APS1070

Foundations of Data Analytics and Machine Learning

Winter 2021

Week 3: Foundations of Learning

- *K-Nearest Neighbours*
- *Decision Trees*
- *Clustering*

Sinisa Colic, PhD, PEng

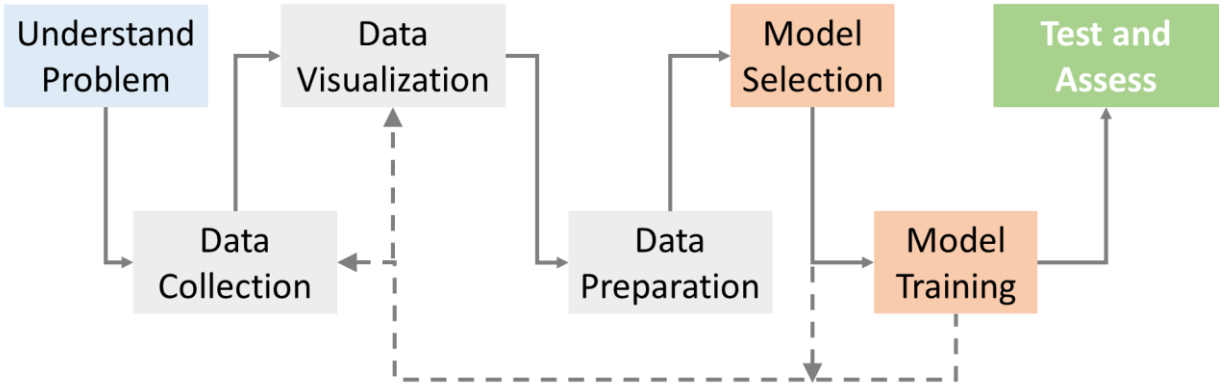
Slide Attribution

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

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

Last Time

- Python Checklist and Expectations
- Algorithms and Big O Notation
- Python for Data Science
  - NumPy
  - Matplotlib
  - Pandas



Agenda

Today's focus is on **Foundations of Learning**

- Machine Learning Taxonomy
- k-Nearest Neighbours
- Assessing Performance
- Decisions Trees
- Clustering Strategies
  - k-Means
  - Density-Based Clustering
  - Agglomerative Clustering

Supervised Learning

Unsupervised Learning

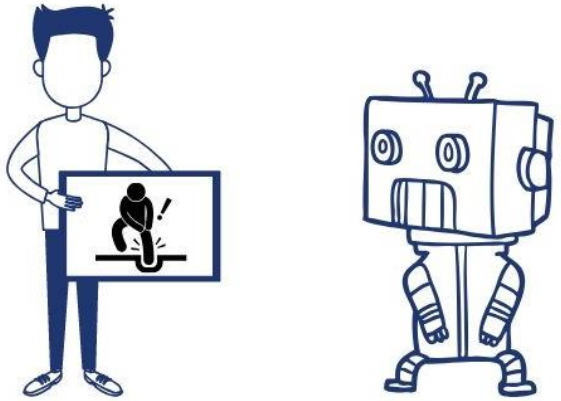
Machine Learning Taxonomy

- Machine learning is about learning from the data and can be categorized based on the following:
  - Supervised vs Unsupervised
  - Classification vs Regression
  - Parametric vs non-Parametric
  - Instance-based vs Model-based

- Today we will explore the above categories using **instance-based learning**.
- Model-based learning will be covered in the 2nd half of the course.

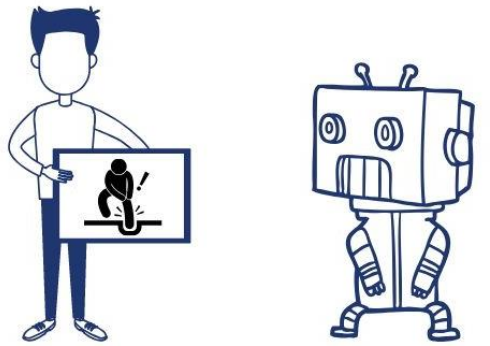
Classification and Regression

- **Classification:** discrete output or 1 of N
  - Example: Sentiment classification a tweet:
    - Input: tweet text
    - Output: whether the tweet is happy or sad
- **Regression:** real-valued or continuous value
  - Examples: Age prediction given a headshot:
    - Input: headshot image
    - Output: person's age

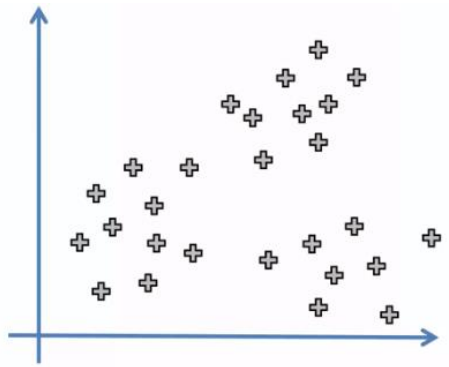


Supervised vs Unsupervised

- **Supervised Learning:** Learning model that maps an input to an output based on example input-output pairs
  - e.g., classification and regression



- **Unsupervised Learning:** Learning in the absence of a teacher, or feedback on what the output should be.
  - e.g., clustering



Parametric vs Nonparametric

- **Parametric:**
  - Have a fixed number of parameters
  - Estimate of fixed parameters improves with more data
  - Make strong assumptions about the data
- **Nonparametric:**
  - Number of parameters grow with # samples.
  - Size depends on # samples.
  - Complexity grows with # samples.
- Trade-offs between parametric and non-parametric algorithms are in computational cost and accuracy.

Instance-Based vs Model-Based Learning

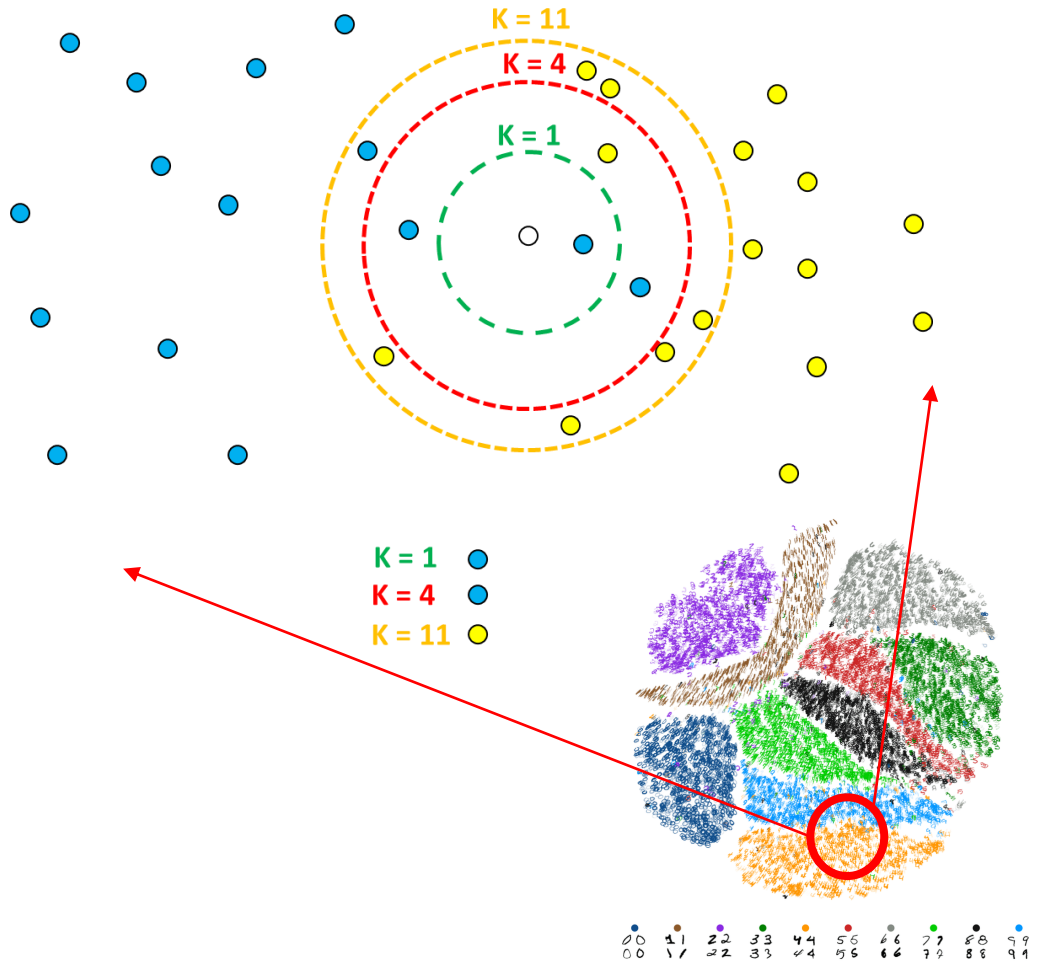
- **Instance-Based:**
  - system learns the examples by heart, then generalizes to new cases by using a similarity/distance measure to compare them to the learned examples.

- **Model-Based:**
  - build a model of these examples and then use that model to make predictions.
    - more details in weeks 9 to 11

k-Nearest-Neighbours (Supervised Learning)

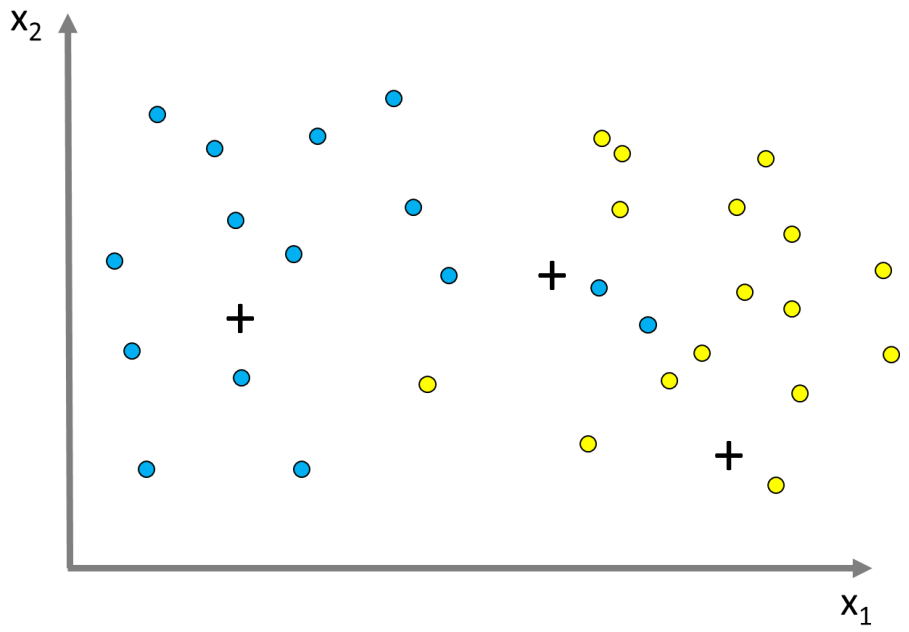
k-Nearest Neighbour Classification

- Distance-based supervised learning approach
- Instance-based learning or “lazy” learning (just stores the training data)
- Flexible and makes no assumptions on data distribution (**nonparametric**)
- Typically used for classification, but can also be extended to regression



k-Nearest Neighbour Classification

- Works on multidimensional data
- For visualization purposes we will use a 2-dimensional example
- + represents unknown test samples that we want to classify



Simple Algorithm

- We can make the assumption that similar samples will be located close together
- Simple Algorithm

Calculate distance

Obtain the nearest neighbour

Determine labels

Algorithm

- For a single nearest neighbour

1. Find example $(\mathbf{x}^*, t^*)$ (from the stored training set) closest to $\mathbf{x}$. That is:

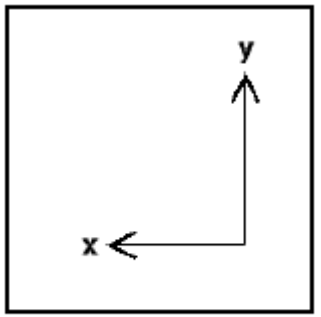
$$\mathbf{x}^* = \underset{\mathbf{x}^{(i)} \in \text{train. set}}{\text{argmin}} \ \text{distance}(\mathbf{x}^{(i)}, \mathbf{x})$$

2. Output $y = t^*$

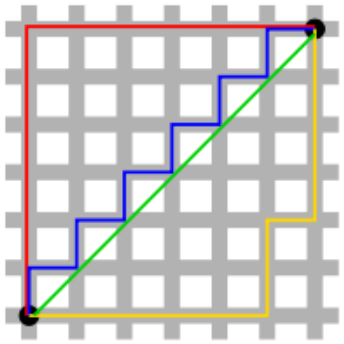
How do we measure distance?

$$\left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p}$$

p = 1, Manhattan Distance

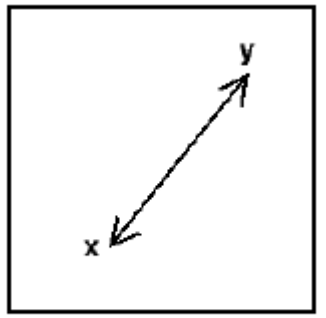


Manhattan



Source — Taxicab geometry Wikipedia

p = 2, Euclidean Distance



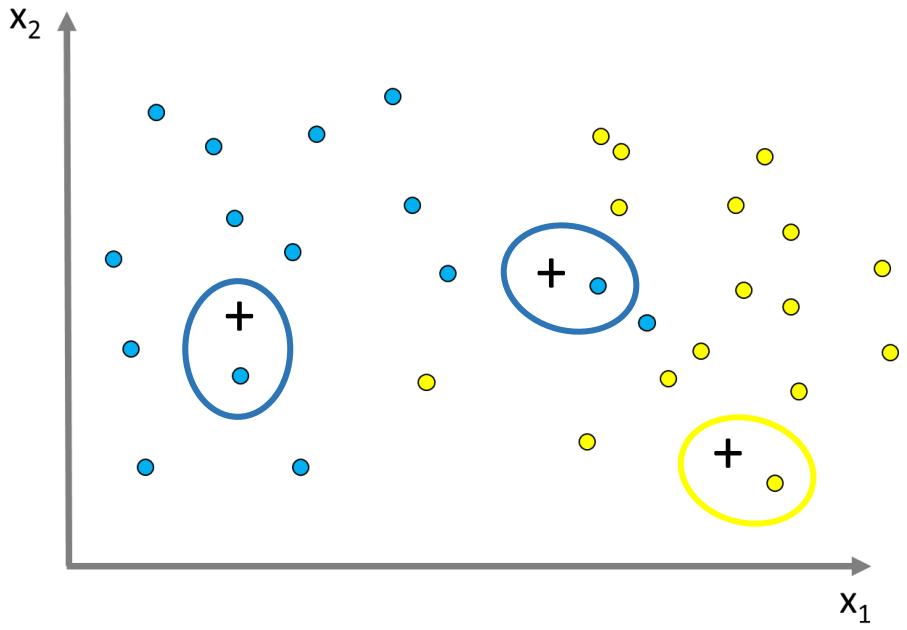
Euclidean

Dimension of x or y

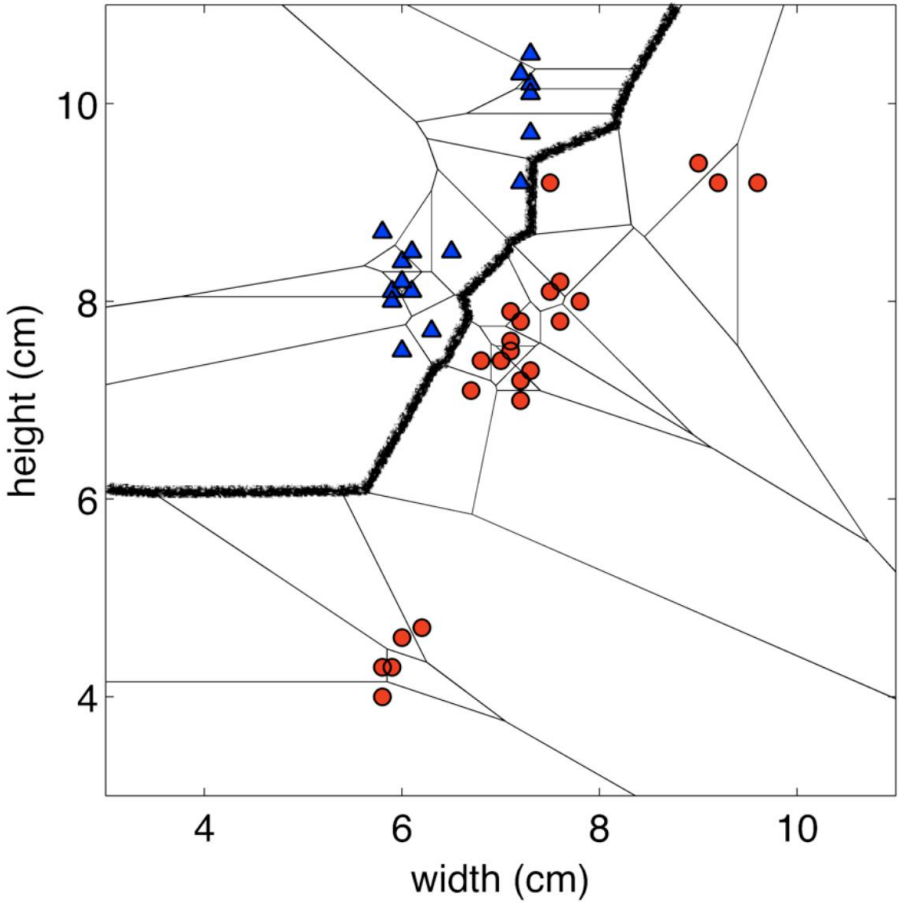
$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

1-Nearest Neighbour

- Using k = 1 classify the test samples “+”
- The labels for the training set are coloured blue and yellow
- The examples are 2-dimensional
- Apply Euclidian distance



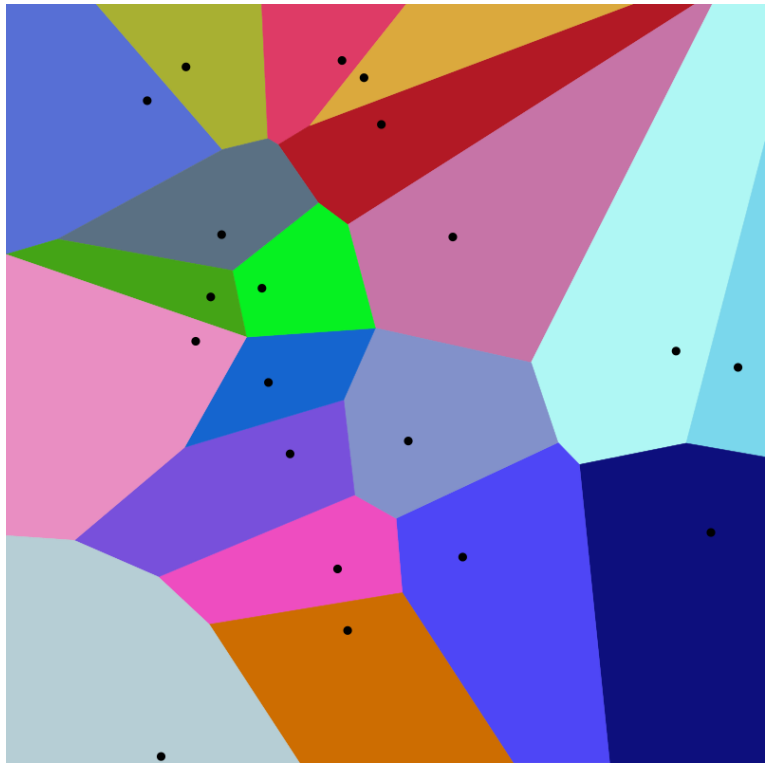
Decision Boundary



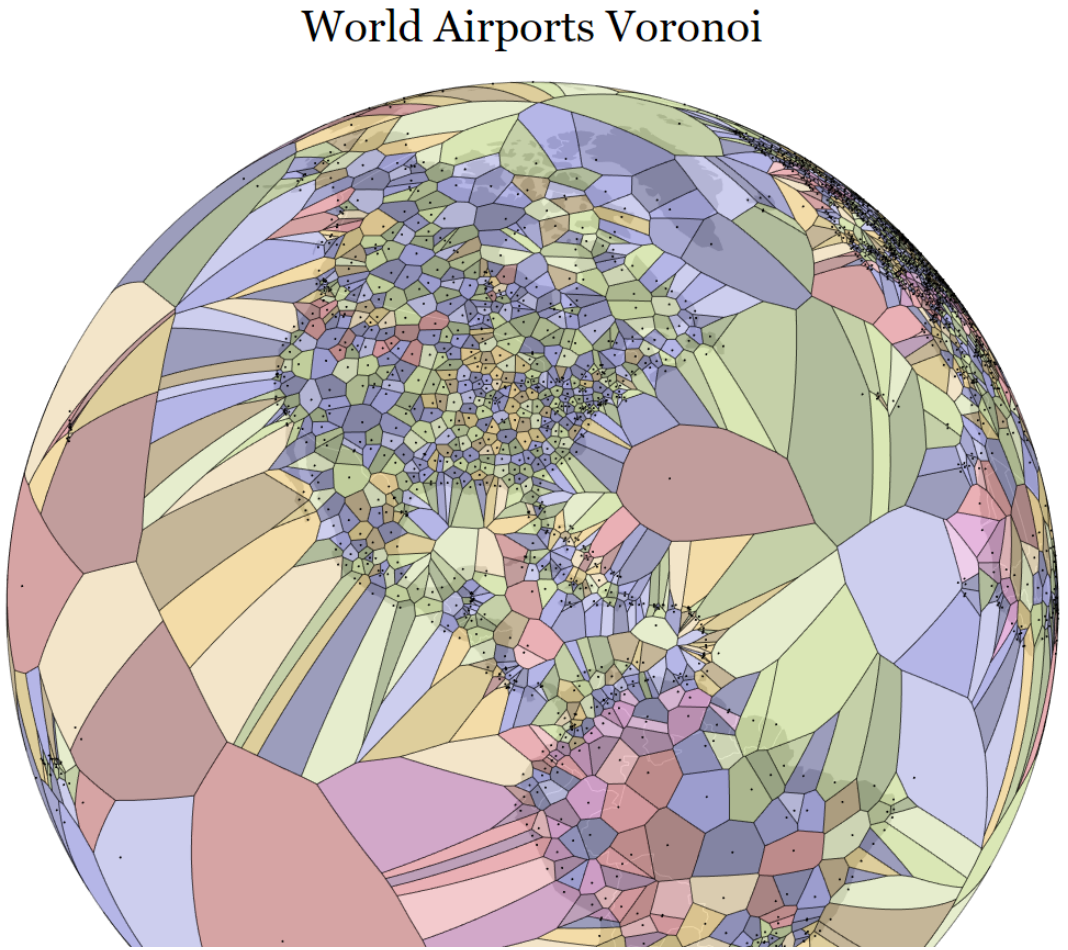
- Can generate arbitrary test points on the plane and apply kNN
- The boundary between regions of input space assigned to different categories.

