Mathematics of Data

INFO-4604, Applied Machine Learning

University of Colorado Boulder

September 5, 2017

Prof. Michael Paul

Goals

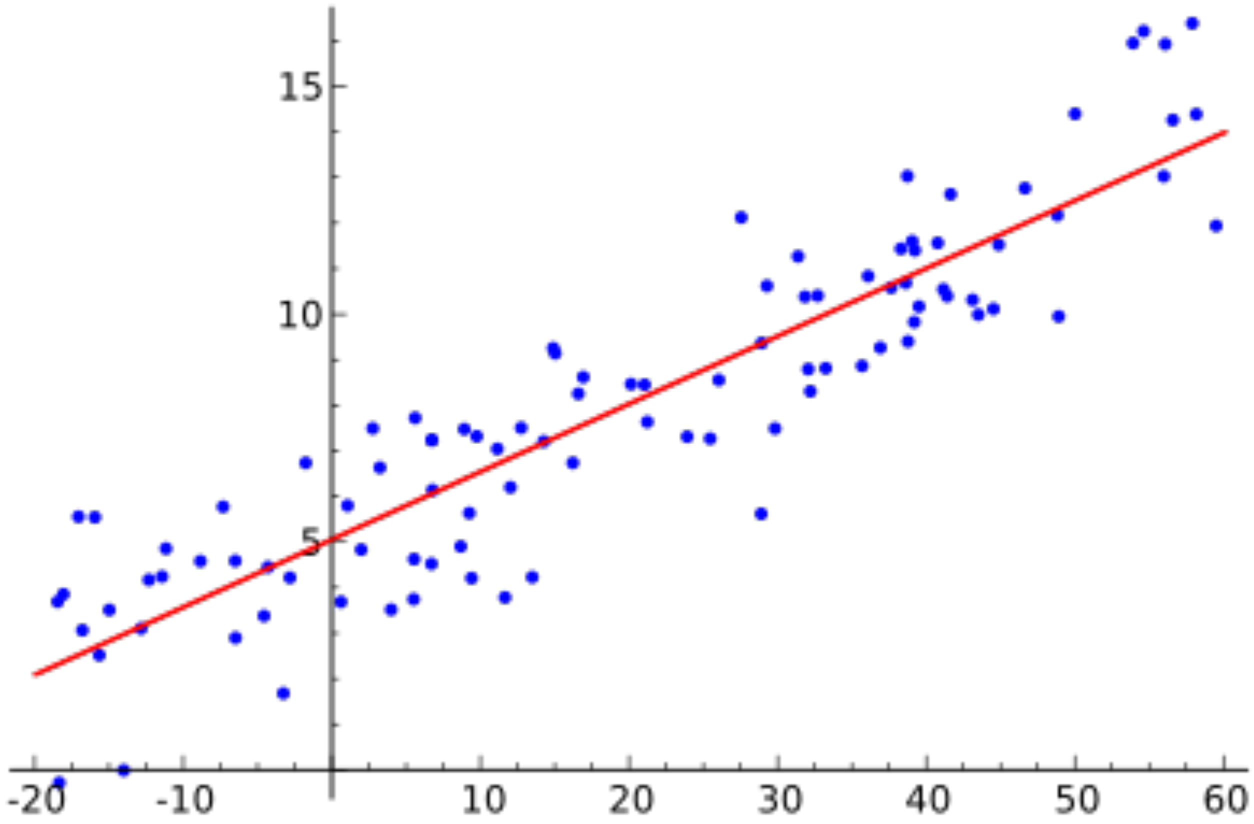
- In the intro lecture, every visualization was in 2D
  - What happens when we have more dimensions?
- Vectors and data points
  - What does a feature vector look like geometrically?
  - How to calculate the distance between points?
  - Definitions: vector products and linear functions

Two new algorithms today

- K-nearest neighbors classification
  - Label an instance with the most common label among the most similar training instances
- K-means clustering
  - Put instances into clusters to which they are closest (in a geometric space)

