APS1070

Foundations of Data Analytics and Machine Learning

Winter 2021

Week 3: Foundations of Learning

- *End-to-end Machine Learning*
- *K-Nearest Neighbours Recap*
- *Decision Trees*
- *Clustering*

Sinisa Colic and Samin Aref

Slide Attribution

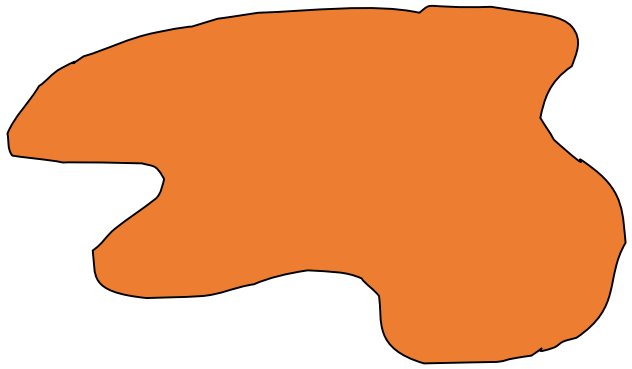
These slides contain materials from various sources. Special thanks to the following authors:

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

Last Time

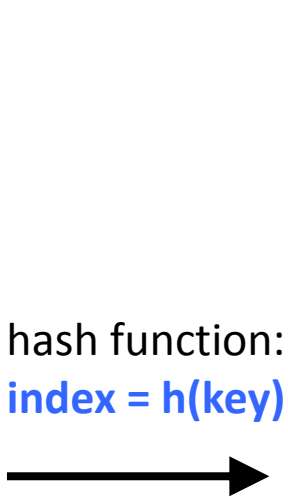
- Python Checklist and Expectations
- Algorithms and Big O Notation

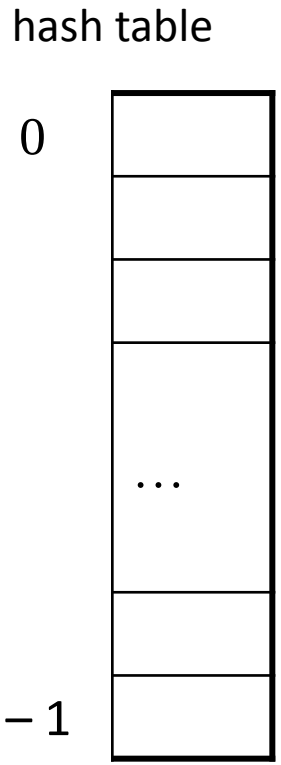
Hash Tables



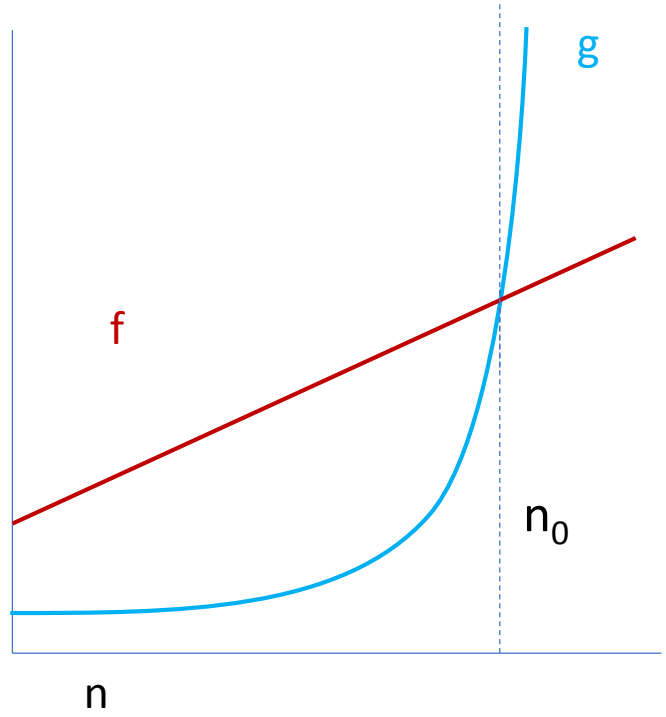
key space (e.g., integers, strings)

hash function:
index = h(key)





Big-Oh Notation



$f(n) \leq c\, g(n)$ for all $n \geq n_0$

Agenda

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

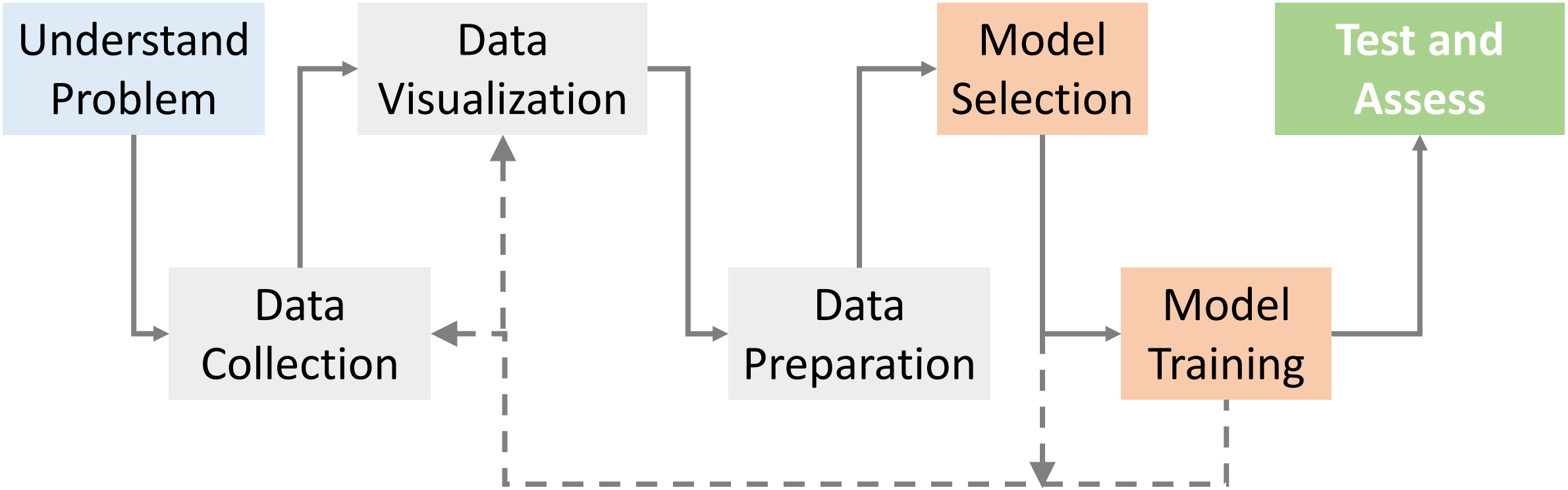
- End-to-end Machine Learning Revisited
- k-Nearest Neighbours Recap
- Assessing Performance
- Decisions Trees

Supervised Learning

- Clustering Strategies
  - k-Means
  - Density-Based Clustering
  - Agglomerative Clustering

Unsupervised Learning

End-to-End Machine Learning



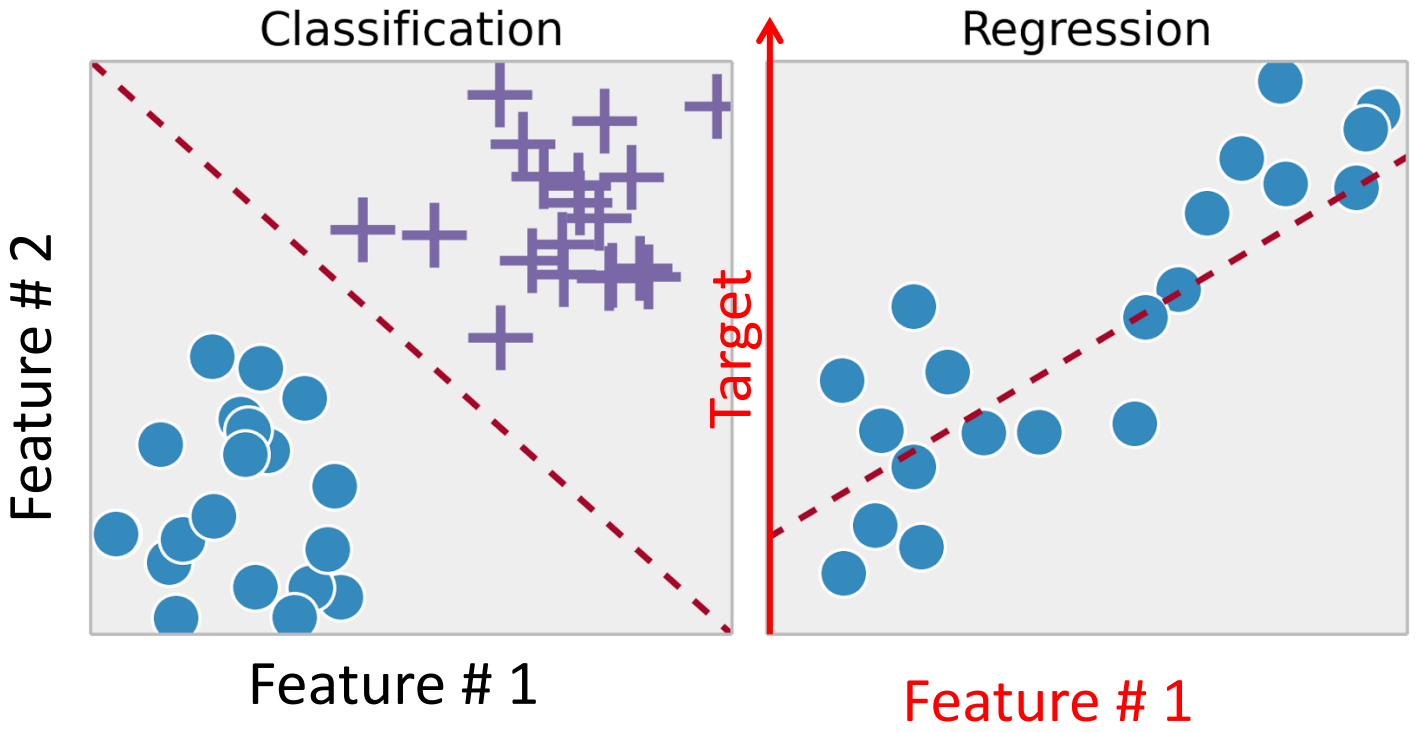
Understand the Problem

- Often, we need to make some sort of decisions (predictions)
- Two common types of decisions that we make are:
  - Classification
    - Discrete number of possibilities
  - Regression
    - Continuous number of real-valued possibilities

	Supervised	Unsupervised
Discrete	classification	clustering
Continuous	regression	dimensionality reduction

Classification vs. Regression

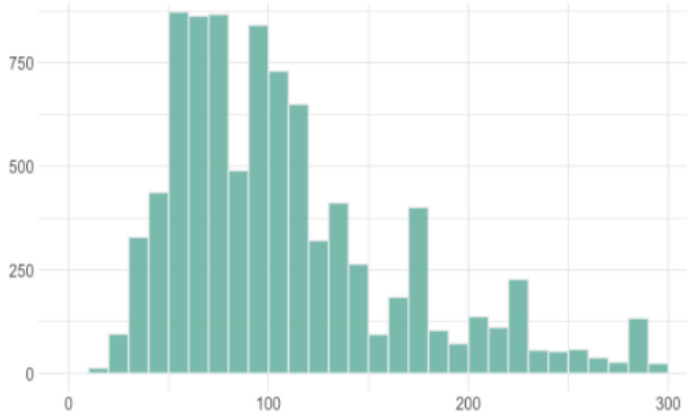
- **Classification:** Discrete target
  - Separate the Dataset
    - Apples or oranges?
    - Dog or Cat?
    - Handwritten digit recognition
- **Regression:** Continuous Target
  - Fit the dataset
    - Price of a house
    - Revenue of a company
    - Age of a tree



Understand the Problem

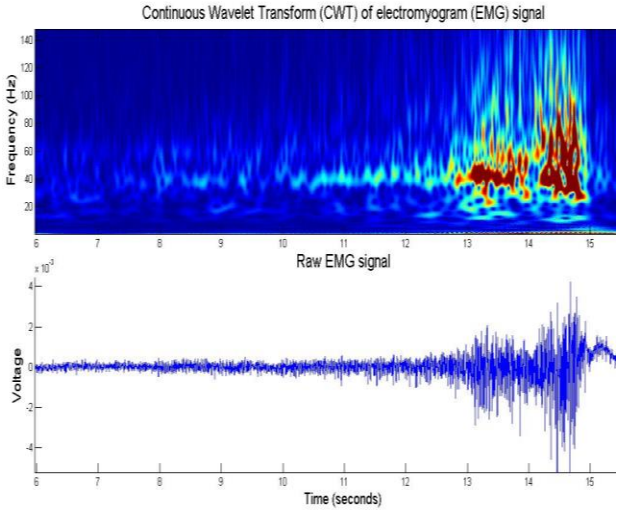
Input data is represented by features that can come in many forms:

- Raw pixels
- Histograms
- Tabular data
- Spectrograms
- …



	Gulf Respondents			Non-Gulf Respondents		
	Sample			Sample		
	Eats Oysters	Does Not	Census Range	Eats Oysters	Does Not	Census Range
N =	444	72		269	35	
Age 65+ (%)	0.23	0.21	0.08–0.16	0.20	0.29	0.10–0.14
Male (%)*,†	0.47	0.32	0.50–0.53	0.67	0.63	0.49–0.53
White (%)	0.65	0.68	0.33–0.73	0.69	0.77	0.30–0.85
Persons per household	2.48	2.29	2.30–2.70	2.58	2.49	2.10–2.80
High school graduate (%)	0.96	0.96	0.72–0.93	0.95	1.00	0.80–0.93
College graduate (%)*	0.50	0.43	0.24–0.49	0.58	0.46	0.22–0.58
Household income category*	$40,000 –$49,999	$40,000 –$49,1000	$30,858 –$52,971	$75,000 –$84,999	$60,000 –$74,999	$34,800 –$78,378

Notes: Asterisk (*) indicates significant difference between sample Gulf and non-Gulf oyster eaters. Dagger (†) indicates significant difference between sample Gulf oyster eaters and noneaters.
Source: U.S. Census Bureau (2016).

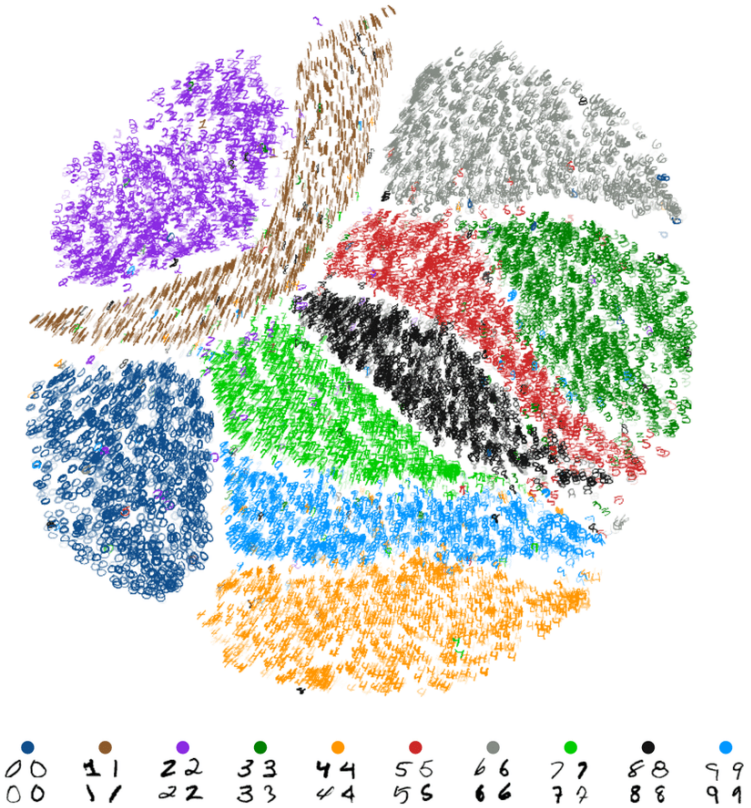


Data Exploration

- Understand your data through visualization
- Assess the difficulty of the problem

- You have a data set D = {($x^{(i)}$,$y^{(i)}$)}
- You want to learn y = f(x) from D
  - more precisely, you want to minimize error in predictions

- Q: What kind of model (algorithm) do you need for identifying handwritten digits?



MNIST low-dimensional projection

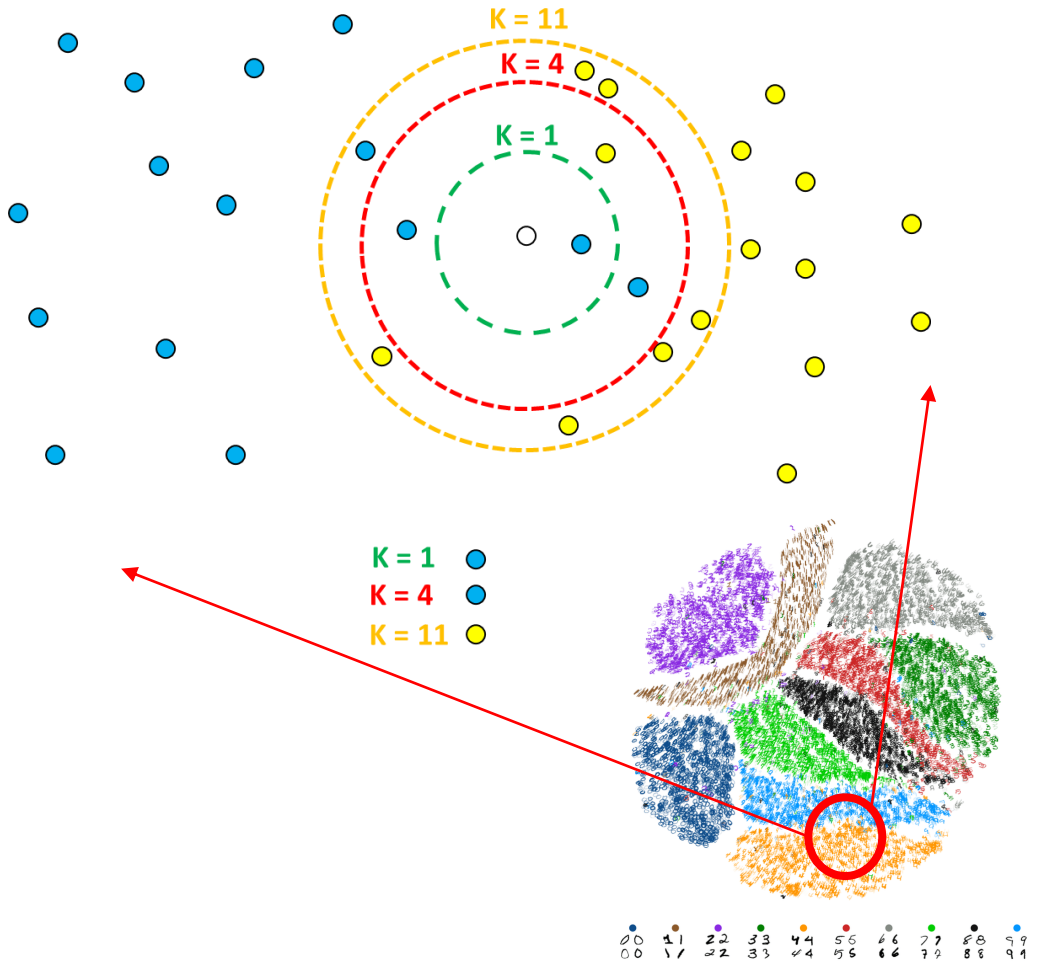
Model Selection

Many classifiers to choose from

- Support-Vector Machine (SVM)
- Logistic Regression
- Random Forests
- Naive Bayes
- Bayesian network
- K-Nearest Neighbour
- (Deep) Neural networks
- Etc.

Model Selection

- Often the easiest algorithm to implement is k-Nearest Neighbours
- Match to similar data using a distance metric



Q: What happens as we increase #data?

Q: What about as #data approaches infinity?

Test and Assess

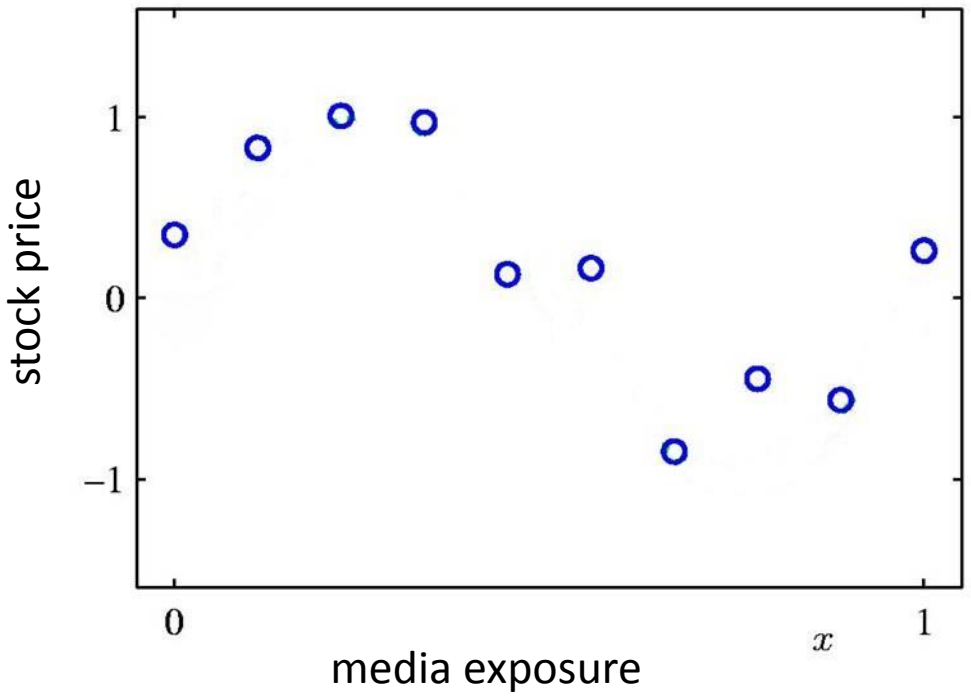
- Unlike us, computers have no trouble with memorization.
- The real question is, how well does our algorithm make predictions on new data?

- We need a way to measure how well our algorithm (model) **generalizes** to new, never before seen, data.

Regression Example

- Let's look at a more concrete example…
- Given noisy sample data (blue), we want to find the polynomial that generated the data

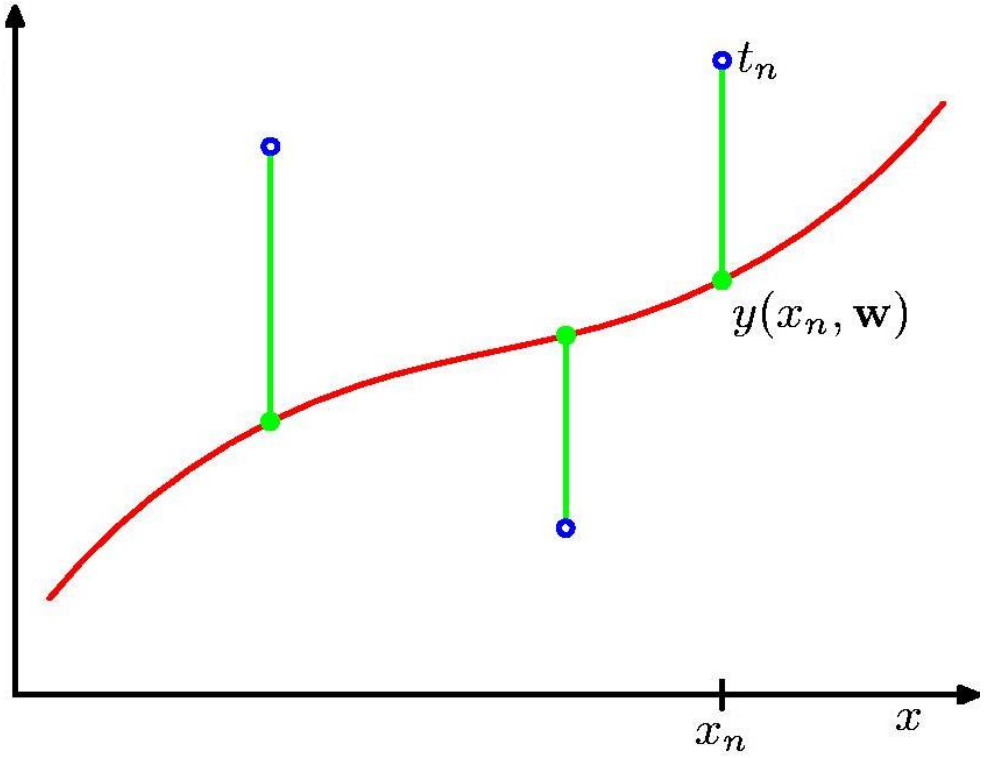
- Q: What kind of a problem is this?



From PRML (Bishop, 2006)

Mean Squared Error

- Need to first define our error term, in this case we can use the **mean squared error (MSE)**:
- Error is measured by finding the squared error in the prediction of y(x) from x.
  - The error for the red polynomial is the sum of the squared vertical errors

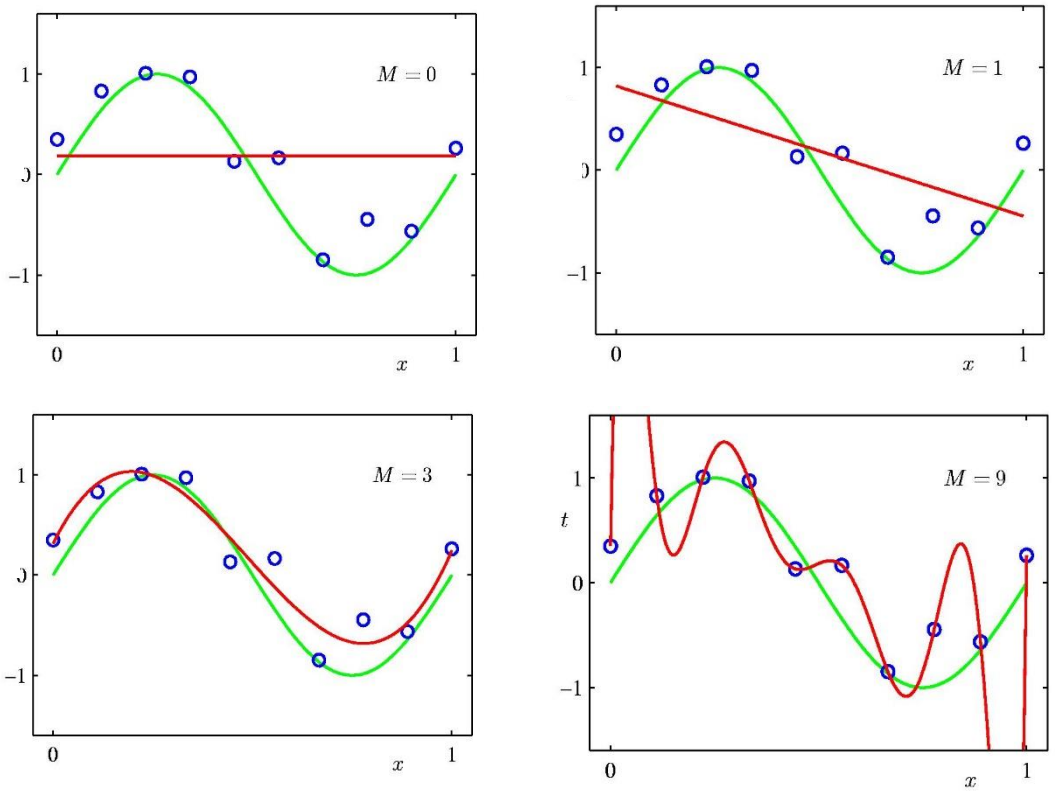


From PRML (Bishop, 2006)

Fitting the Data

Q: Which polynomial fits the data best?