Voronoi Diagrams (k=1)

Euclidian distance

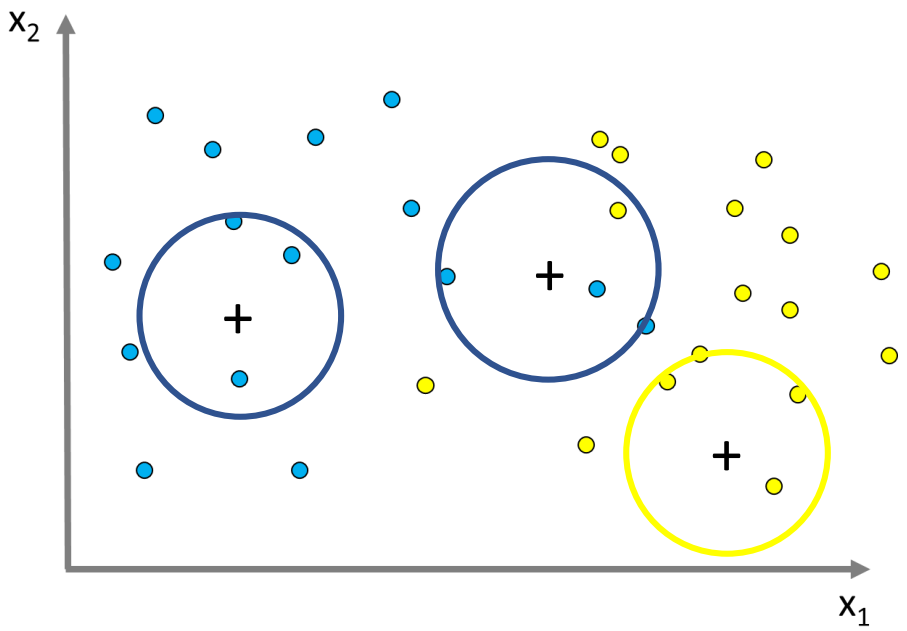


Source: Wikipedia

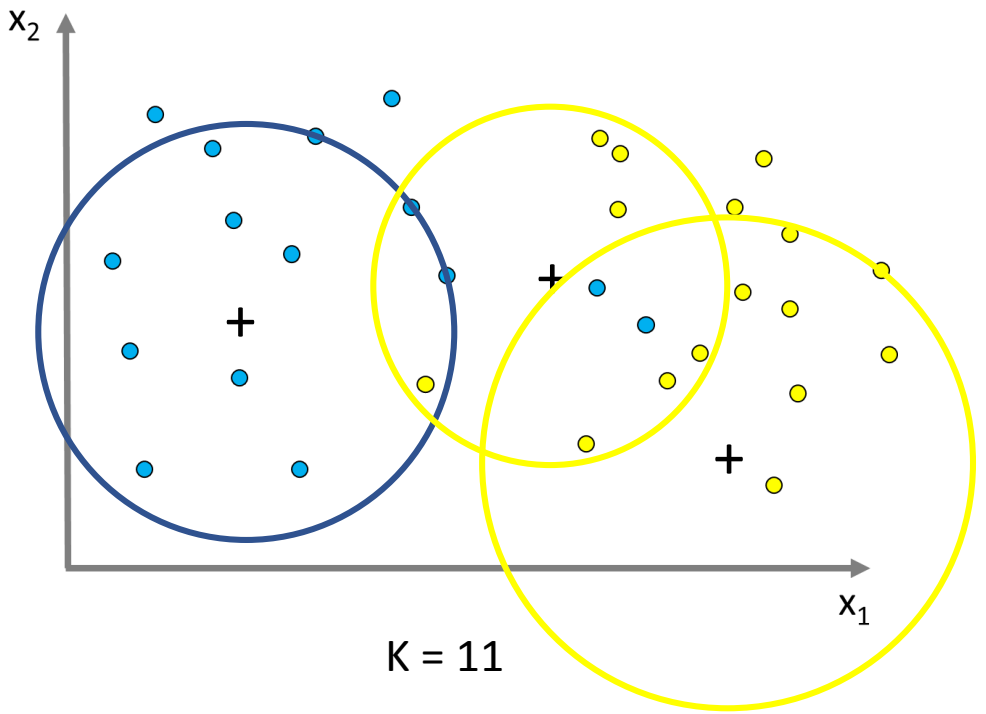


Spherical Voronoi – Source: Jason Davies

Nearest Neighbour (k > 1)



K = 3



K = 11

- Q: Why would we not consider 2-Nearest Neighbours?

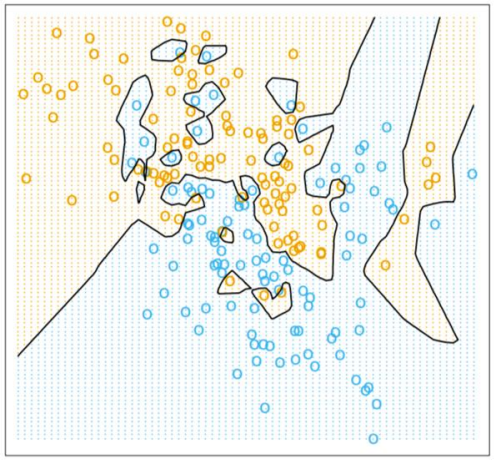
Selection of k

- Q: What happens if we let k be very small?
- Q: What happens if we let k be very large?

Tradeoffs in choosing k?

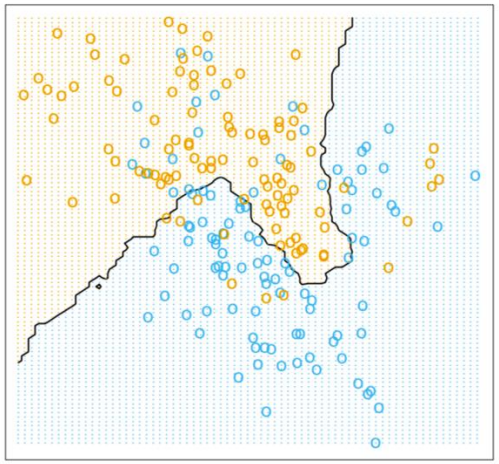
- **Small k**
  - Good at capturing fine-grained patterns
  - May overfit, i.e. be sensitive to random noise

Excellent for training data, not that good for new data, too complex



- **Large k**
  - Makes stable predictions by averaging over lots of examples
  - May underfit, i.e. fail to capture important regularities

Not that good for training data, not good for new data, too simple



What is the best k?

k is a hyperparameter

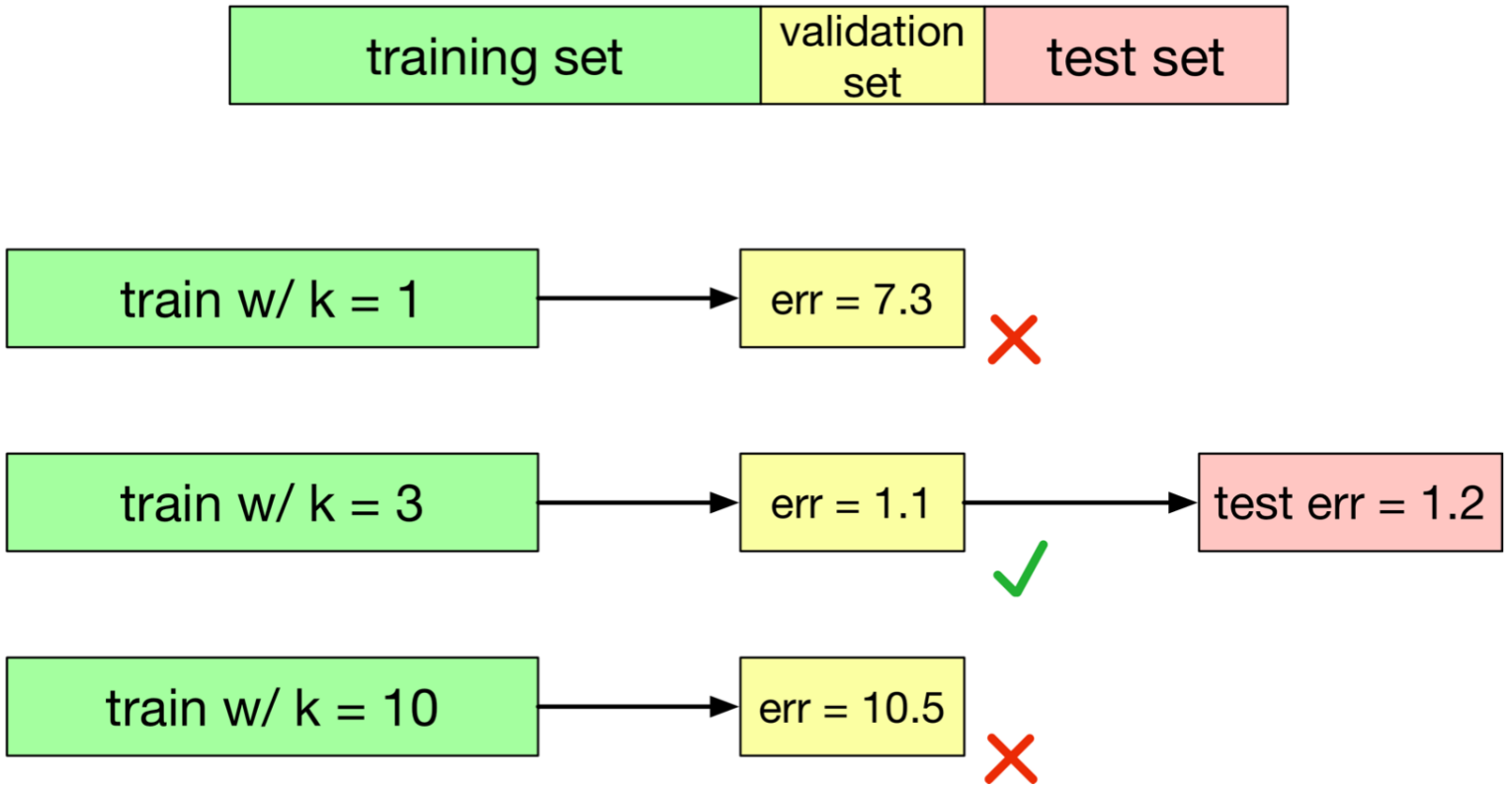
- Try different values, and see which works best on the test set?

What is the best k?

- Select the k based on the best performance on the validation set, report the results on the test set
- Generally, the performance on the validation set will be better than on the test set

- Q: How do you think it will perform on the training set?

Training, Validation and Test Set

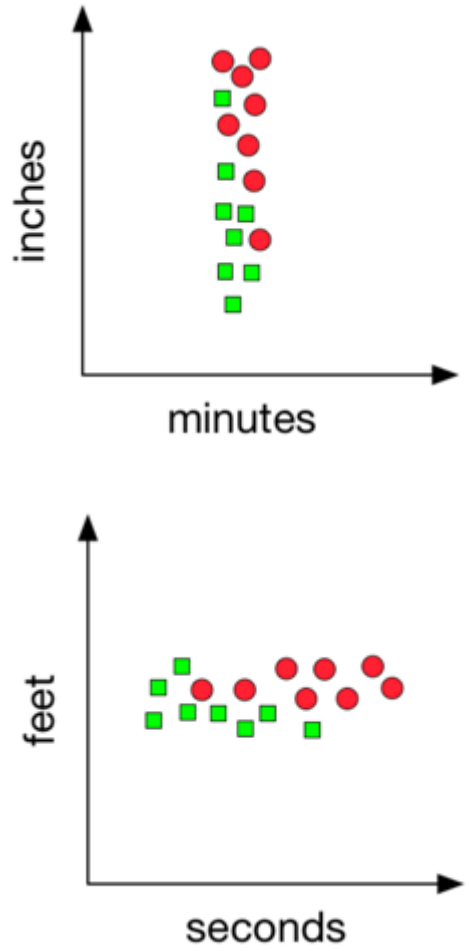


Normalization

- Nearest Neighbours can be sensitive to the ranges of different features
- Often, the units are arbitrary
- **Simple fix:** normalize each dimension to be zero mean and unit variance (i.e., compute the mean μ_j and standard deviation σ_j, and take,

$$\tilde{x}_j = \frac{x_j - \mu_j}{\sigma_j}$$

- **Caution:** depending on the problem, the scale might be important!



KNN Code Example (Google Colab)

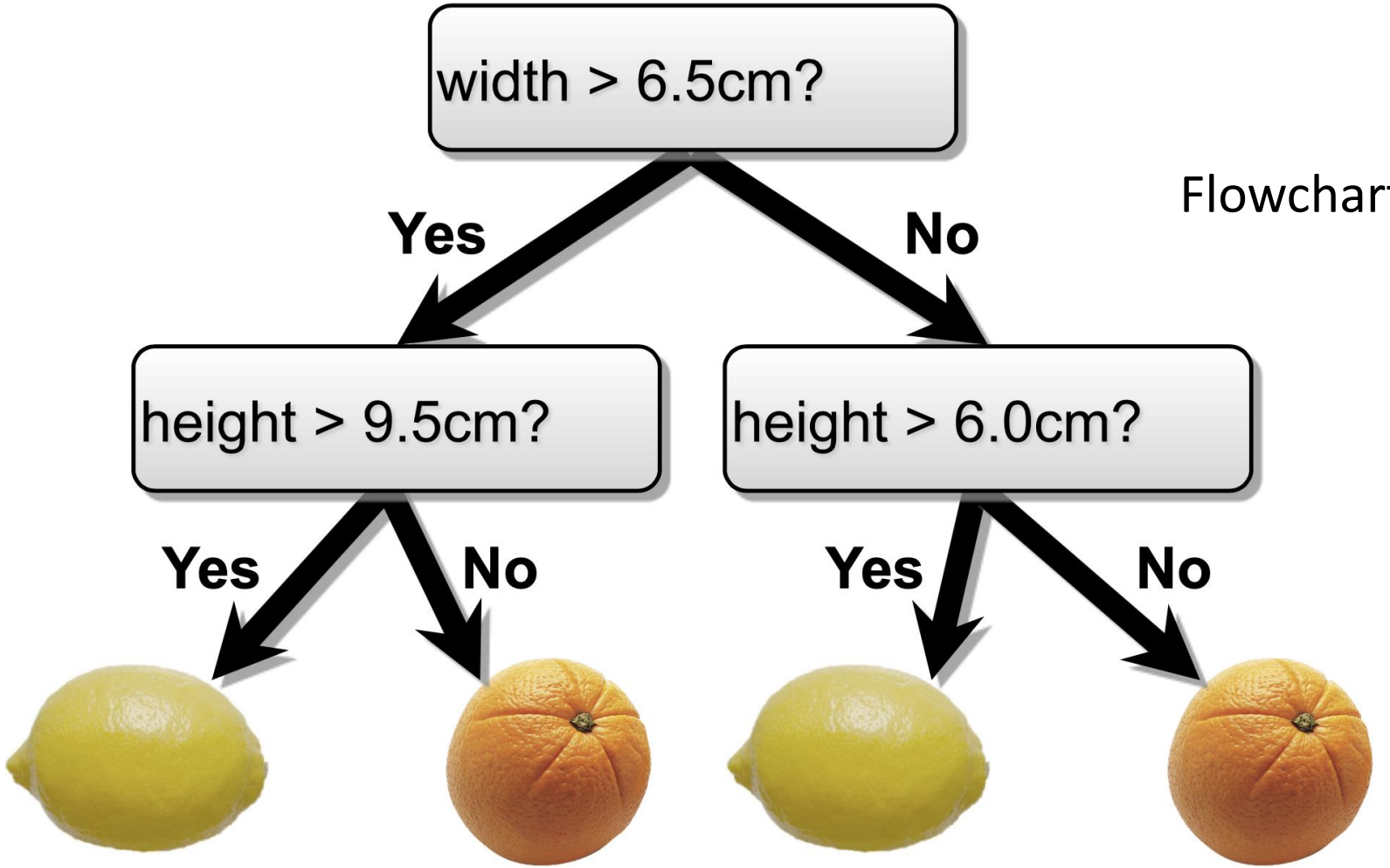
Decision Trees (Supervised Learning)

Decision Trees

- A rule-based supervised learning algorithm
- Powerful algorithm capable of fitting complex datasets.
- Can be applied to classification (discrete) and regression (continuous) tasks.
- Highly interpretable!

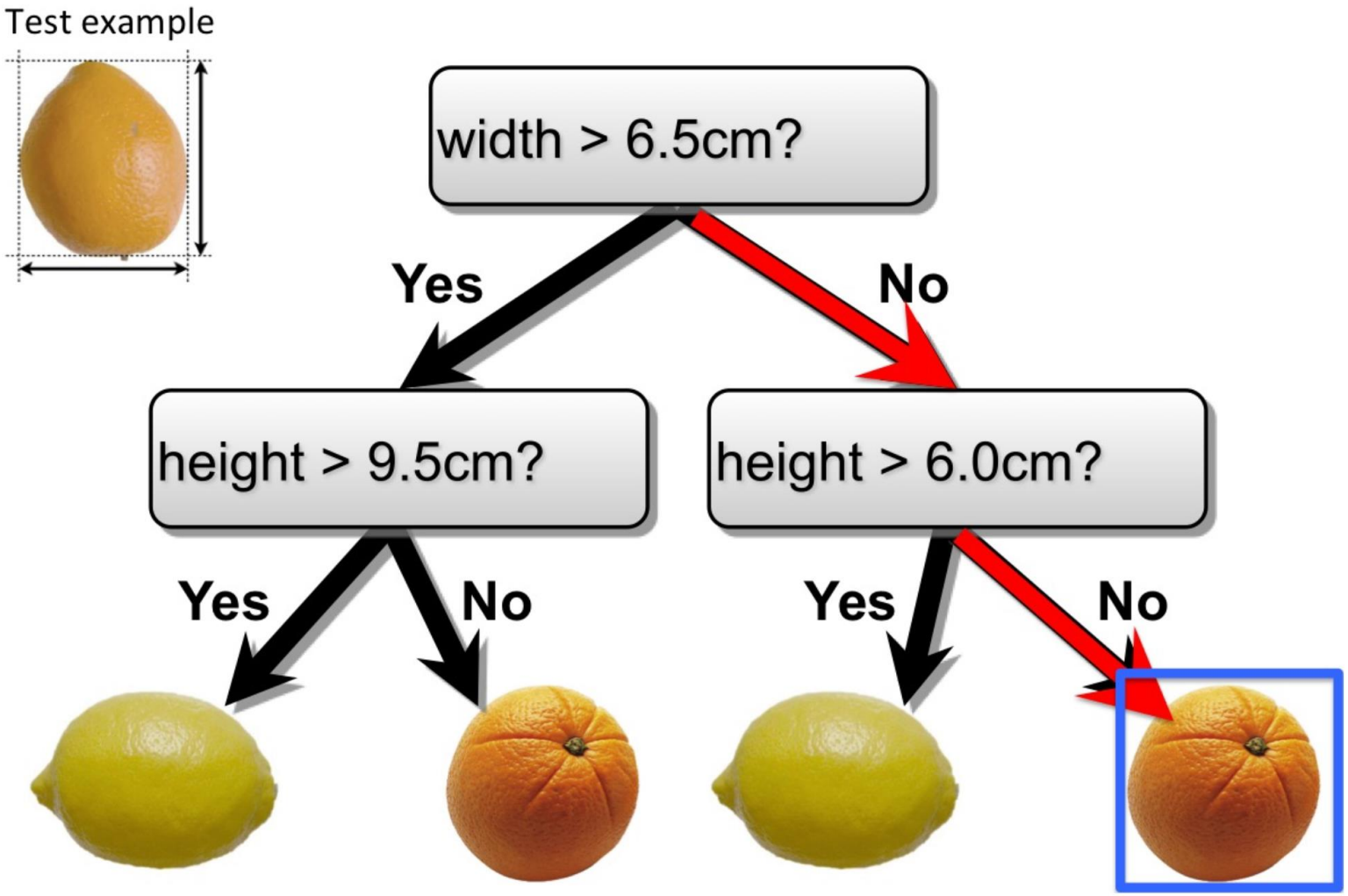
- A fundamental component of Random Forests which are one of the most used Machine Learning algorithms today

Lemon Vs. Orange!



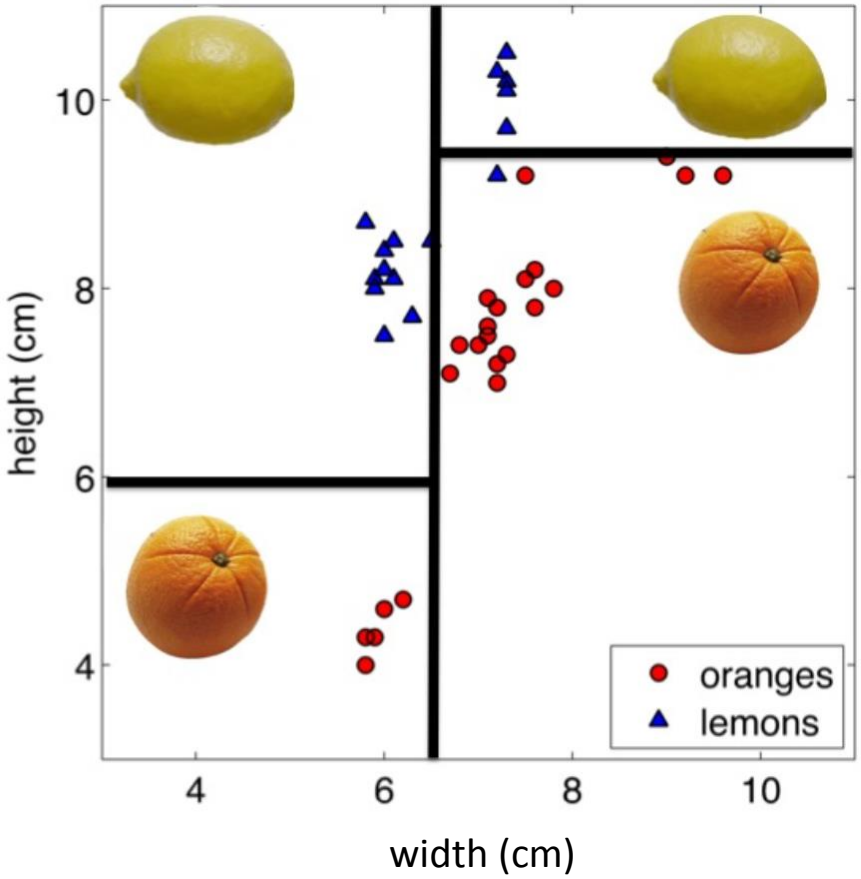
Flowchart-like structure!