Both require a way to measure the similarity/distance between instances

Linear Regression



Linear Functions

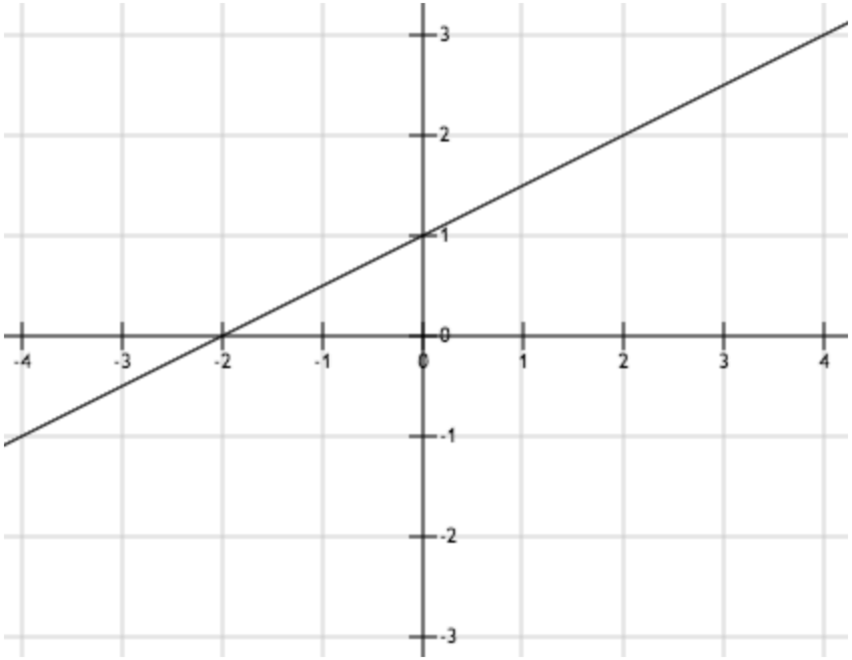
General form of a line:

intercept

$f(x) = mx + b$

slope

$y = \frac{1}{2}x + 1$



Linear Functions

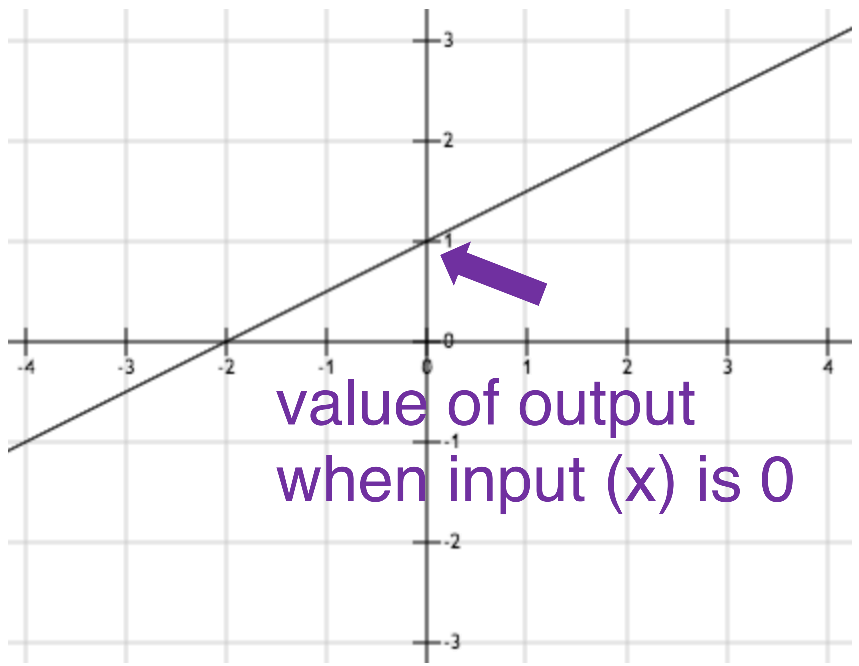
General form of a line:

intercept

$$f(x) = mx + b$$

slope

$y = ½x + 1$



Linear Functions

General form of a line:

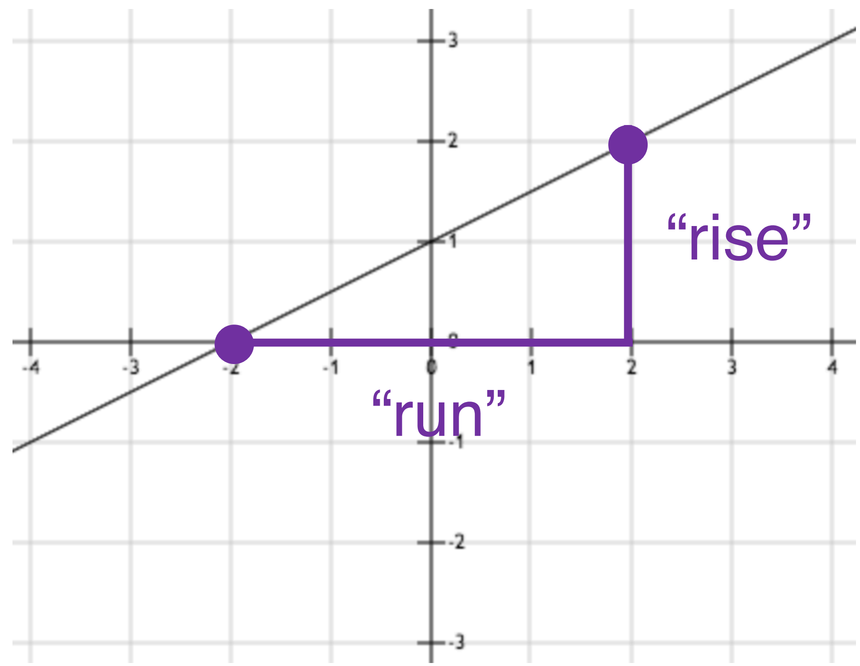
intercept

$$f(x) = mx + b$$

slope

= “rise over run”

$y = ½x + 1$



Linear Functions

General form of a line:

$$\underbrace{f(x) = mx + b}$$

intercept (b)

slope (m)

m and b are called **parameters**

- They are constant (once specified)
- Also called **coefficients**

x is the **argument** of the function

- It is the input to the function

Linear Functions

Machine learning involves learning the *parameters* of the predictor function

In linear regression, the predictor function is a linear function

- But the parameters are unknown ahead of time
- Goal is to learn what the slope and intercept should be

(How to do that is a question we'll answer next week)