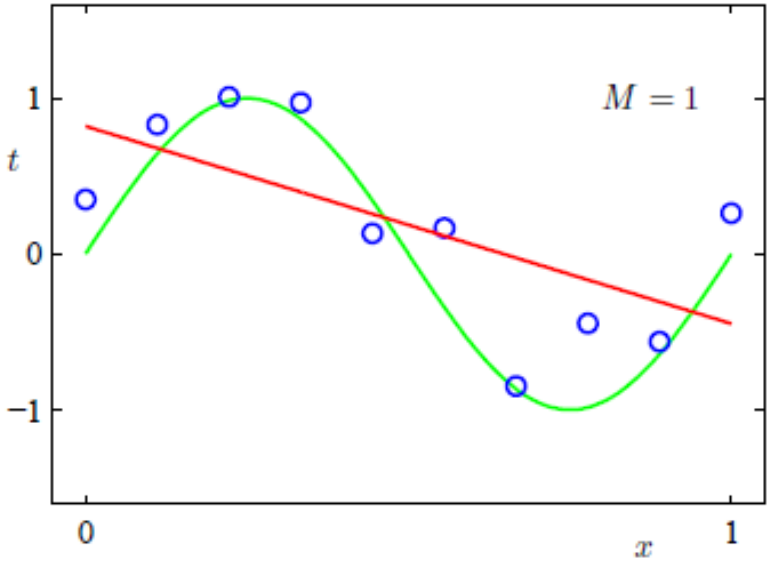
- based on error term?
- based on test data?



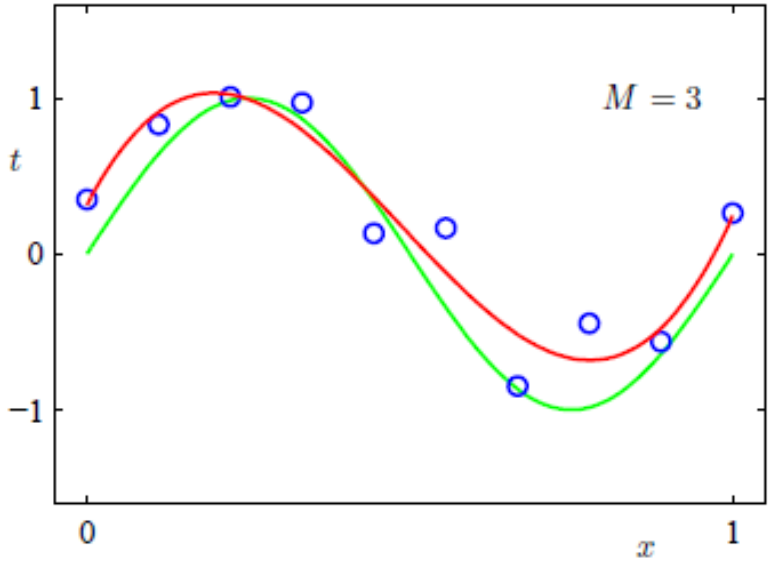
From PRML (Bishop, 2006)

Overfitting vs Underfitting

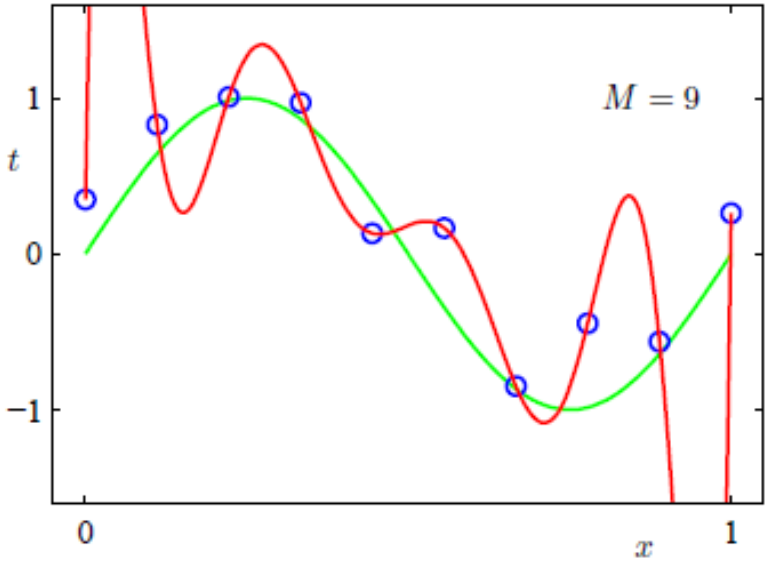
High training error
and High test error (Underfit)



Acceptable training error
and test error



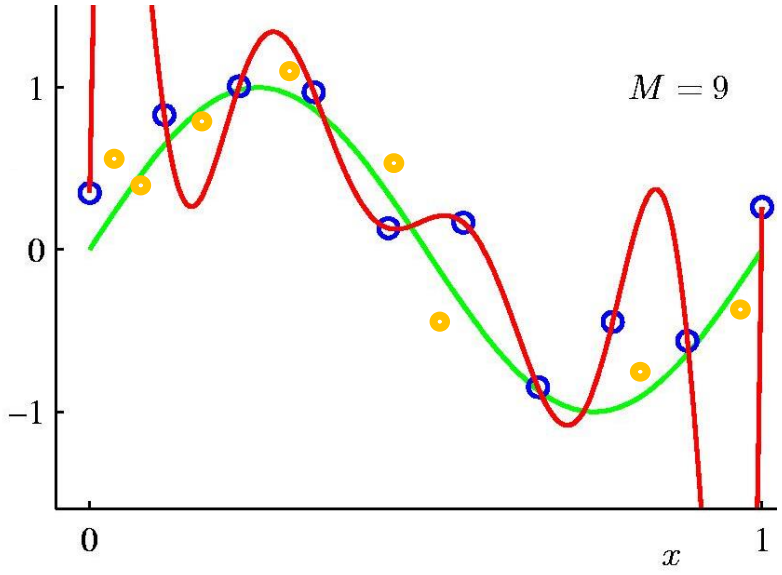
Perfect training error (zero)
and high test error (Overfit)

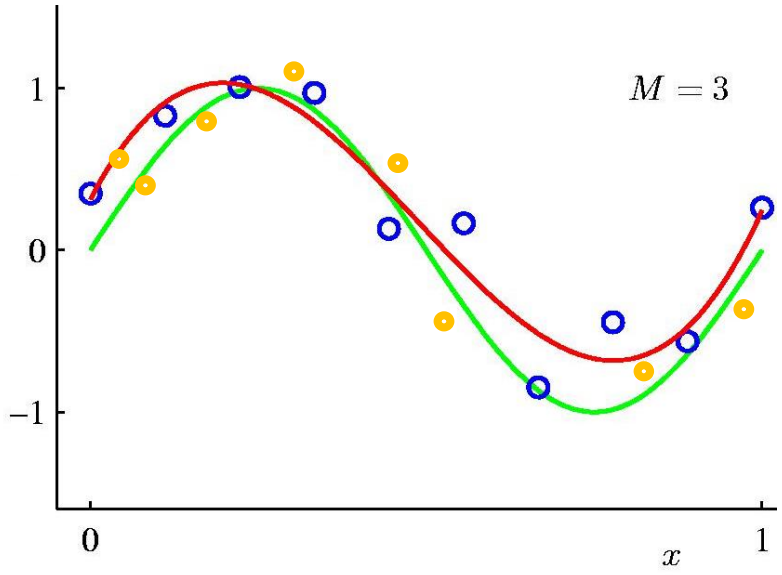


From PRML (Bishop, 2006)

Generalization

- Giving the model a greater capacity (more complexity) to fit the data… does not necessarily help
- How do we evaluate the model performance?





From PRML (Bishop, 2006)

Verify model on New Data

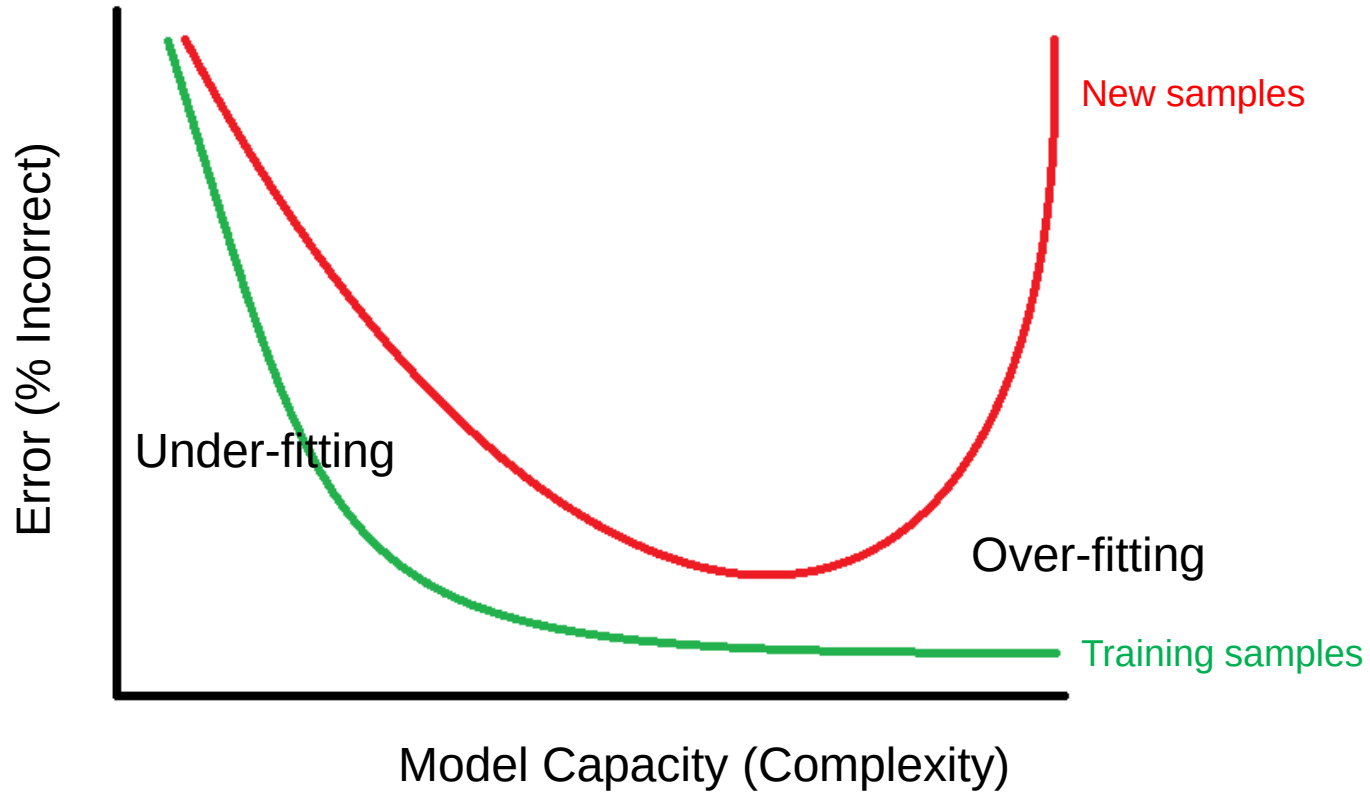
Overfitting

- In brief: **overfitting** is fitting characteristics of training data that do not generalize to future test data

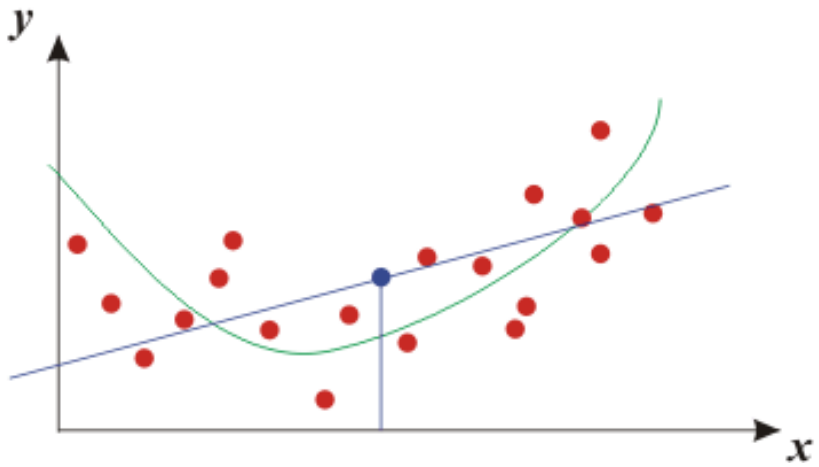
- Central problem in machine learning
- Particularly problematic if #data << #parameters
- … don’t have enough data to “identify” parameters

Generalization

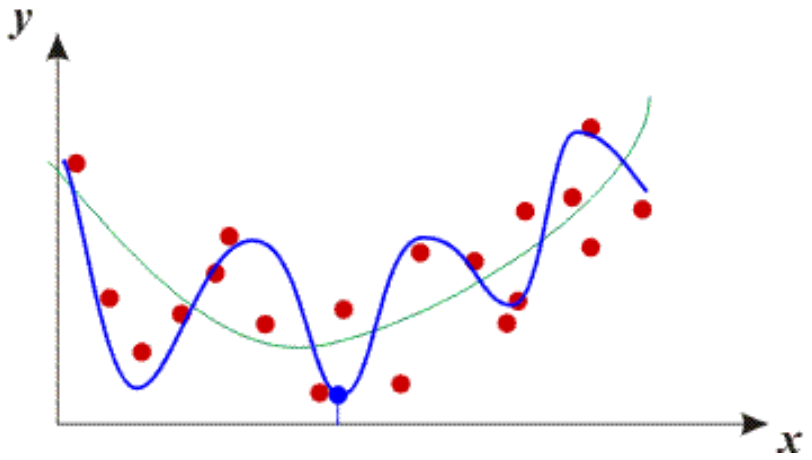
- Machine learning is a game of balance, with our objective being to **generalize** to all possible future data



Bias-Variance Trade-off



- Models with too few parameters are inaccurate because of a **large bias** (not enough flexibility).



- Models with too many parameters are inaccurate because of a **large variance** (too much sensitivity to the sample).

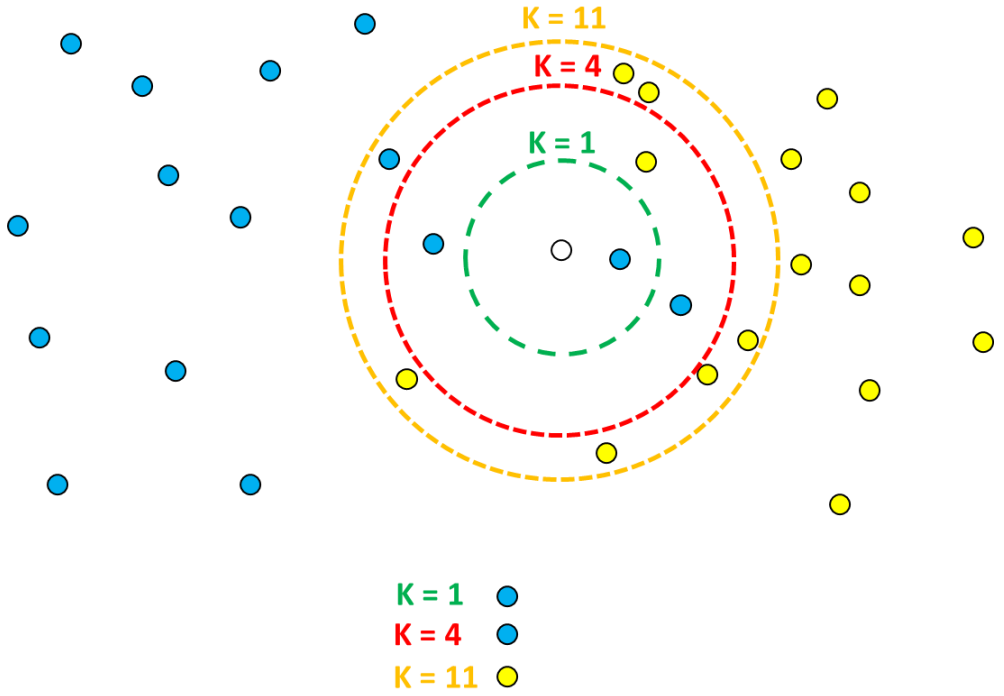
Inductive Bias

- Let's avoid making assumptions about the model (polynomial order)
  - Assume for simplicity that D = $\{(x^{(i)},y^{(i)})\}$ is noise free
  - $x^{(i)}$'s in D only cover small subset of input space x
- **Q: What's the best we can do?**
  - If we've seen $x=x^{(i)}$ report $y=y^{(i)}$
  - If we have not seen $x= x^{(i)}$, can't say anything (no assumptions)
- This is called rote learning… boring, eh?
  - **Key idea: you can't generalize to unseen data w/o assumptions!**
- Thus, key to ML is generalization
  - **To generalize, ML algorithm must have some inductive bias**
  - Bias usually in the form of a restricted model (hypothesis) space
  - Important to understand restrictions (and whether appropriate)

Inductive Bias

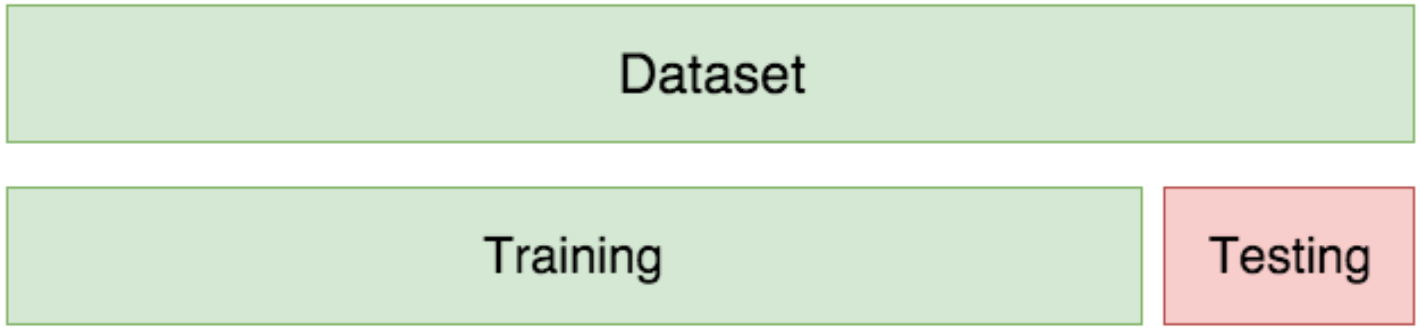
Example: Nearest neighbors

- We suppose that most of the cases in a small neighborhood in feature space belong to the same class. Given a case for which the class is unknown, **we assume that it belongs to the same class as the majority in its immediate neighborhood.**
- This is the inductive bias used in the k-nearest neighbors algorithm.
- The assumption is that cases that are near each other tend to belong to the same class.



Training and Testing Data

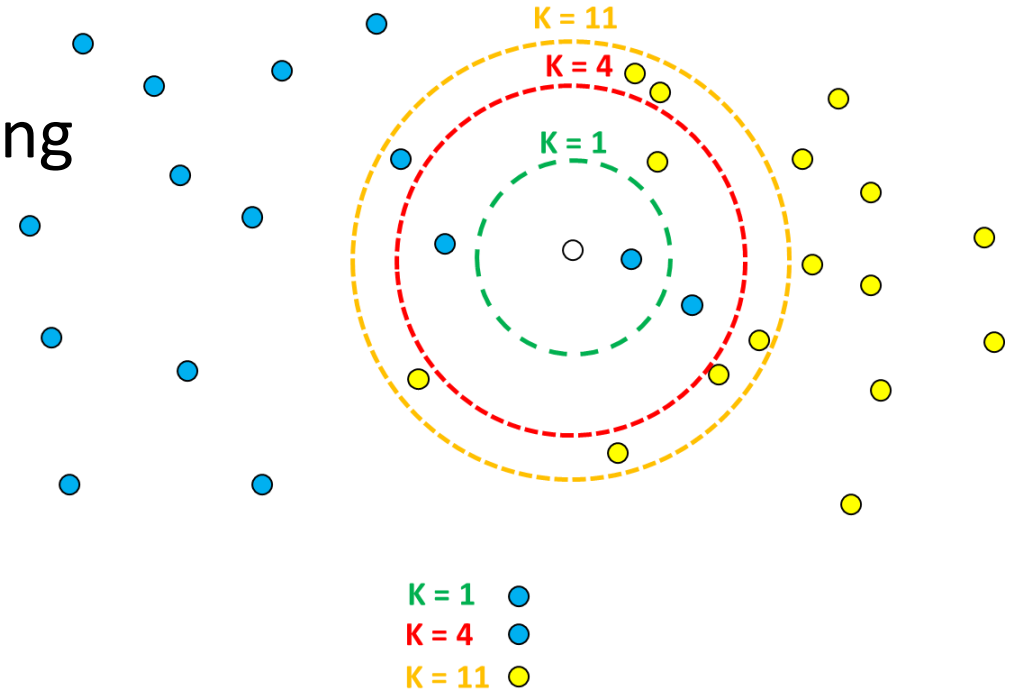
- Track generalization error by splitting data into training and testing
  - 80% training and 20% testing
- More data = better model
  - Would like to use all our data for training, however we need some way to evaluate our model



The problem with tracking test accuracy

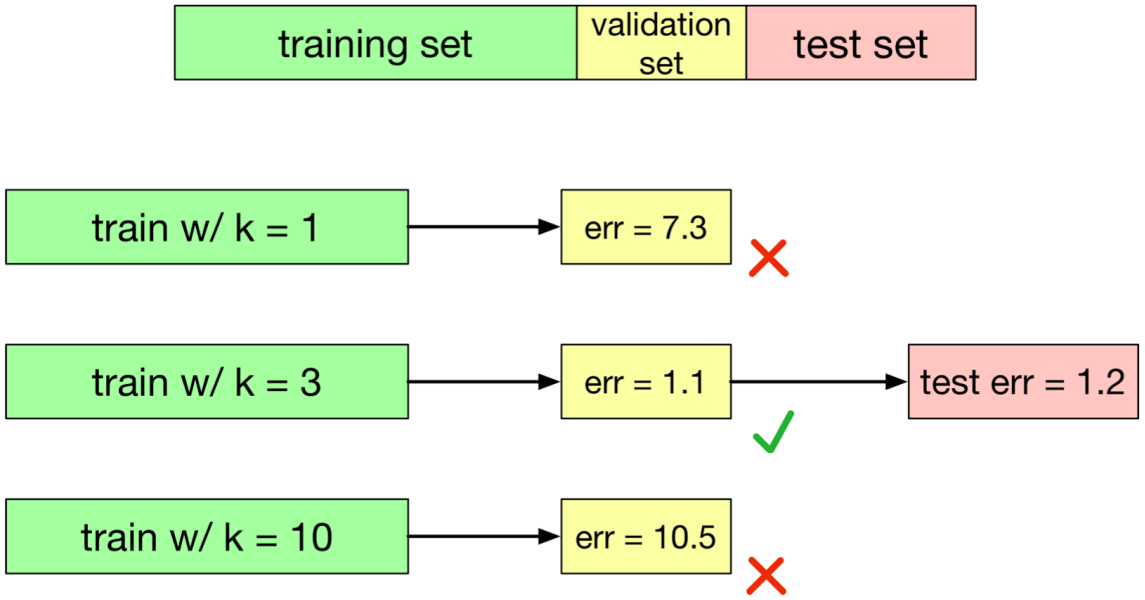
- What K should be?