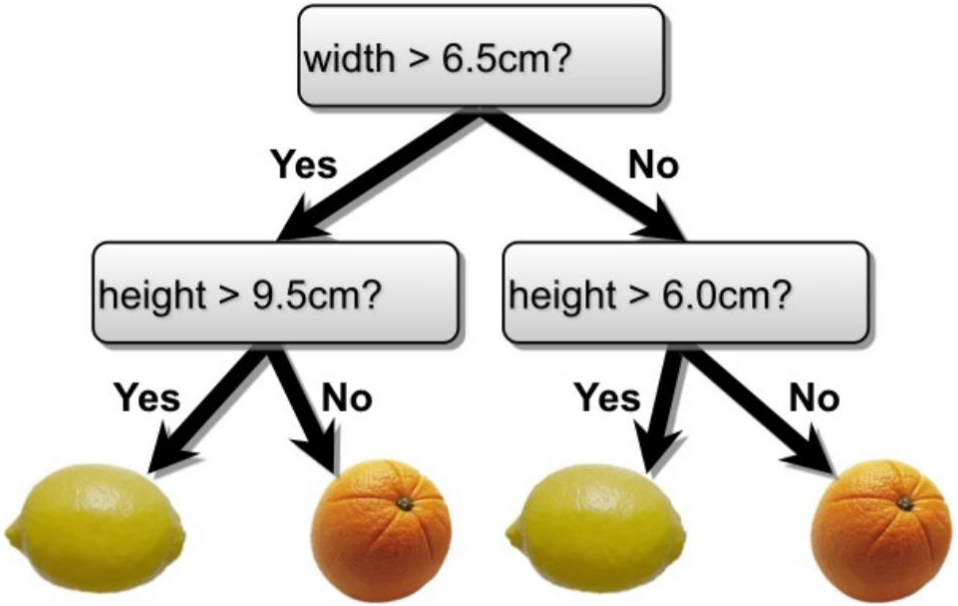
Test example



Constructing a Decision Tree

- 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*							
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*							
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*							
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*							
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*							

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).

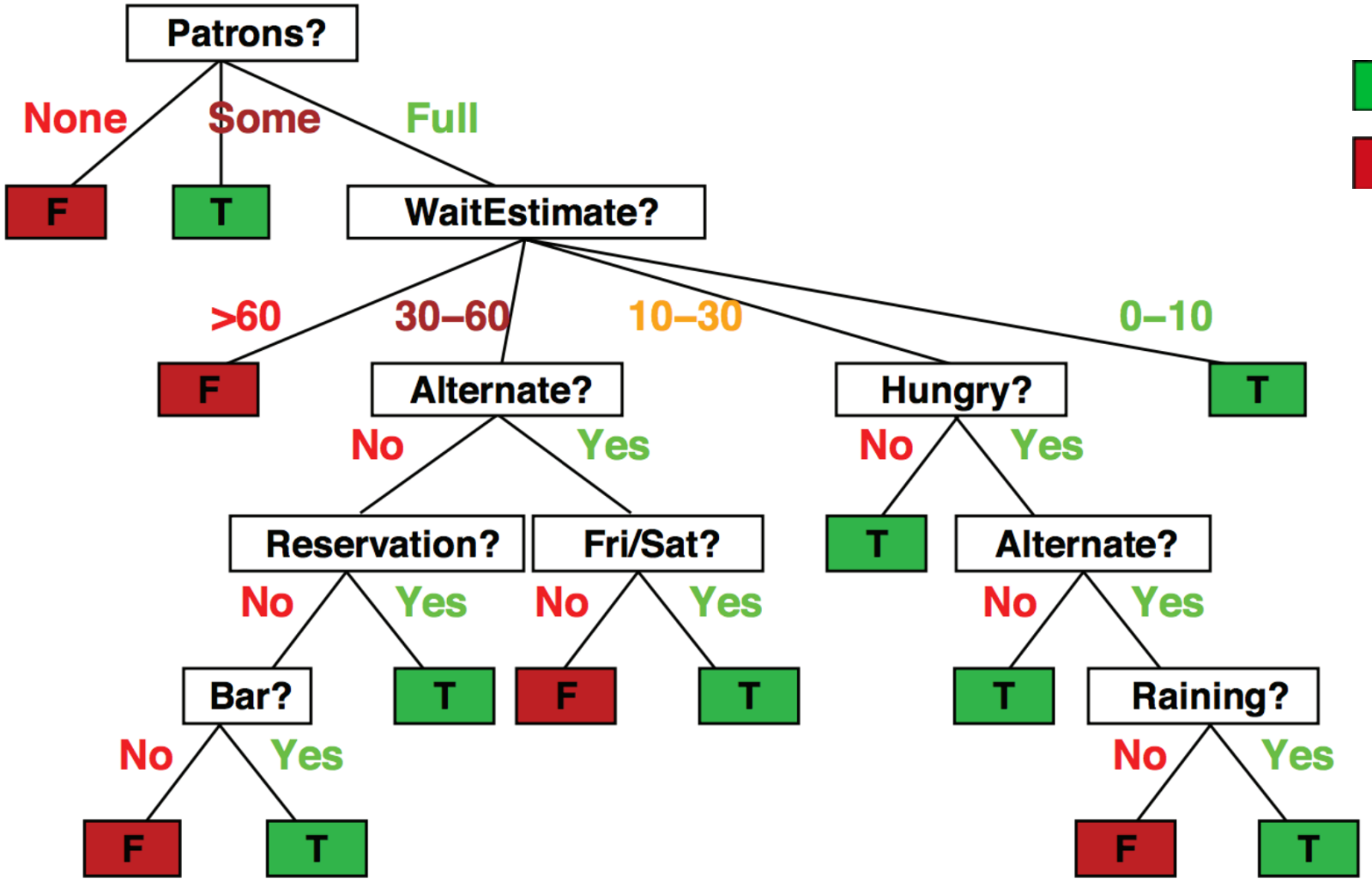
What if the attributes are discrete?

Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*						
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*						
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*						

Attributes: Features (inputs)!
Discrete or Continuous

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).

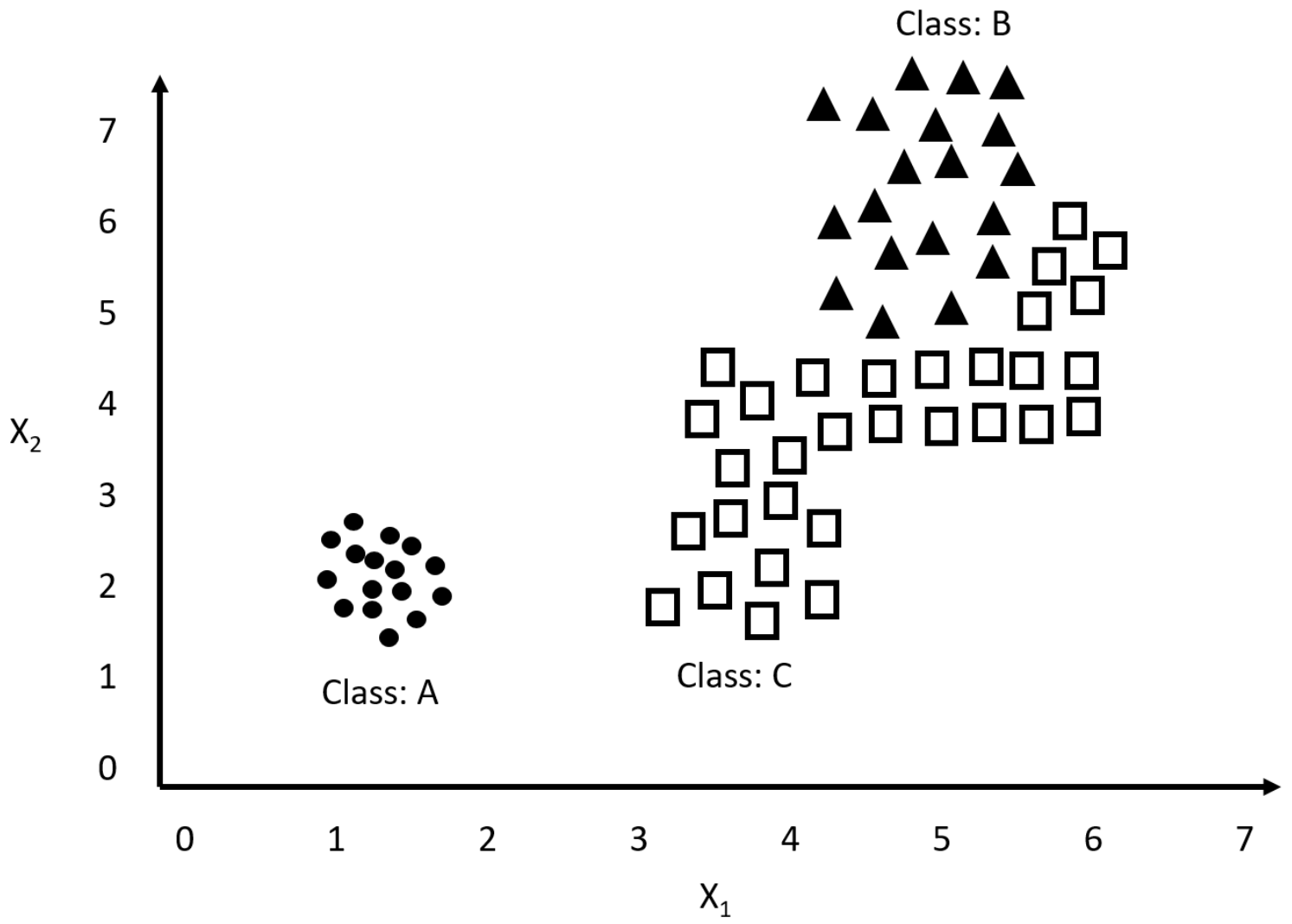
Output is Discrete



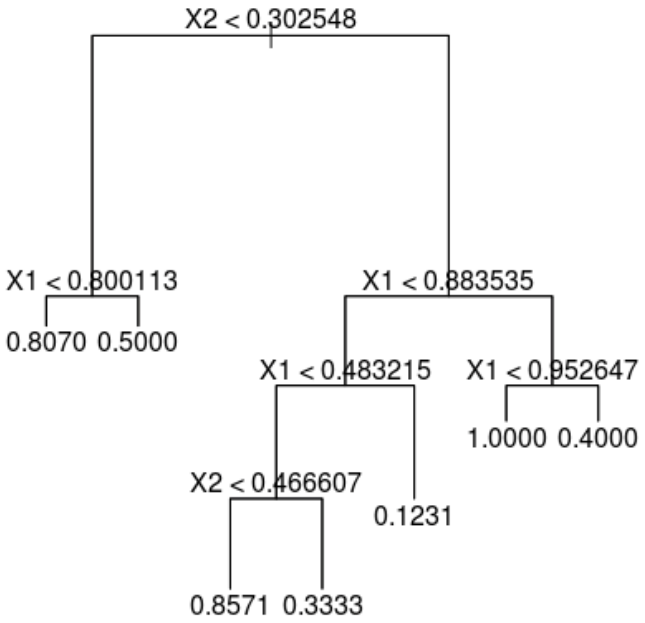
T — Wait for table

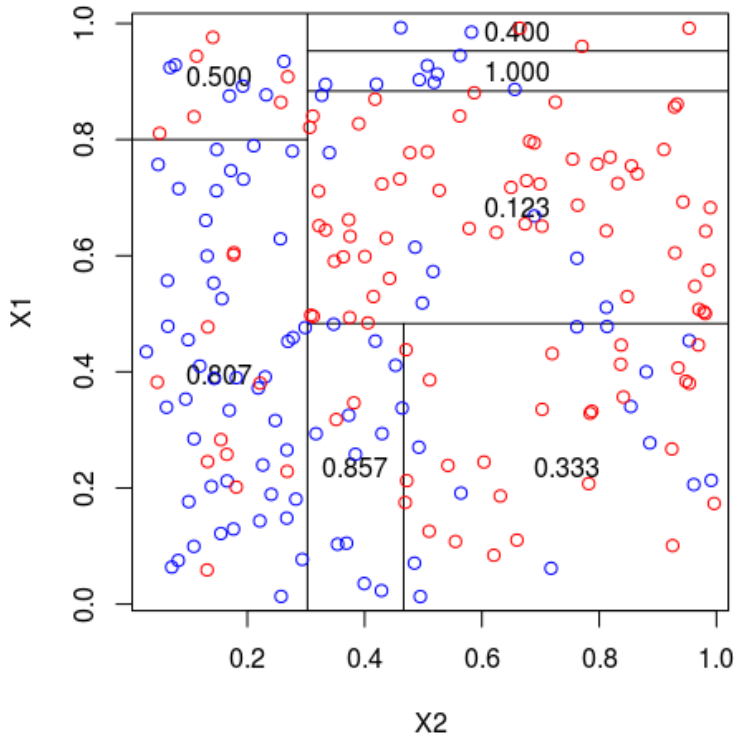
F — Go somewhere else

Example Problem



Output is Continuous (Regression)





- Instead of predicting a class at each leaf node, predict a **value based on the average** of all instances at the leaf node.

Source: GDCoder

Summary: Discrete vs Continuous Output

- **Classification Tree**:
  - discrete output
  - output node (leaf) typically set to the **most common value**

- **Regression Tree**:
  - continuous output
  - output node (leaf) value typically set to the **mean value** in data

Generalization

- Decision trees **can fit any function arbitrarily closely**
- Could potentially create a leaf for each example in the training dataset
- Not likely to generalize to test data!

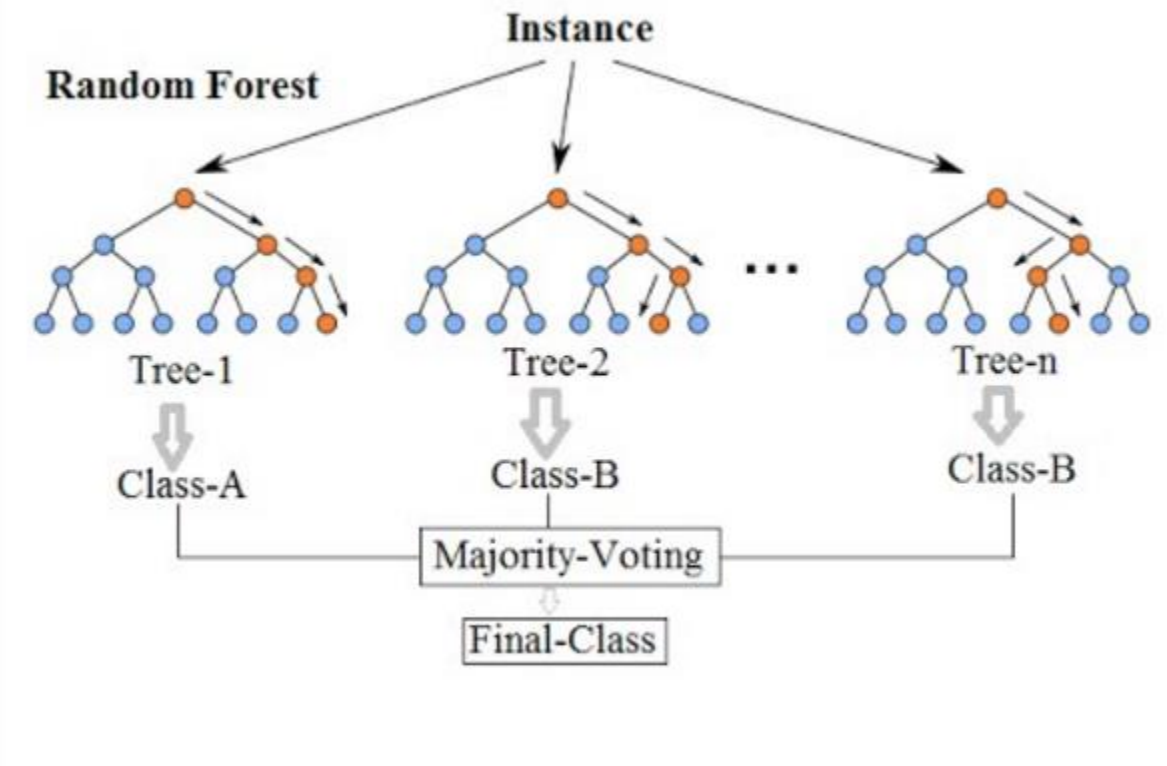
- Need some way to prune the tree!

Managing Overfitting

- Add parameters to reduce potential for overfitting
- Parameters include:
  - depth of tree
  - minimum number of samples

Random Forests

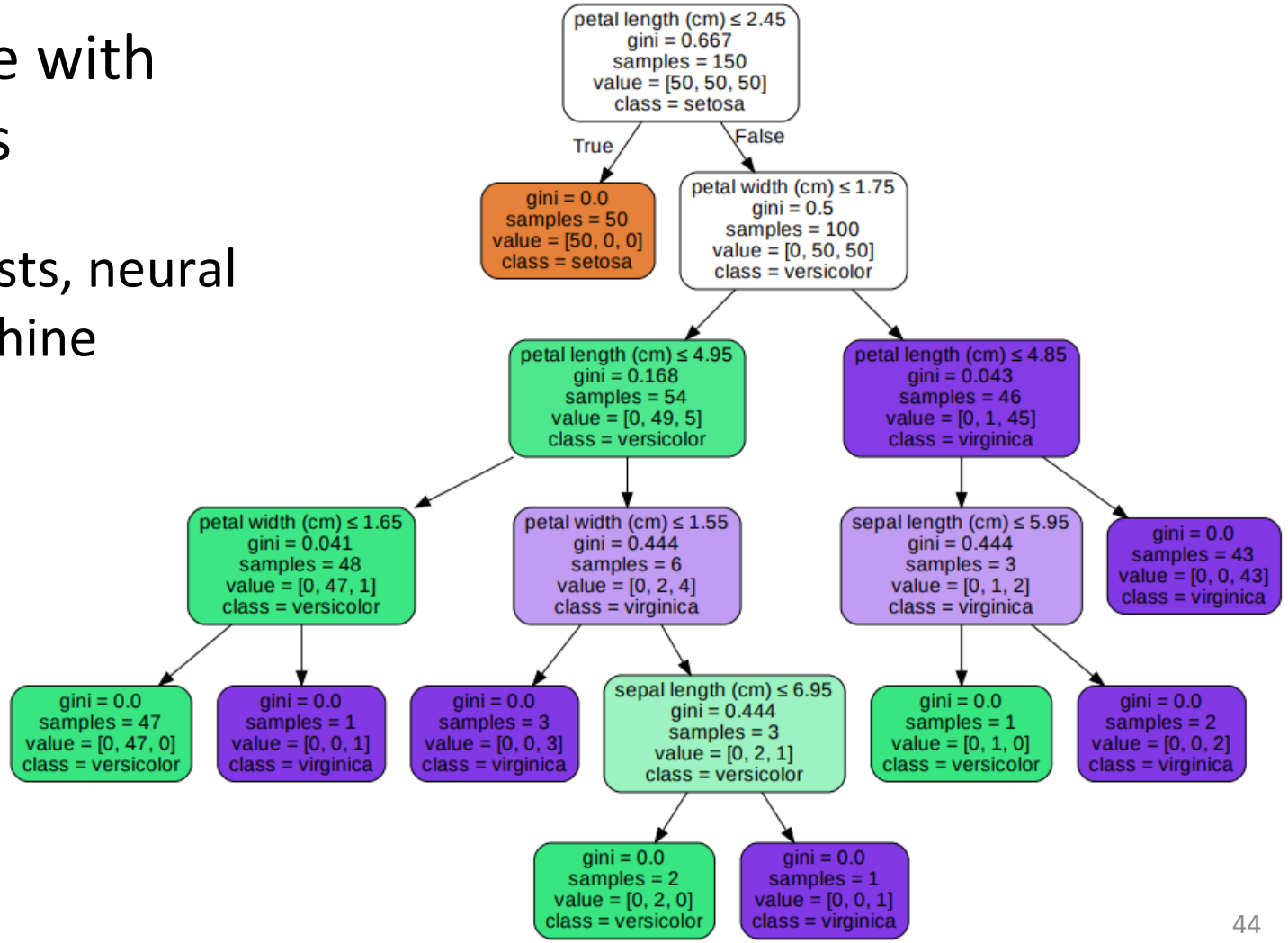
- One of the most popular variants of decision trees
- Addresses overfitting by training multiple trees on subsampling of features among other things
- Majority vote of all the trees is used to make the final output



Source: Venkata Jagannath

Model Interpretation

- Decision Trees are intuitive with **easy to interpret** decisions
- …not the case for random forests, neural networks and many other machine learning algorithms



Comparison to k-NN

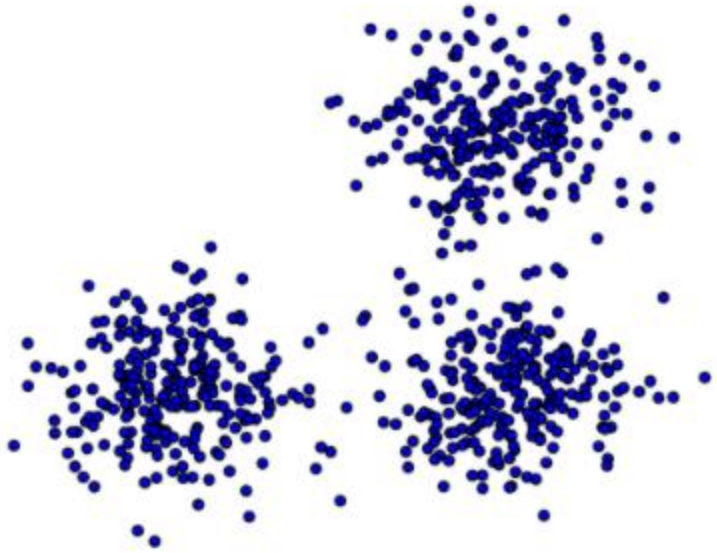
- There are many advantages of Decision Trees over k-Nearest Neighbours:
  - Good with discrete attributes
  - Robust to scale of inputs (does not require normalization)
  - Easily handle missing values
  - Good at handling lots of attributes, especially when only a few are important
  - Fast test time
  - More interpretable
  - Decision trees not good at handling rotations in data

Decision Trees Code Example (Google Colab)

Clustering Strategies (Unsupervised Learning)

Clustering

- Clustering algorithms group samples/instances based on similarities in features
  - **Input:** set of samples/instances described by features
  - **Output:** assigned cluster (group) for each sample/instance
- Clustering are **unsupervised techniques** and do not have access to the sample/instance labels



Clustering Strategies

- **k-Means Clustering**
  - Assigns each point to the nearest cluster center
- Density-Based Clustering
  - Limits the separation distance between cluster points and maintains cluster density
- Agglomerative clustering
  - Assumes each point is a cluster and iteratively merges the closest clusters

K-Means

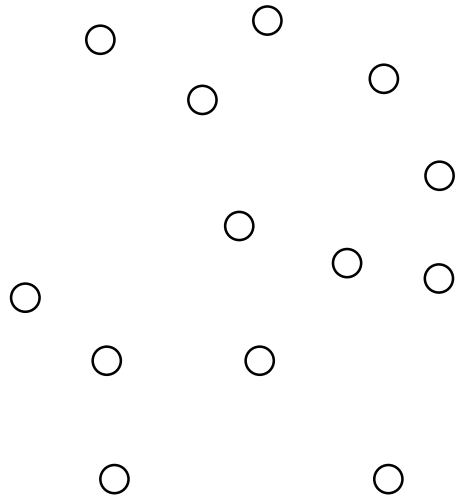
- Most well-known clustering method
- Distance-based **unsupervised** learning algorithm, NOT to be confused with k-NN.

