\mathbf{m}_1, \ldots, \mathbf{m}_K$ to random values
- Repeat until convergence (until assignments do not change):
  - Assignment: Each data point $\mathbf{x}^{(n)}$ assigned to nearest mean

    $$\hat{k}^n = arg\min_k d(\mathbf{m}_k, \mathbf{x}^{(n)})$$

    (with, for example, L2 norm: $\hat{k}^n = arg\min_k ||\mathbf{m}_k - \mathbf{x}^{(n)}||^2$)

    and Responsibilities (1-hot encoding)

    $$r_k^{(n)} = 1 \longleftrightarrow \hat{k}^{(n)} = k$$

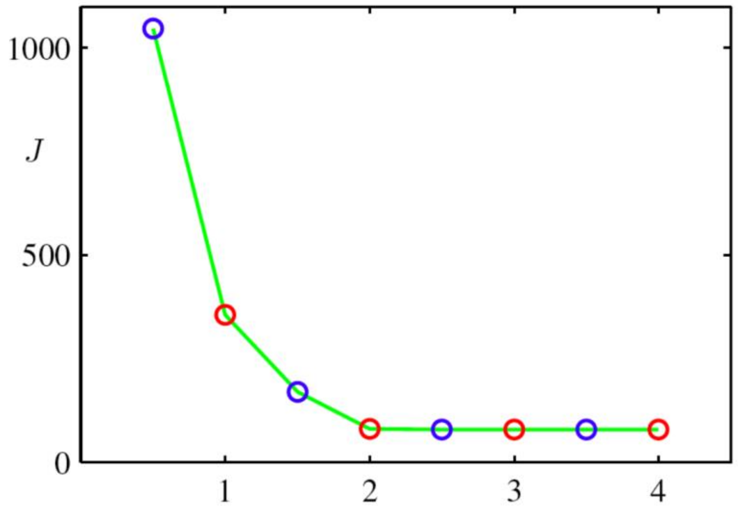
  - Refitting: Model parameters, means are adjusted to match sample means of data points they are responsible for:

    $$\mathbf{m}_k = \frac{\sum_n r_k^{(n)} \mathbf{x}^{(n)}}{\sum_n r_k^{(n)}}$$

Slide credit: Ethan Fetaya, James Lucas, Emad Andrews

K-means Convergence

- Whenever an assignment is changed, the sum squared distances J of data points from their assigned cluster centers is reduced.
- Whenever a cluster center is moved, J is reduced.
- Test for convergence: If the assignments do not change in the assignment step, we have converged (to at least a local minimum).