- If we track test error/accuracy in our training curve, then:
  - We may make decisions about model architecture using the test accuracy and make the testing meaningless…
  - **…then the final test accuracy will not be a realistic estimate of how our model will perform on a new data set!**



Validation Set

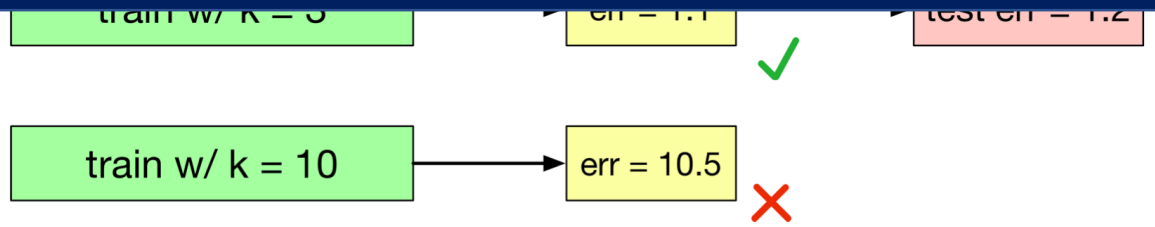
- We still want to track the loss/accuracy on a data set not used for training
- **Idea**: set aside a separate data set, called the **validation set**
  - Track validation accuracy in the training curve
  - Make decisions about model architecture using the validation set



Validation Set

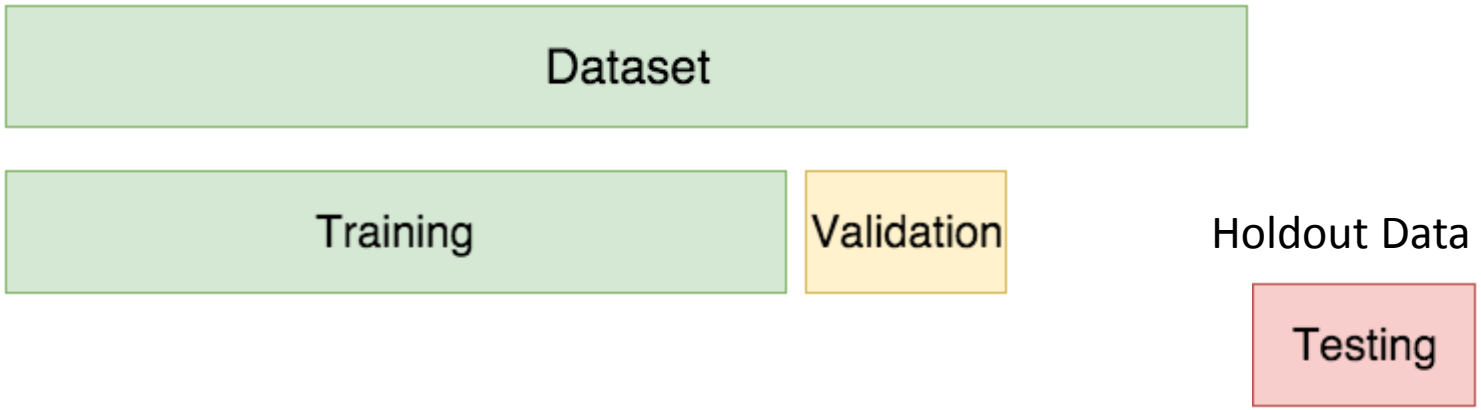
- We still want to track the loss/accuracy on a data set not used for training
- **Idea**: set aside a separate data set, called the **validation set**

K is a hyperparameter.
We tune hyperparameters using the validation set



Validation and Holdout Data

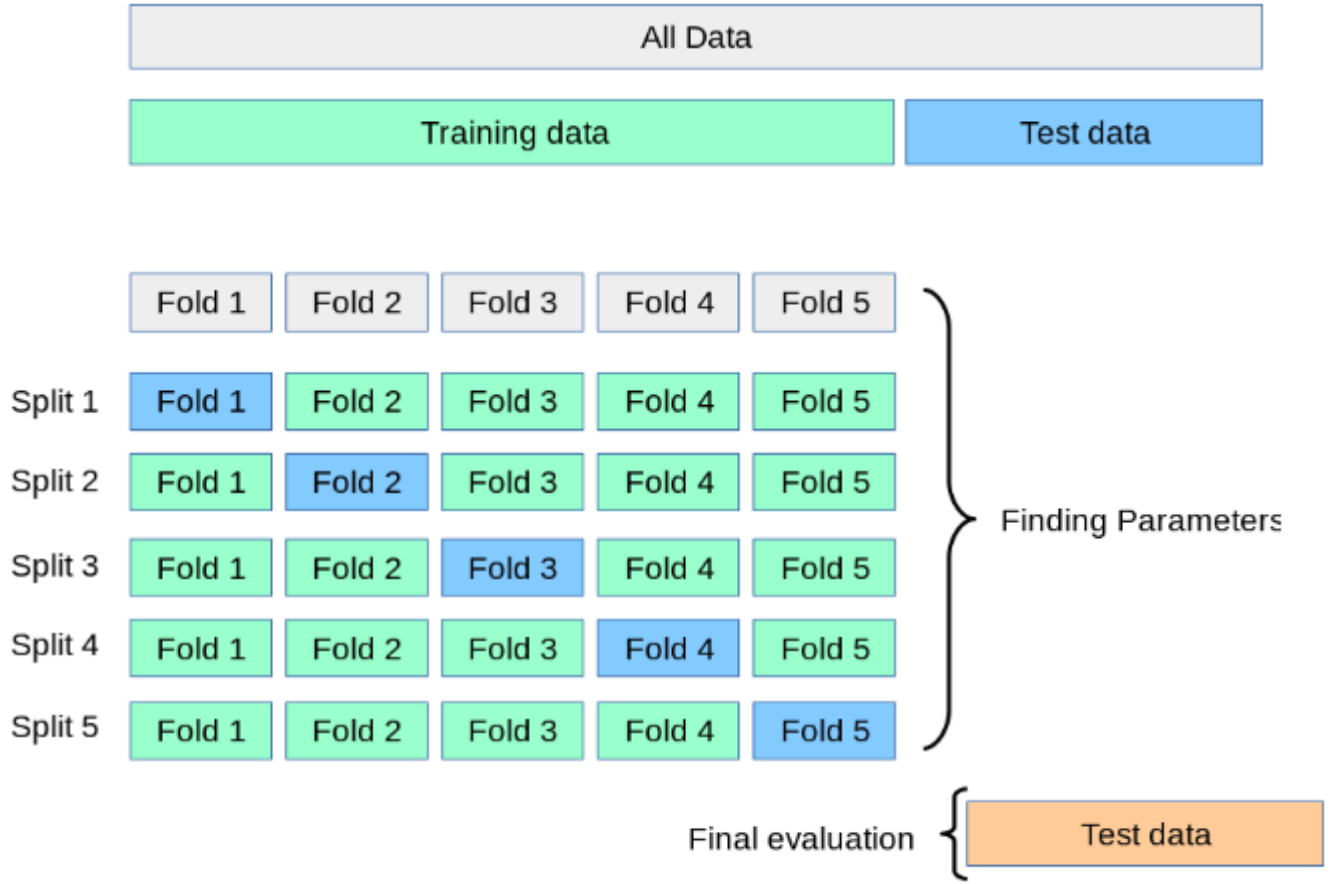
- Training, Validation and Testing Data
  - Less data for your training model
- Ideally use the holdout data only once
  - **Requires a great deal of discipline** to not look at the holdout data



Q: How do we select the data that goes into training, validation and testing? Can that affect performance?

Cross-Validation

- Splitting training and validation data into several folds during training
- This is known as **k-fold Cross-Validation**
- Model parameters selected based on average achieved over *k* folds



Source: scikit-learn

Data Processing

- Q: You test your model on new data and you find it fails to predict certain samples. Why could be happening?

Training Data

Test Data

Data Augmentation

- For example, how can your algorithms (models) predict on rotations if it has never seen a rotated sample?

- Apply Data Augmentation!
  - translation,
  - scaling,
  - rotation,
  - reflection,
  - ...

Linear Algebra to the Rescue!

Source: https://morioh.com/p/928228425a08

More Data Processing

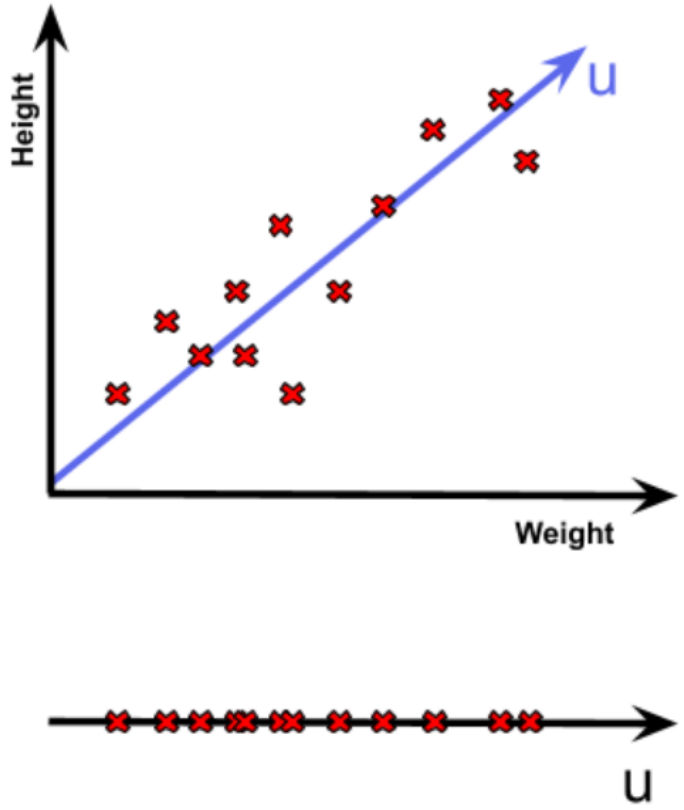
- Q: Large input feature size (short and wide data) is problematic? Why do you think that is?

- **Curse of dimensionality**!
  - As features grow you require more model capacity (complexity) to represent the data
  - Models of greater complexity require exponentially more training data

Dimensionality Reduction

Solution:

- Reduce the number of features using dimensionality reduction
  - Principal Component Analysis
  - more details provided in weeks 7 and 8

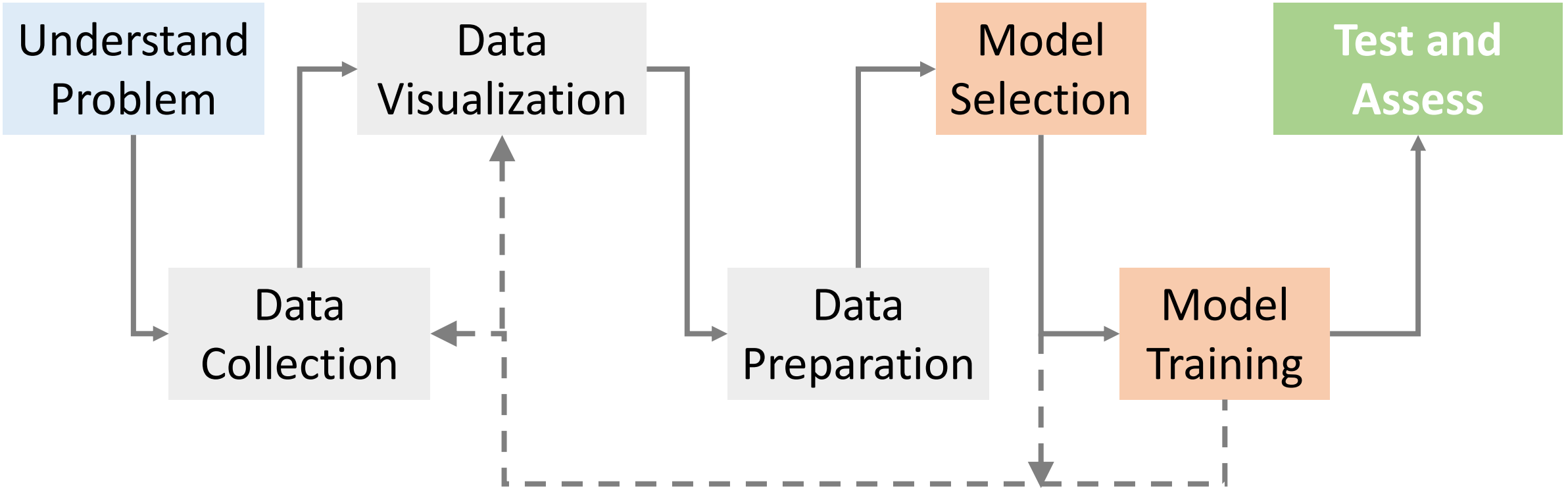


Source: Data Courses

Deep Learning

- Principle Component Analysis (PCA) is limited to **linear transformations**
- Deep Learning techniques can be used to learn and apply **nonlinear transformations** for dimensionality reduction
  - More detail on model-based machine learning techniques in weeks 9 – 11

Roadmap for the rest of APS1070



End-to-end machine learning is just one piece of the pie. The concepts we'll cover in this course have utility that goes far beyond machine learning.

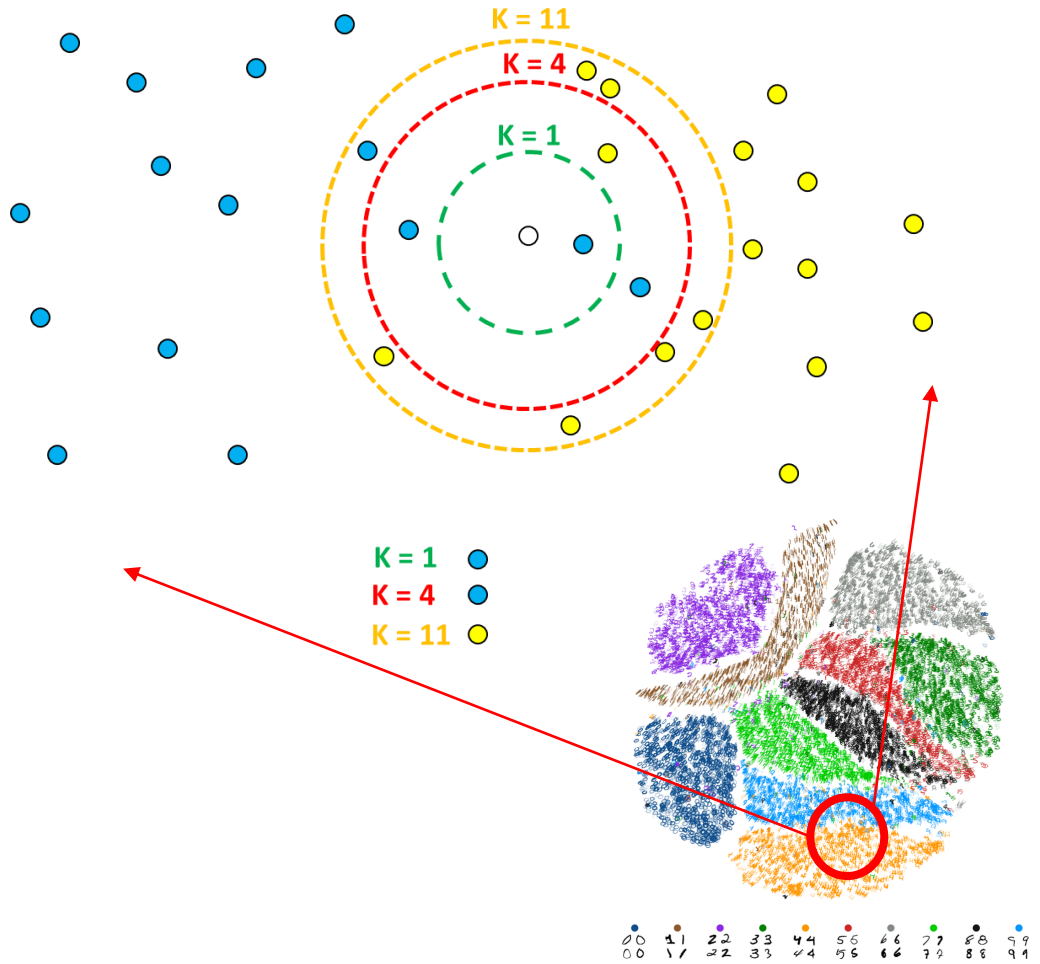
Titanic Sample Code

1. End-to-end machine learning
2. Python Libraries
   - NumPy
   - Matlplotlib
   - Pandas
   - Scikit-Learn

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



Instance-Based vs Model-Based Learning

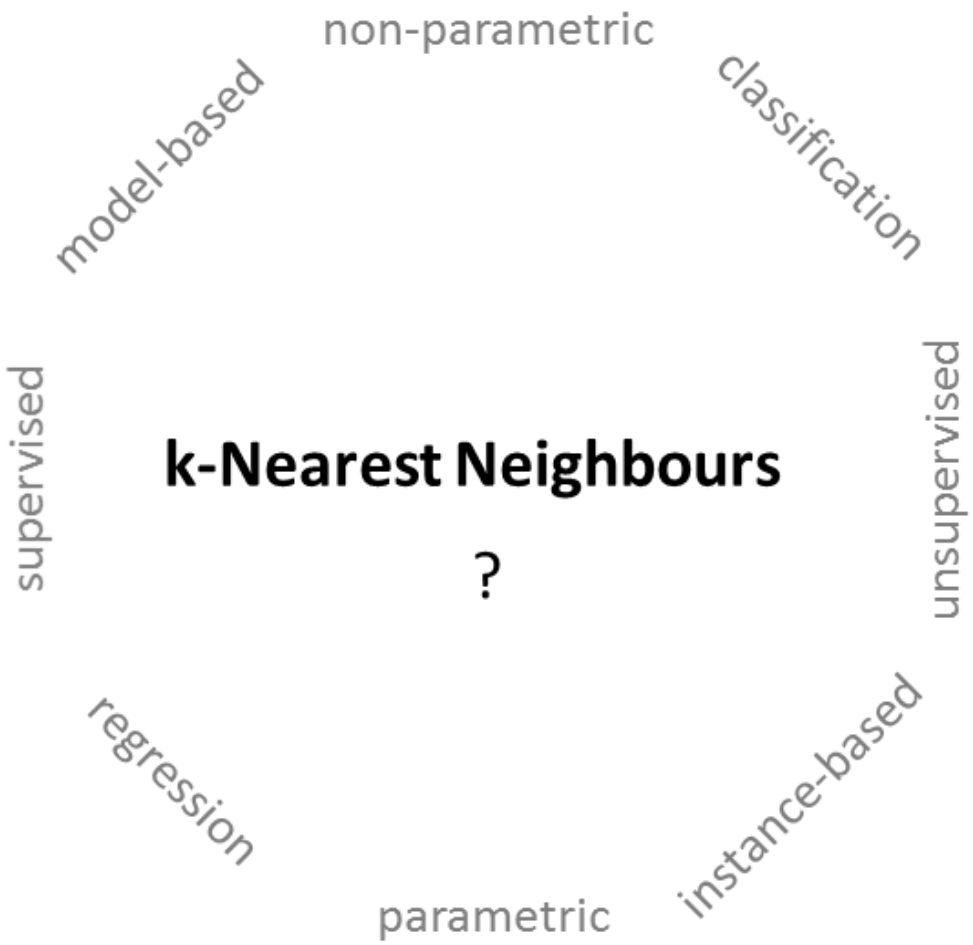
- **Instance-Based:**
  - system learns the examples by heart, then generalizes to new cases by using a similarity/distance measure to compare them to the learned examples.

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

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.

Example: kNNs Summary



Q: How can you use kNNs for Regression Problems?

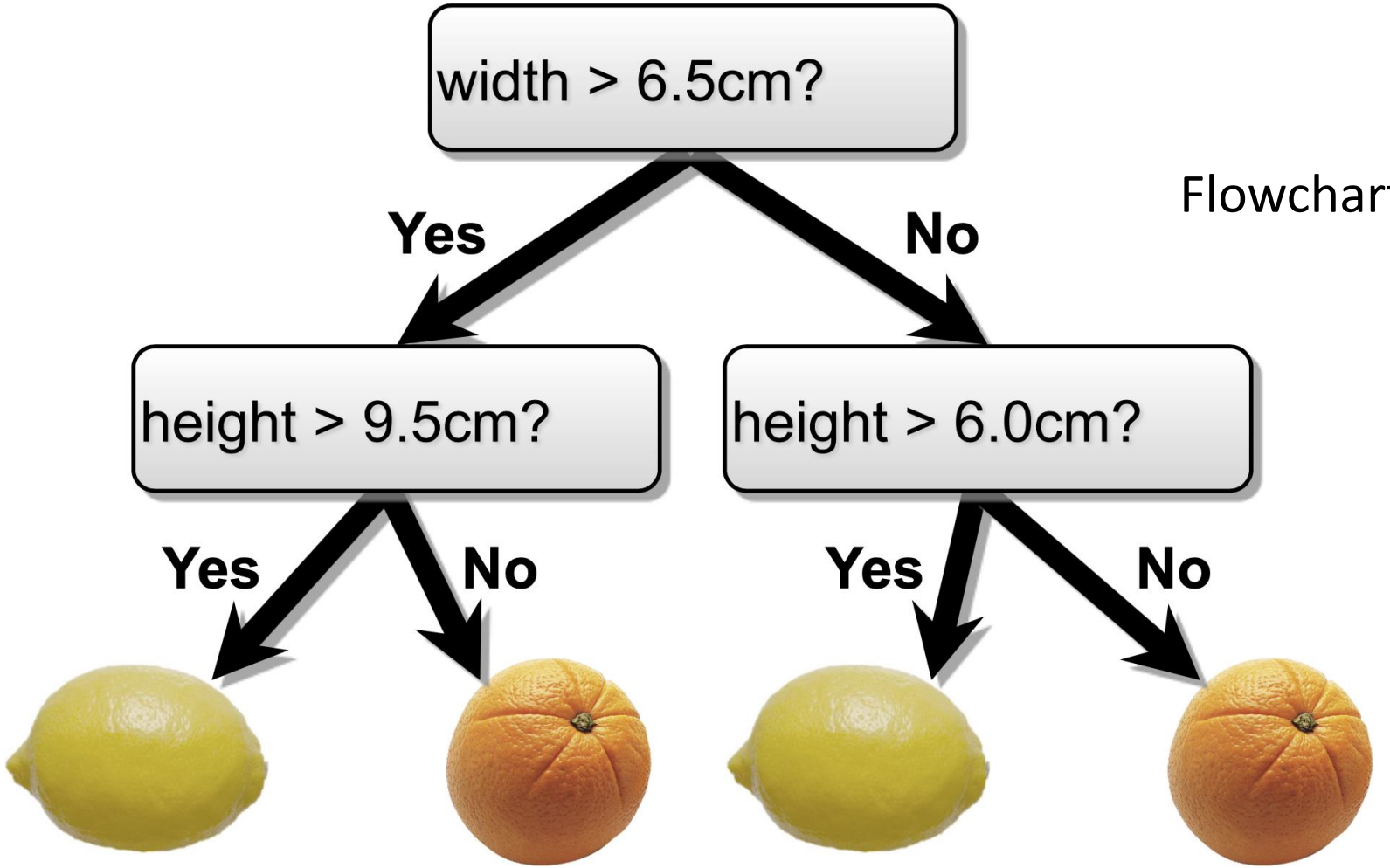
Decision Trees (Supervised Learning)

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!

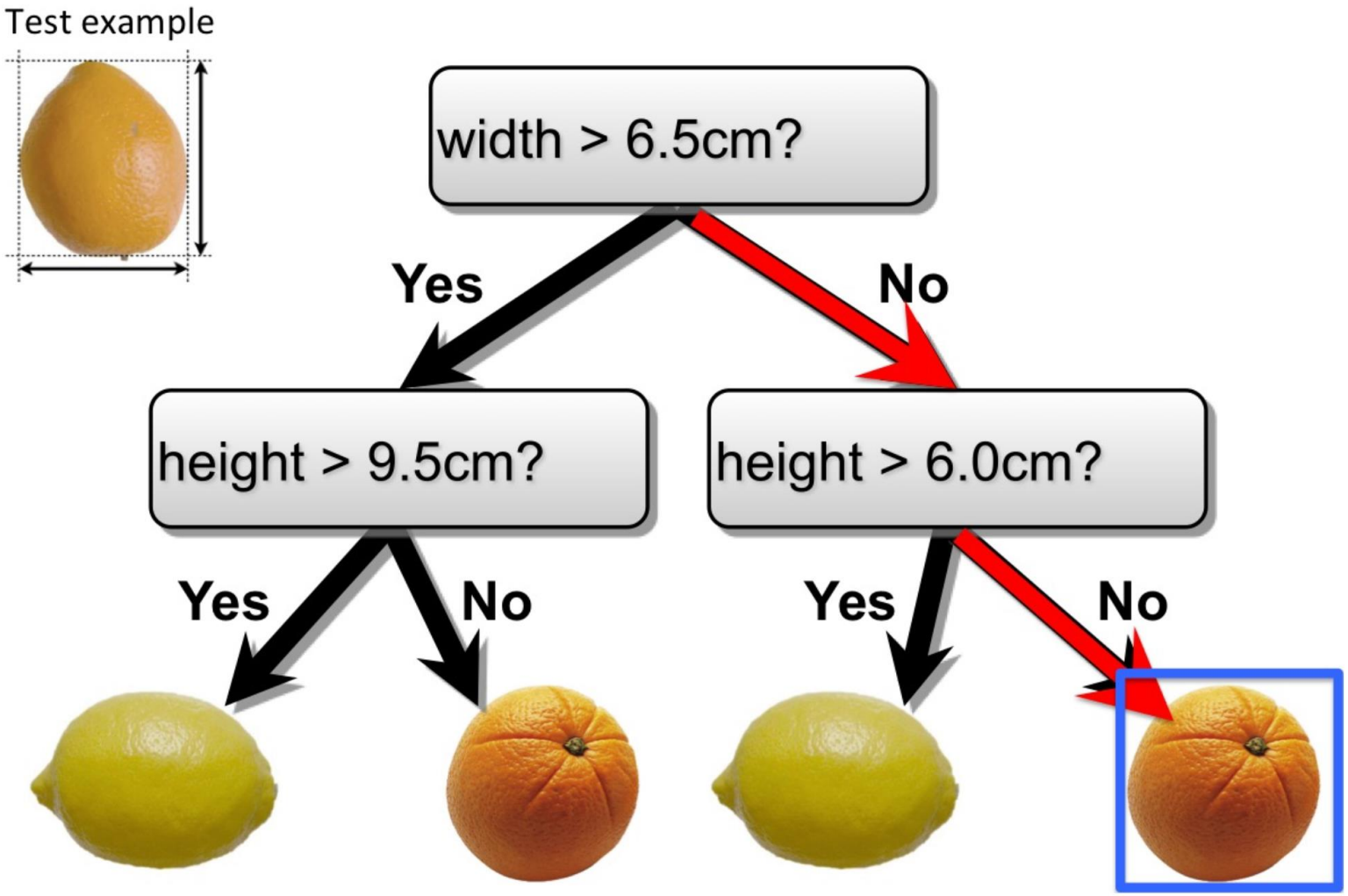
- A fundamental component of Random Forests which are one of the most used Machine Learning algorithms today

Lemon Vs. Orange!



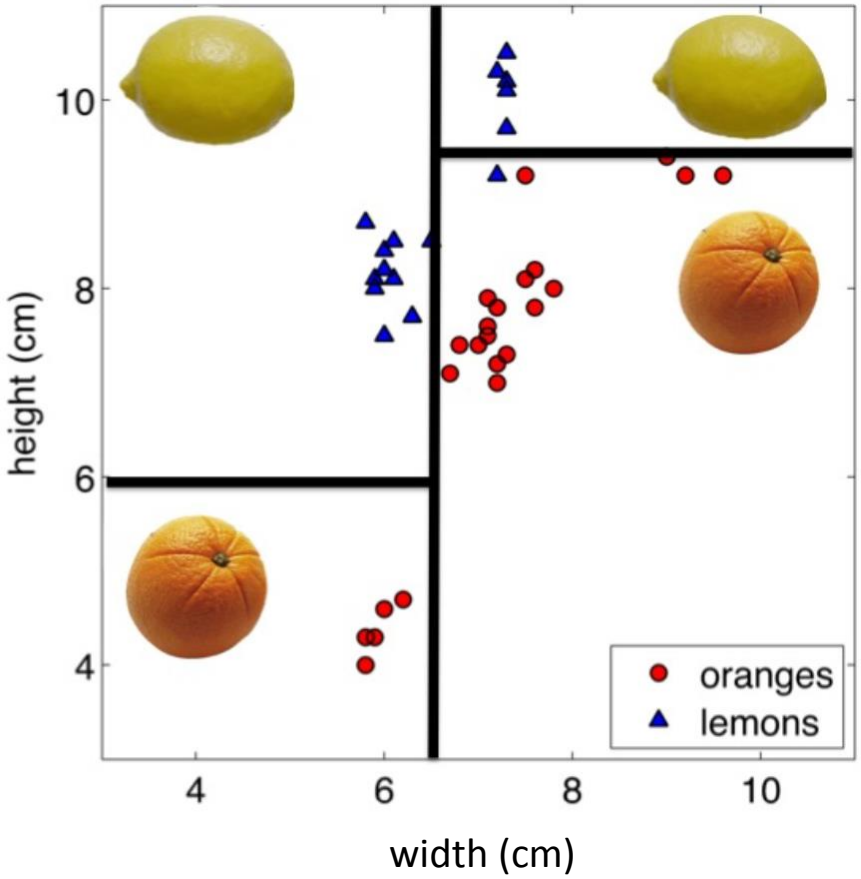
Flowchart-like structure!