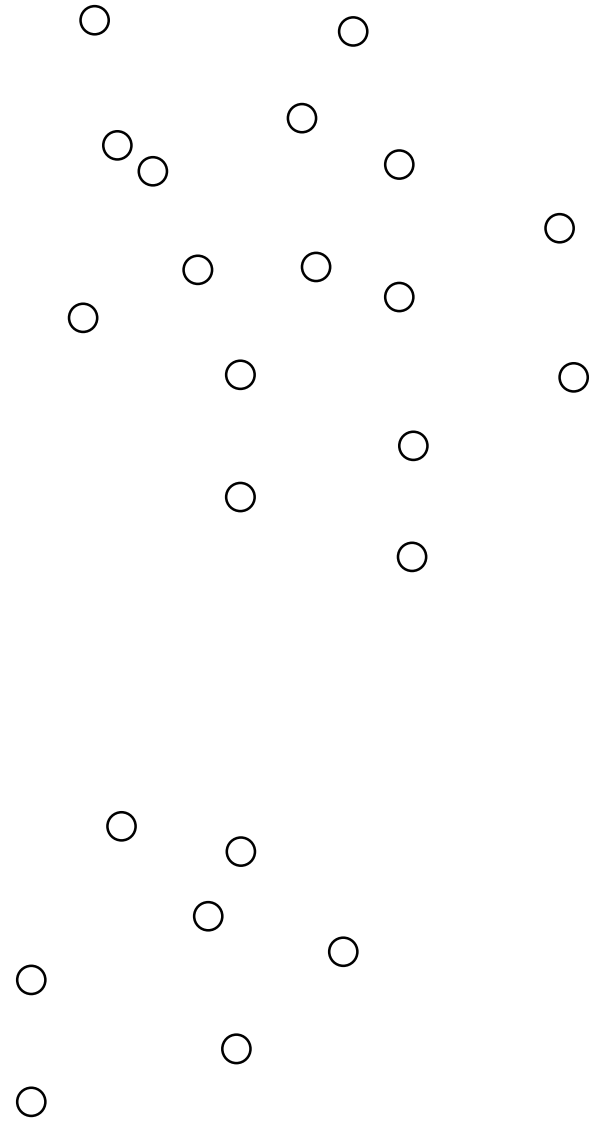
- **Algorithm:**
  1. Assign each sample/instance to its closest mean
  2. Update the means based on the assignment
  3. Repeat until convergence

- **Requires:**
  - Selection of the number of clusters ‘k’ (hyperparameter)
  - Centre of each cluster is randomly initialized at the start

K-Means Example

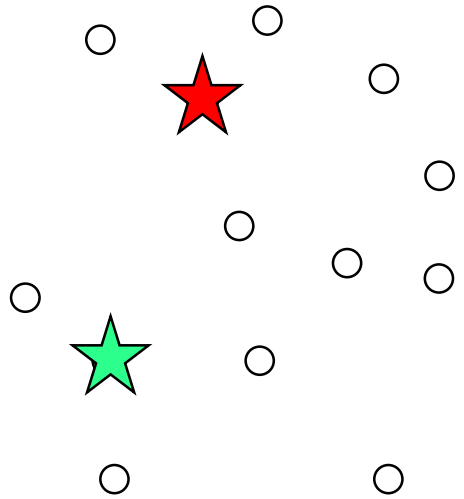
- K is number of clusters
- STEP 1: Guess center locations
- STEP 2: Map out what data point is closest to what center
- STEP 3: Center moves to the centroid of all points it “owns”
- REPEAT!

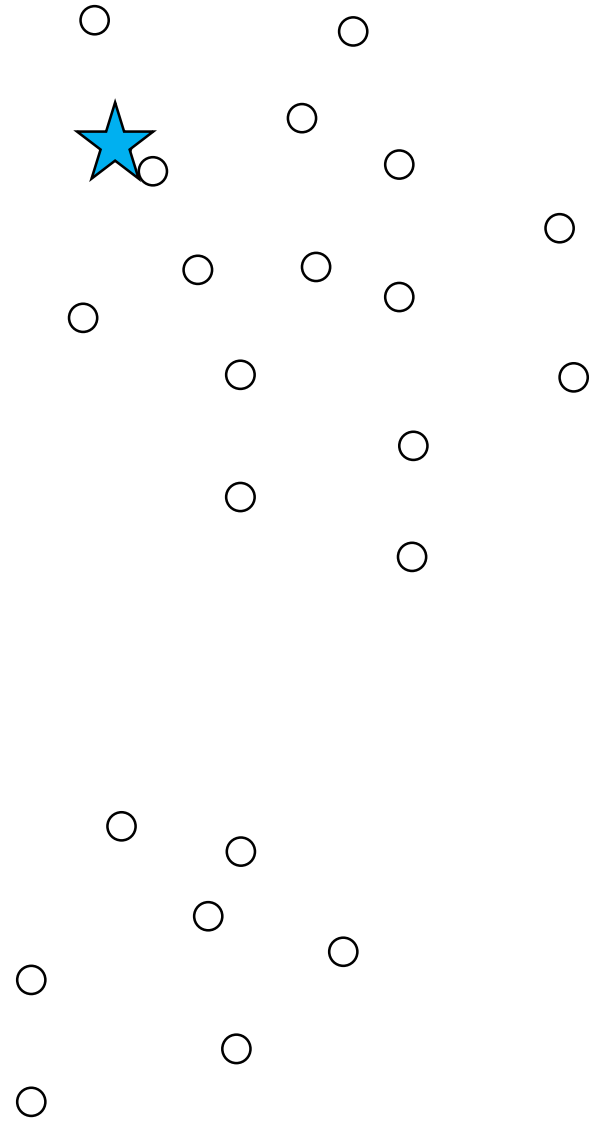




K-Means Example

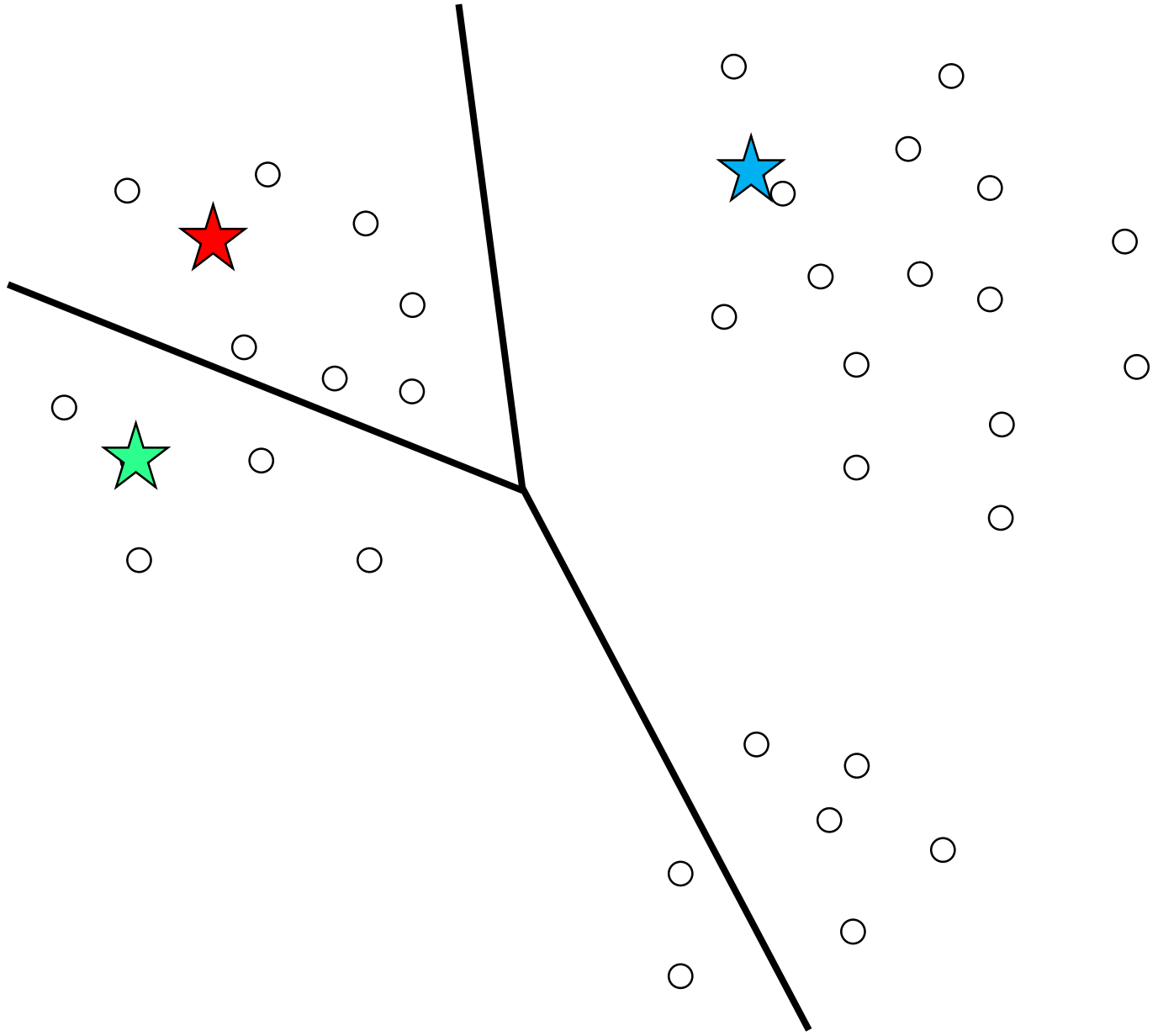
- K is number of clusters
- STEP 1: Guess center locations
- STEP 2: Map out what data point is closest to what center
- STEP 3: Center moves to the centroid of all points it “owns”
- REPEAT!





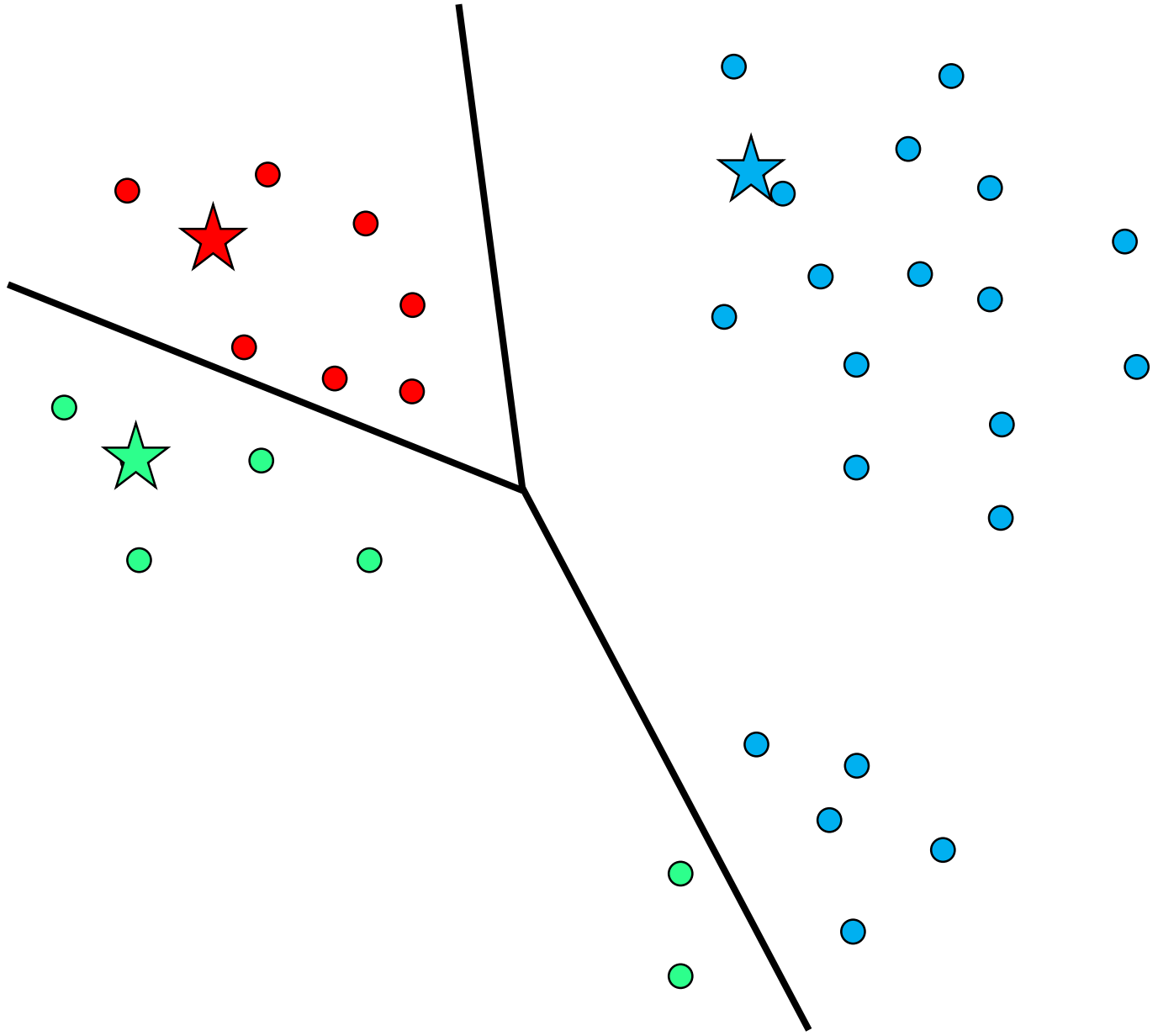
K-Means Example

- 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!



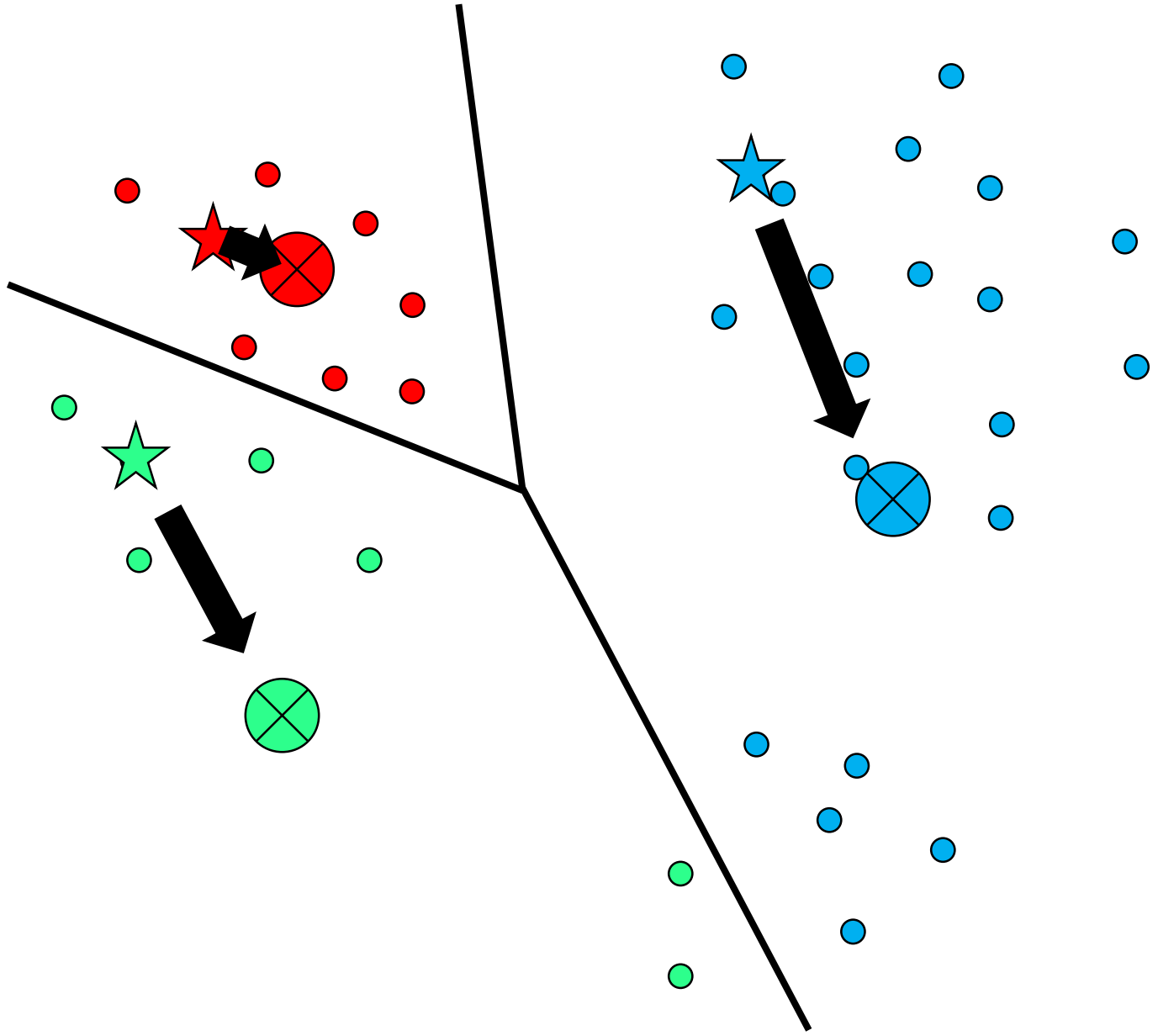
K-Means Example

- 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!



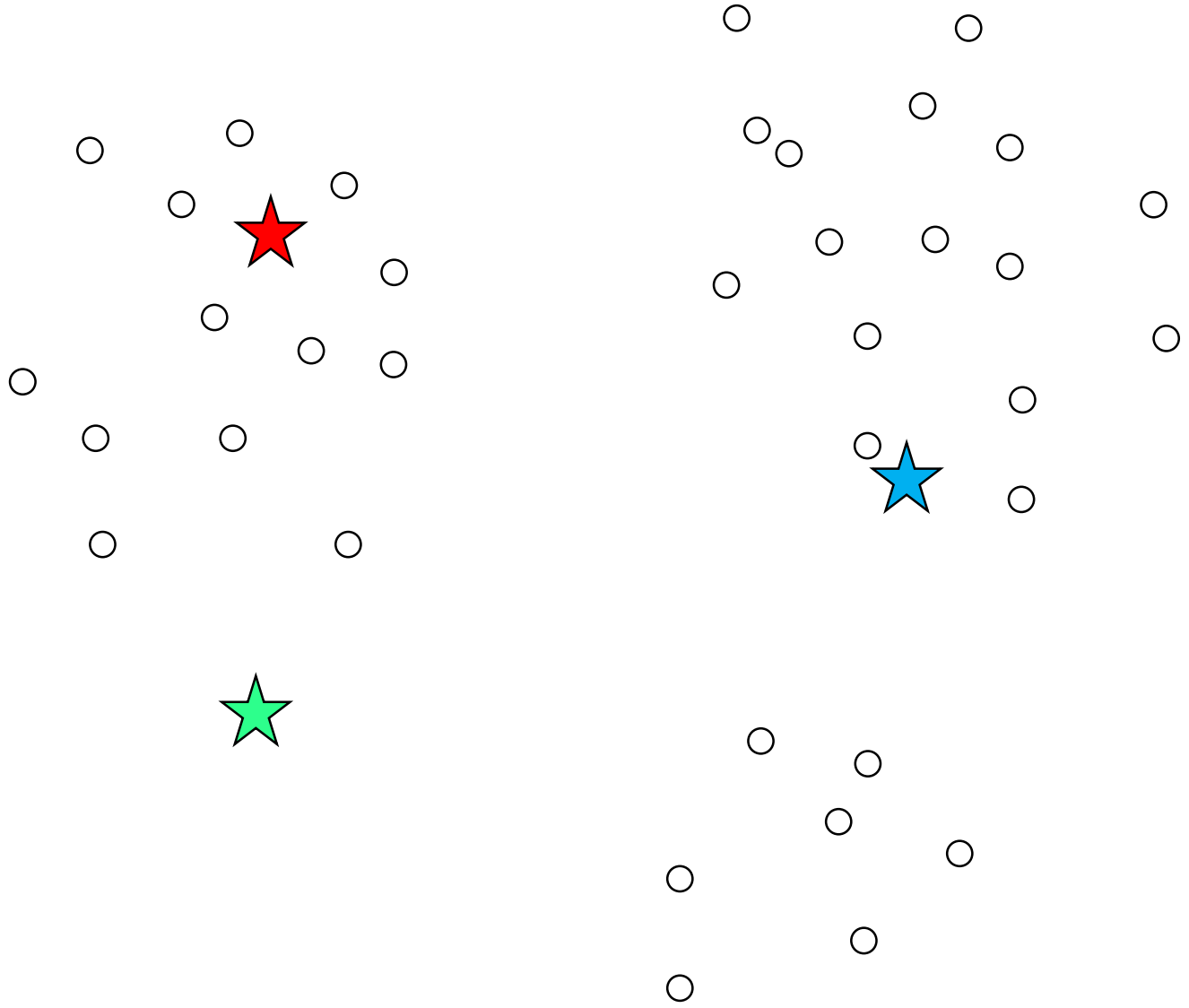
K-Means Example

- 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!