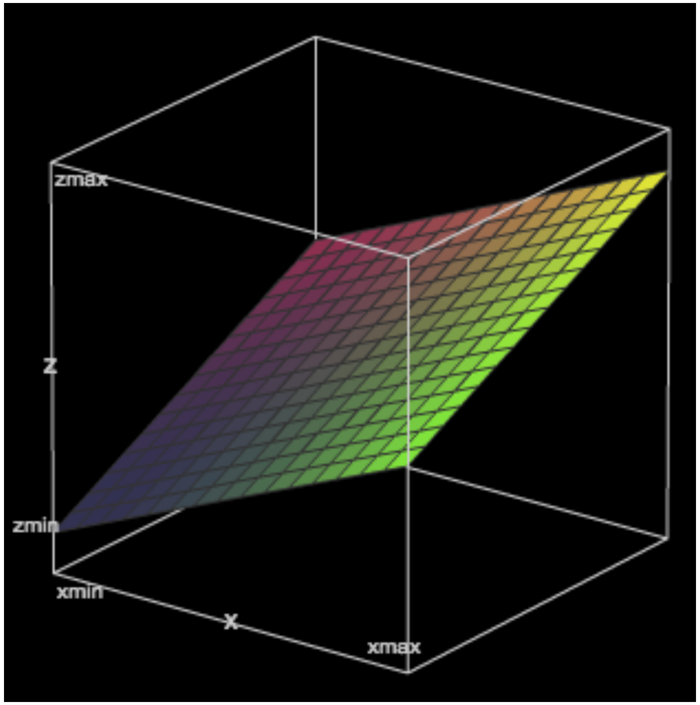
Linear Functions

Linear functions can have more than one argument

$$f(x_1,x_2) = m_1x_1 + m_2x_2 + b$$

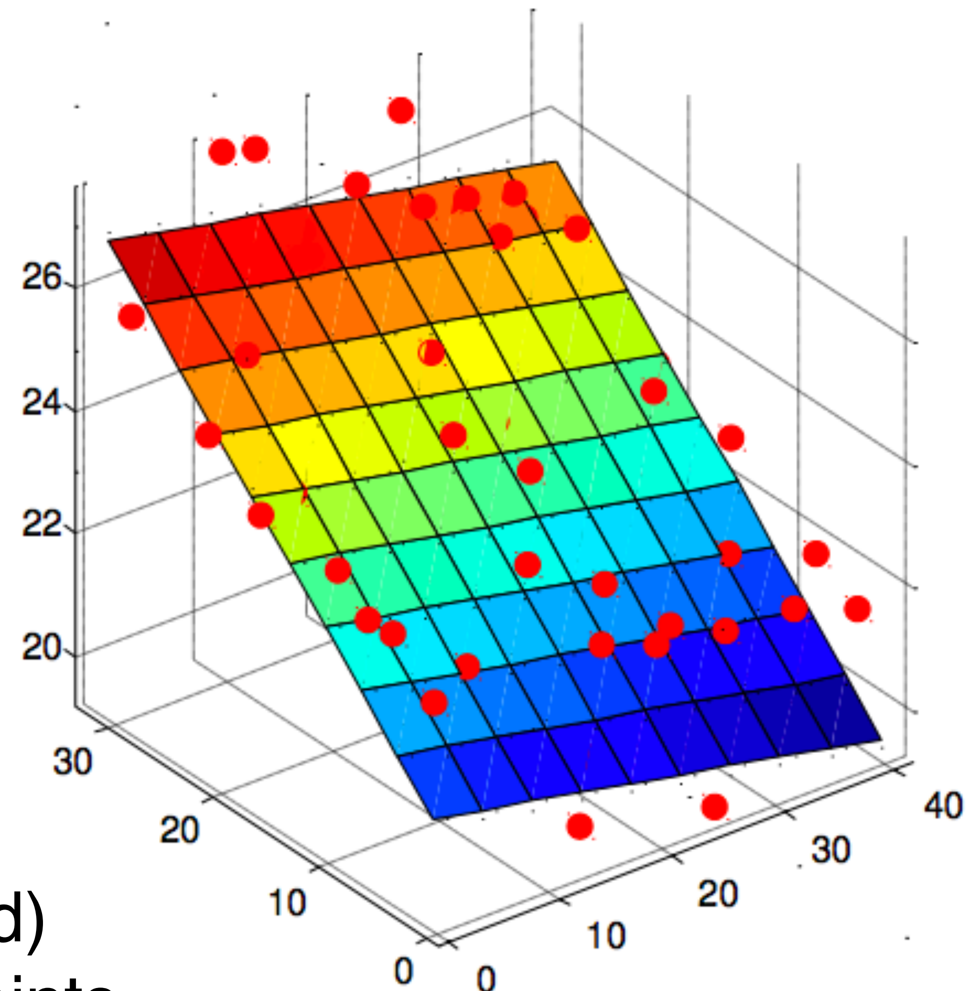
- One variable: line
- Two variables: plane