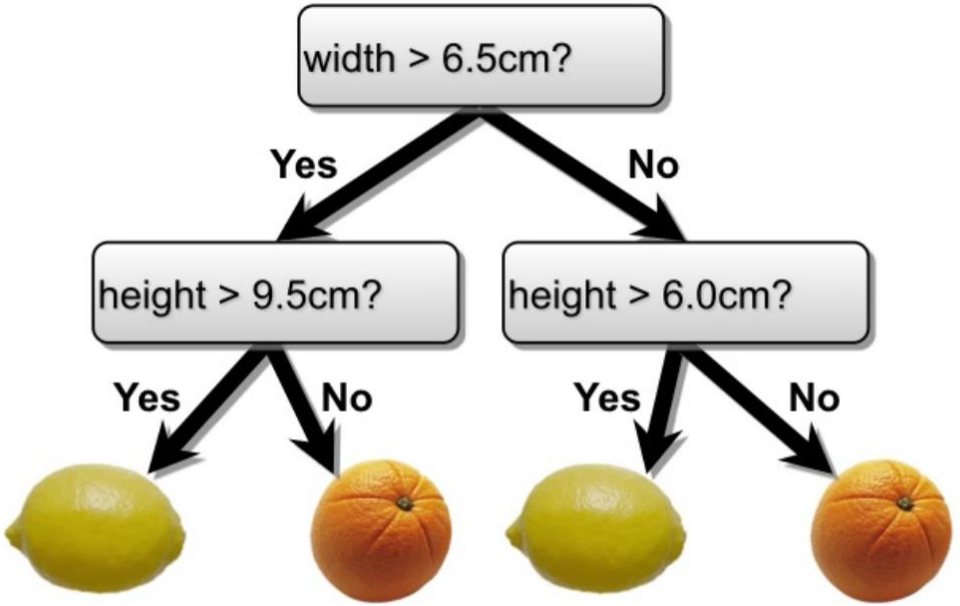
Test example



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).

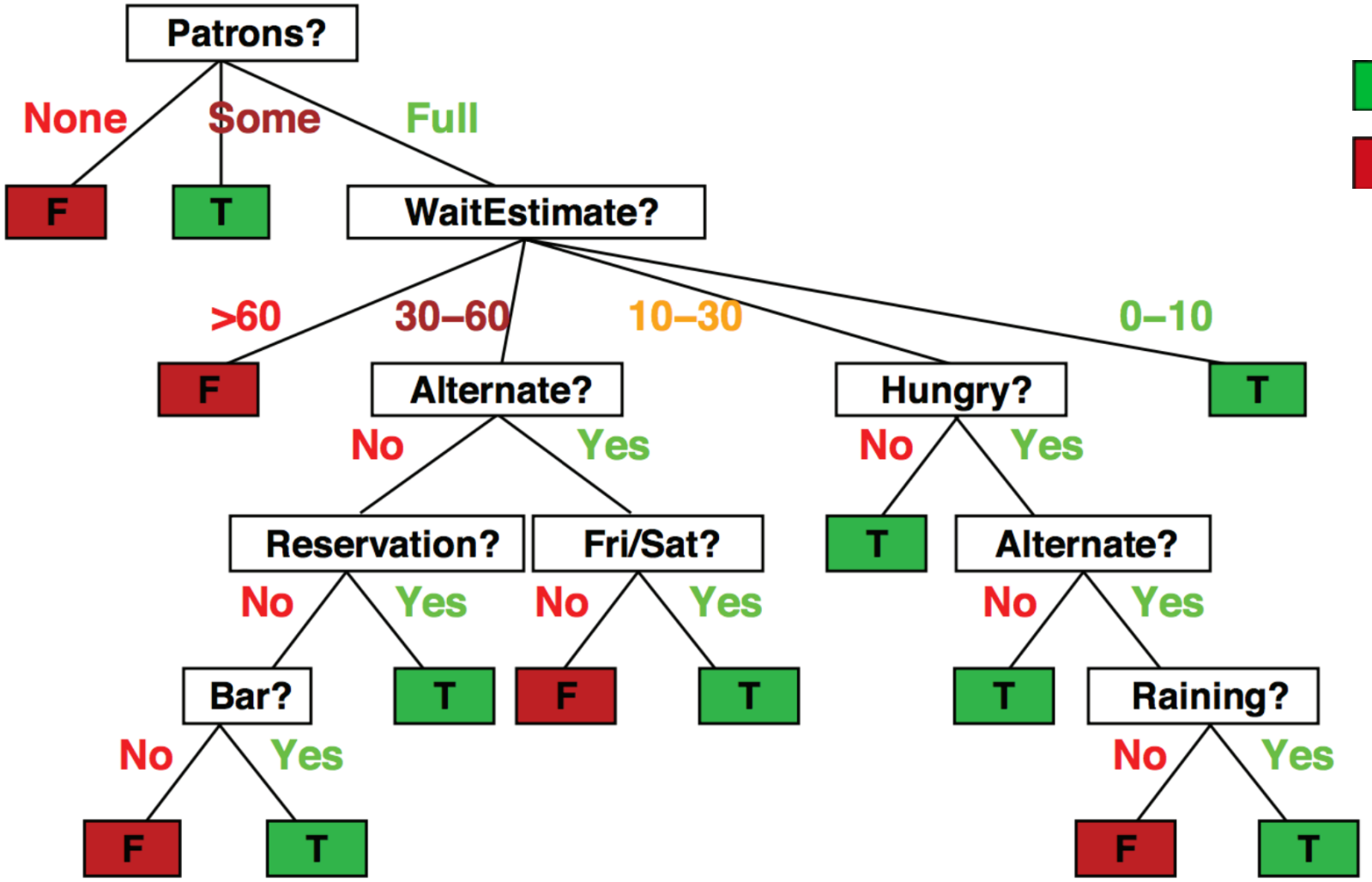
What if the attributes are discrete?

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).

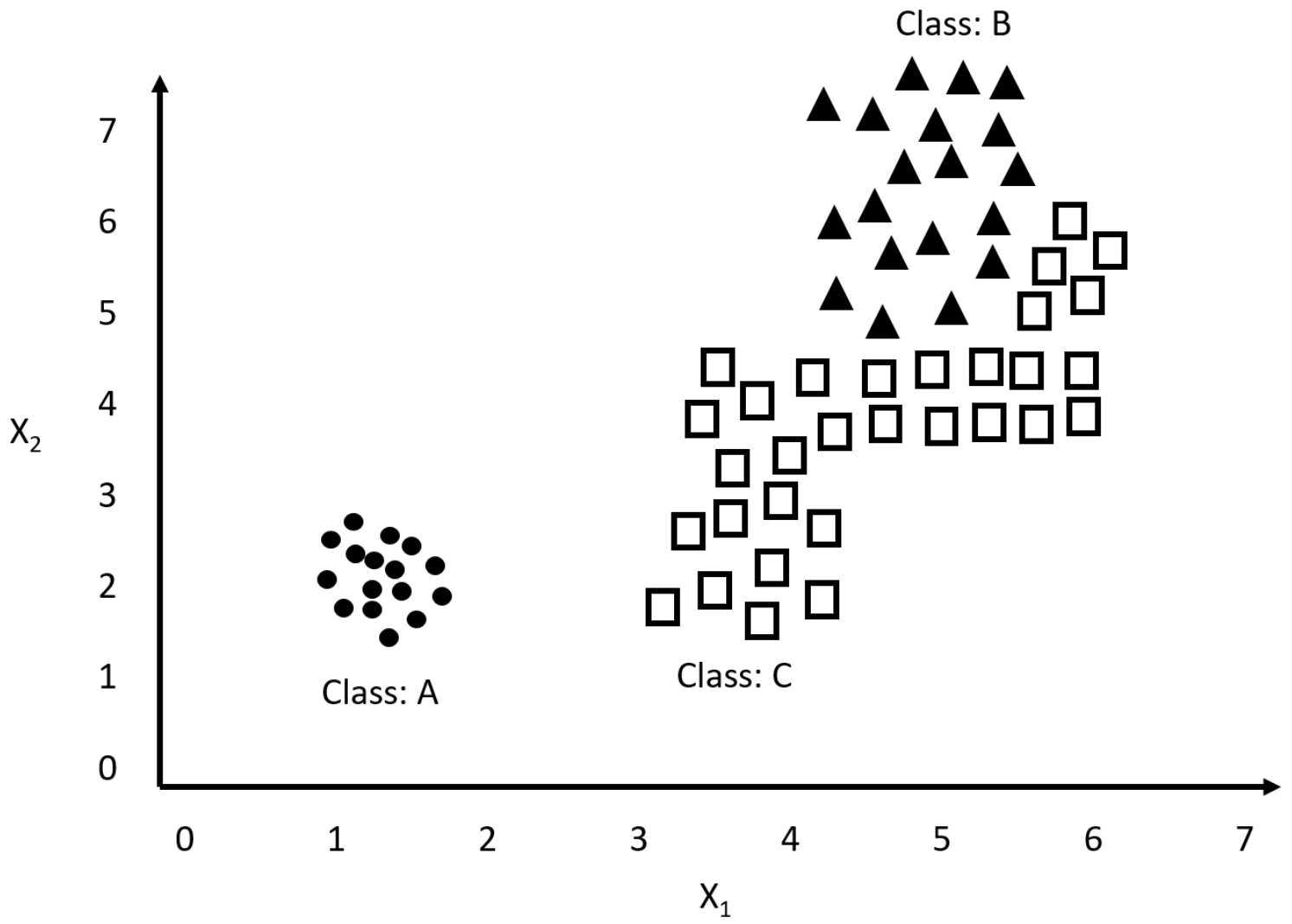
Output is Discrete



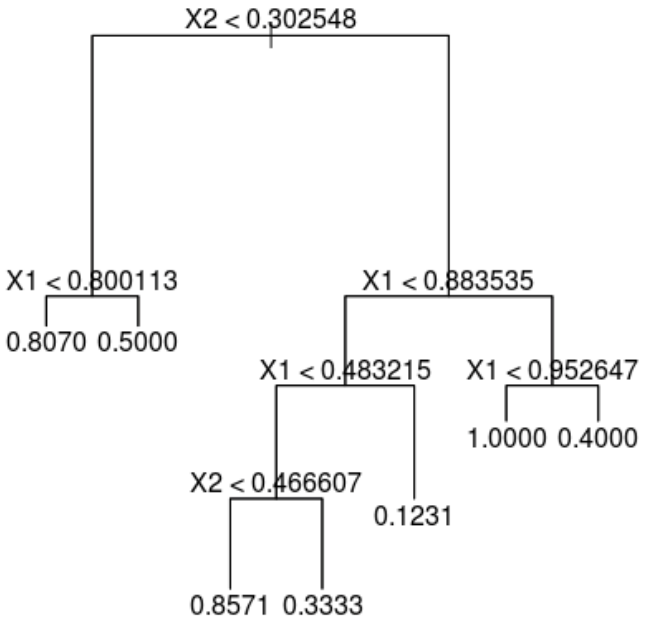
T — Wait for table

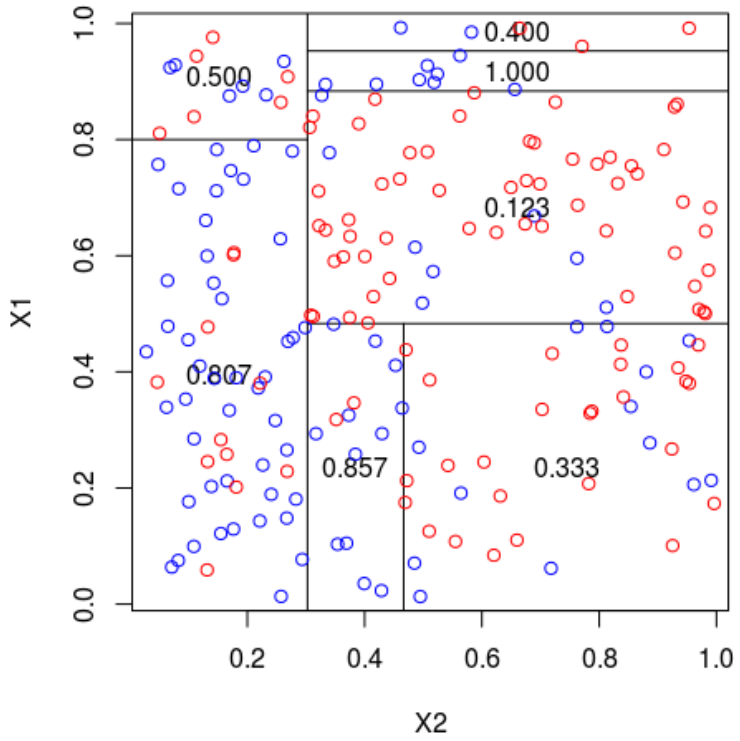
F — Go somewhere else

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!

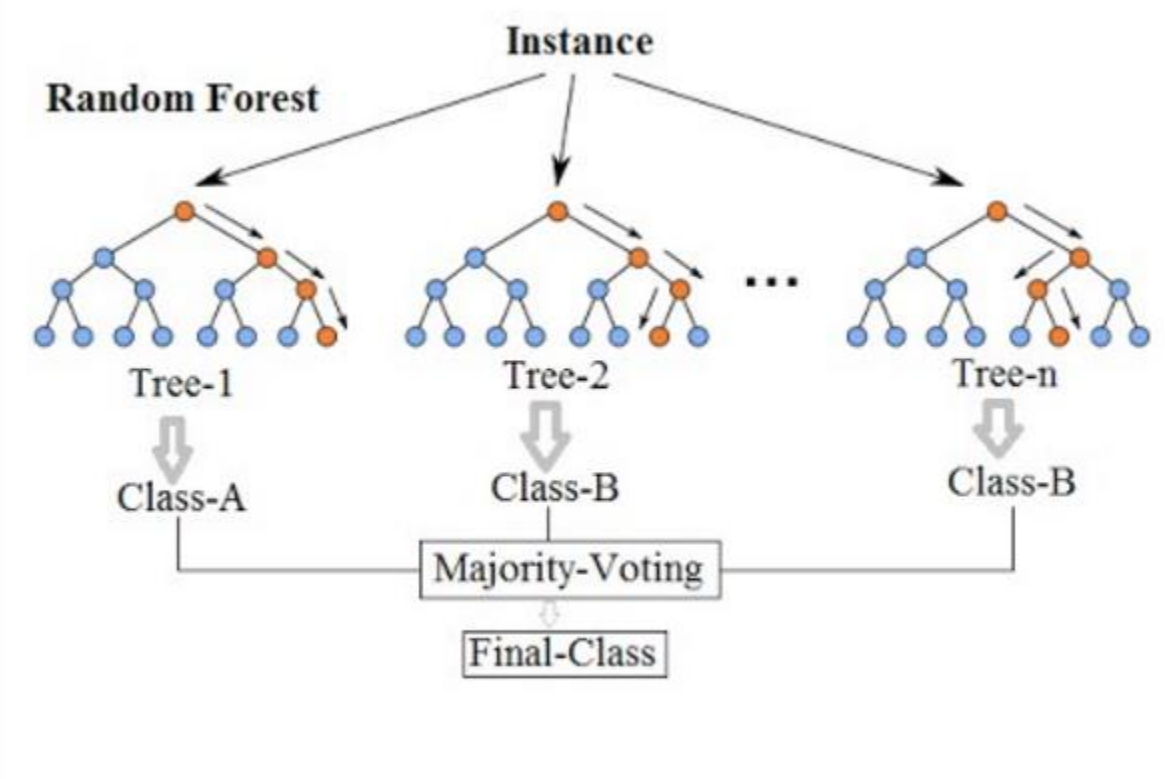
- Need some way to prune the tree!

Managing Overfitting

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

Random Forests

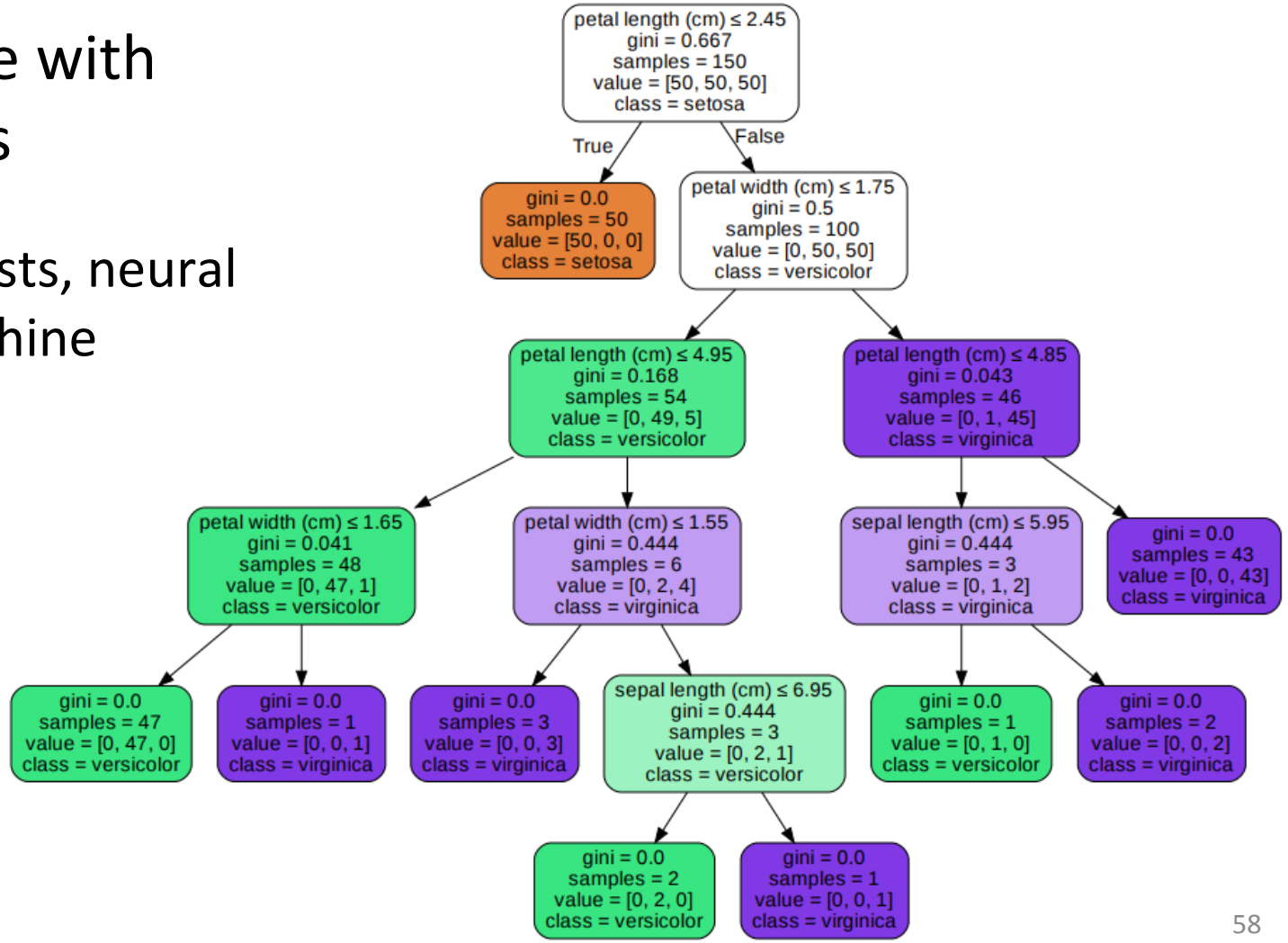
- One of the most popular variants of decision trees
- Addresses overfitting by training multiple trees on subsampling of features among other things
- Majority vote of all the trees is used to make the final output



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

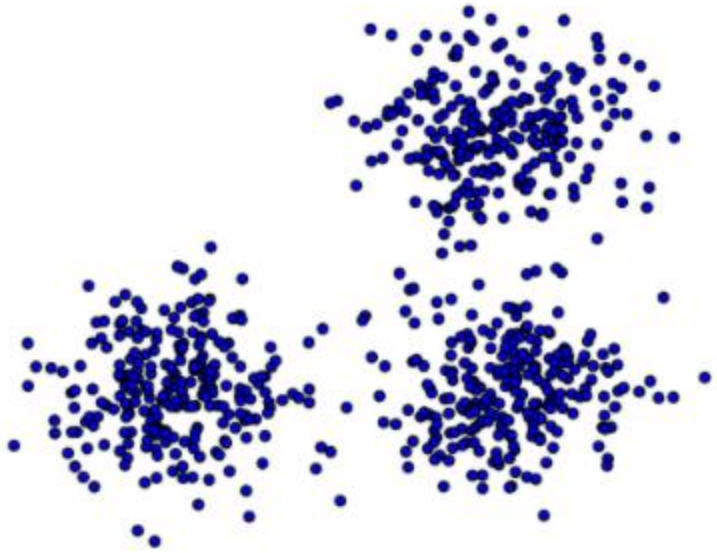
- There are many advantages of Decision Trees over k-Nearest Neighbours:
  - Good with discrete attributes
  - Robust to scale of inputs (does not require normalization)
  - Easily handle missing values
  - Good at handling lots of attributes, especially when only a few are important
  - Fast test time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

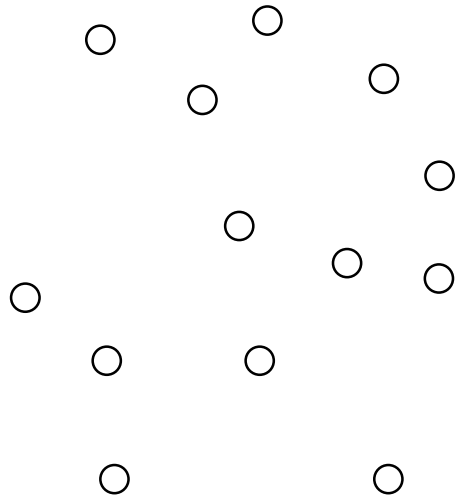
- Most well-known clustering method
- Distance-based **unsupervised** learning algorithm, NOT to be confused with k-NN.

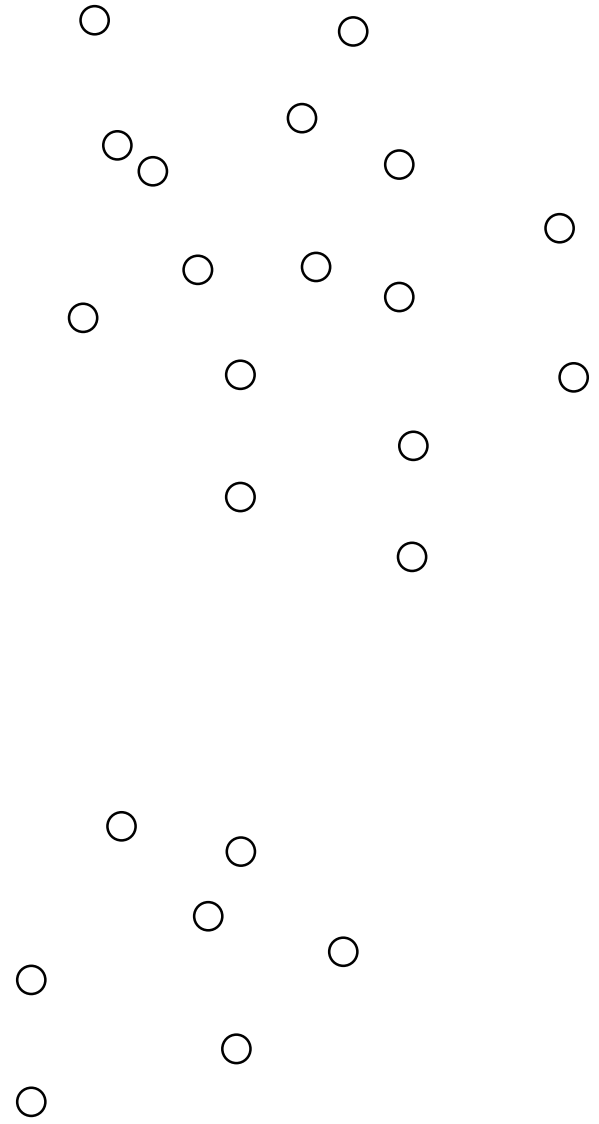
- **Algorithm:**
  1. Assign each sample/instance to its closest mean
  2. Update the means based on the assignment
  3. Repeat until convergence

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

K-Means Example

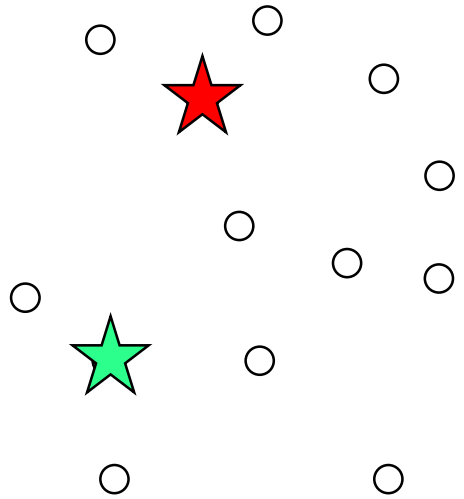
- K is number of clusters
- STEP 1: Guess center locations
- STEP 2: Map out what data point is closest to what center
- STEP 3: Center moves to the centroid of all points it “owns”
- REPEAT!