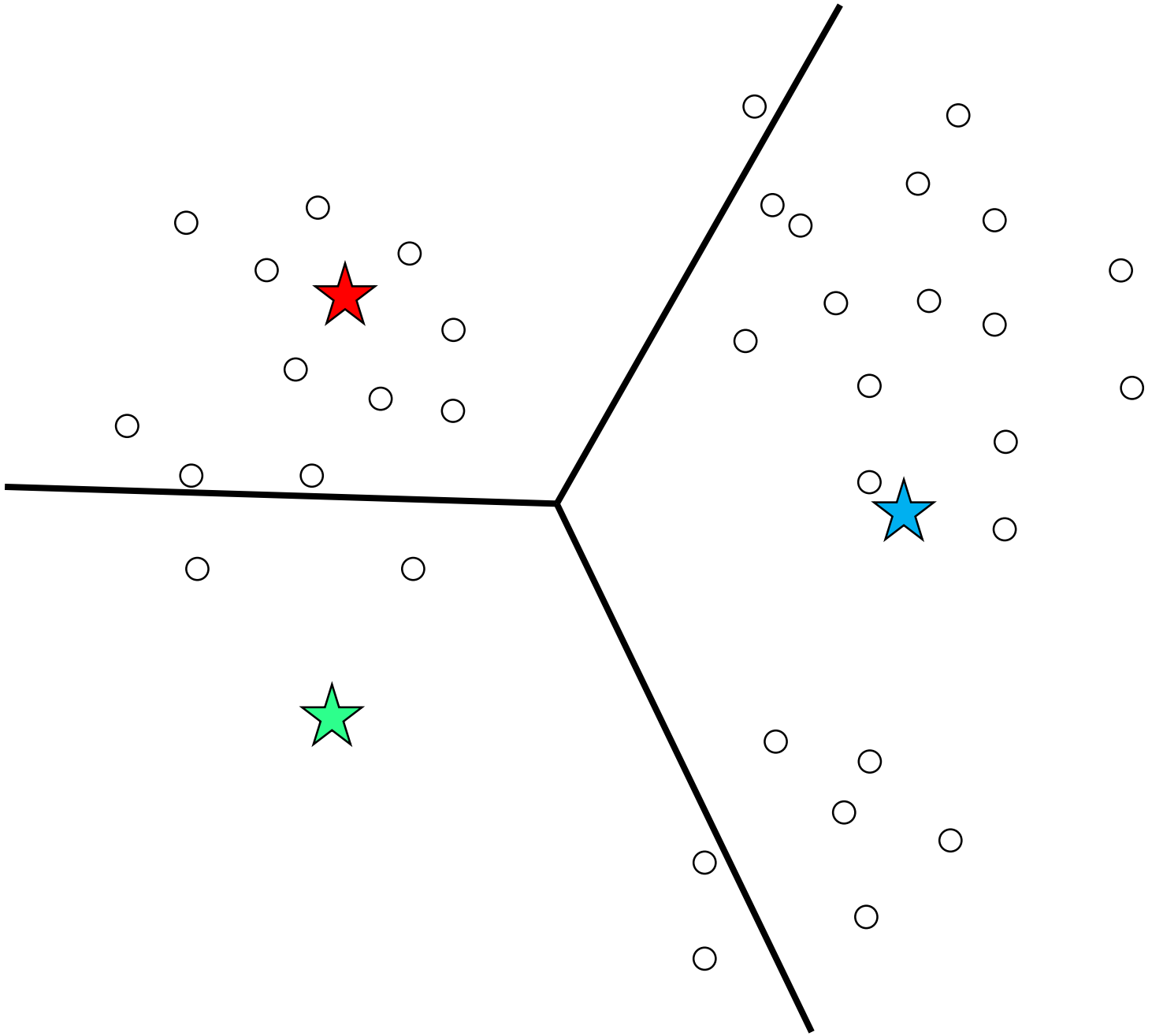
K-means Example

- 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!



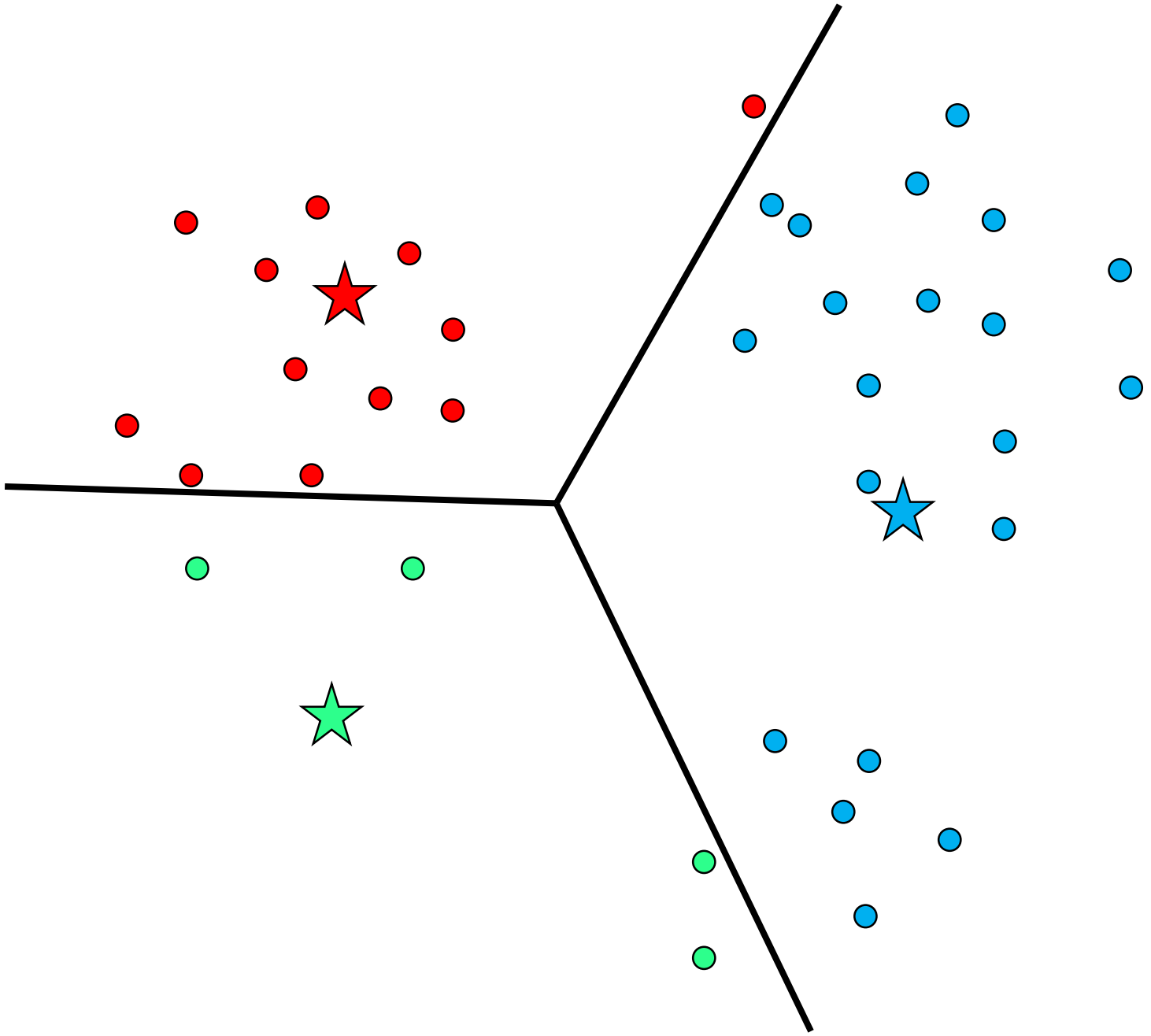
K-means Example

- 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!



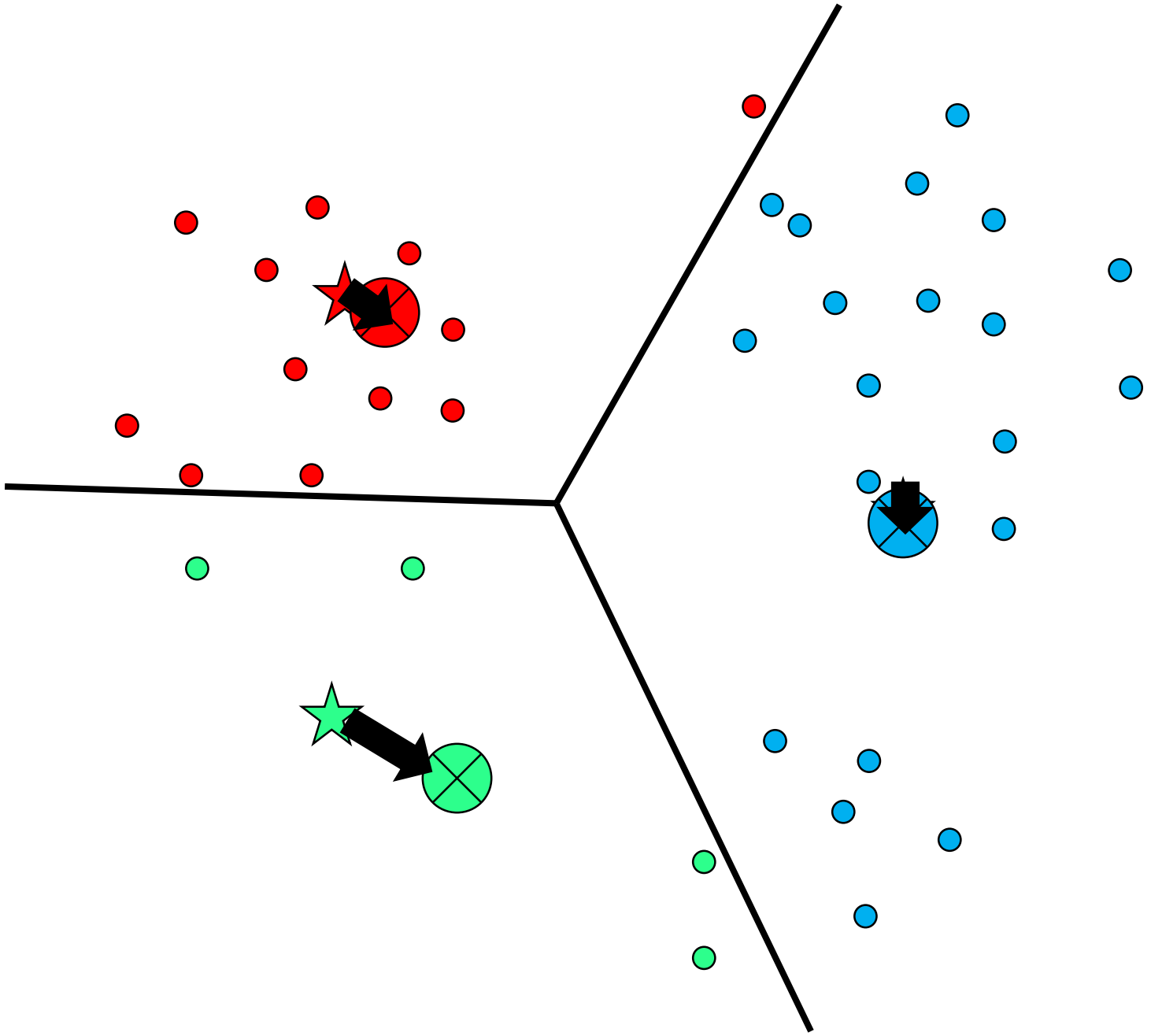
K-means Example

- 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!



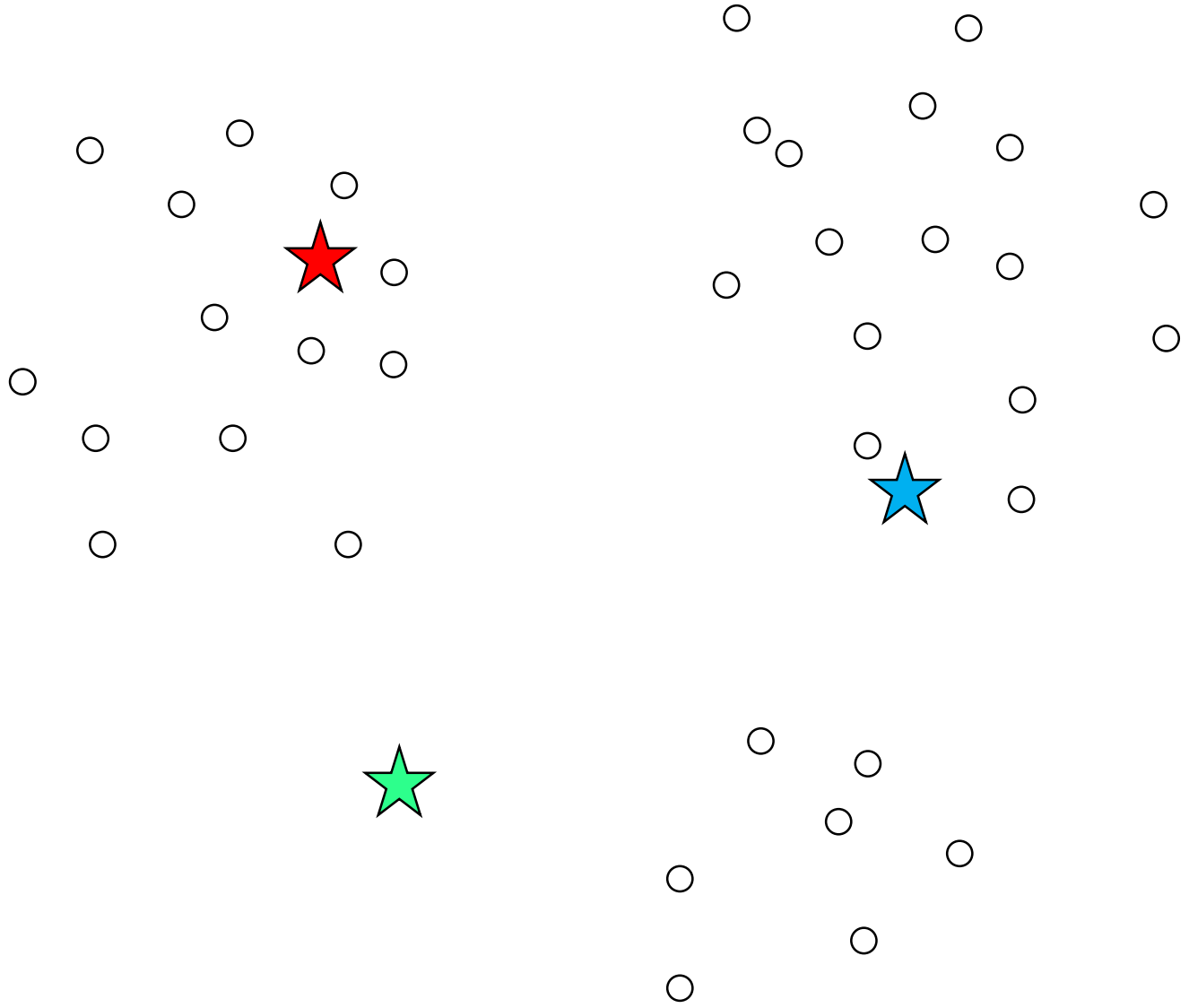
K-means Example

- 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!



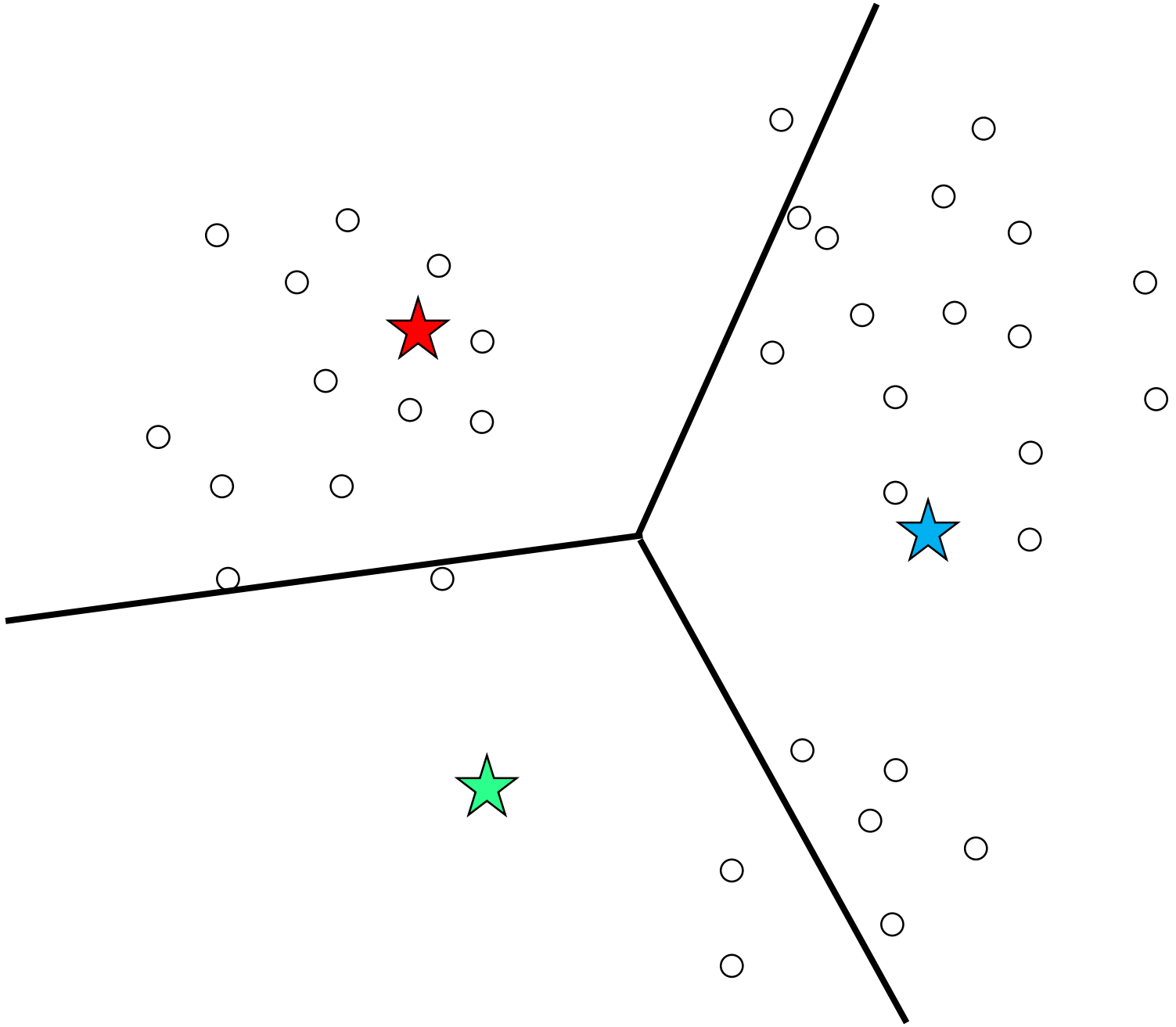
K-means Example

- 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!



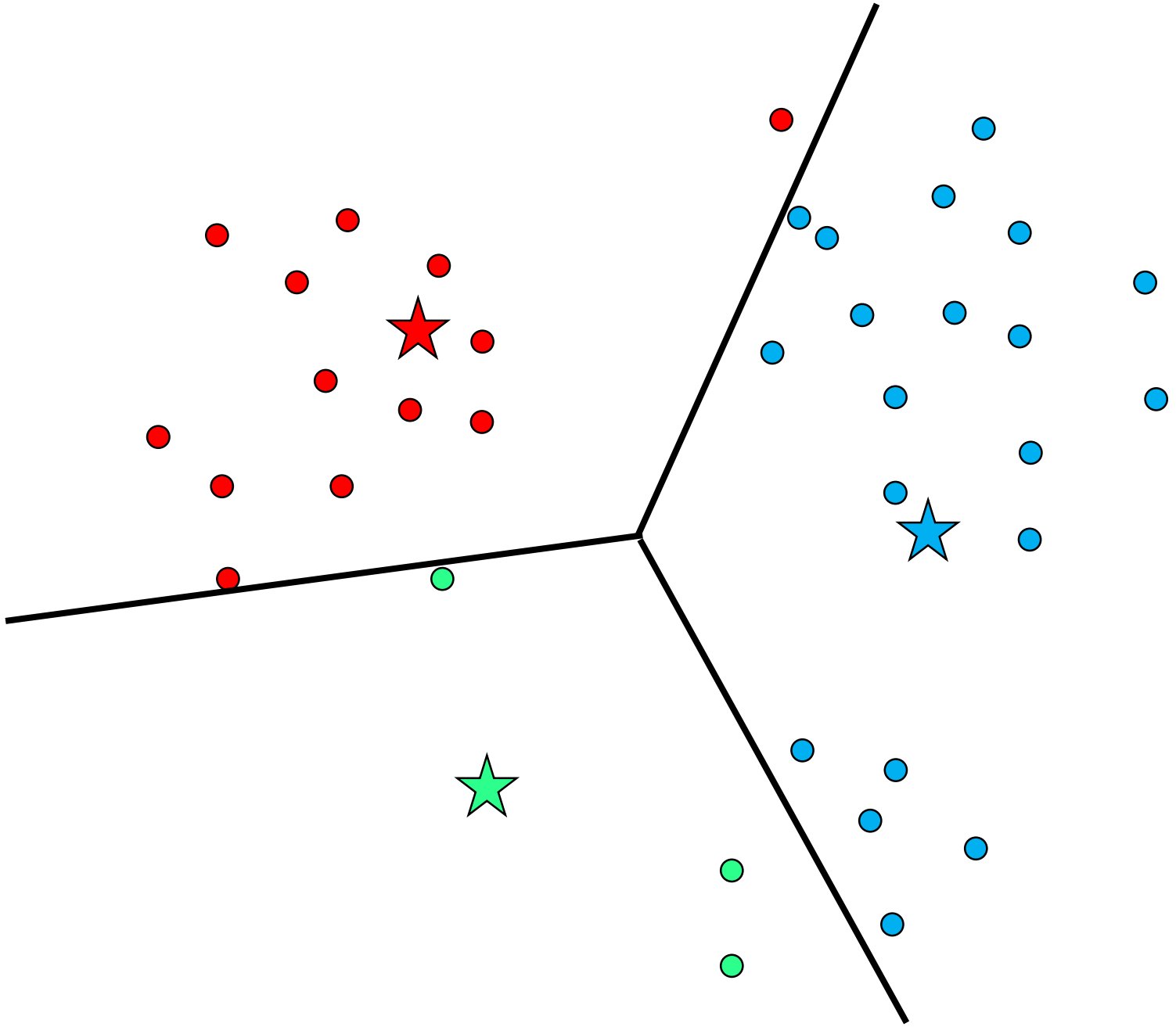
K-means Example

- 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!