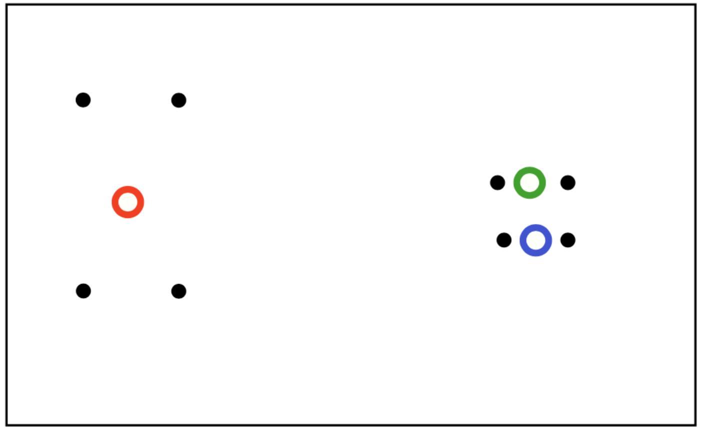


Slide credit: Ethan Fetaya, James Lucas, Emad Andrews

Local Minima

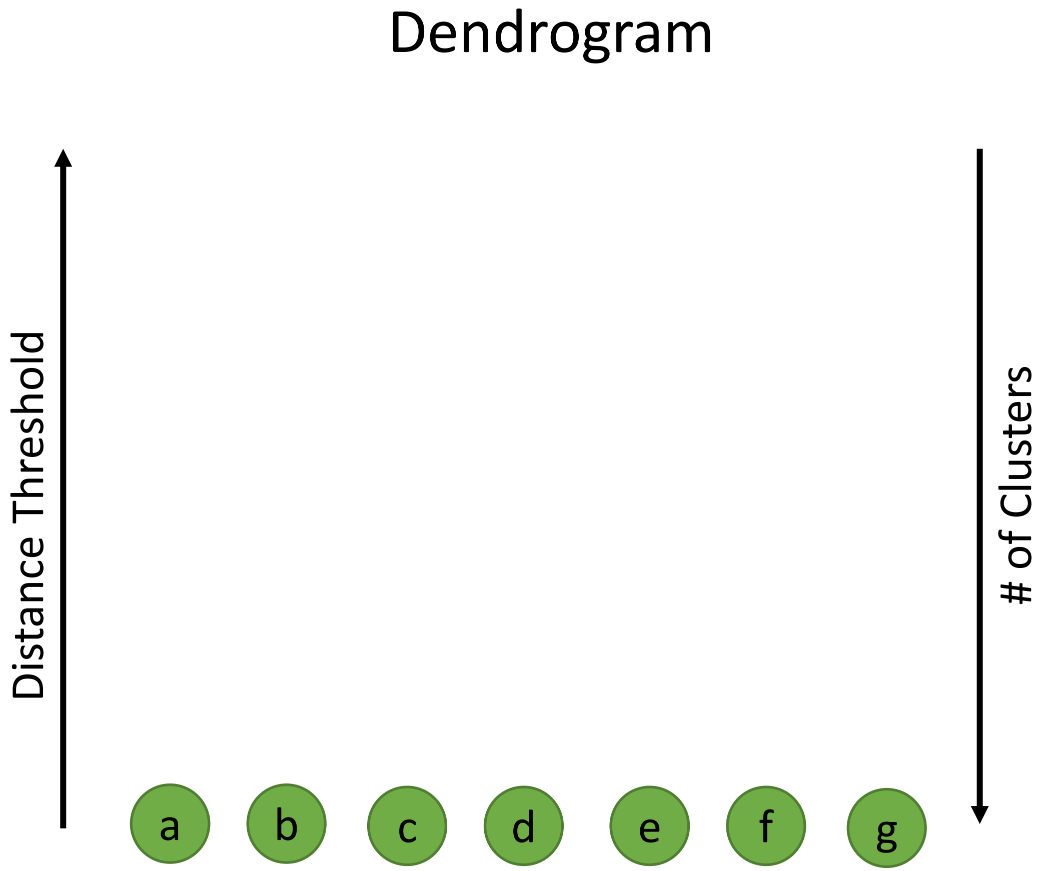
- The objective J is non-convex (so coordinate descent on J is not guaranteed to converge to the global minimum)
- There is nothing to prevent k-means getting stuck at local minima.
- We could try many random starting points
- We could try non-local split-and-merge moves:
  - Simultaneously merge two nearby clusters
  - and split a big cluster into two