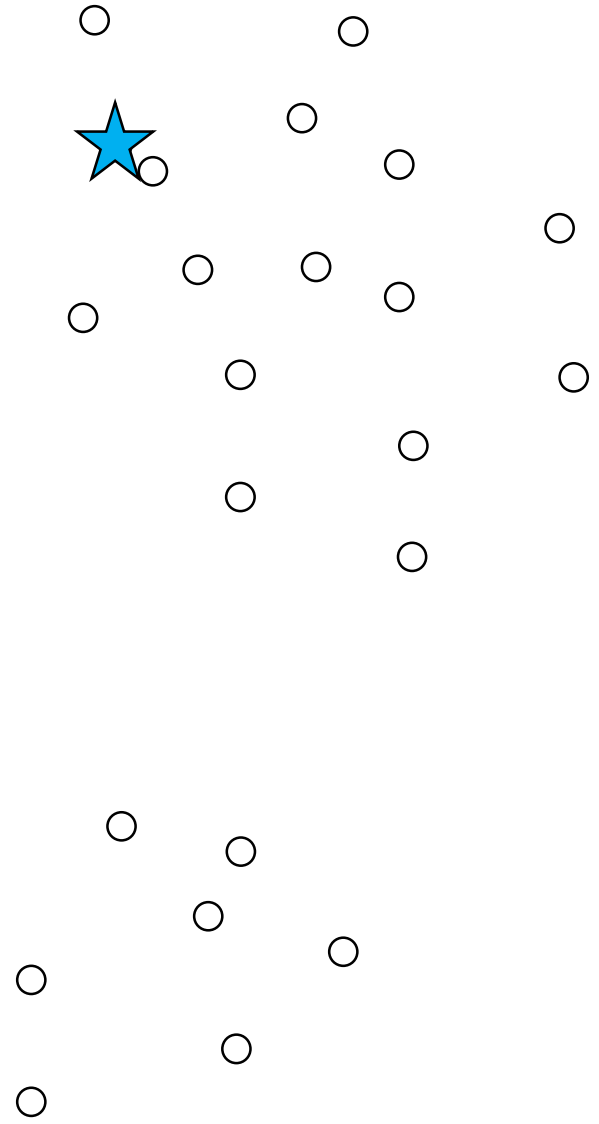




K-Means Example

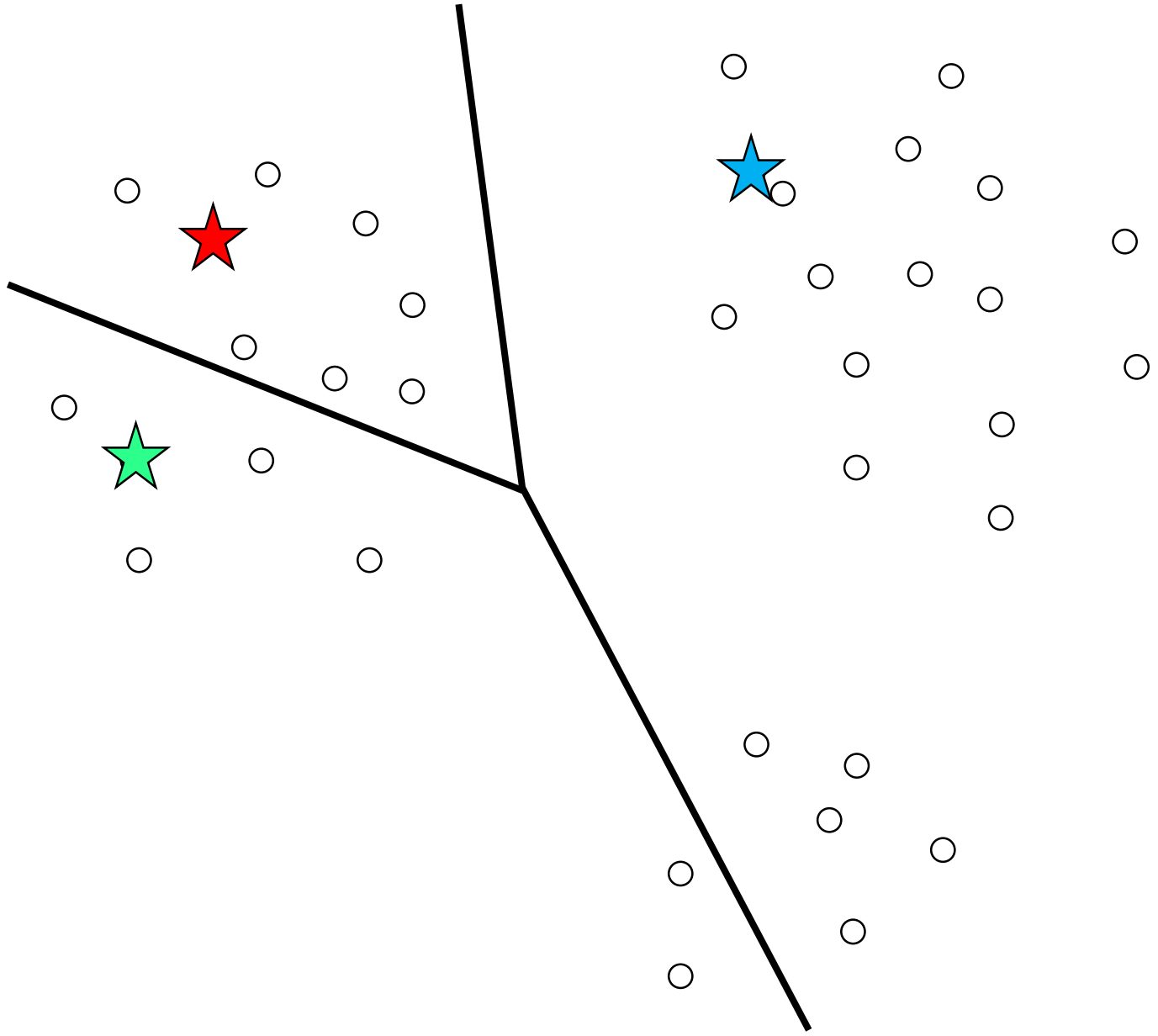
- K is number of clusters
- STEP 1: Guess center locations
- STEP 2: Map out what data point is closest to what center
- STEP 3: Center moves to the centroid of all points it “owns”
- REPEAT!





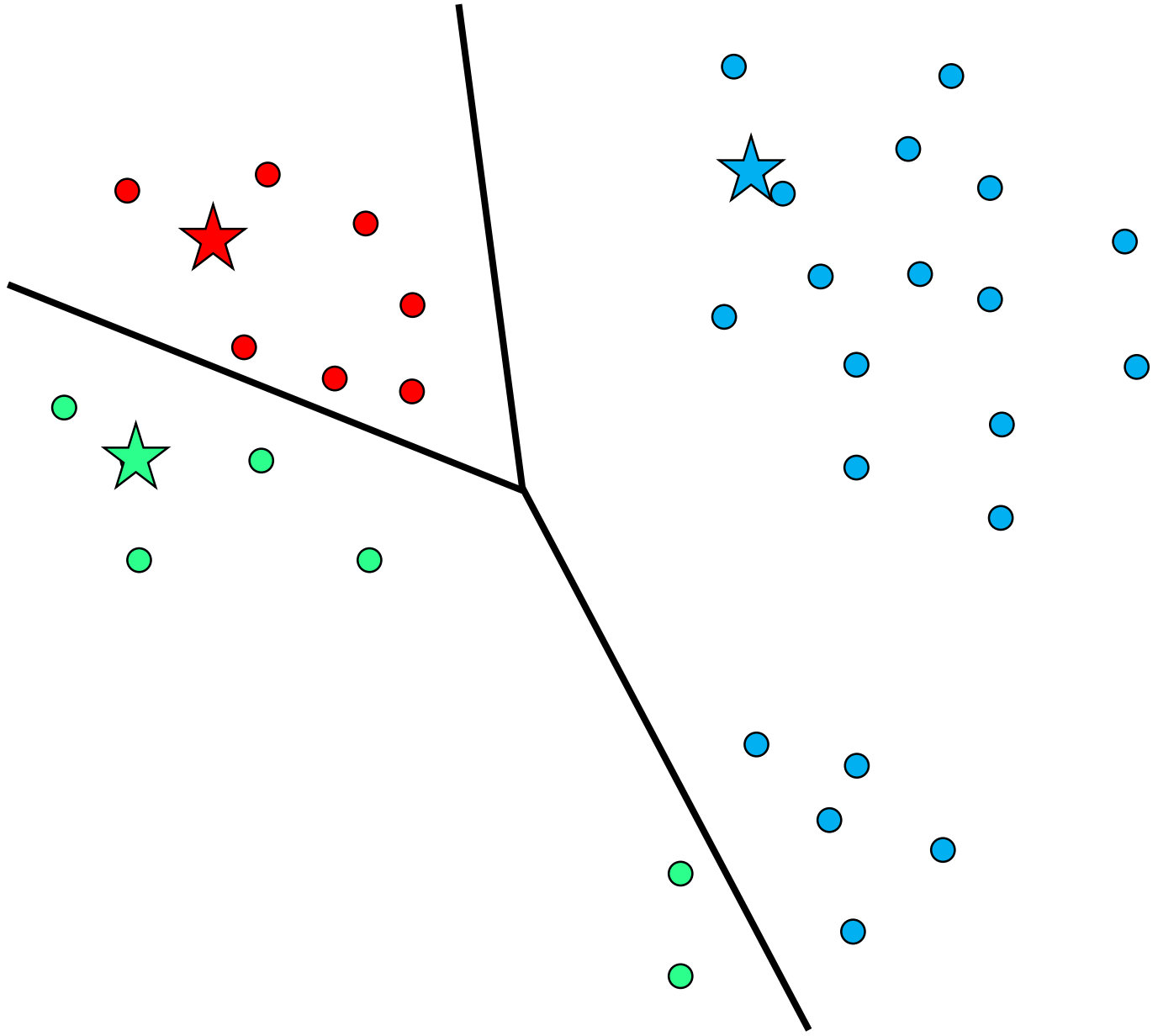
K-Means Example

- 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!



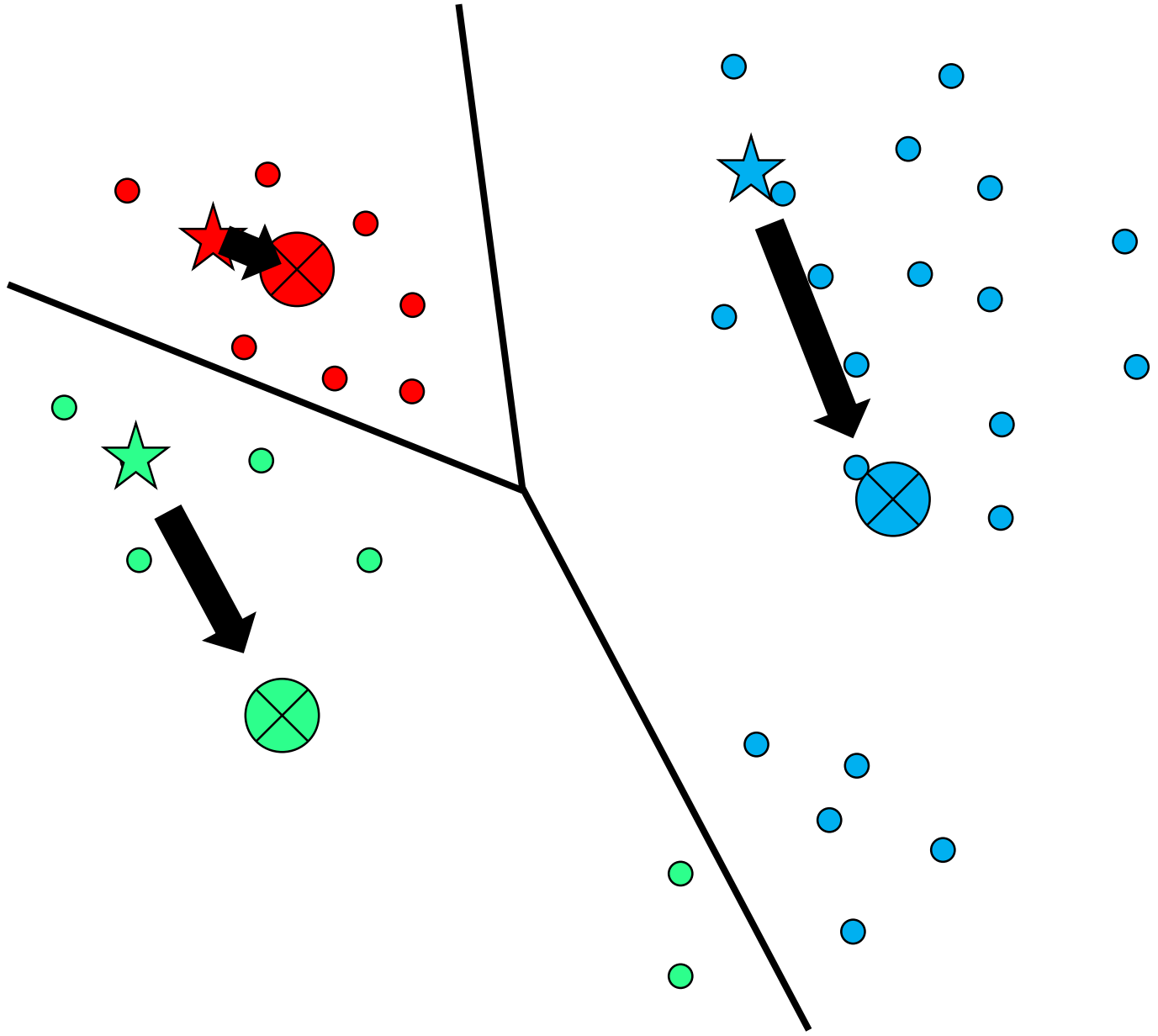
K-Means Example

- 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!



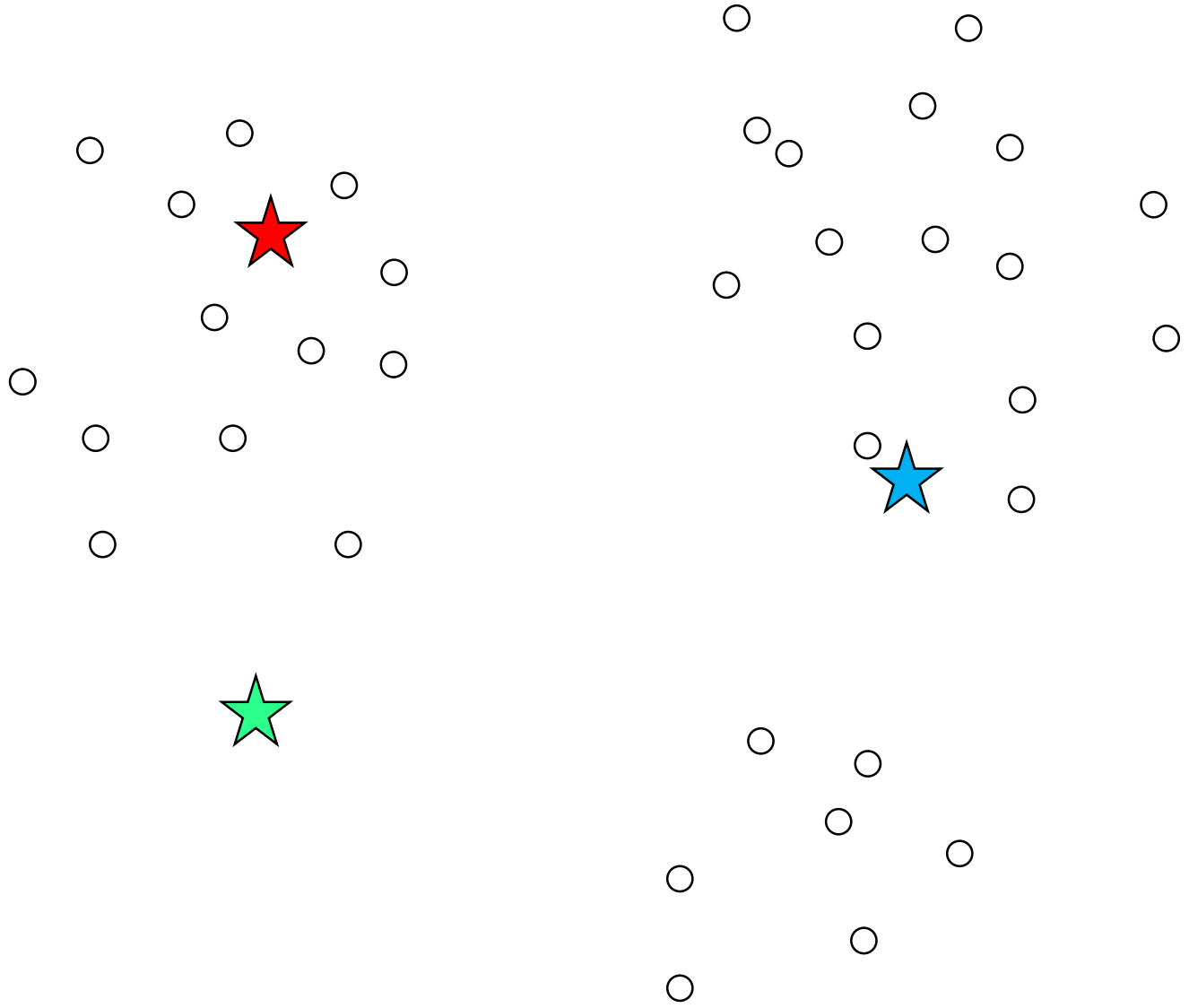
K-Means Example

- 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!



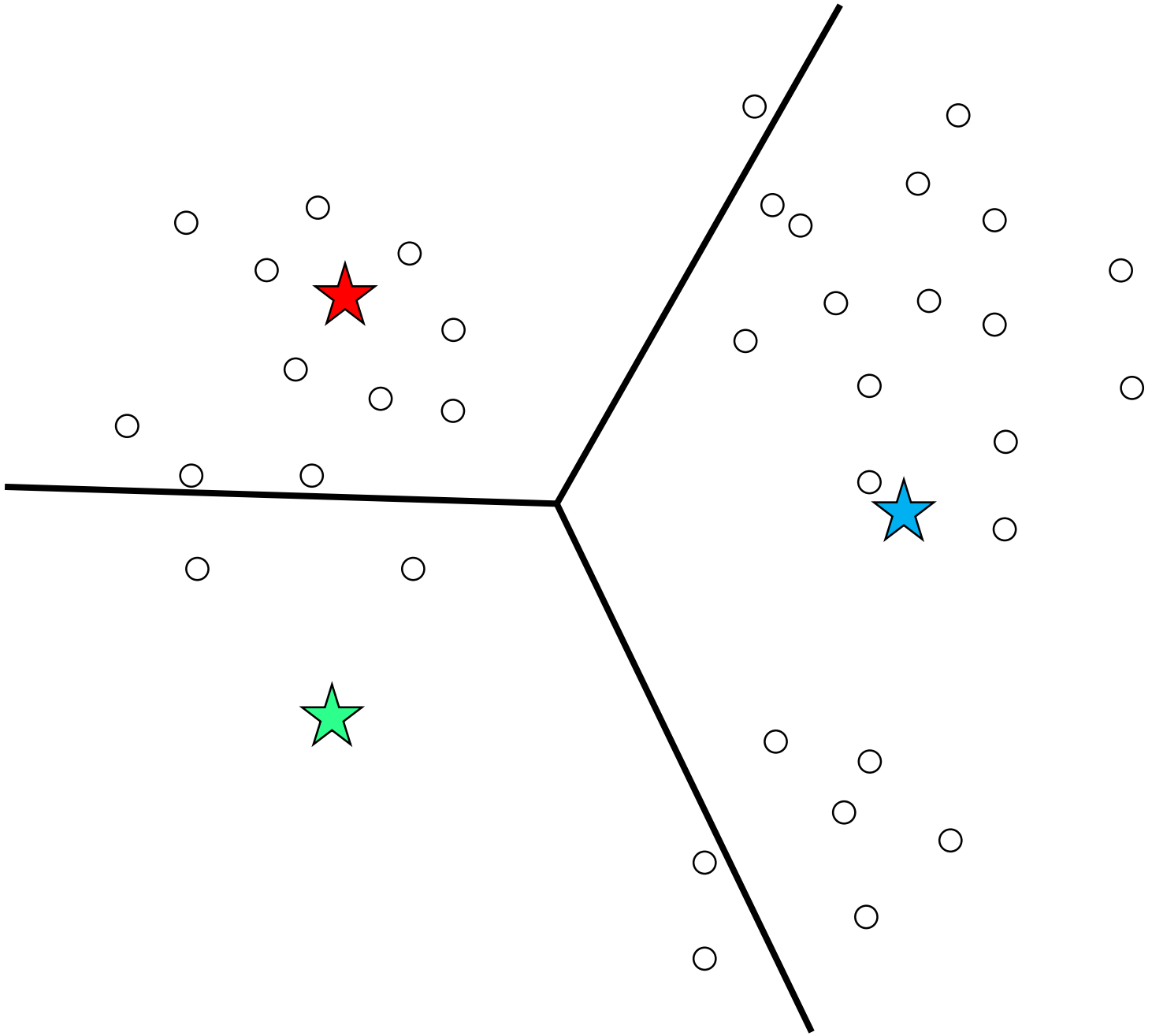
K-means Example

- 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!



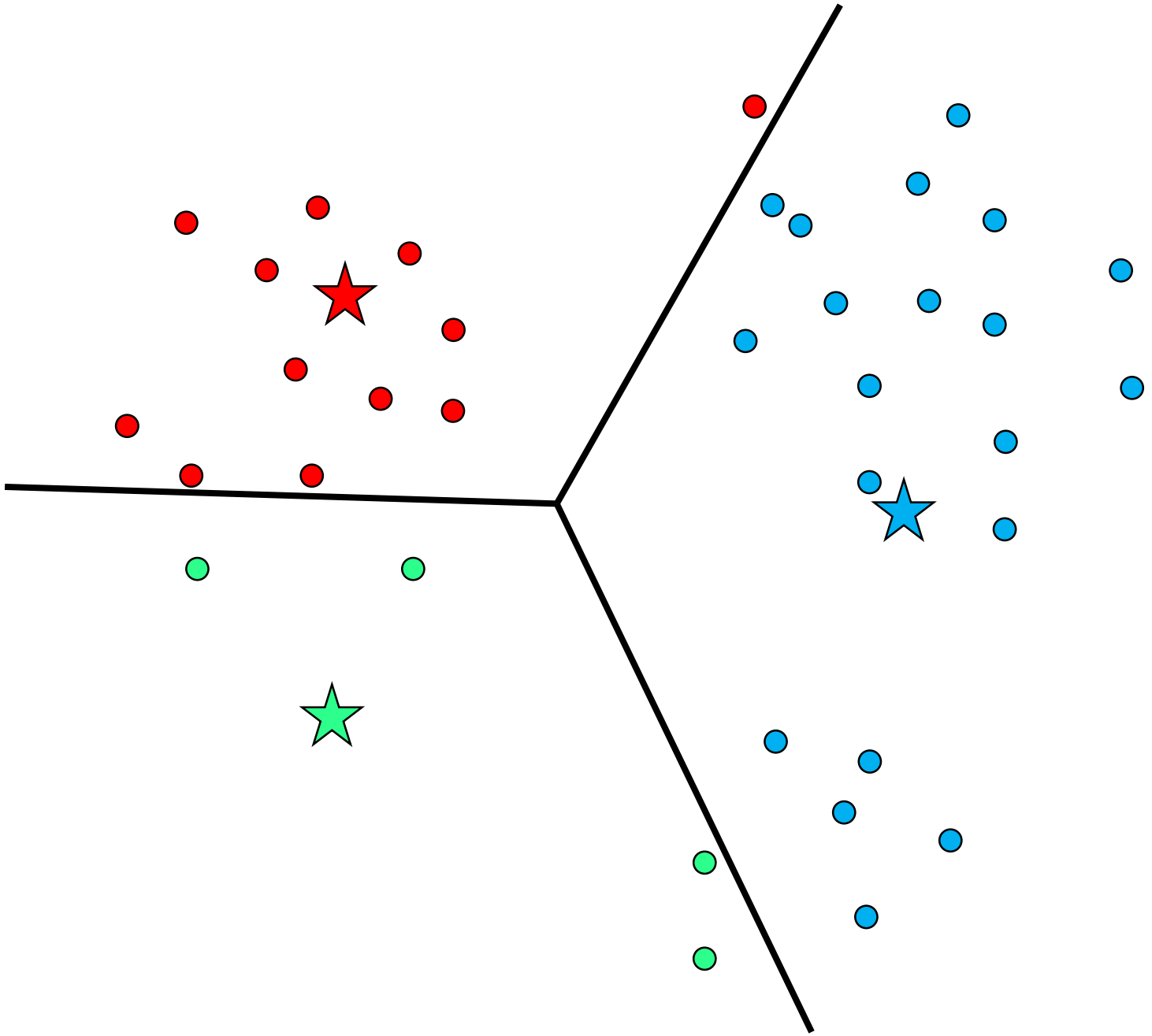
K-means Example

- 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!



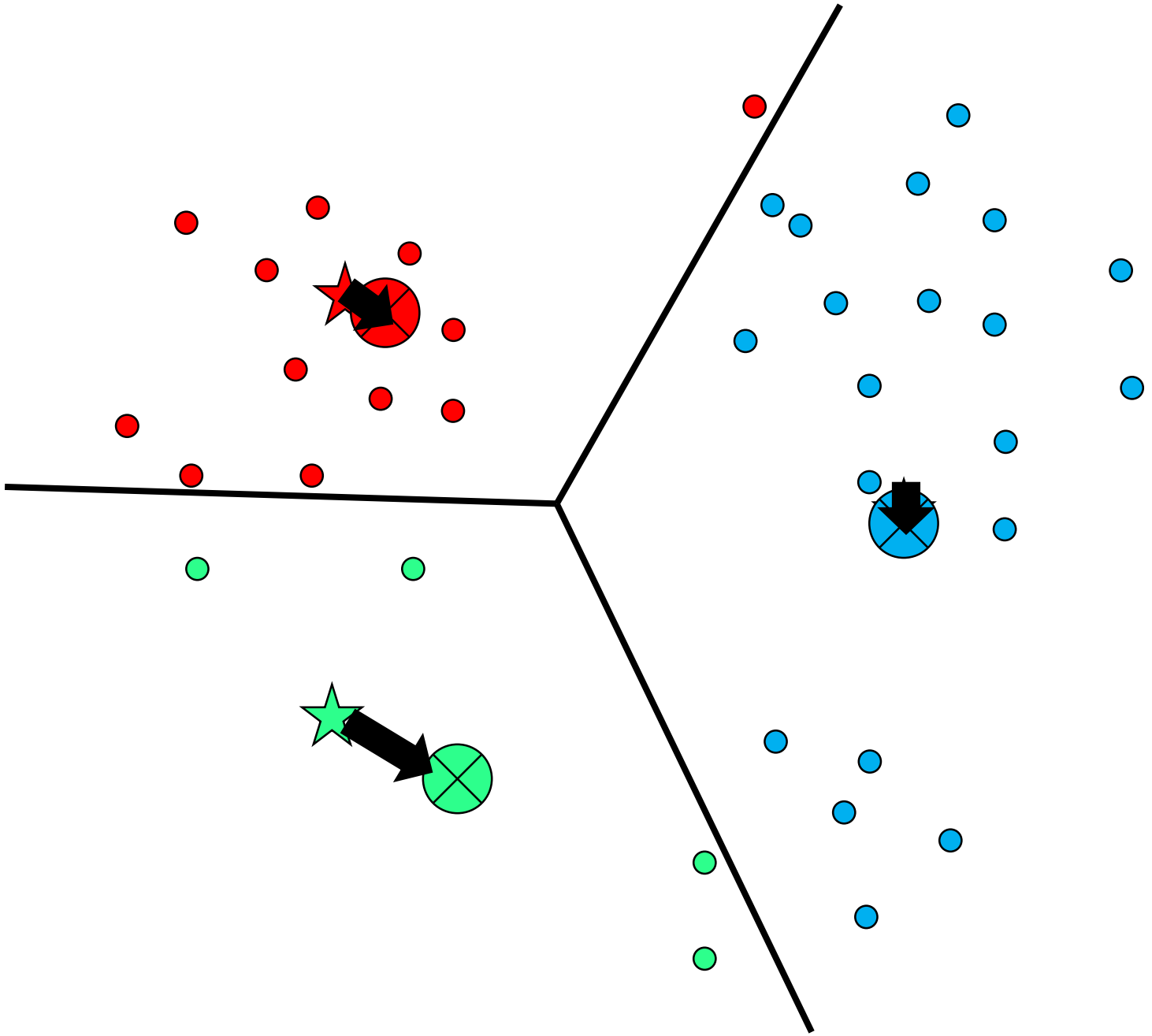
K-means Example

- 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!