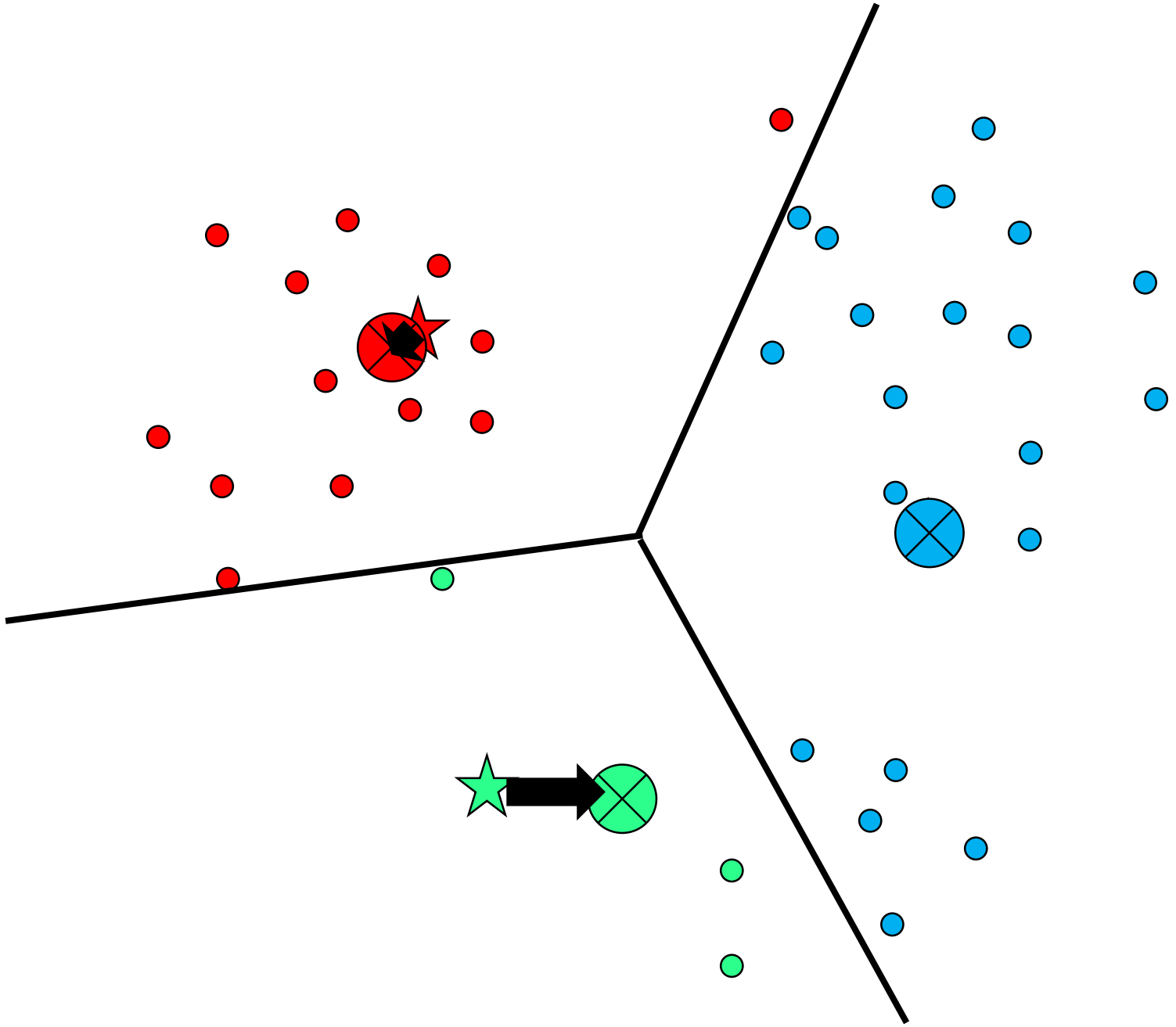
K-means Example

- 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!



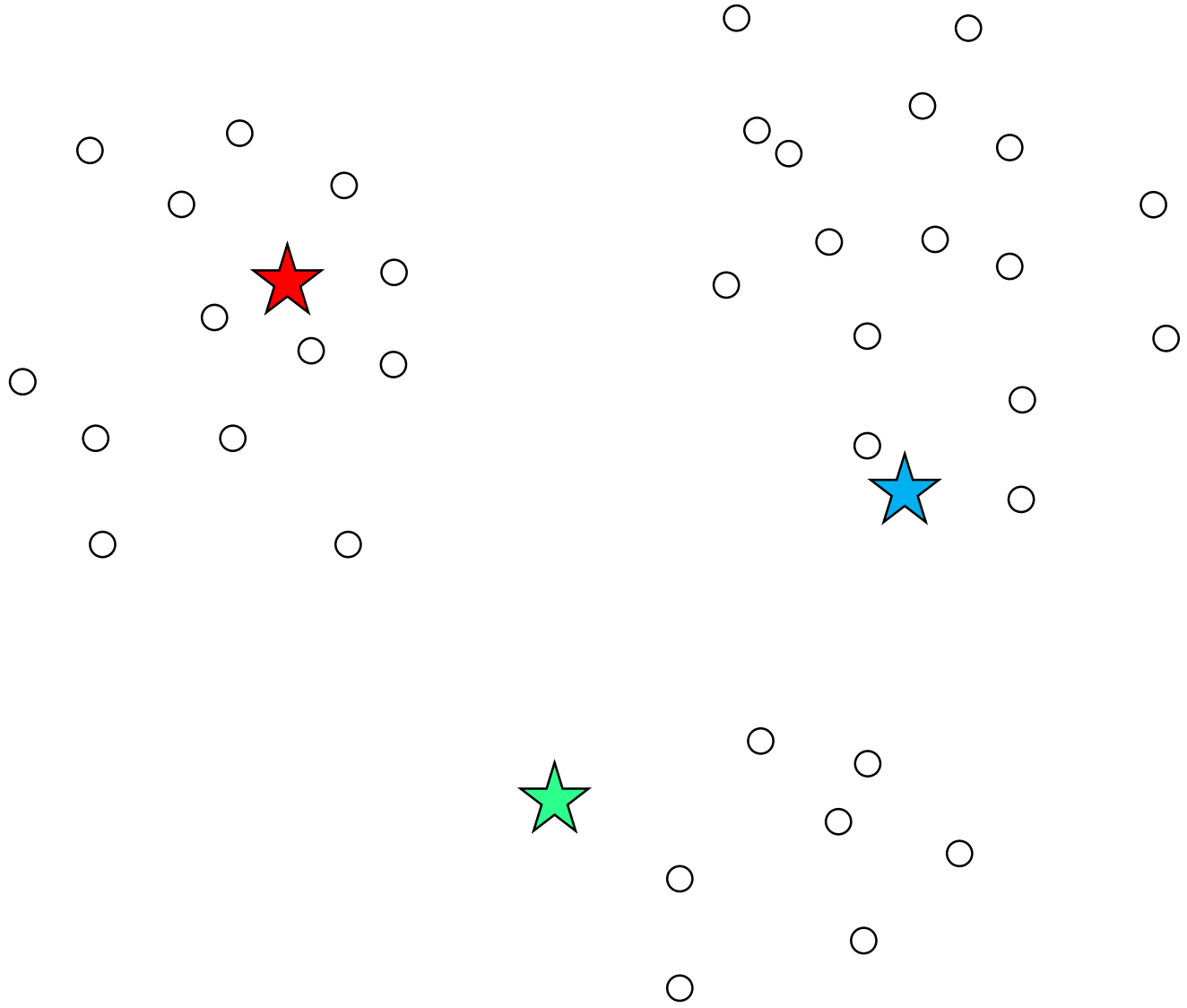
K-means Example

- 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!



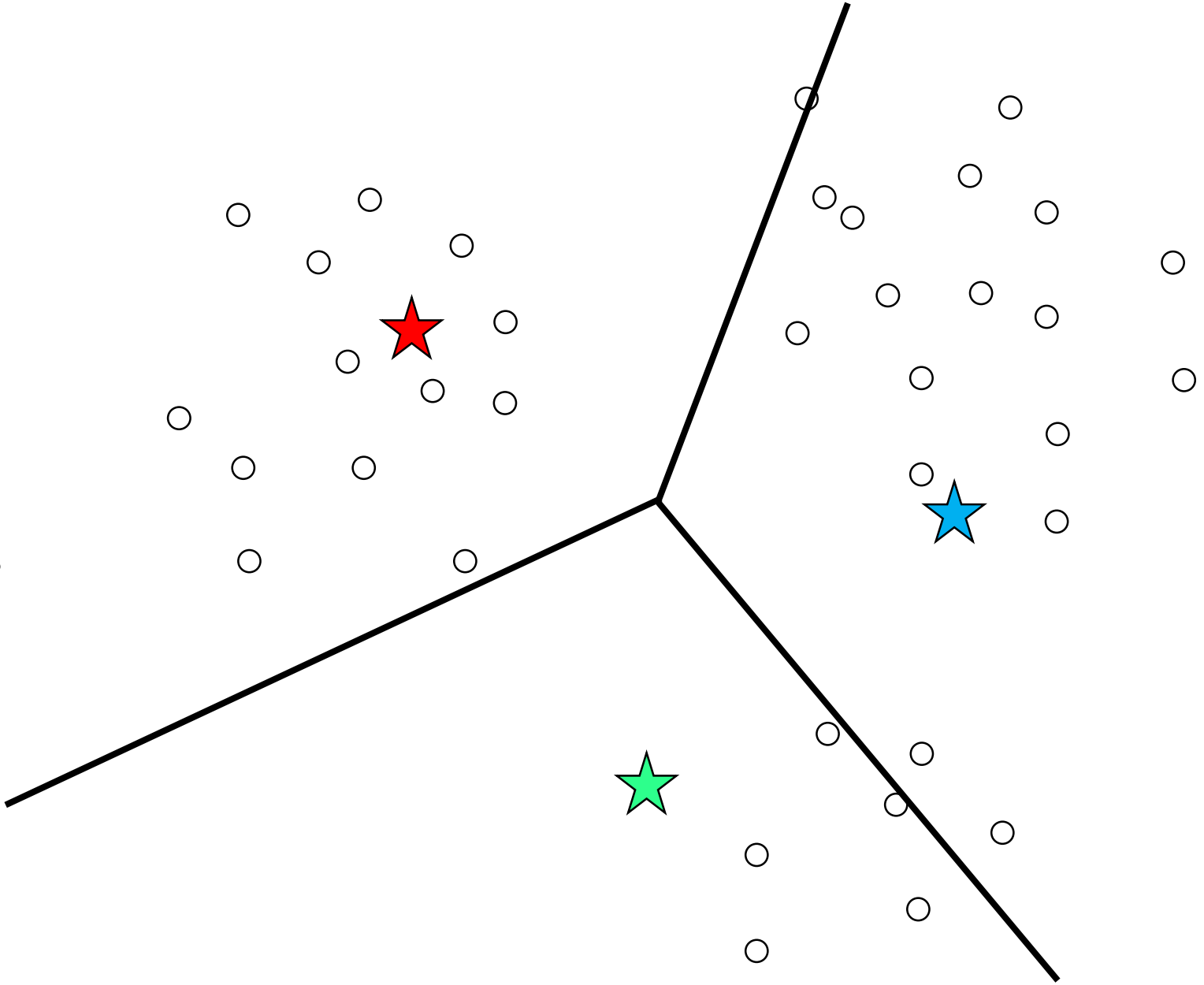
K-means Example

- 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!



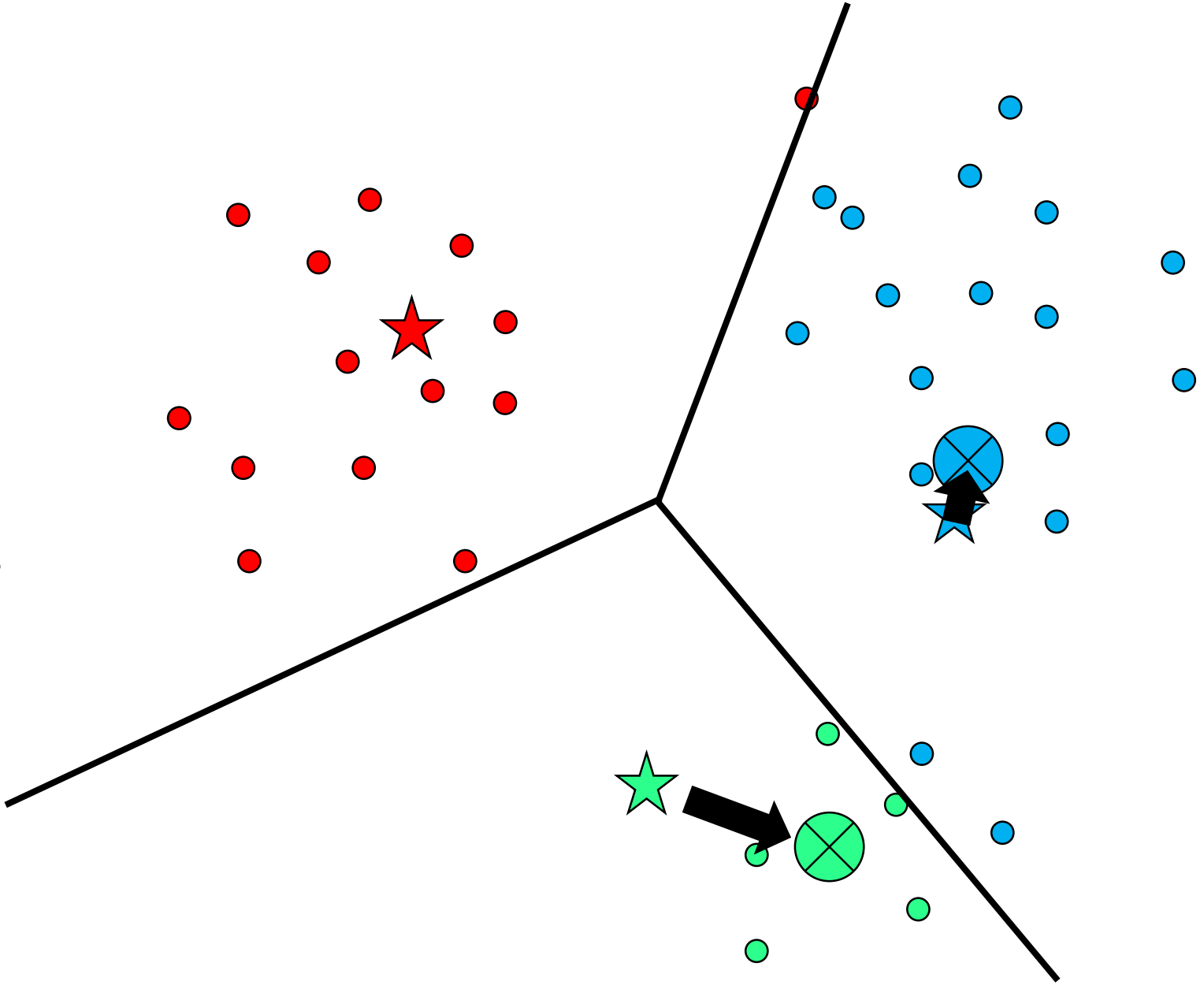
K-means Example

- 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!



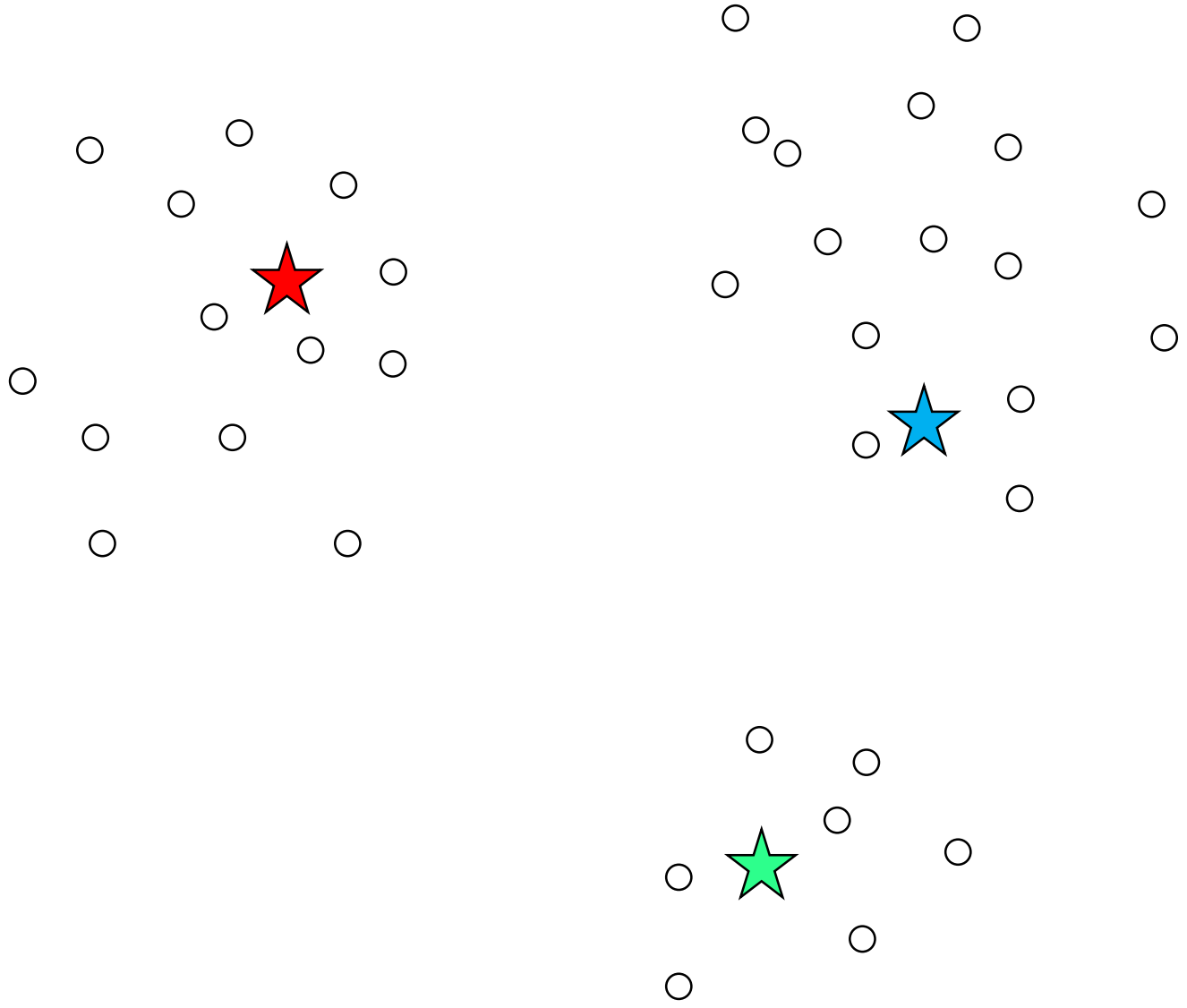
K-means Example

- 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!



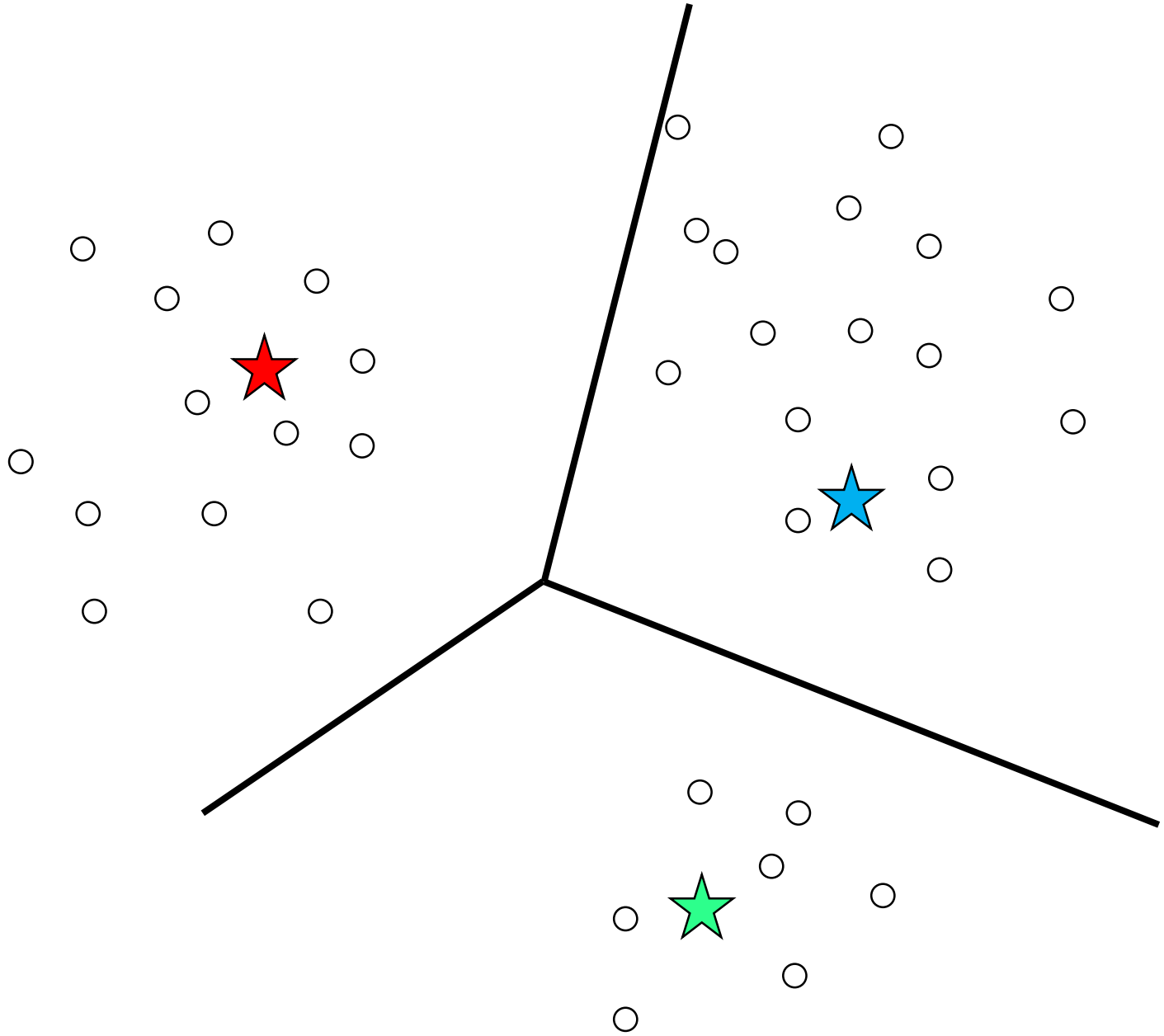
K-means Example

- 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!