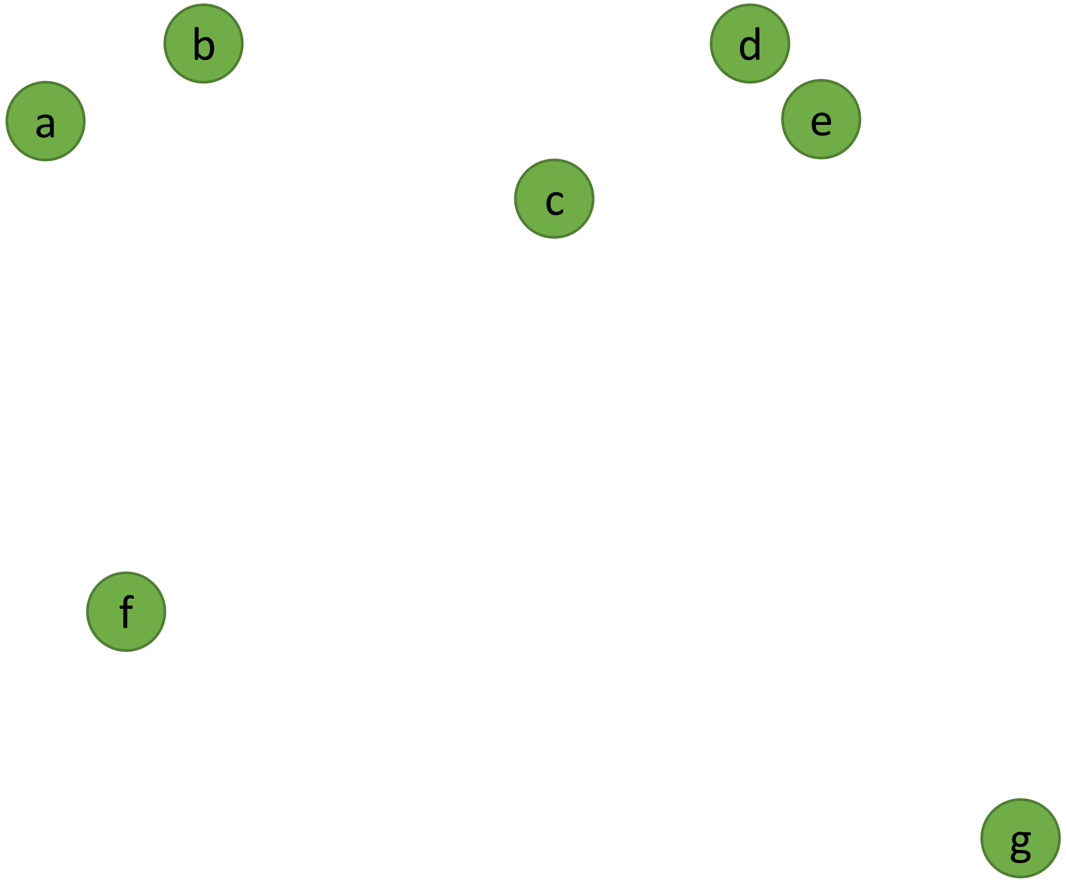
A bad local optimum



Slide credit: Ethan Fetaya, James Lucas, Emad Andrews

Agglomerative clustering





Credit – Victor Lavrenko

Python & Data

Missing Values

There are a number of methods to deal with missing values in the data frame:

df.method()	description
dropna()	Drop missing observations
dropna(how='all')	Drop observations where all cells is NA
dropna(axis=1, how='all')	Drop column if all the values are missing
dropna(thresh = 5)	Drop rows that contain less than 5 non-missing values
fillna(0)	Replace missing values with zeros
isnull()	returns True if the value is missing
notnull()	Returns True for non-missing values

Missing Values

- When summing the data, missing values will be treated as zero
- If all values are missing, the sum will be equal to NaN
- Missing values in GroupBy method are excluded (just like in R)
- Many descriptive statistics methods have *skipna* option to control if missing data should be excluded . This value is set to *True* by default (unlike R)

Aggregation Functions in Pandas

Aggregation - computing a summary statistic about each group, i.e.

- compute group sums or means
- compute group sizes/counts

Common aggregation functions:

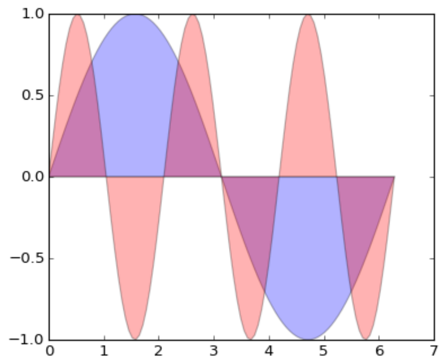
min, max
count, sum, prod
mean, median, mode, mad
std, var

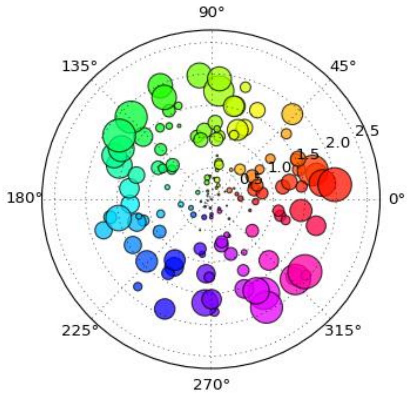
Plotting and Visualization

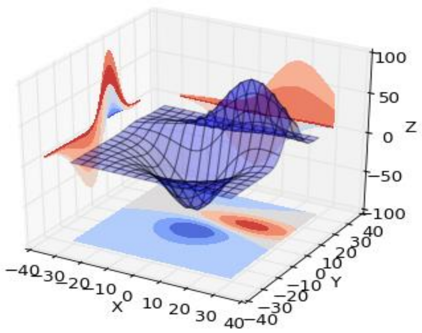
Matplotlib

We're going to continue our discussion of scientific computing with matplotlib.

Matplotlib is an incredibly powerful (and beautiful!) 2-D plotting library. It's easy to use and provides a huge number of examples for tackling unique problems.