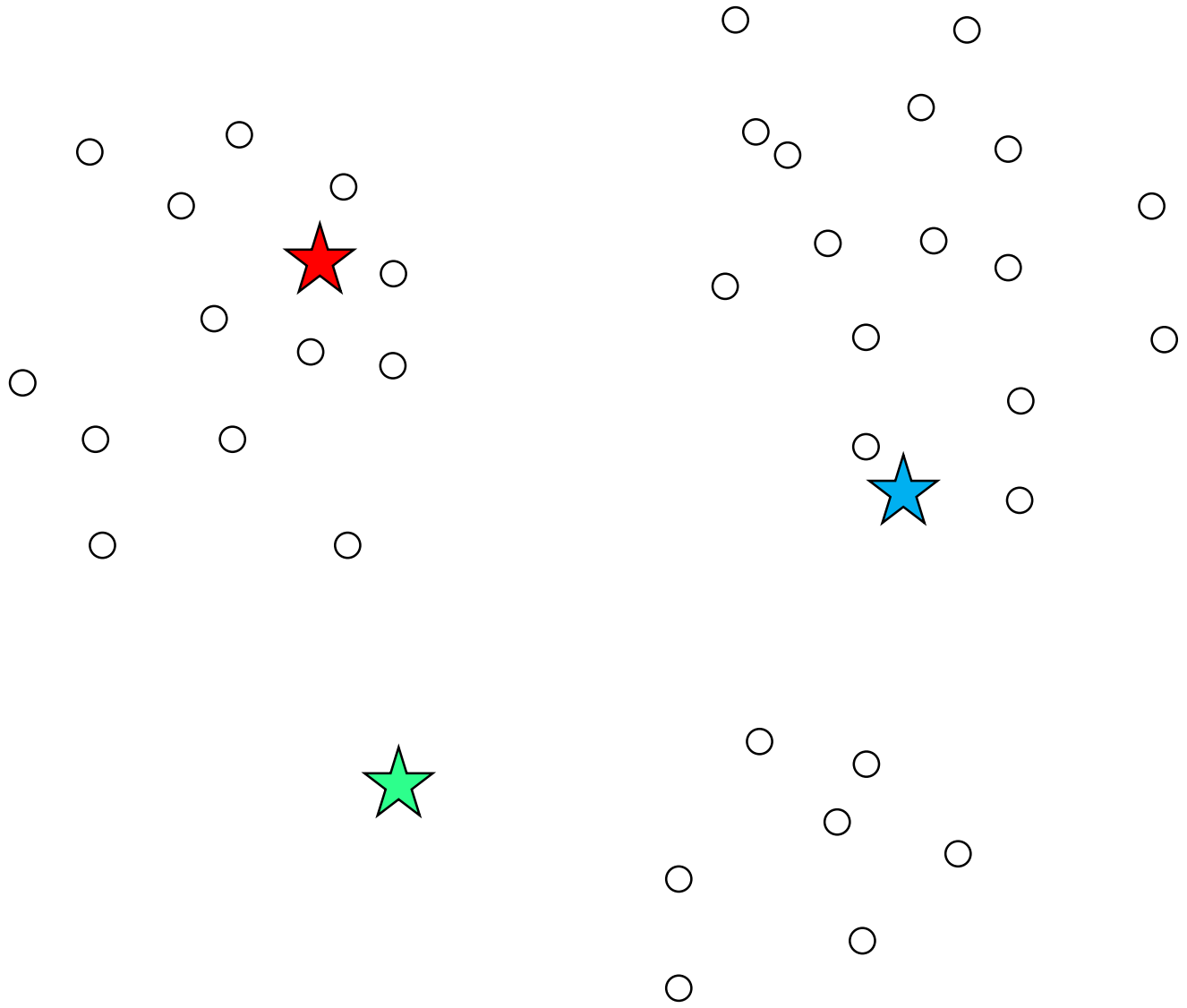
K-means Example

- 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!



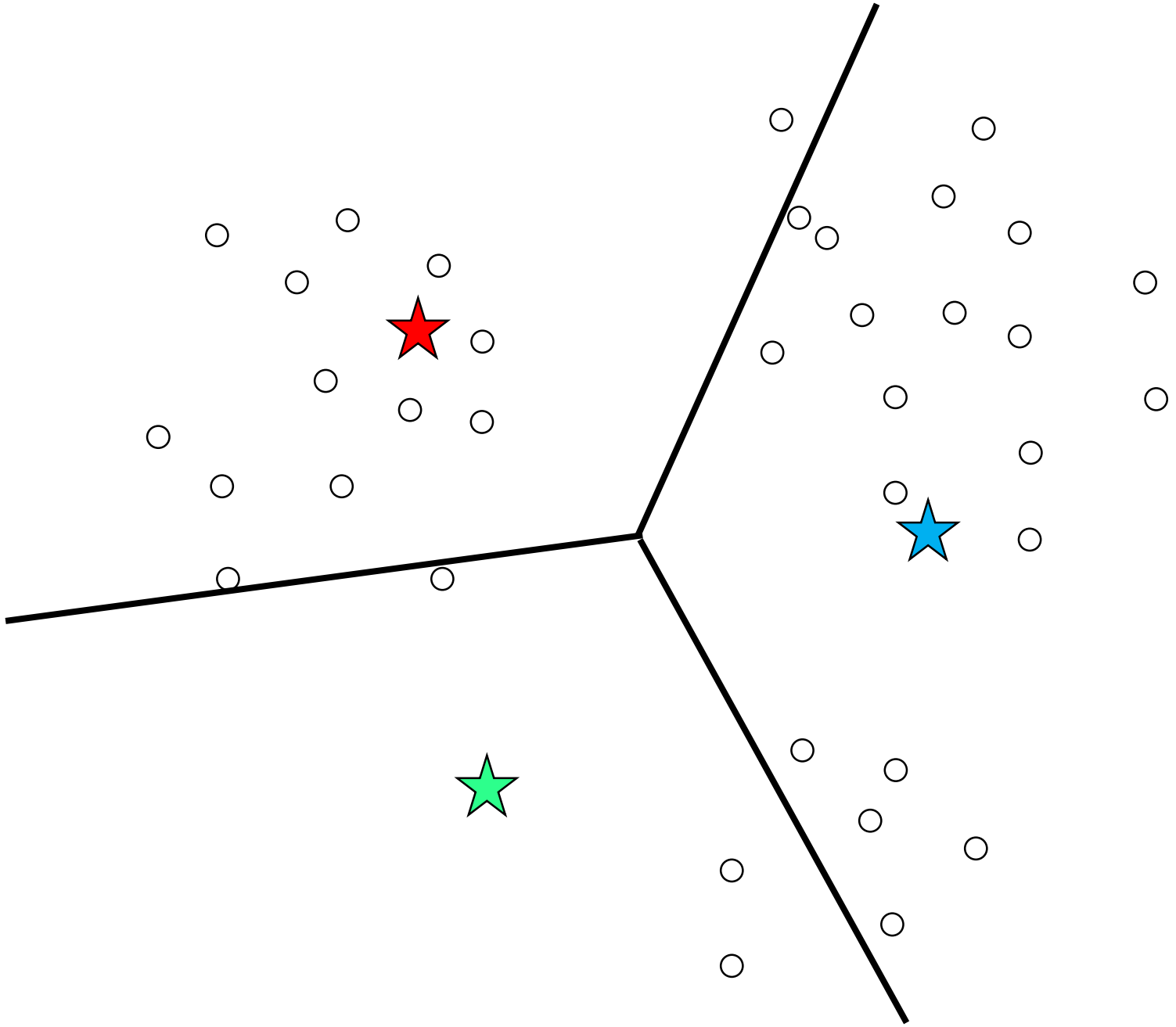
K-means Example

- 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!



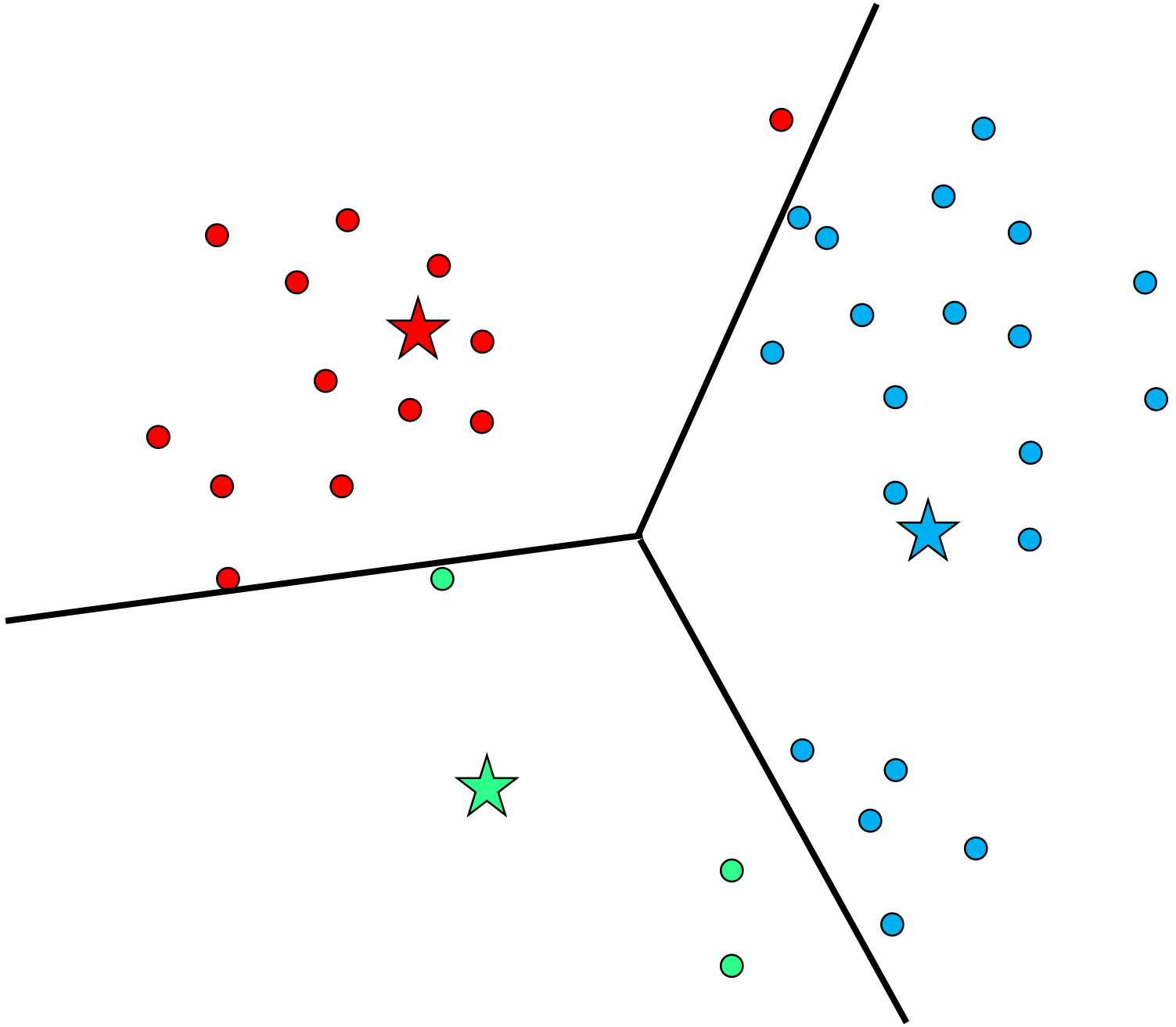
K-means Example

- 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!



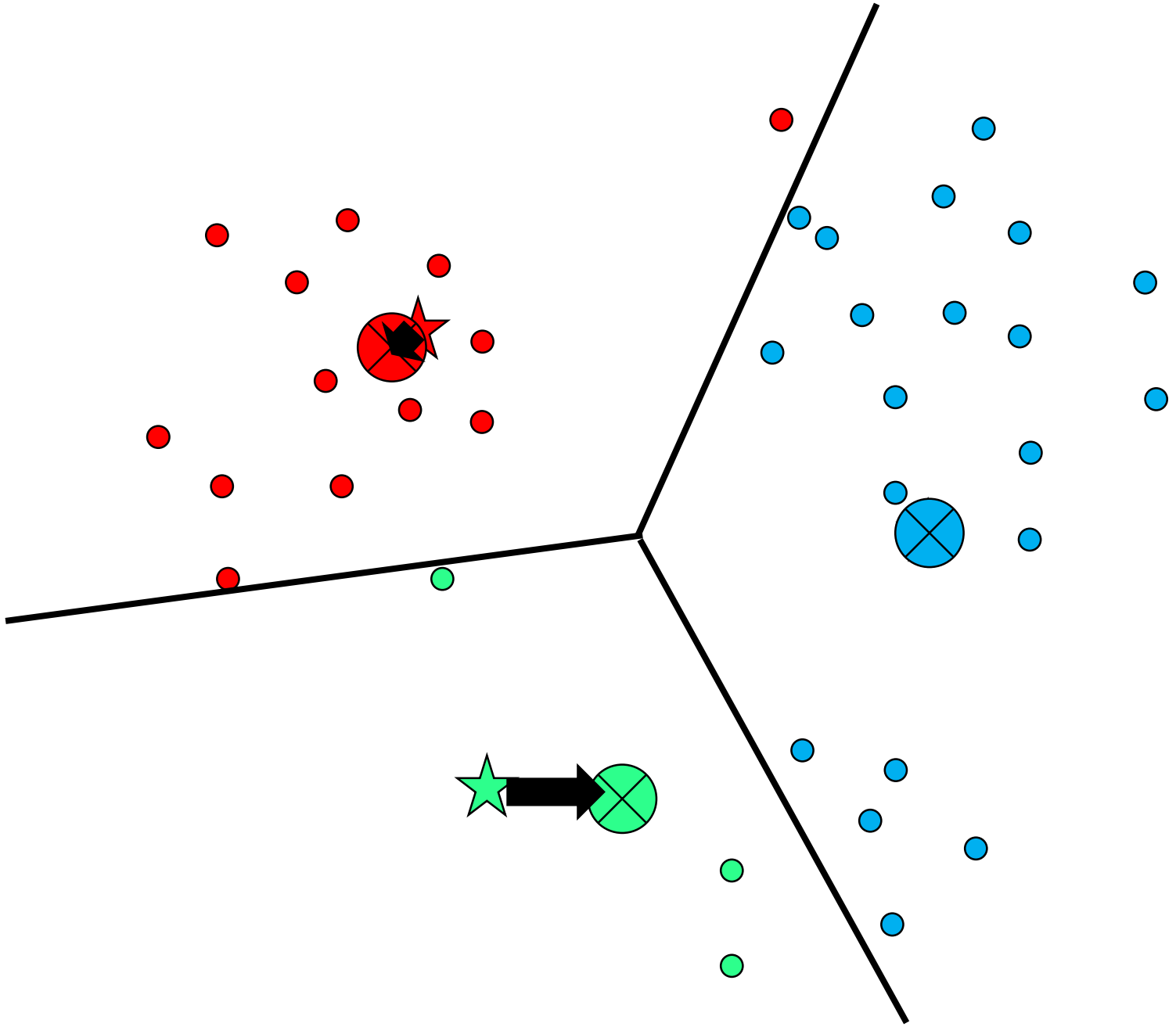
K-means Example

- 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!



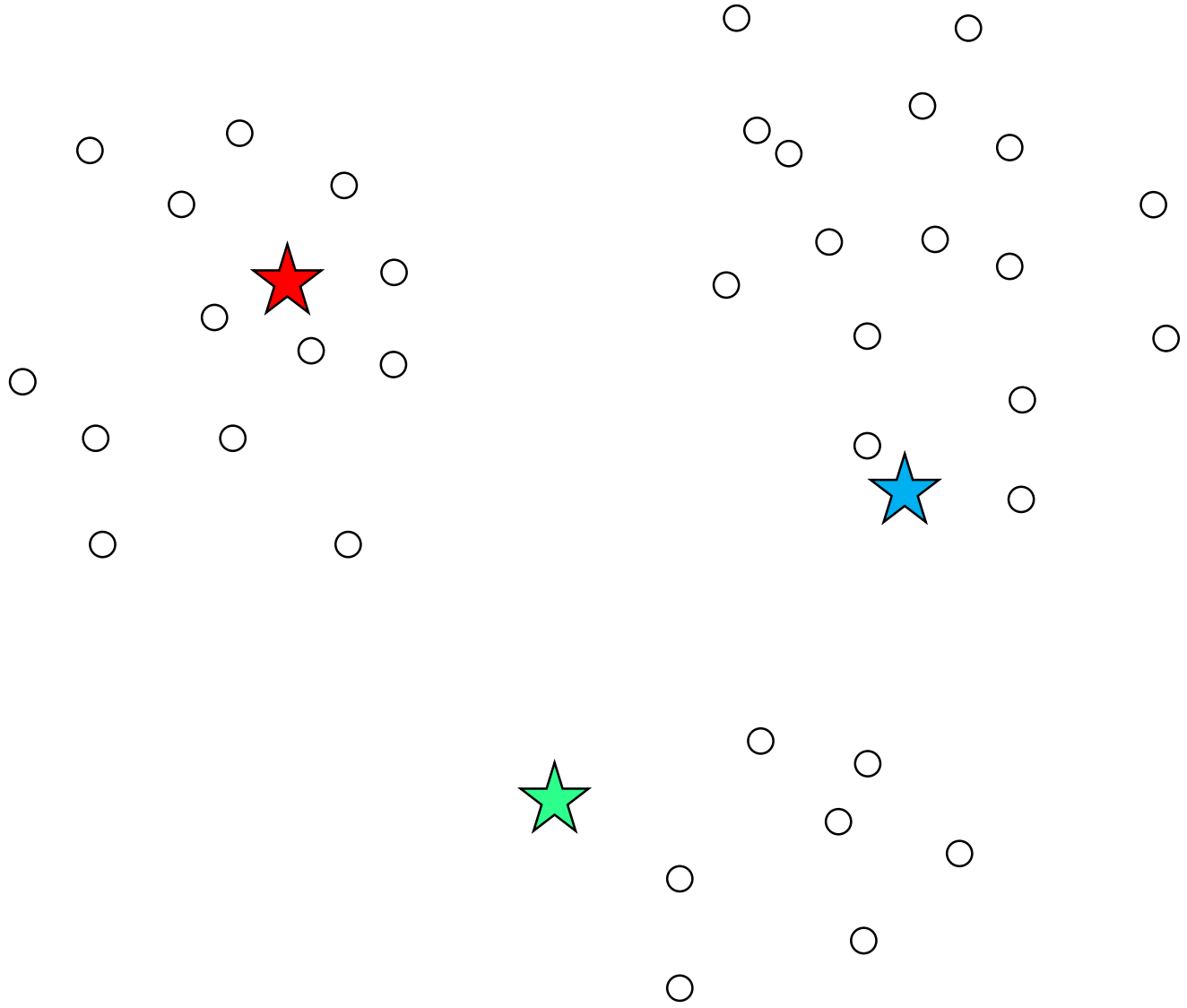
K-means Example

- 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!



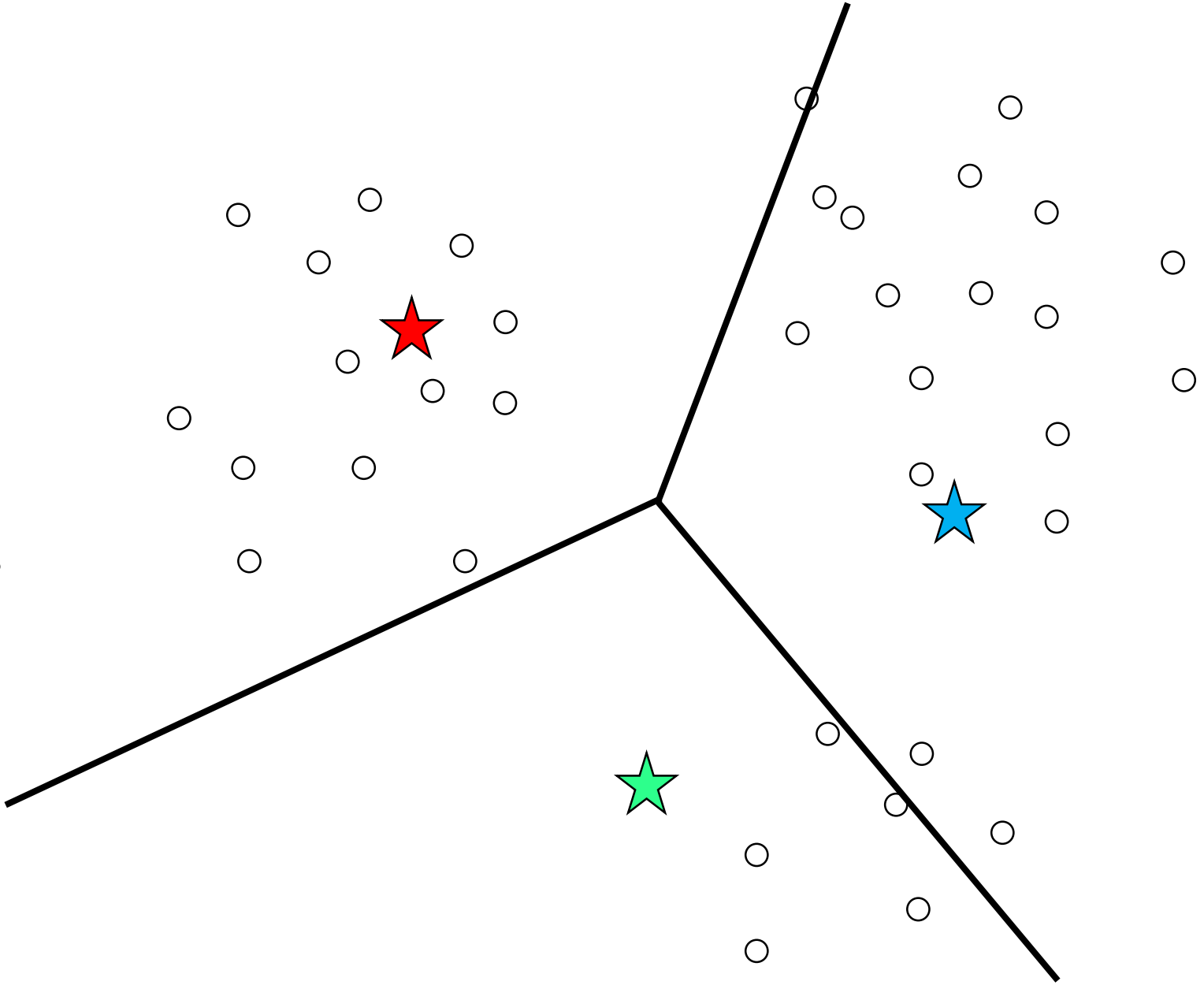
K-means Example

- 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!



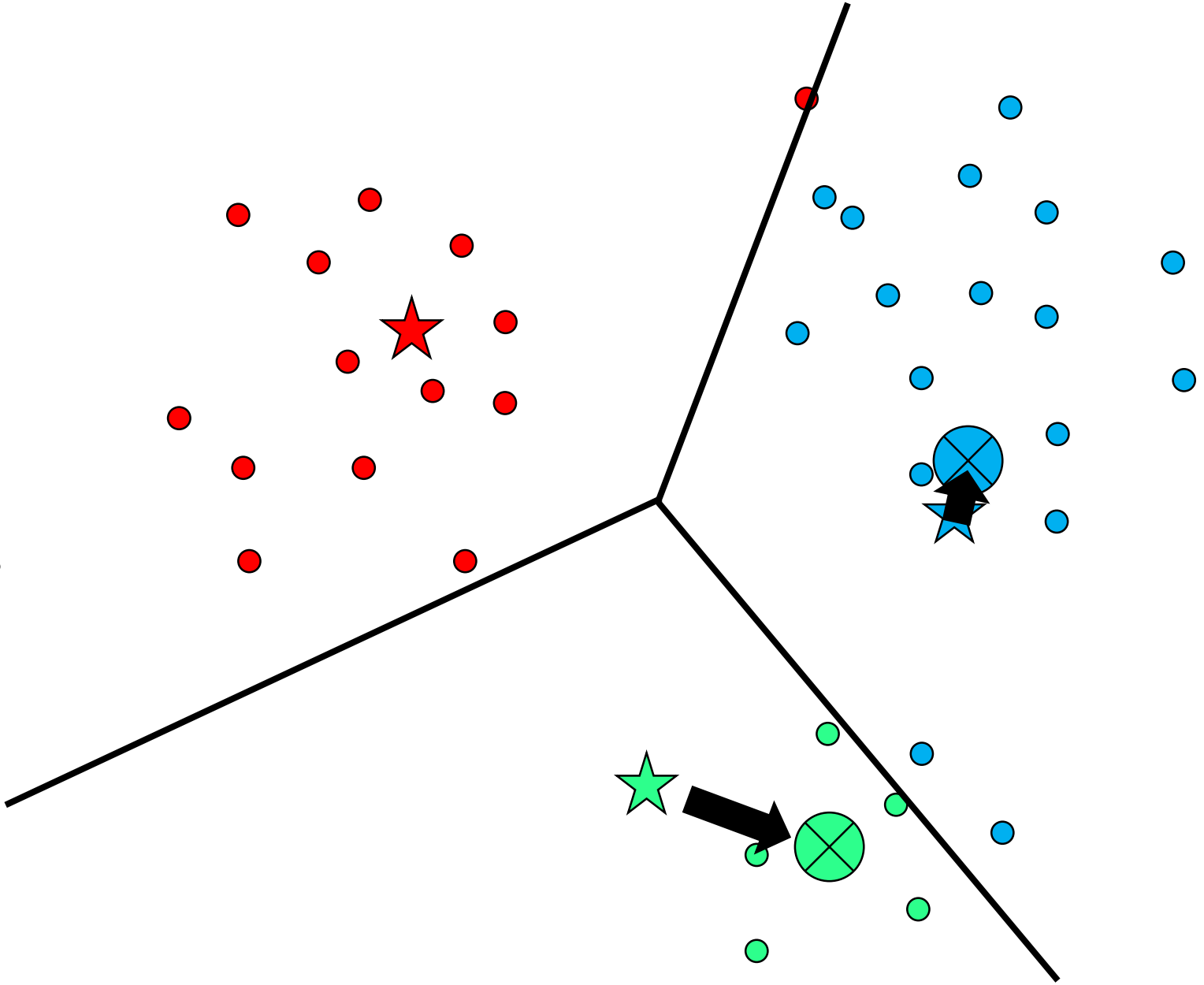
K-means Example

- 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!