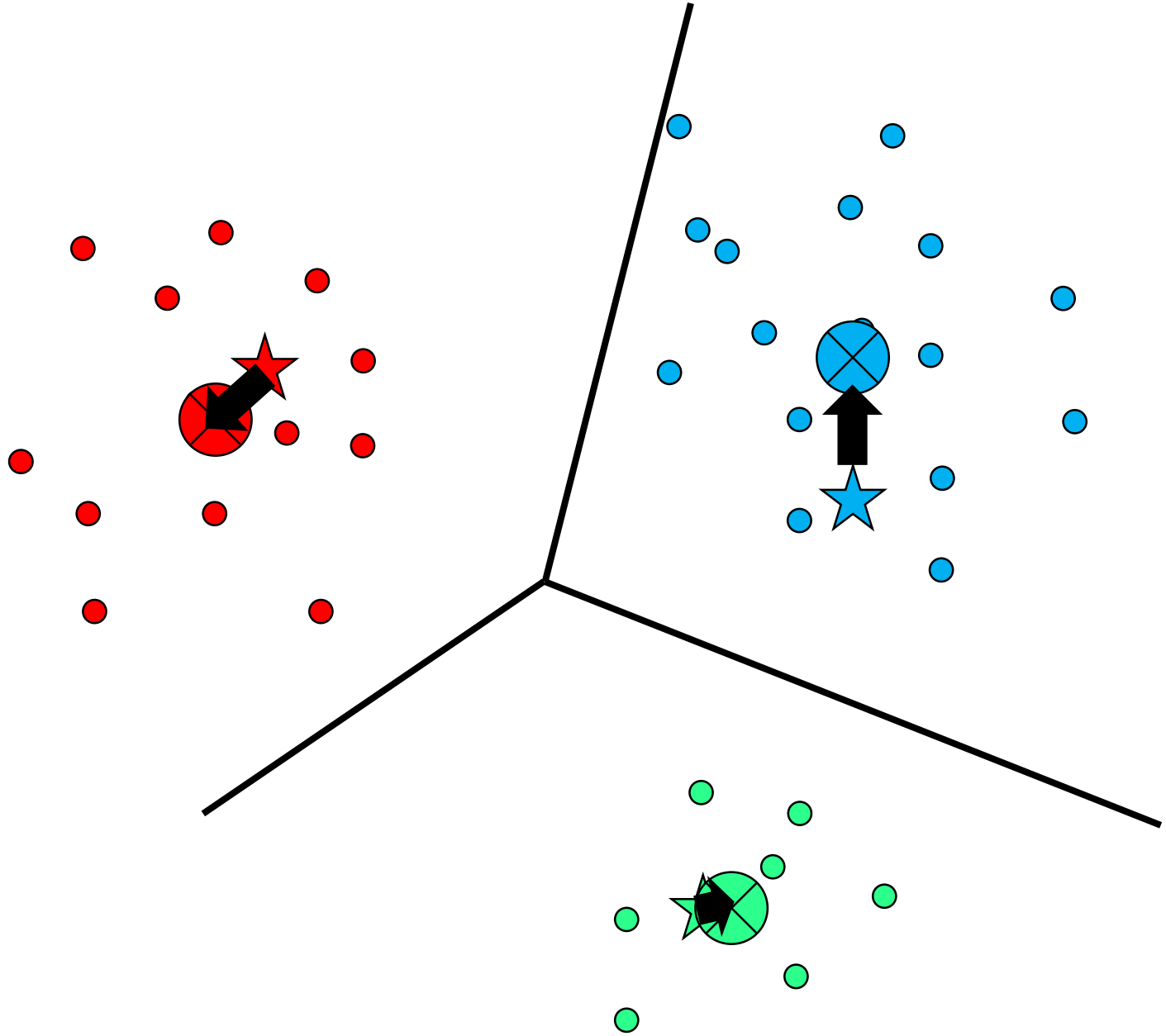
K-means Example

- 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!



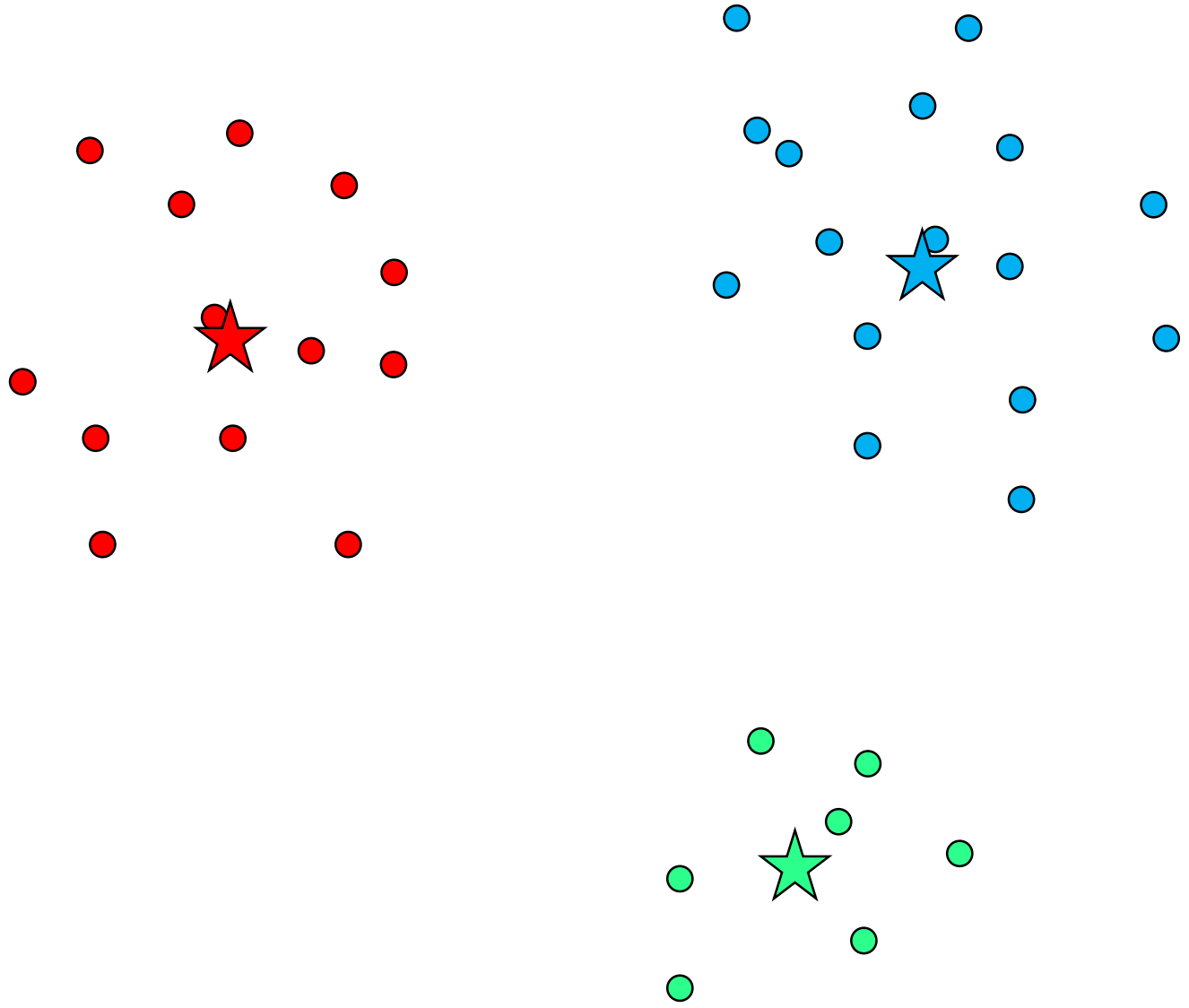
K-means Example

- 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!

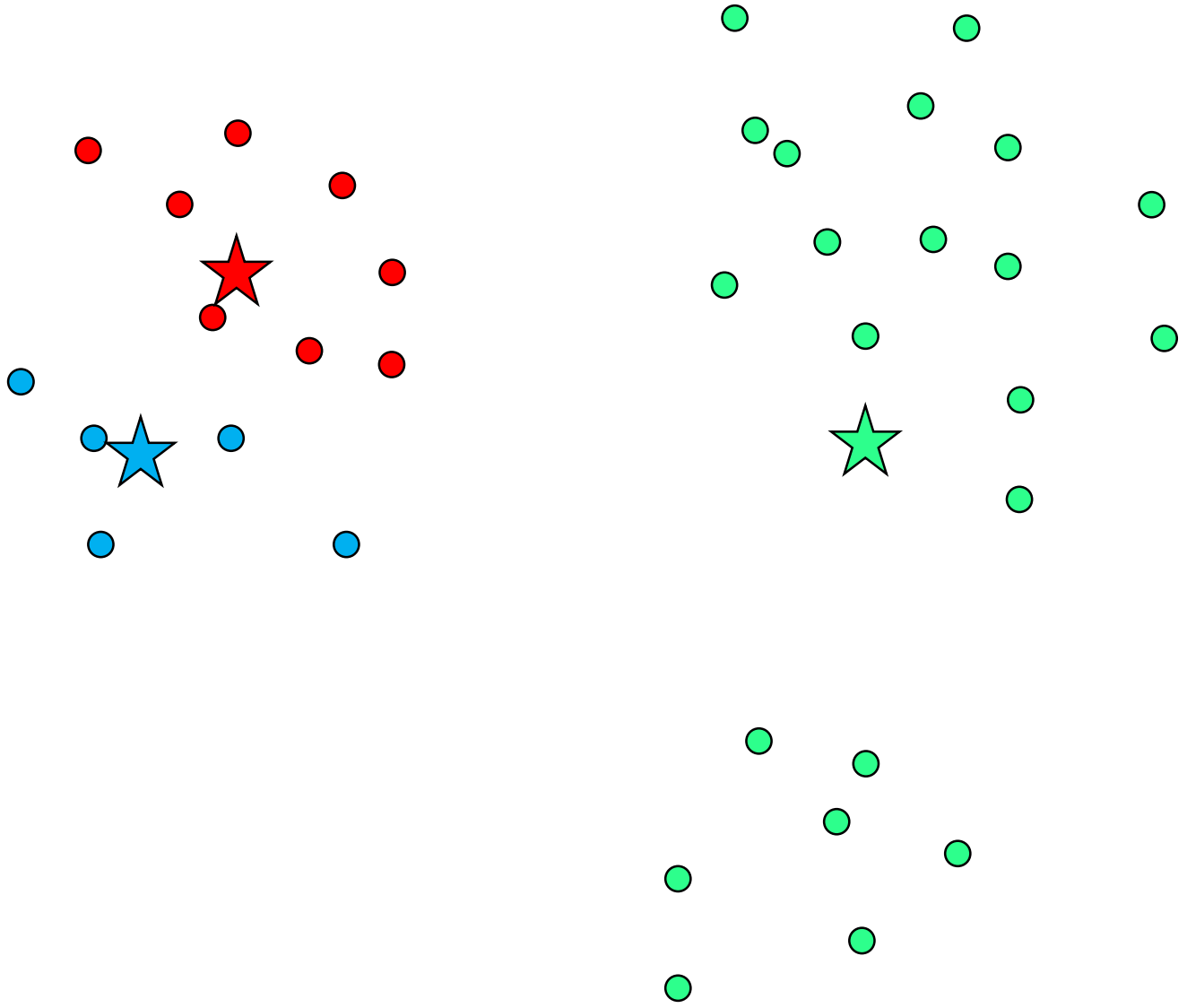


K-means Example

- 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!



Doesn’t always work… can get stuck!



Algorithm

K-means Objective:

Find cluster centers $\mathbf{m}$ and assignments $\mathbf{r}$ 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 $\mathbf{m}_k$)

Note: N points, K clusters

Source: Ethan Fetaya, James Lucas, Emad Andrews

If the notation is confusing, you can come back to it after we cover Linear Algebra

- 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)

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

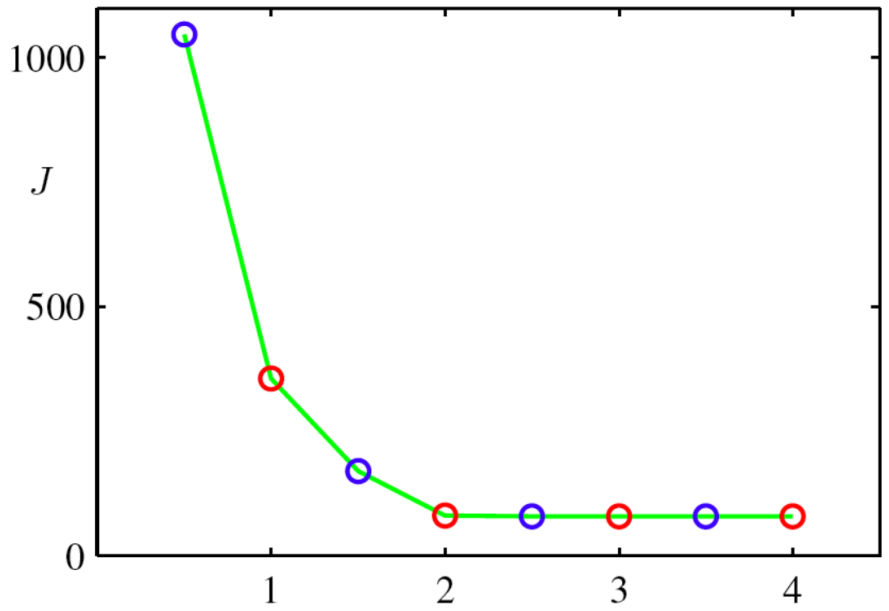
  - Refitting: Model parameters, means are adjusted to match sample means of data points they are responsible for:

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

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 centres is reduced
- Whenever a **cluster 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).



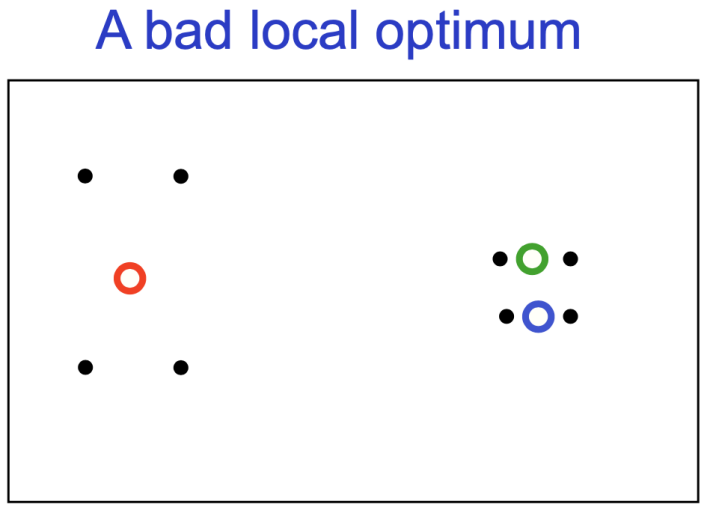
Slide credit: Ethan Fetaya, James Lucas, Emad Andrews

Local Minima

- There is nothing to prevent k-means from getting stuck at a local minimum

Options for avoiding local minimum:

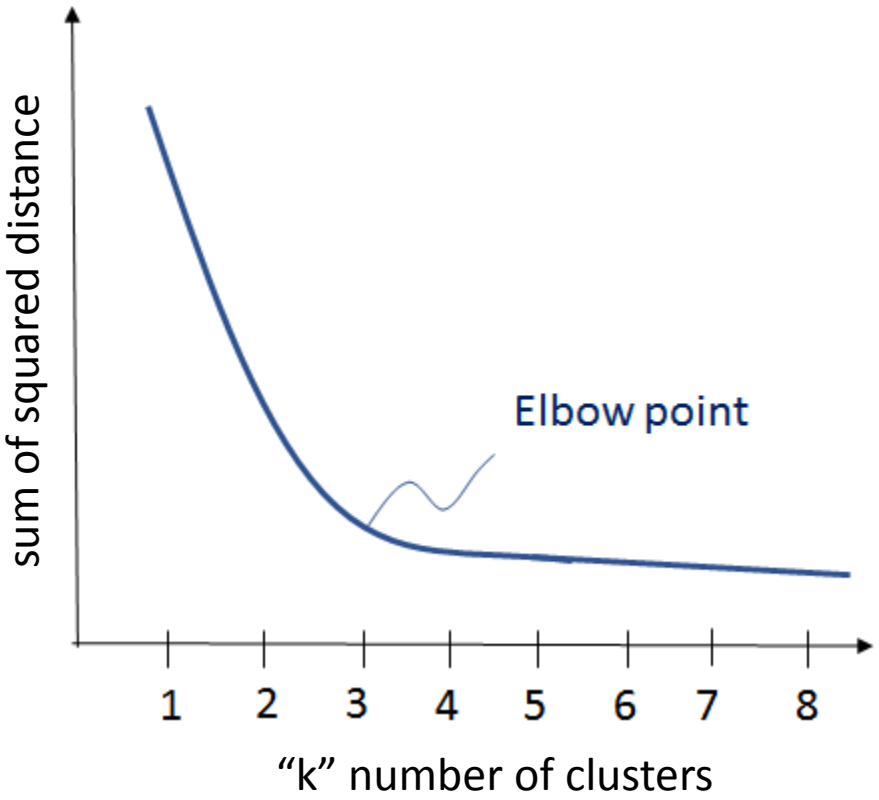
- we could try many random starting points
- split a big cluster into two
- merge to nearby clusters



Slide credit: Ethan Fetaya, James Lucas, Emad Andrews

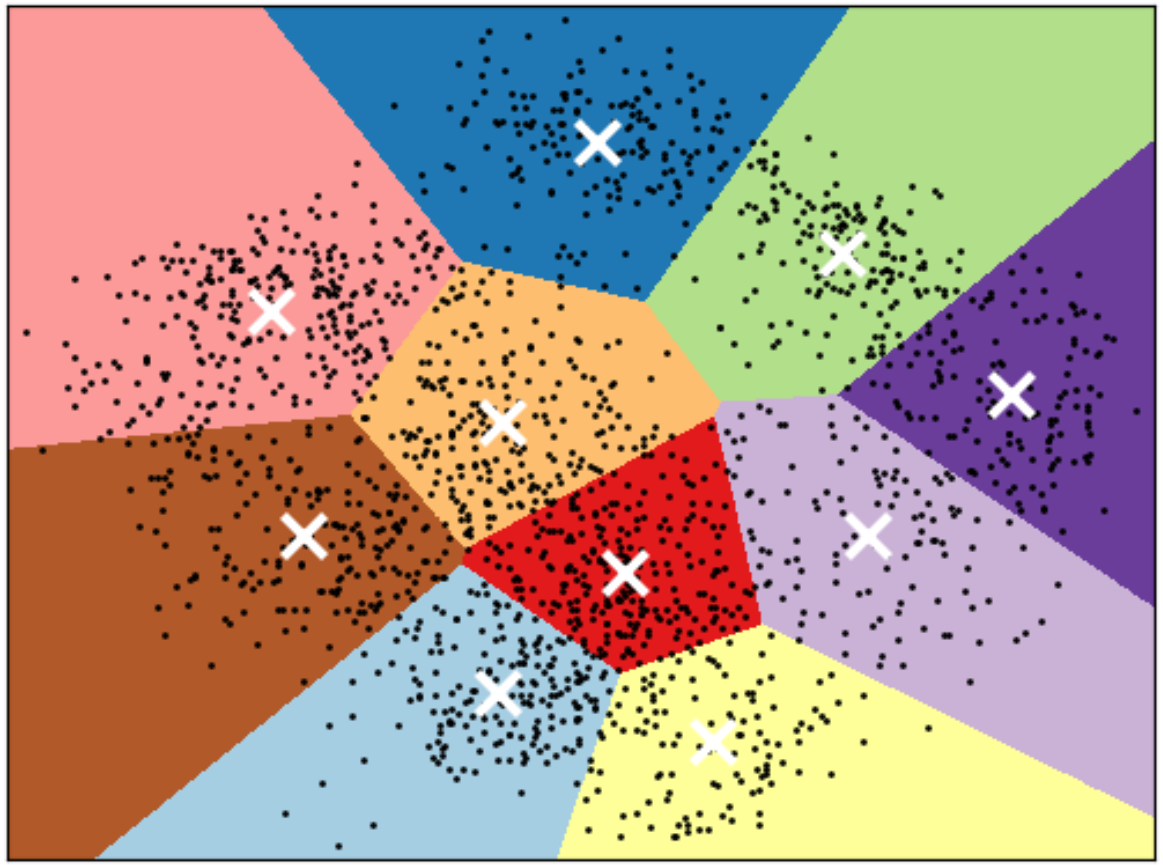
How to choose number of clusters (k)?

- As k increase the sum of squared distance goes towards zero
- If the plot looks like an arm, then the elbow of the arm is the optimal k
  - e.g., the elbow is at k=5 indicating the optimal number of clusters is 5



Shape of K-Means Clusters

- K-means split the space according on the closest mean:

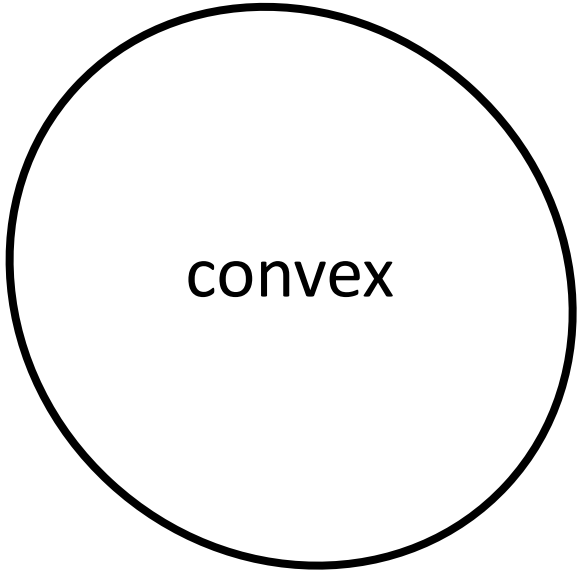


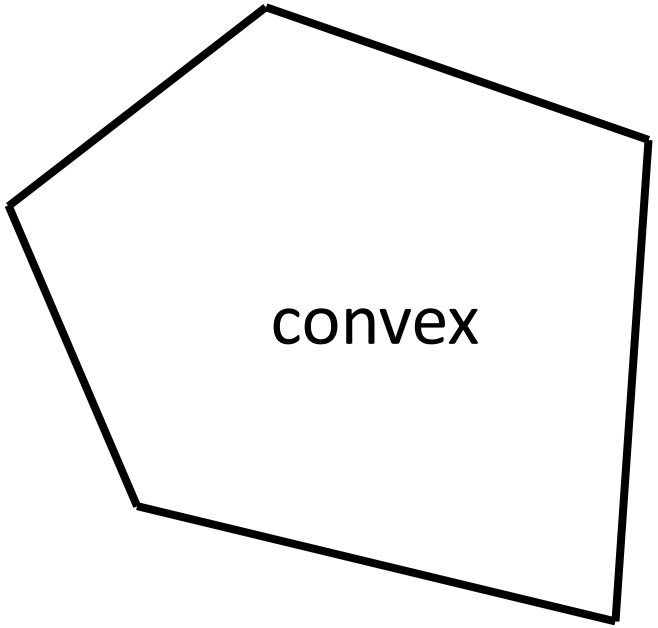
Source: scikit-learn

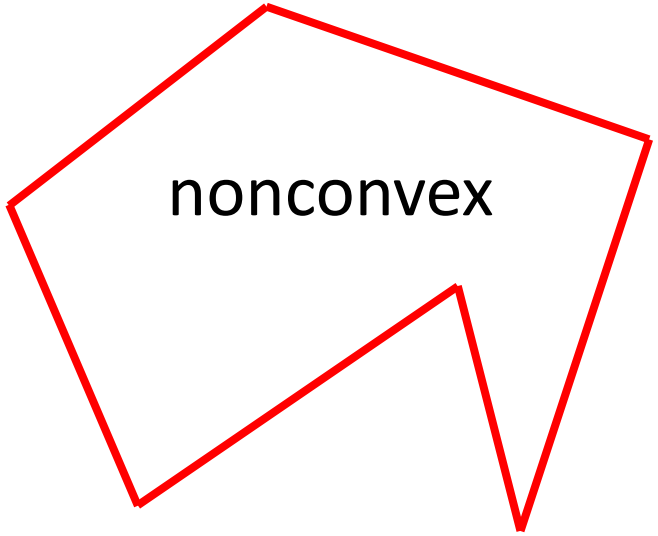
- Note that the clusters form convex regions.