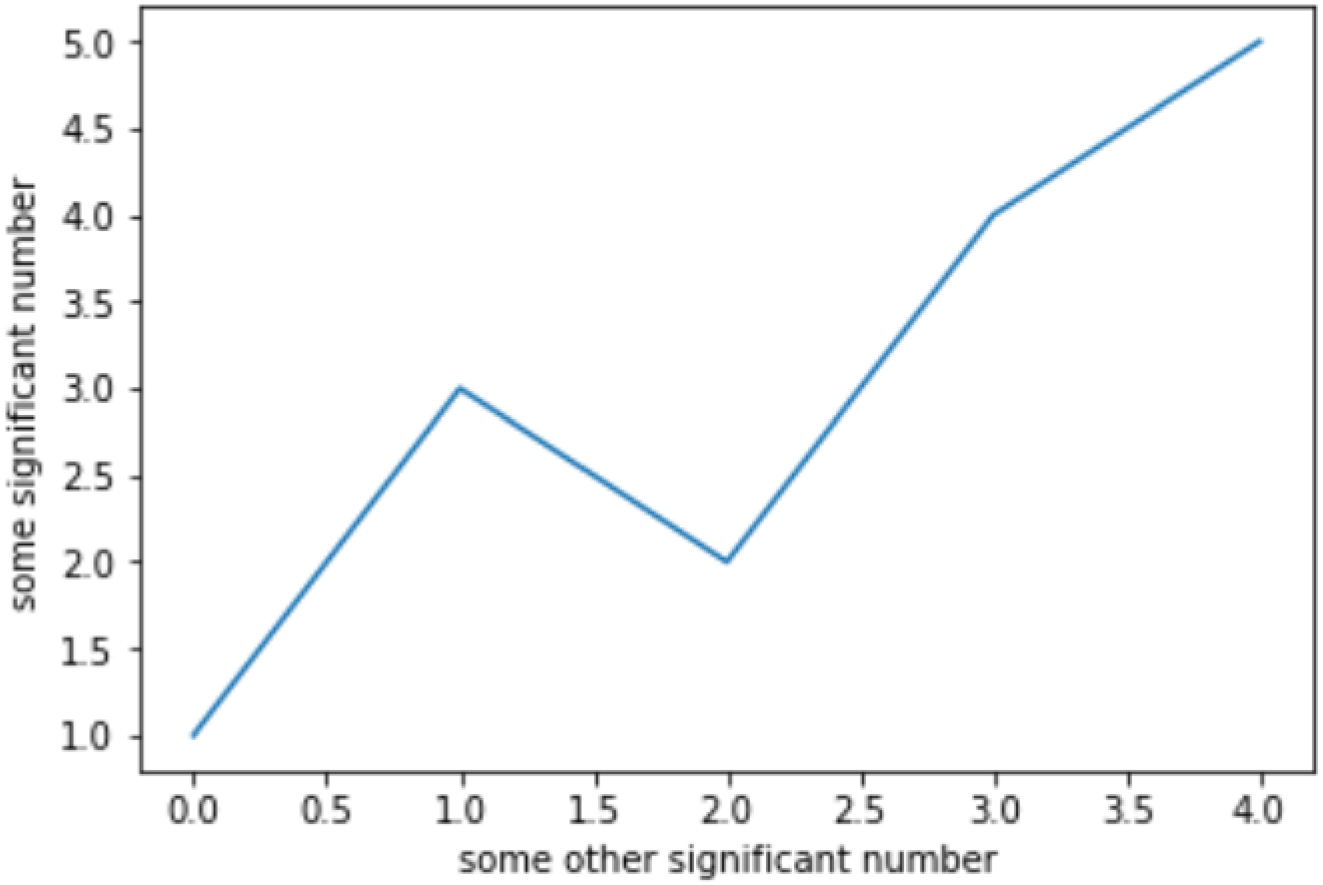


pyplot

At the center of most matplotlib scripts is pyplot. The pyplot module is stateful and tracks changes to a *figure*. All pyplot functions revolve around creating or manipulating the state of a figure.

When a single sequence object is passed to the plot function, it will generate the x-values for you starting with 0.

```
import numpy as np
import matplotlib.pyplot as plt
plt.plot([1, 3, 2, 4, 5])
plt.ylabel('some significant number')
plt.xlabel('some other significant number')
plt.show()
```



pyplot

- The plot function can actually take any number of arguments.
- The format string argument associated with a pair of sequence objects indicates the color and line type of the plot (e.g. ‘bs’ indicates blue squares and ‘ro’ indicates red circles).
- Generally speaking, the `x_values` and `y_values` will be numpy arrays and if not, they will be converted to numpy arrays internally.
- Line properties can be set via keyword arguments to the plot function. Examples include `label`, `linewidth`, `animated`, `color`, etc…

pyplot

- The `text()` command can be used to add text in an arbitrary location
- `xlabel()` adds text to x-axis.
- `ylabel()` adds text to y-axis.
- `title()` adds title to plot.
- `clear()` removes all plots from the axes.

All methods are available on pyplot and on the axes instance generally.

Graphics

	description
distplot	histogram
barplot	estimate of central tendency for a numeric variable
violinplot	similar to boxplot, also shows the probability density of the data
jointplot	Scatterplot
regplot	Regression plot
pairplot	Pairplot
boxplot	boxplot
swarmplot	categorical scatterplot
factorplot	General categorical plot