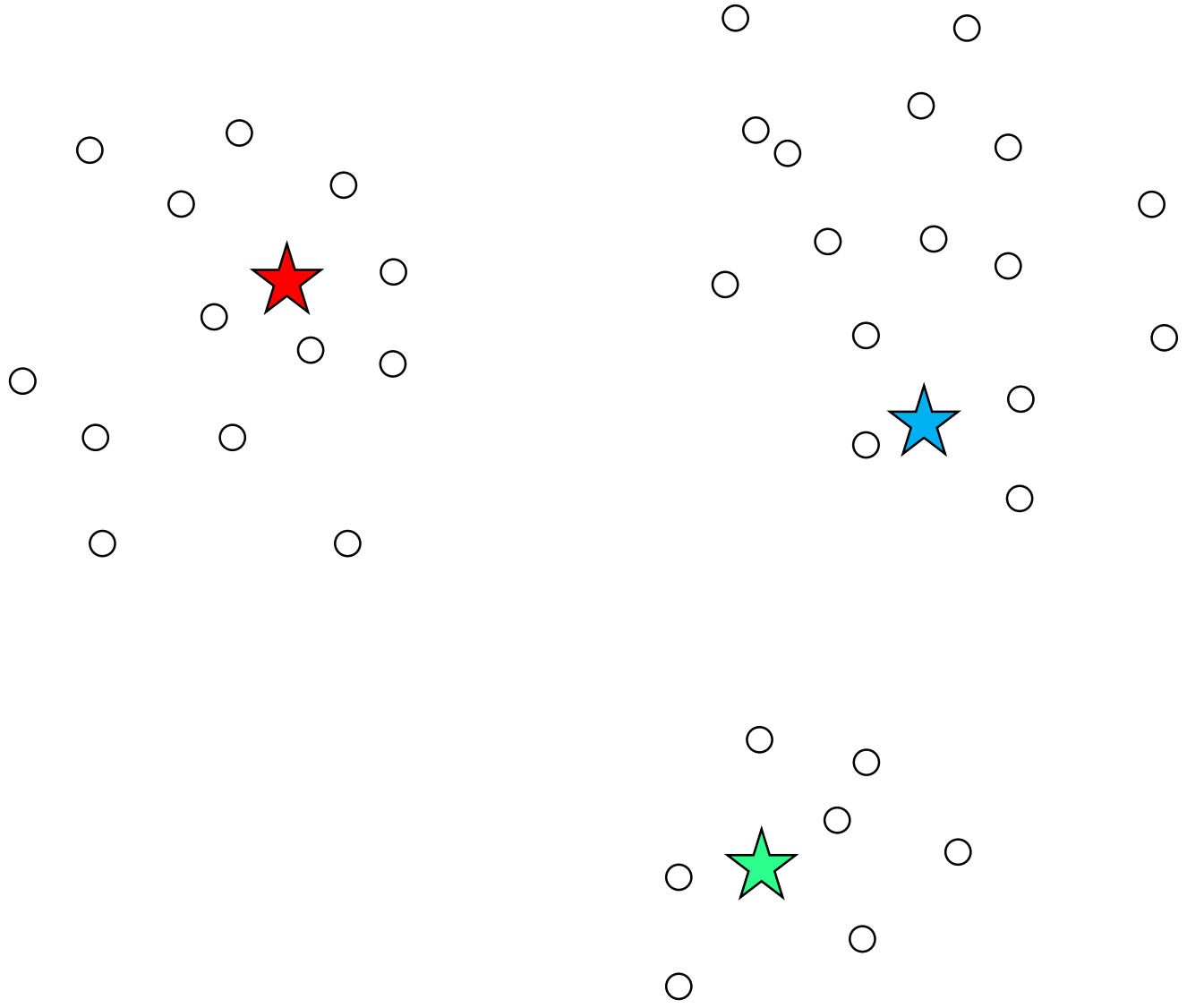
K-means Example

- 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!



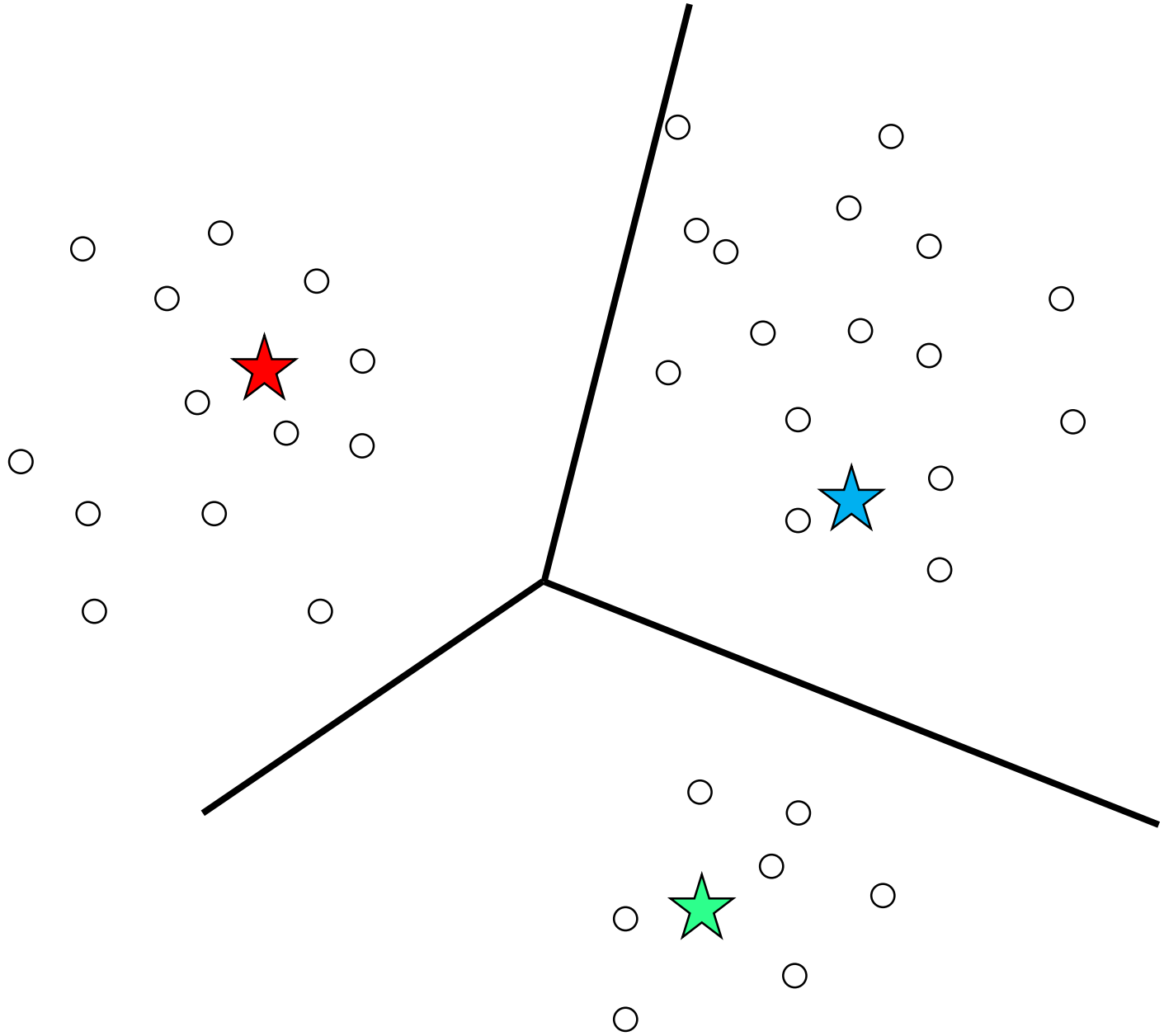
K-means Example

- 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!



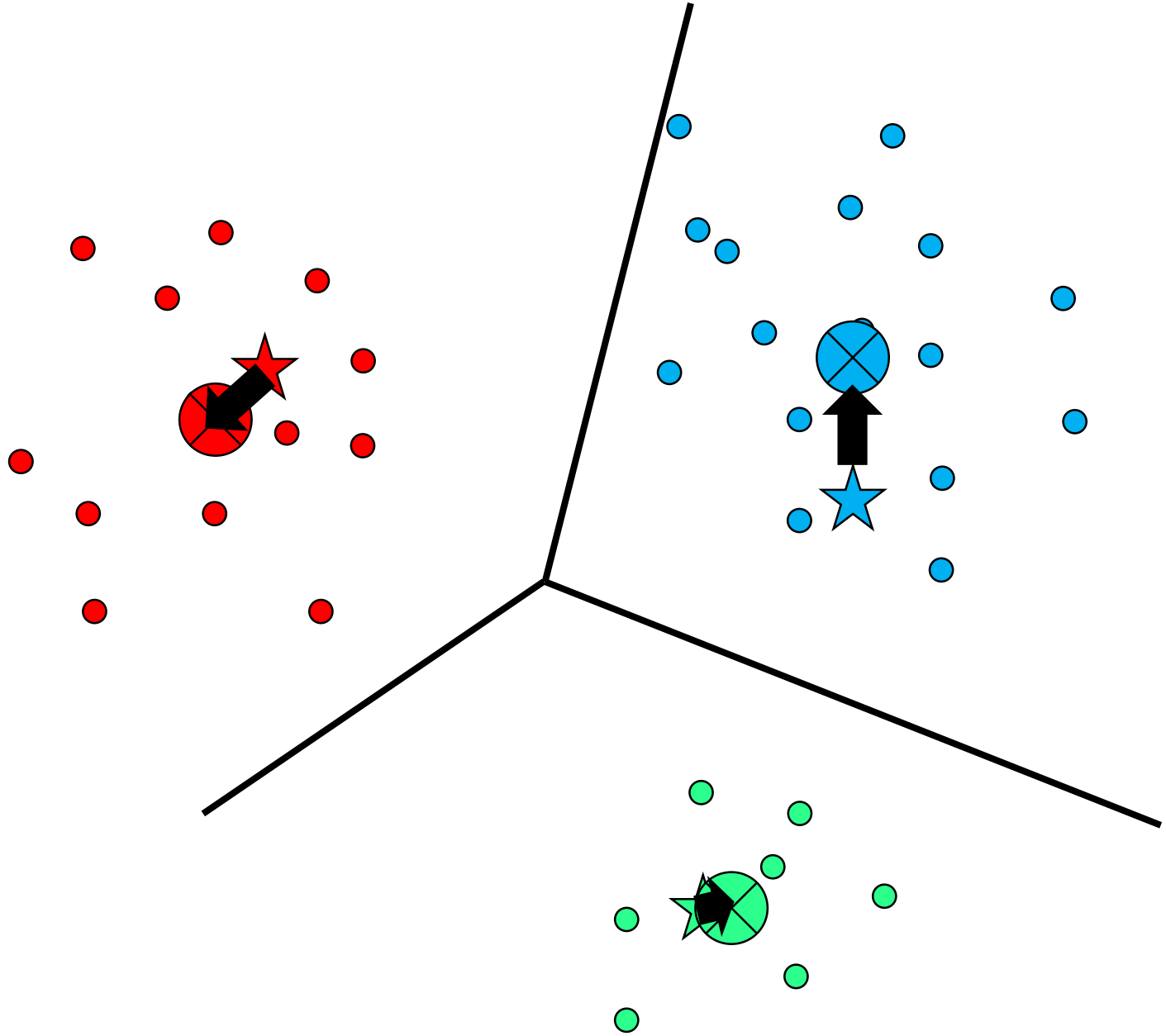
K-means Example

- 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!



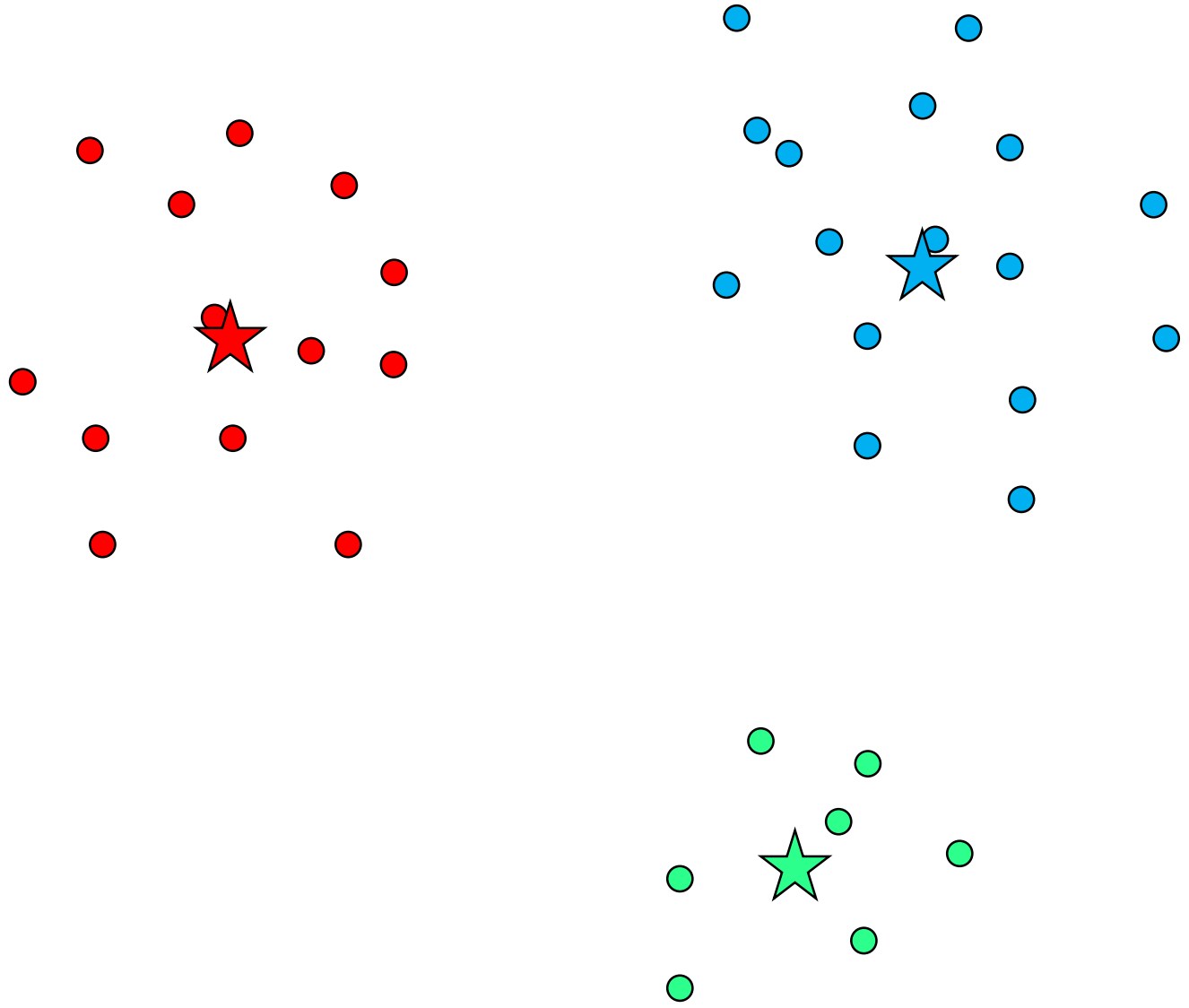
K-means Example

- 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!

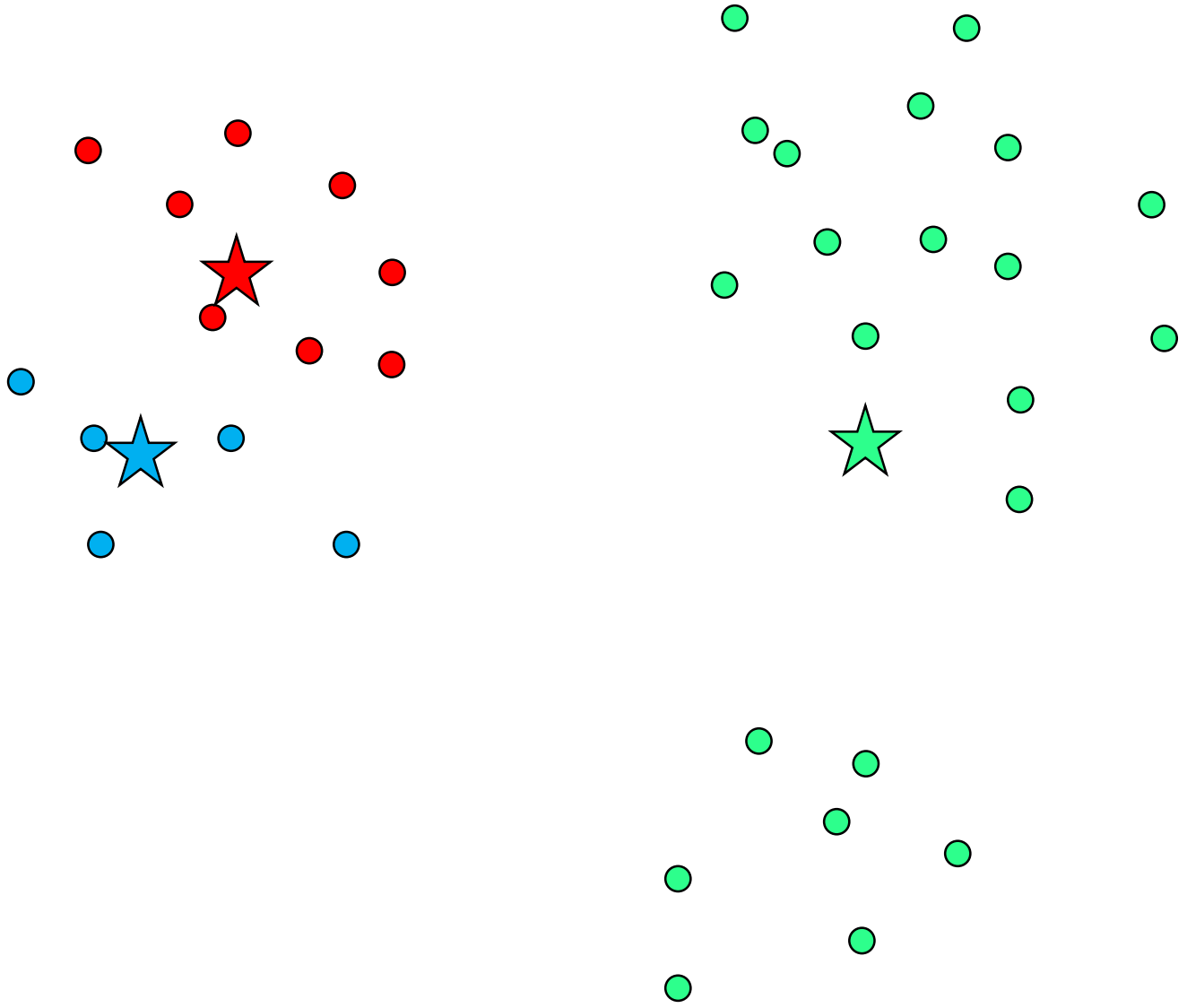


K-means Example

- 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)

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

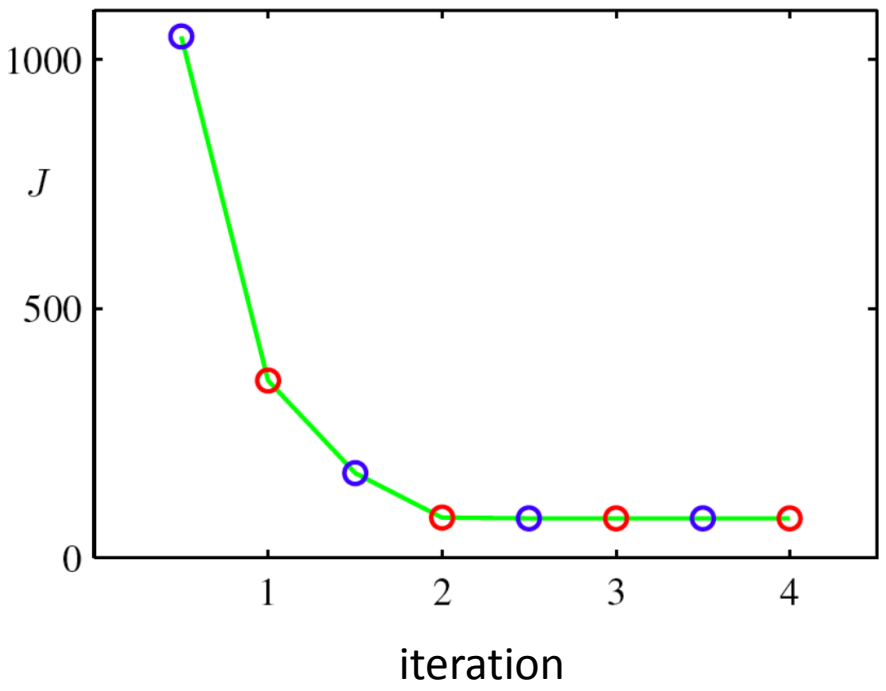
  - Refitting: Model parameters, means are adjusted to match sample means of data points they are responsible for:

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

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).



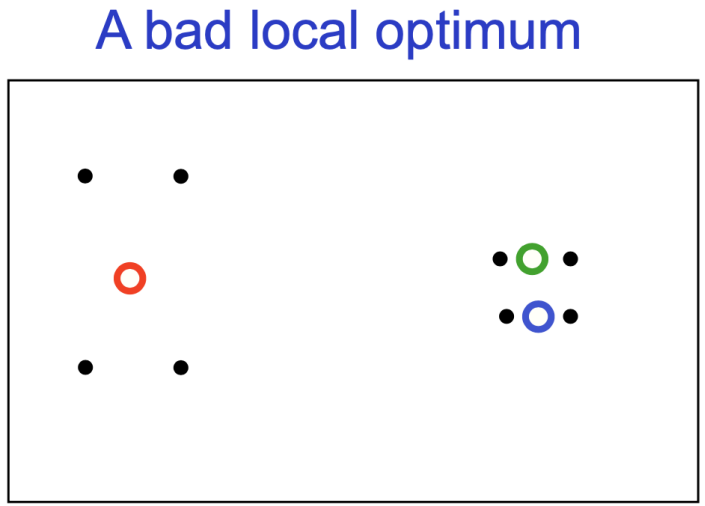
Slide credit: Ethan Fetaya, James Lucas, Emad Andrews

Local Minima

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

Options for avoiding local minimum:

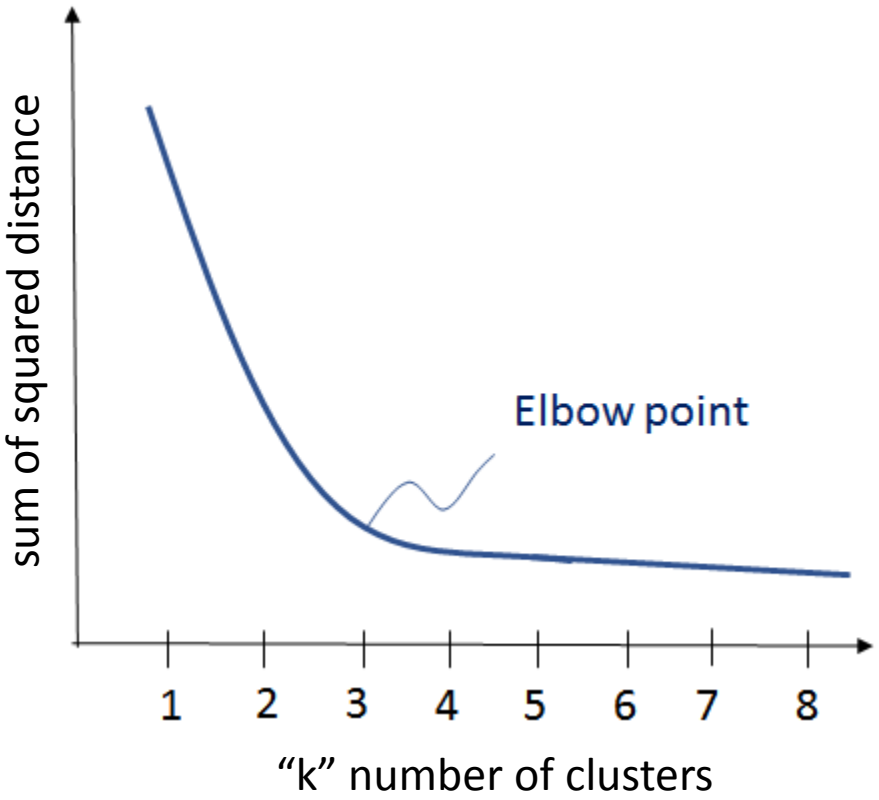
- we could try many random starting points
- split a big cluster into two
- merge to nearby clusters



Slide credit: Ethan Fetaya, James Lucas, Emad Andrews

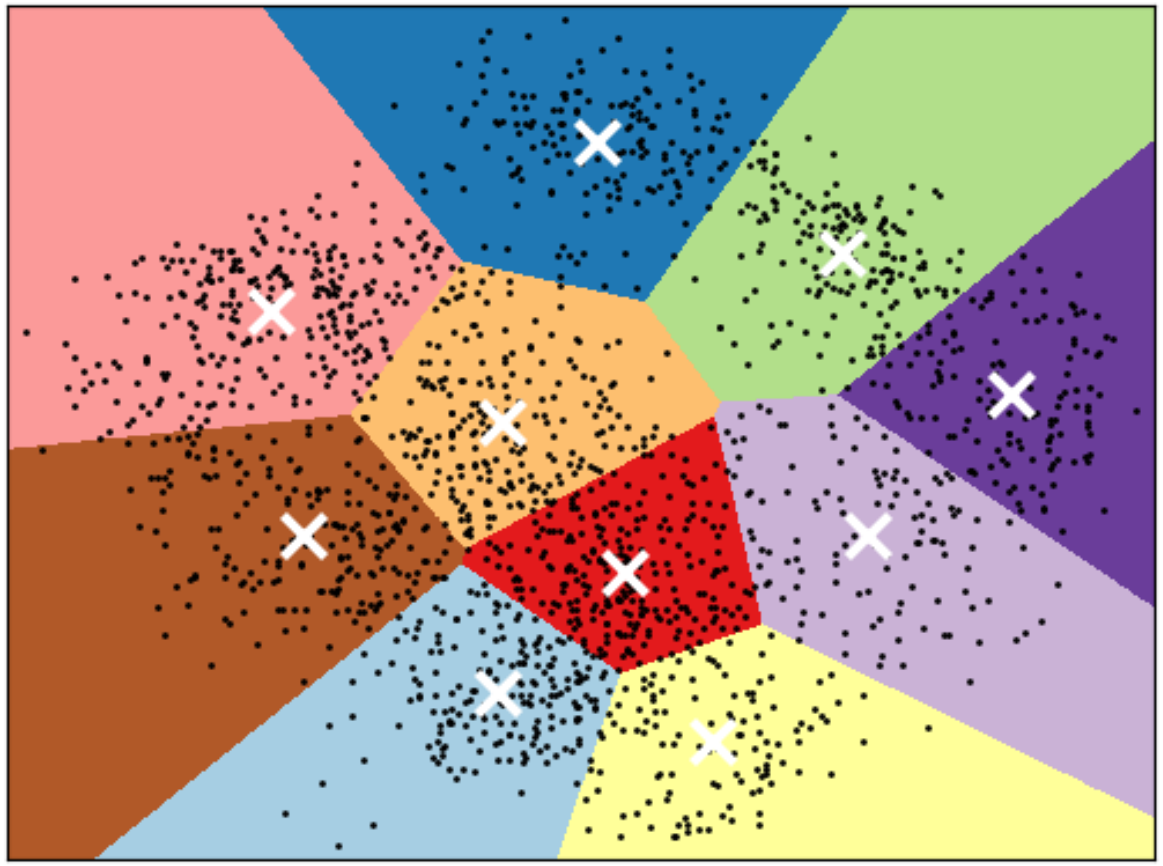
How to choose number of clusters (k)?