Convex Region

- A region is convex if any line between two points in the region remains in the region.

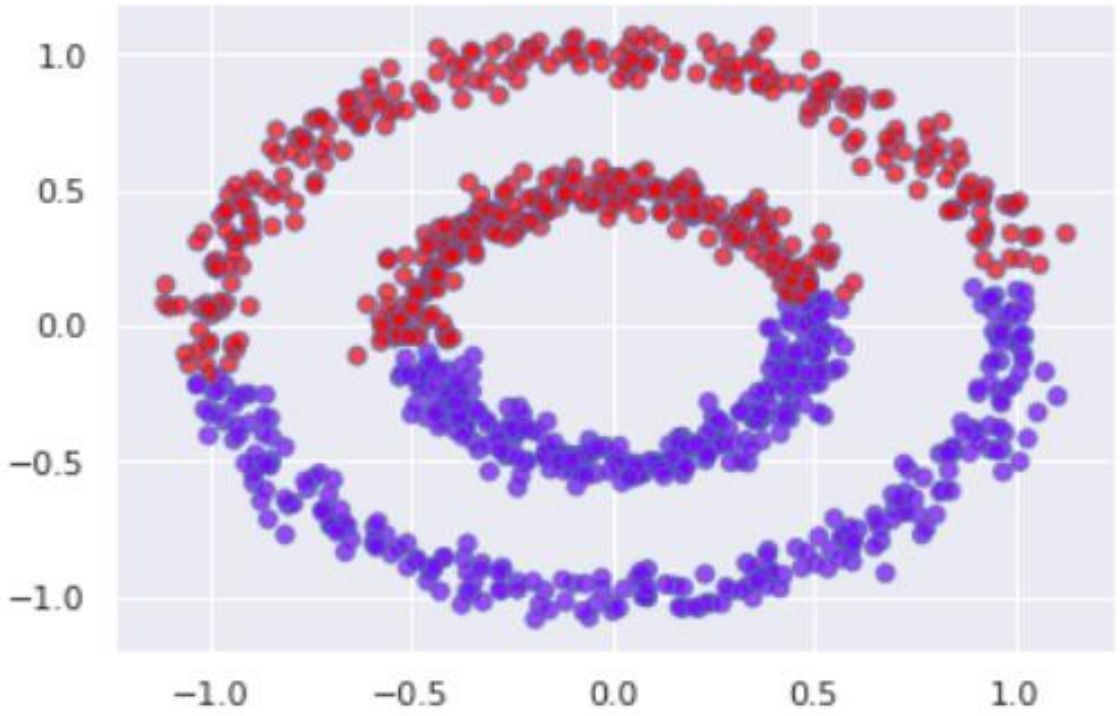






K-Means with Non-Convex Clusters

K-Means (K = 2)

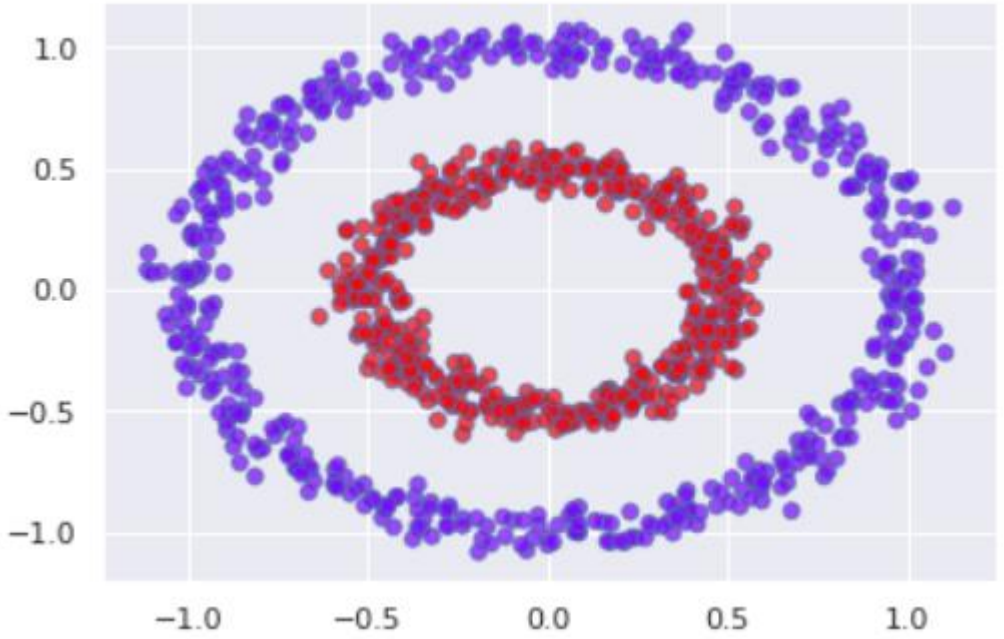


Source: Cory Malkin

K-means is unable to separate non-convex clusters

Density-Based Clustering

- Density-based clustering
  - define clusters based on dense regions
  - cluster number is not set
  - cluster complexity can grow with more data



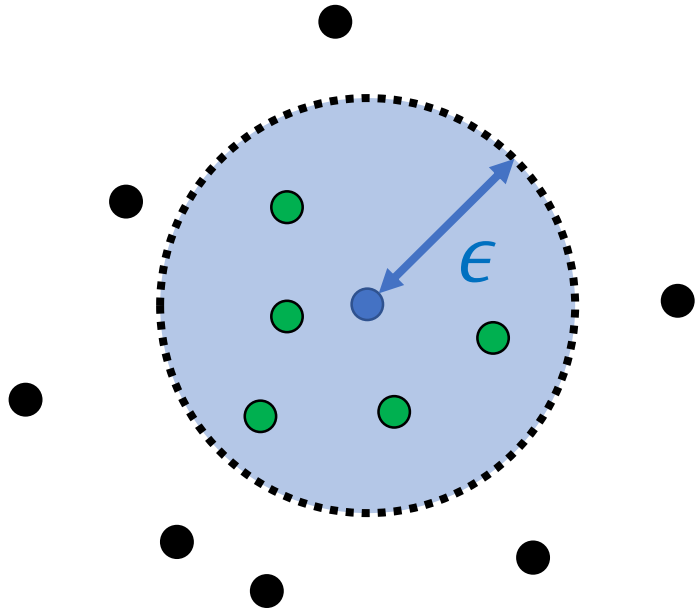
Source: Cory Malkin

- Q: What about samples/instances that don't belong to a cluster?

Density-Based Clustering Algorithm

Algorithm:

- Select unassigned arbitrary point X_i
  - check if X_i is within ε and has at least min_neighbours
    - if no, x_i is will not be included in the cluster
      if yes, expand cluster with x_i
      - Assign all neighbouring points within ε to the cluster
      - Repeat the expand cluster process for each new point that was added

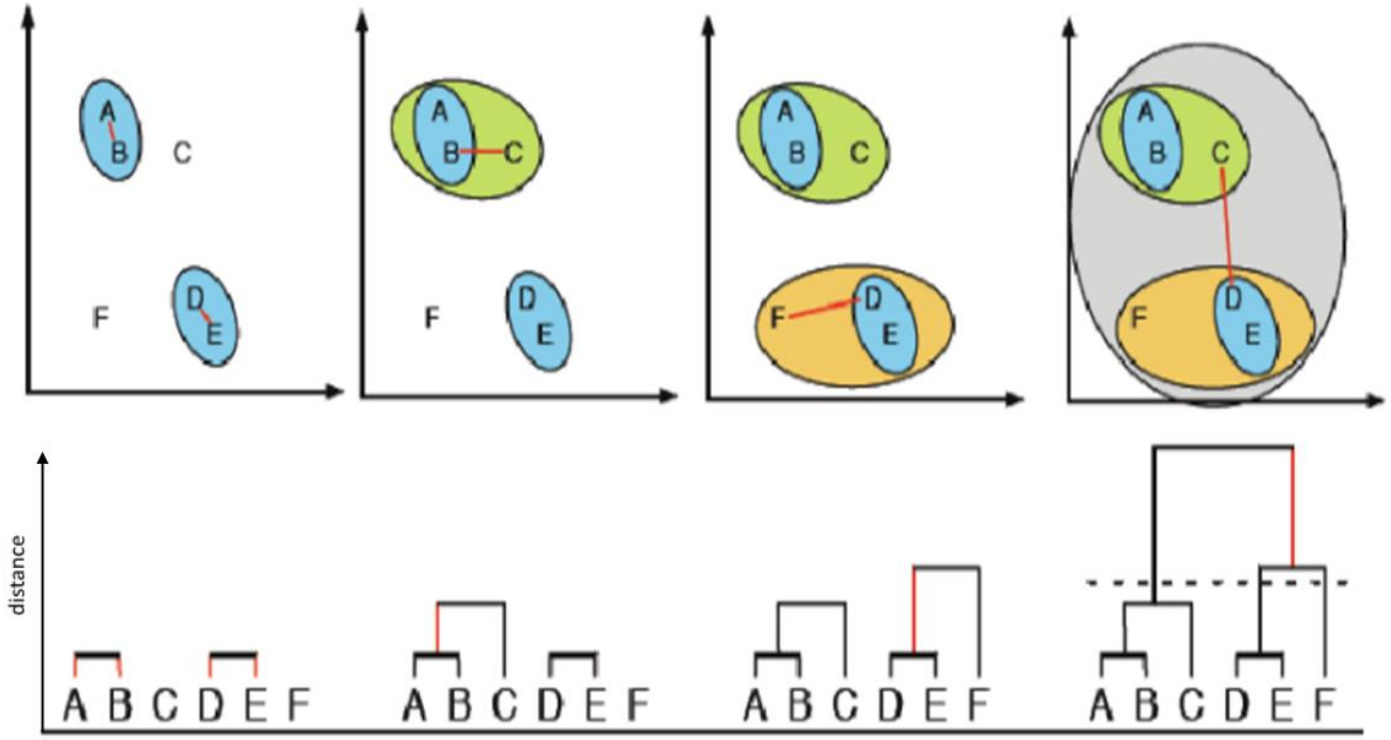


ε (epsilon): distance for deciding if a point is neighbour

min_neighbours: the minimum number of points required

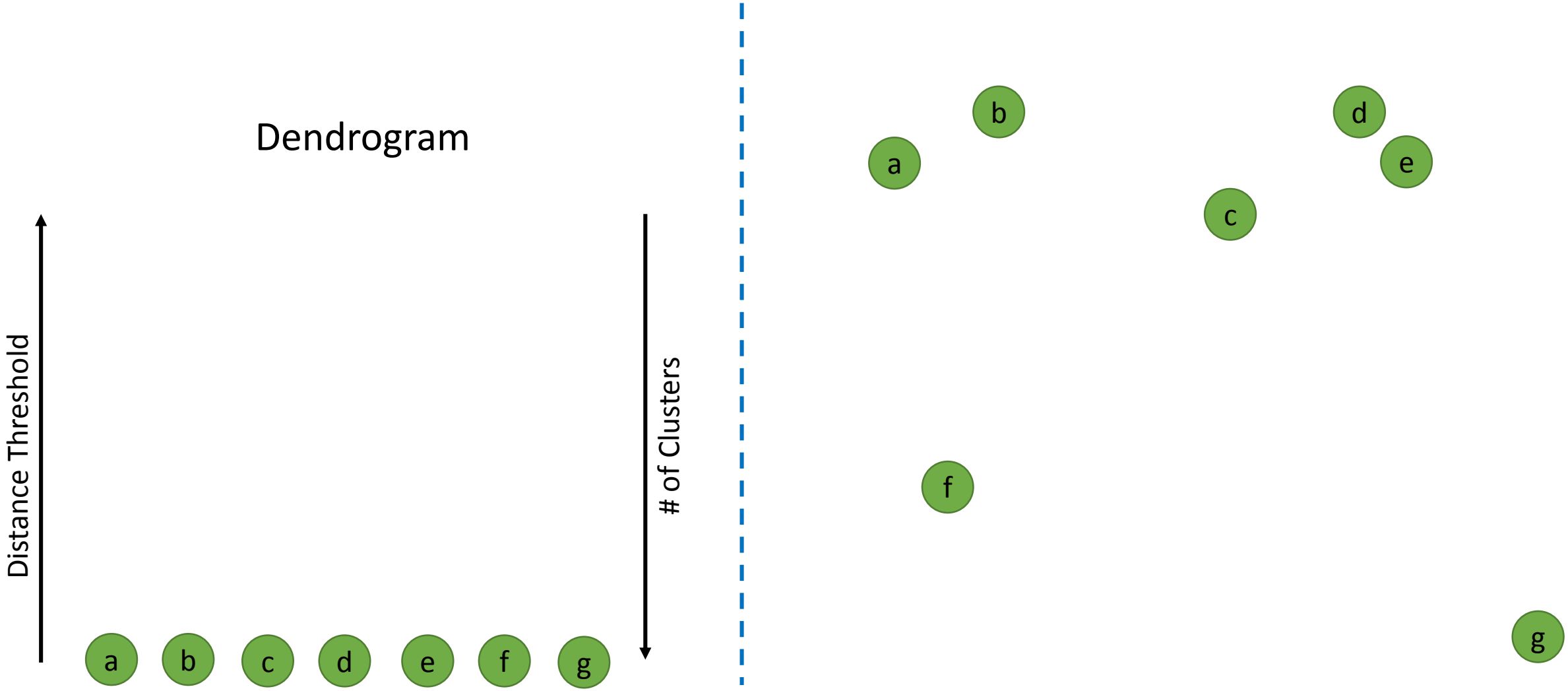
Agglomerative Clustering

- A type of Hierarchical Clustering
- Algorithm:
  1. Starts with each point in its own cluster
  2. Each step merges the two "closest" clusters

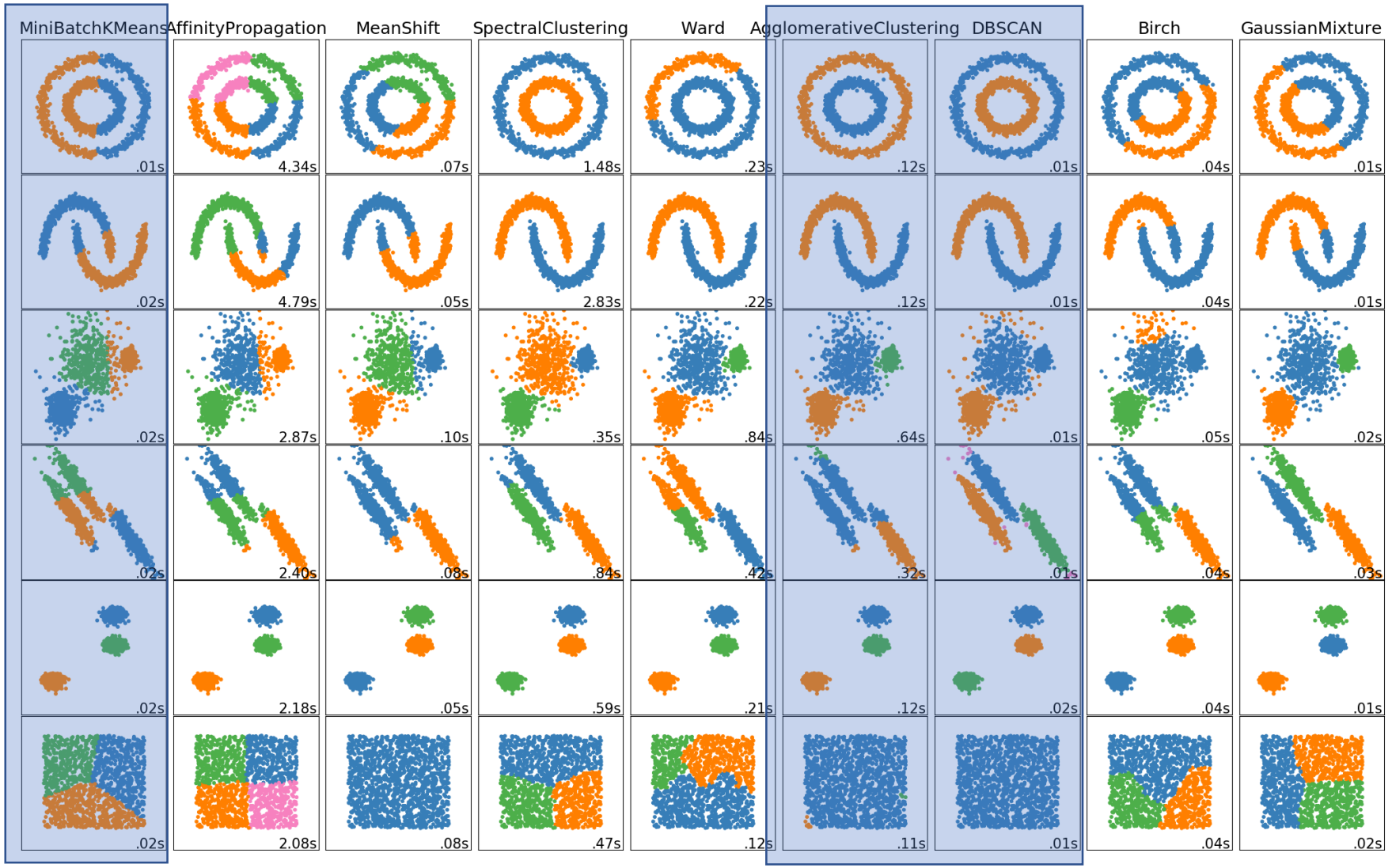


Source: MachineLearningStories

Example: Agglomerative Clustering



Comparison of Clustering Methods



- There are many clustering algorithms to chose from
- Performance will depend on your data

Source: [scikit-learn]

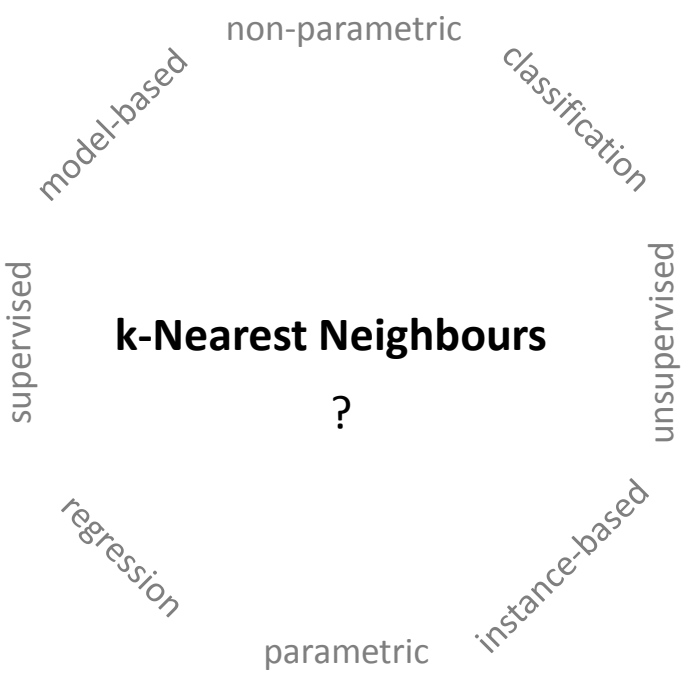
Applications in Computer Vision

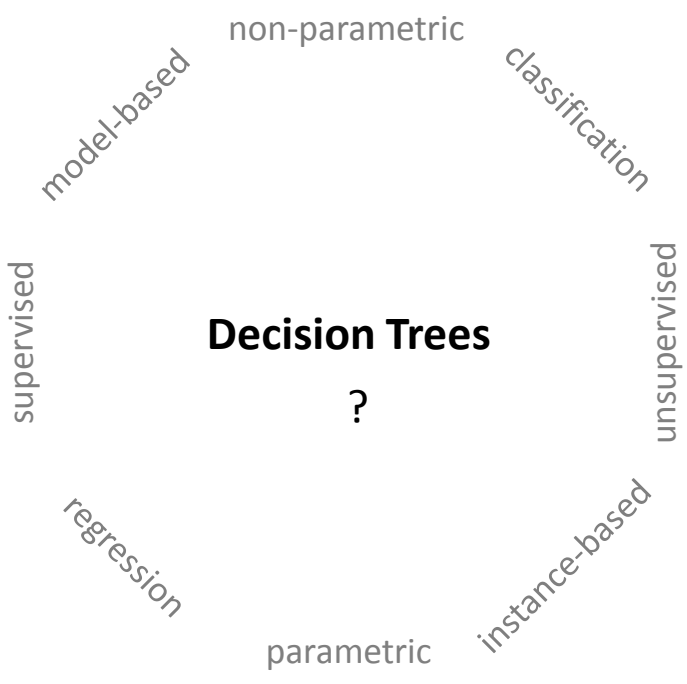
- Replace samples with the mean of their cluster (vector quantization)
- Visual example:
  - Inputs: colour pixels represented as RGB values
  - Outputs: cluster (average RGB) value obtained using k-means
- Q: How can this be applied for compression?
- Q: How can this be applied for image segmentation?

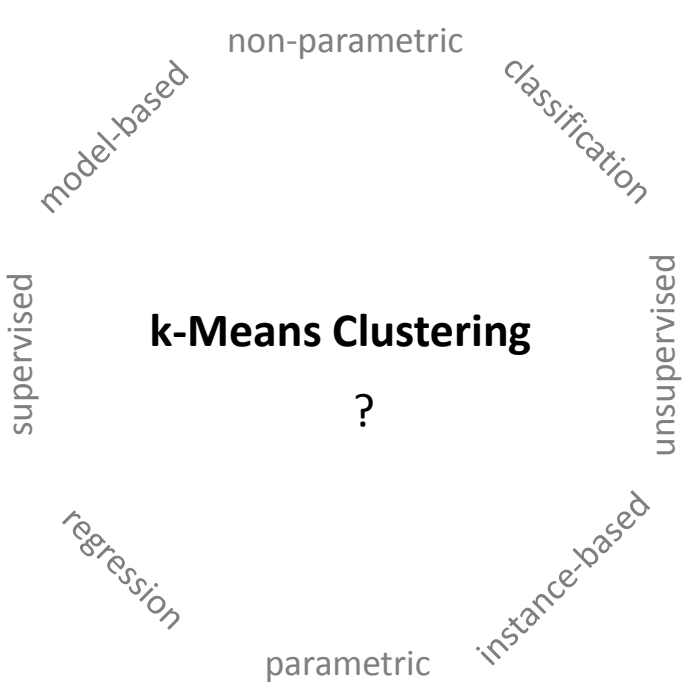
Source: PRML Bishop 2006

K-Means Code Example (Google Colab)

Summary: Instance-Based Learning







- Model-based Learning with be covered in the 2nd half of the course

Next Time

- Week 3 Q/A Support Session
  - Help with Python and Project 1

- Week 4 Lecture – Measuring Uncertainty
  - Probability Theory
  - Summary Statistics
  - Multivariate Gaussians
  - Performance Metrics