- 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:

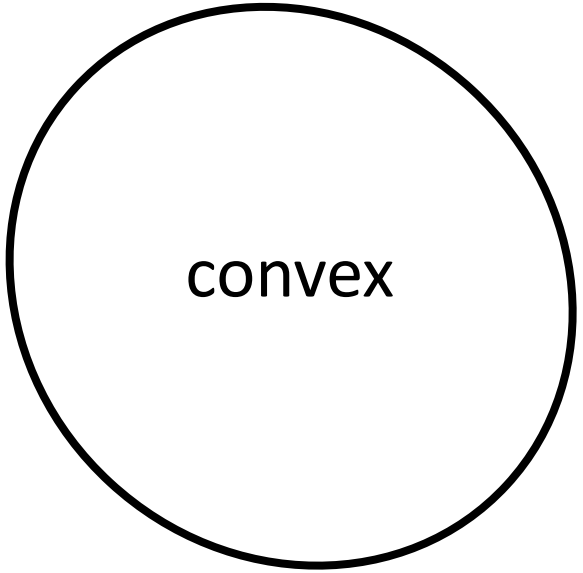


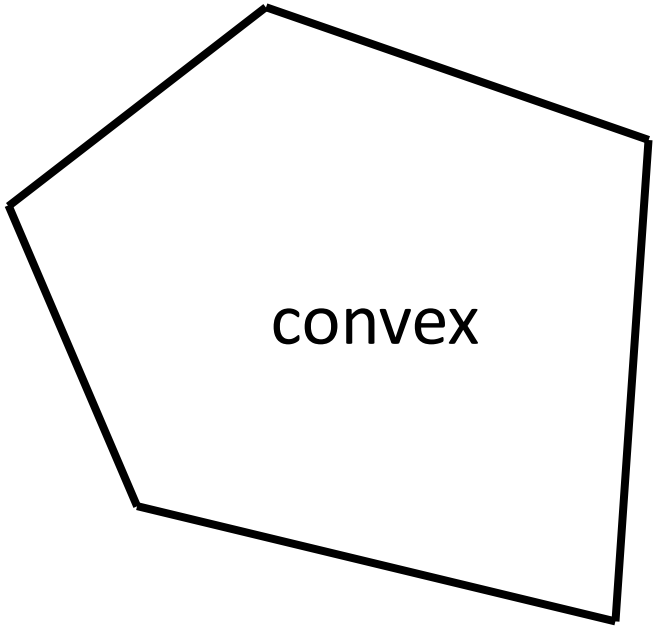
Source: scikit-learn

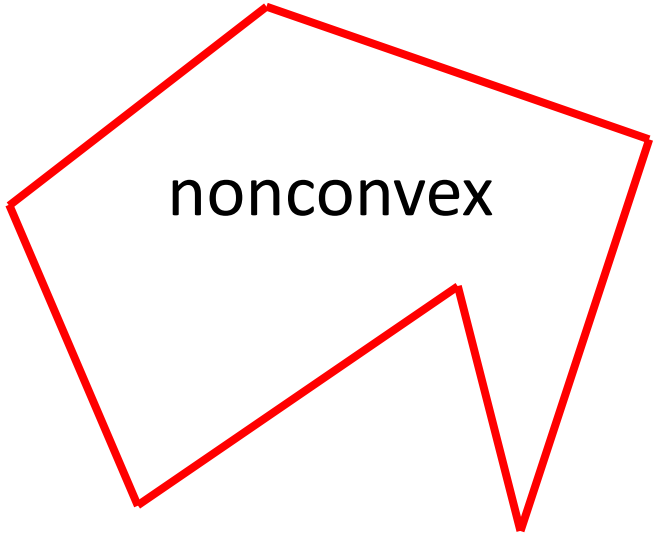
- Note that the clusters form convex regions.

Convex Region

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

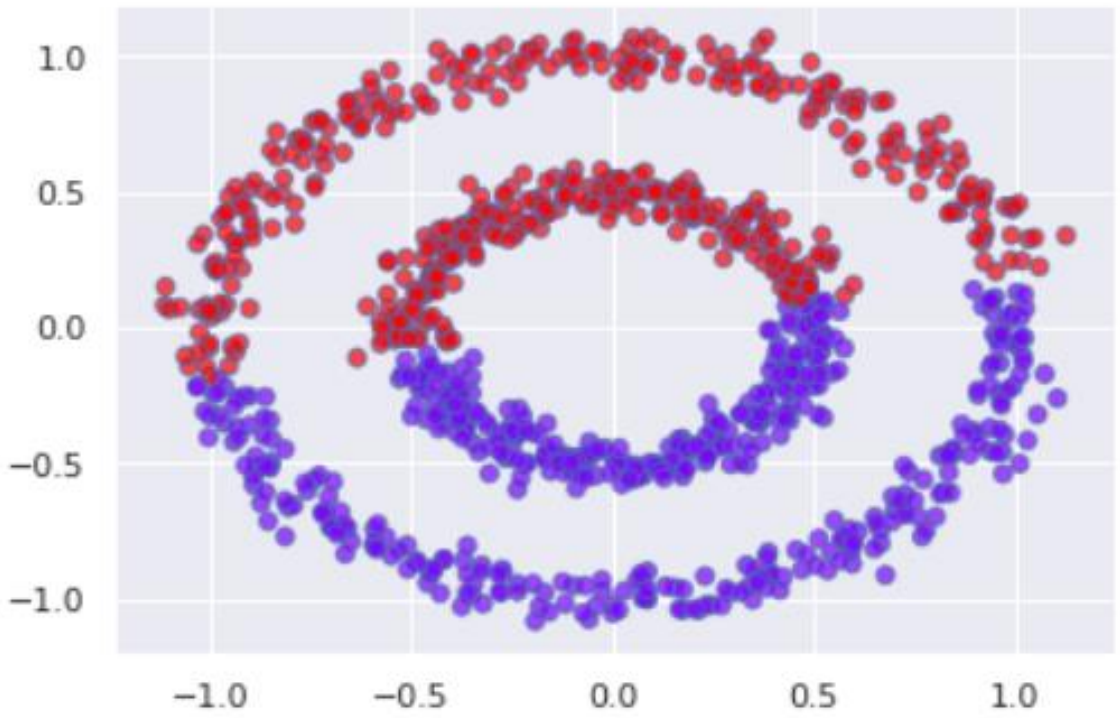






K-Means with Non-Convex Clusters

K-Means (K = 2)

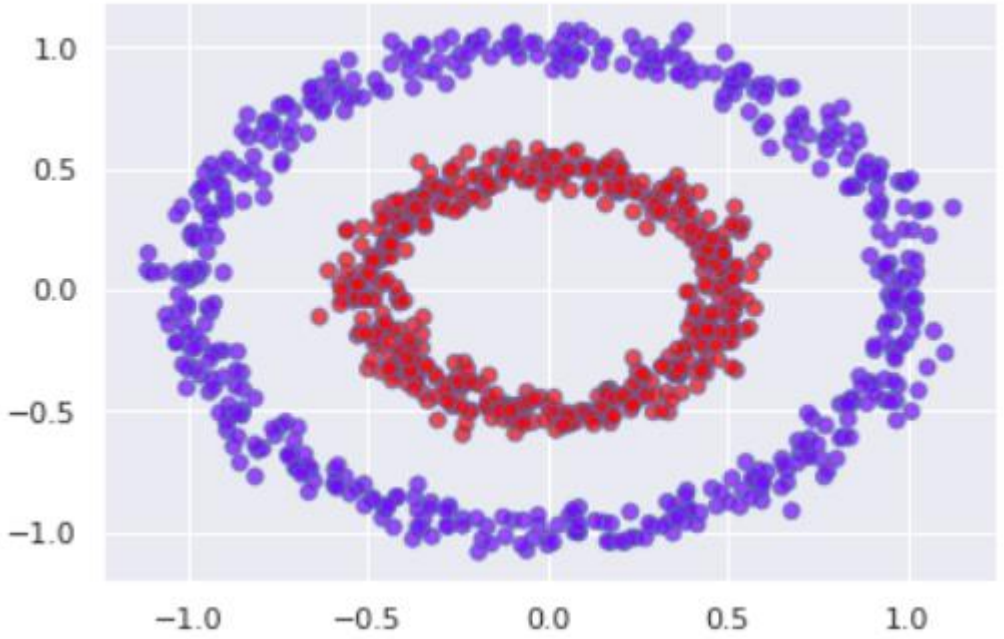


Source: Cory Malkin

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



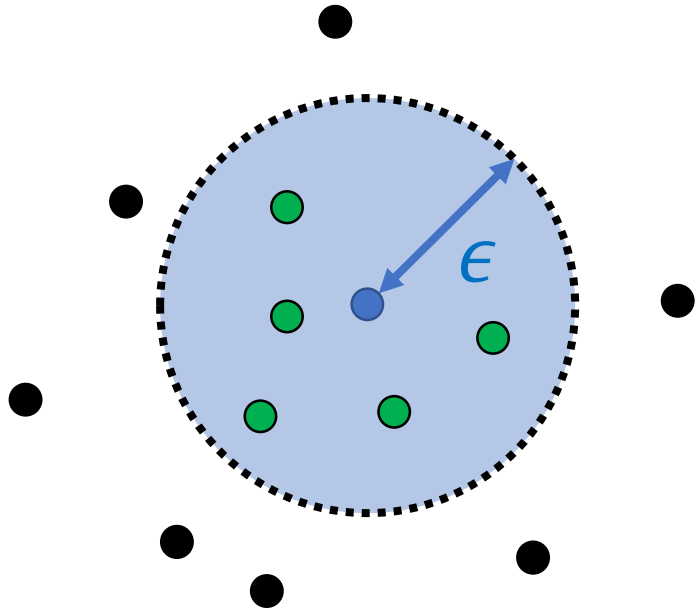
Source: Cory Malkin

- Q: What can we assume about samples/instances that are not assigned 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

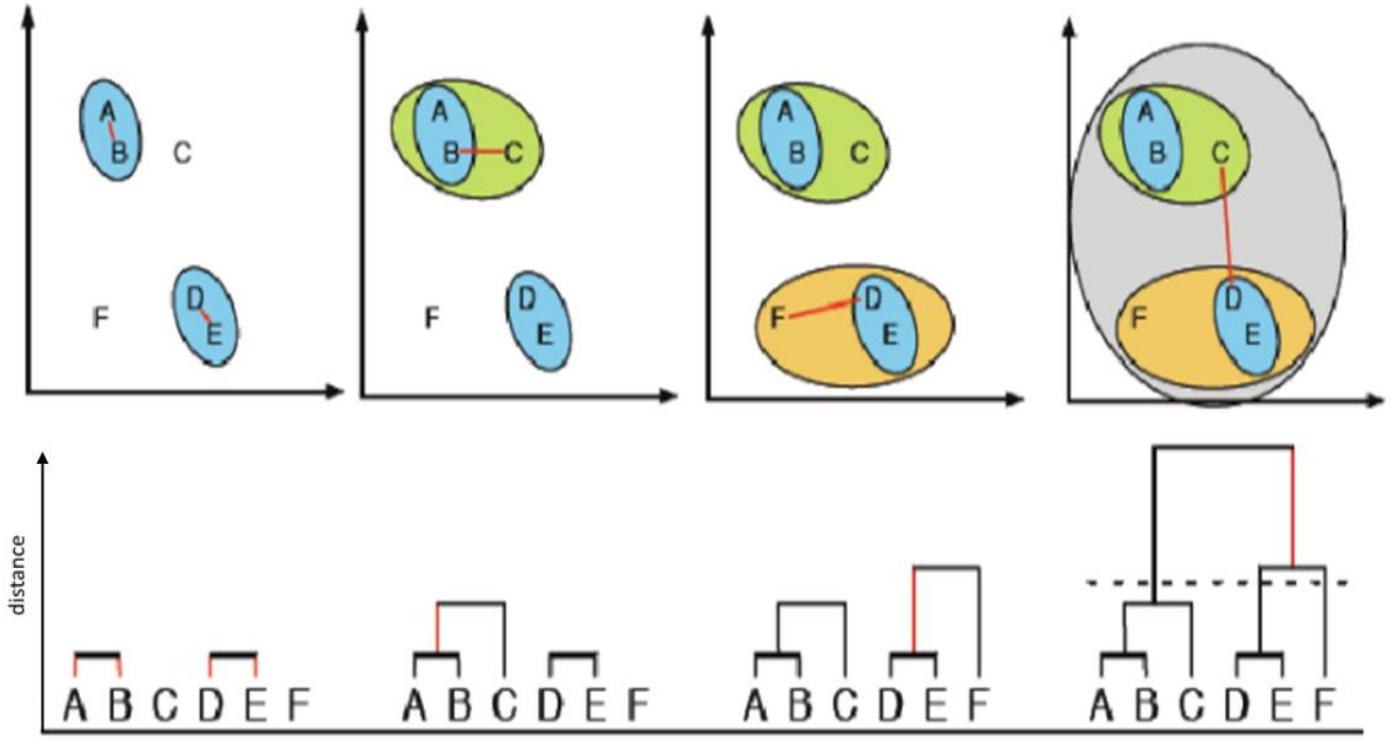


ε (epsilon): distance for deciding if a point is neighbour

min_neighbours: the minimum number of points required

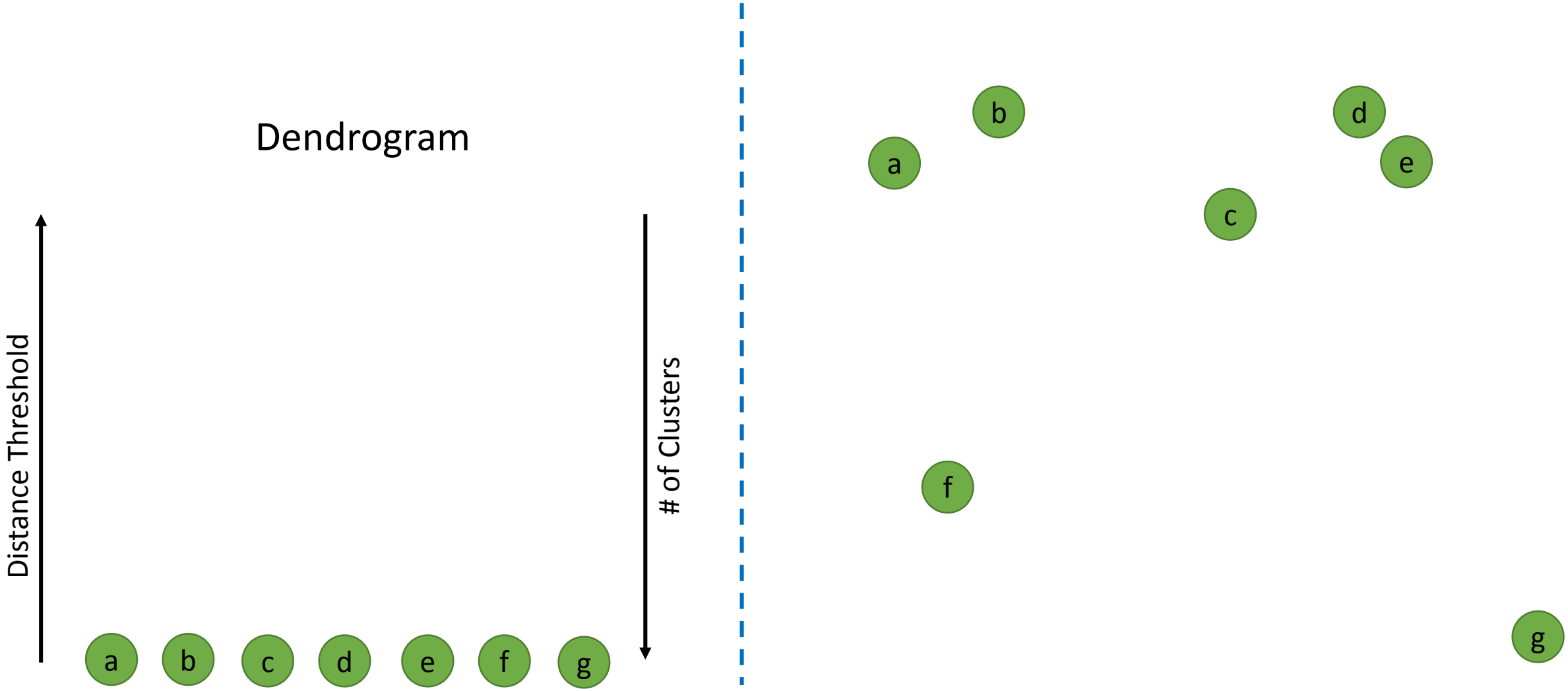
Agglomerative Clustering

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

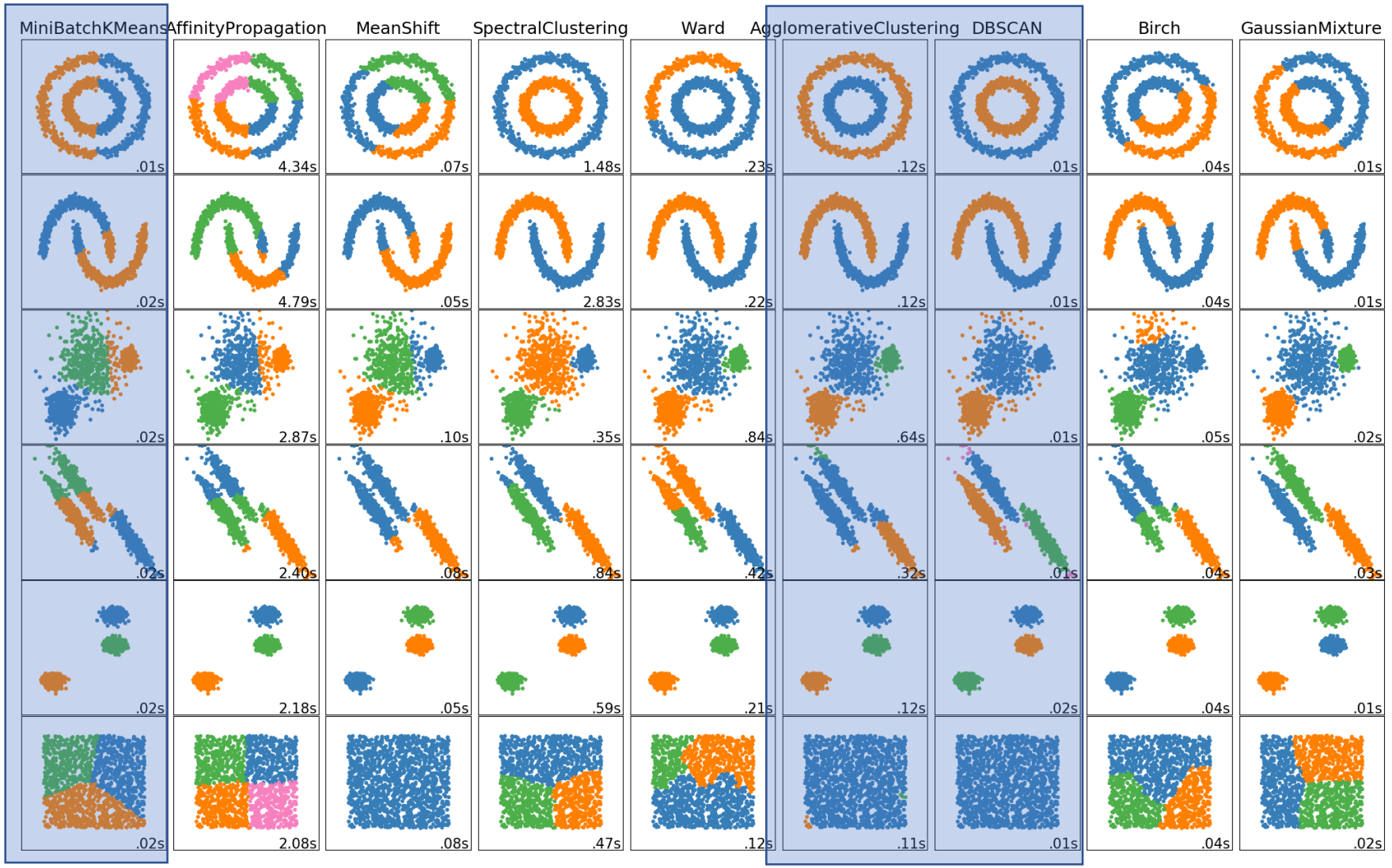


Source: MachineLearningStories

Example: Agglomerative Clustering



Comparison of Clustering Methods



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

Source: [scikit-learn]

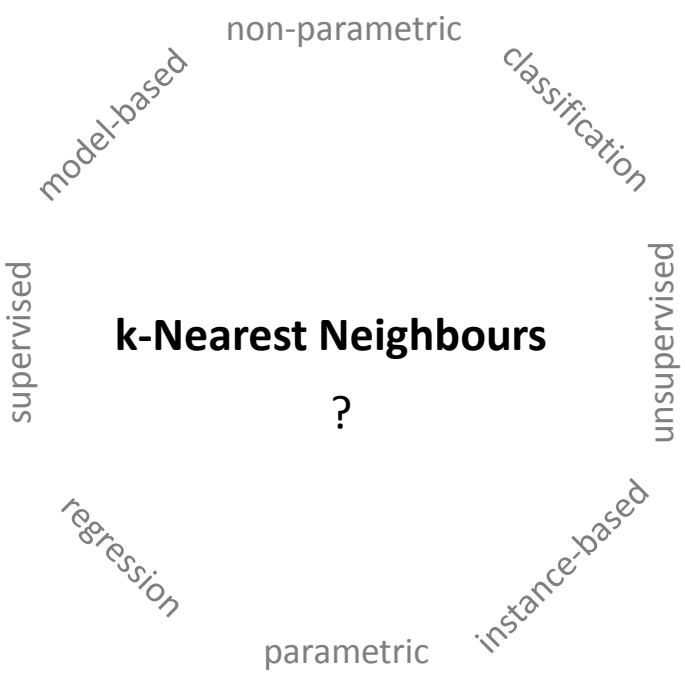
Applications in Computer Vision

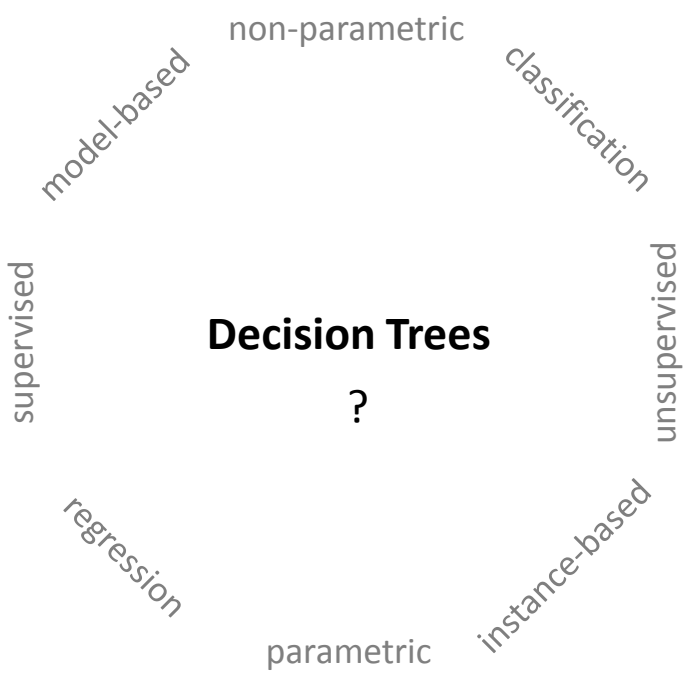
- Replace samples with the mean of their cluster (vector quantization)
- Visual example:
  - Inputs: colour pixels represented as RGB values
  - Outputs: cluster (average RGB) value obtained using k-means
- Q: How can this be applied for compression?
- Q: How can this be applied for image segmentation?

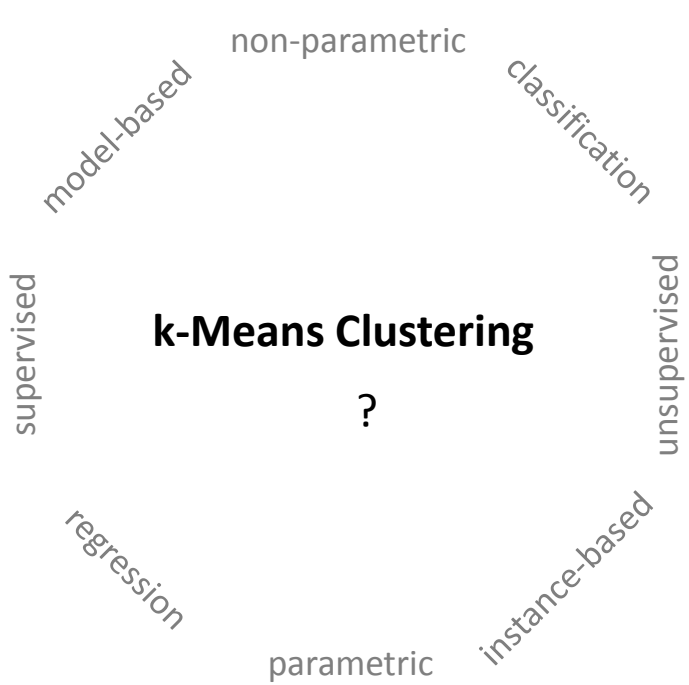
Source: PRML Bishop 2006

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 on Thursday, Jan. 27
  - Help with Python and Project 1
- Reading assignment 3 Due - Jan. 31 at 21:00
- Project 1 Due - Feb. 4 at 23:00
- Week 4 Lecture – Measuring Uncertainty
  - Probability Theory
  - Summary Statistics
  - Multivariate Gaussians
  - Performance Metrics