$y = 2x_1 + 2x_2 + 5$



From: https://www.math.uri.edu/~bkaskosz/flashmo/graph3d/

Linear Regression



- Two input variables (want to predict third)
- Fit a plane to the points

Linear Functions

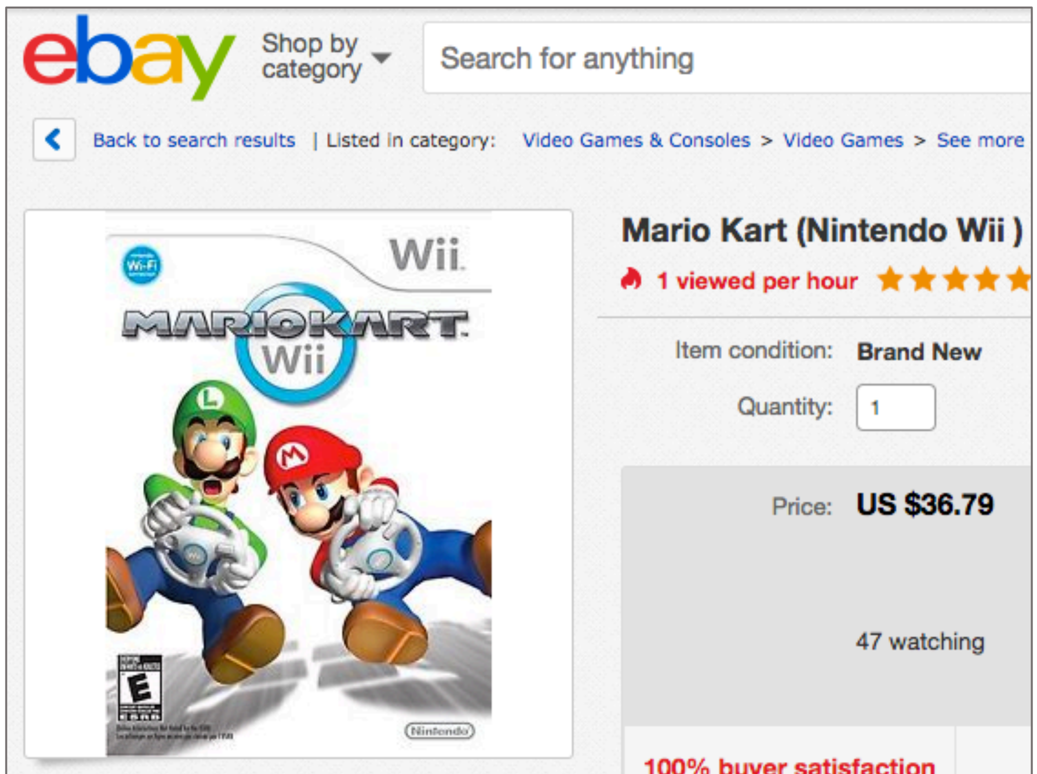
General form of linear functions:

$$f(x_1,\ldots,x_k) = \sum_{i=1}^{k} m_i x_i + b$$

- One variable: line
- Two variables: plane
- In general: **hyperplane**

Linear Regression

How much will Mario Kart (Wii) sell for on eBay? (example from *OpenIntro Stats*, Ch 8)



Linear Regression

How much will Mario Kart (Wii) sell for on eBay?
(example from *OpenIntro Stats*, Ch 8)

Four features:

cond_new	a coded two-level categorical variable, which takes value 1 when the game is new and 0 if the game is used
stock_photo	a coded two-level categorical variable, which takes value 1 if the primary photo used in the auction was a stock photo and 0 if the photo was unique to that auction
duration	the length of the auction, in days, taking values from 1 to 10
wheels	the number of Wii wheels included with the auction (a *Wii wheel* is a plastic racing wheel that holds the Wii controller and is an optional but helpful accessory for playing Mario Kart)

Linear Regression

f(x) = 5.13 *cond_new* + 1.08 *stock_photo*
– 0.03 *duration* + 7.29 *wheels* + 36.21

If you know the values of the four features, you can get a guess of the output (*price*) by plugging them into this function

	price	cond_new	stock_photo	duration	wheels
1	51.55	1	1	3	1
2	37.04	0	1	7	1
⋮	⋮	⋮	⋮	⋮	⋮
140	38.76	0	0	7	0
141	54.51	1	1	1	2

Linear Functions

$$f(x_1,\ldots,x_k) = \sum_{i=1}^{k} m_i x_i \quad + b$$

$$f(x) = \quad 5.13\ \mathit{cond\_new} + 1.08\ \mathit{stock\_photo}$$
$$-\ 0.03\ \mathit{duration} + 7.29\ \mathit{wheels} + 36.21$$

Mapping this to the general form…

$x_1 = \mathit{cond\_new}$	$m_1 = 5.13$	$k = 4$
$x_2 = \mathit{stock\_photo}$	$m_2 = 1.08$	
$x_3 = \mathit{duration}$	$m_3 = -0.03$	
$x_4 = \mathit{wheels}$	$m_4 = 7.29$	$b = 36.21$

Vector Notation

A list of values is called a **vector**

We can use variables to denote entire vectors as shorthand

$\mathbf{m} = \langle m_1, m_2, m_3, m_4 \rangle$

$\mathbf{x} = \langle x_1, x_2, x_3, x_4 \rangle$

Vector Notation

The dot product of two vectors is written as $\mathbf{m}^T\mathbf{x}$ or $\mathbf{m}\bullet\mathbf{x}$, which is defined as:

$$\mathbf{m}^T\mathbf{x} = \sum_{i=1}^{k} m_i x_i$$

Example:

$\mathbf{m} = <5.13, 1.08, -0.03, 7.29>$

$\mathbf{x} = <x_1, x_2, x_3, x_4>$

$\mathbf{m}^T\mathbf{x} = 5.13x_1 + 1.08x_2 - 0.03x_3 + 7.29x_4$

Vector Notation

Equivalent notation for a linear function:

$$f(x_1,\ldots,x_k) = \sum_{i=1}^{k} m_i x_i \quad + b$$

or

$$f(\mathbf{x}) = \mathbf{m}^T\mathbf{x} + b$$

Vector Notation

Terminology:

A **point** is the same as a **vector**
(at least as used in this class)

Vector Notation

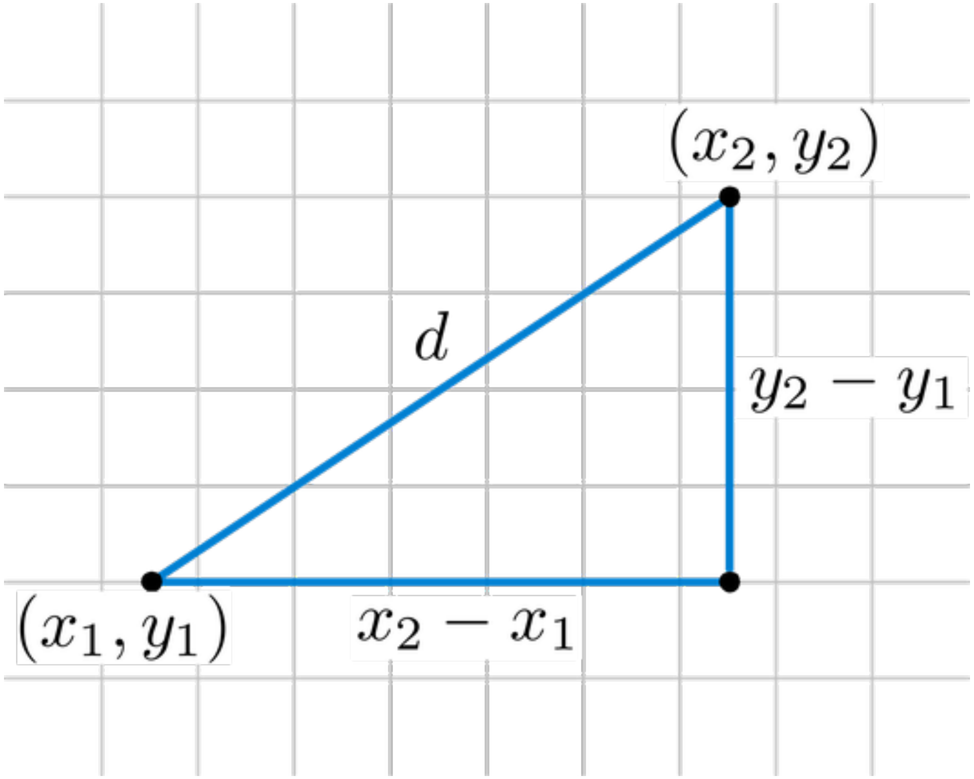
Remember:

In machine learning, the number of dimensions in your points/vectors is the number of *features*

Pause

Distance

How far apart are two points?



Distance

Euclidean distance between two points in two dimensions:

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

In three dimensions (x,y,z):

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

Distance

General formulation of Euclidean distance between two points with *k* dimensions:

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{k} (p_i - q_i)^2}$$

where **p** and **q** are the two points (each represents a k-dimensional vector)

Distance

Example:

$\mathbf{p} = <1.3, 5.0, -0.5, -1.8>$

$\mathbf{q} = <1.8, 5.0, 0.1, -2.3>$

$$d(\mathbf{p}, \mathbf{q}) = \mathrm{sqrt}\Big((1.3-1.8)^2 + (5.0-5.0)^2 + (-0.5-0.1)^2 + (-1.8--2.3)^2 \Big)$$

$$= \mathrm{sqrt}(.86)$$

$$= .927$$

Distance

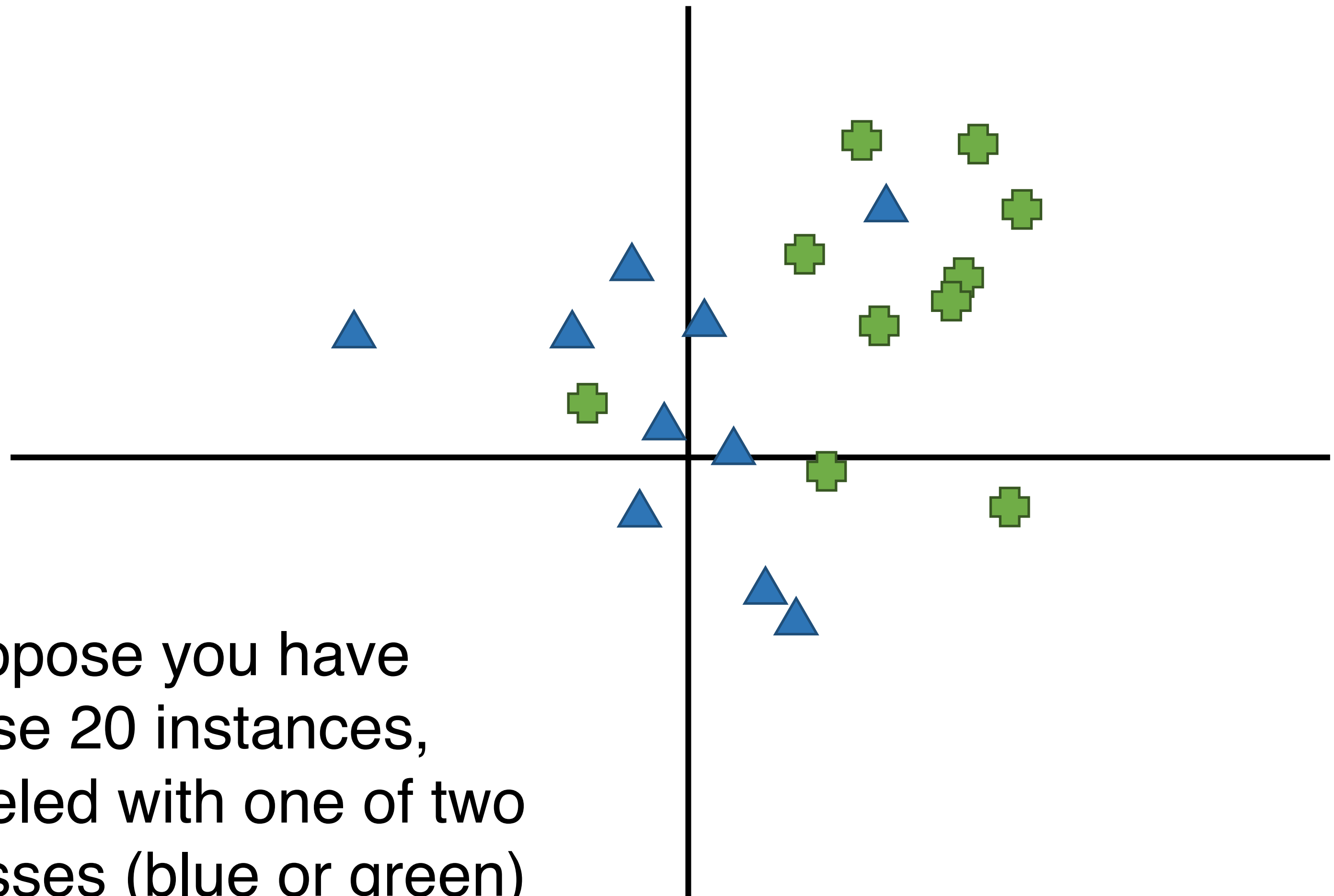
A special case is the distance between a point and zero (the *origin*).

$$d(\mathbf{p}, \mathbf{0}) = \sqrt{\sum_{i=1}^{k} (p_i)^2}$$

This is called the **Euclidean norm** of **p**

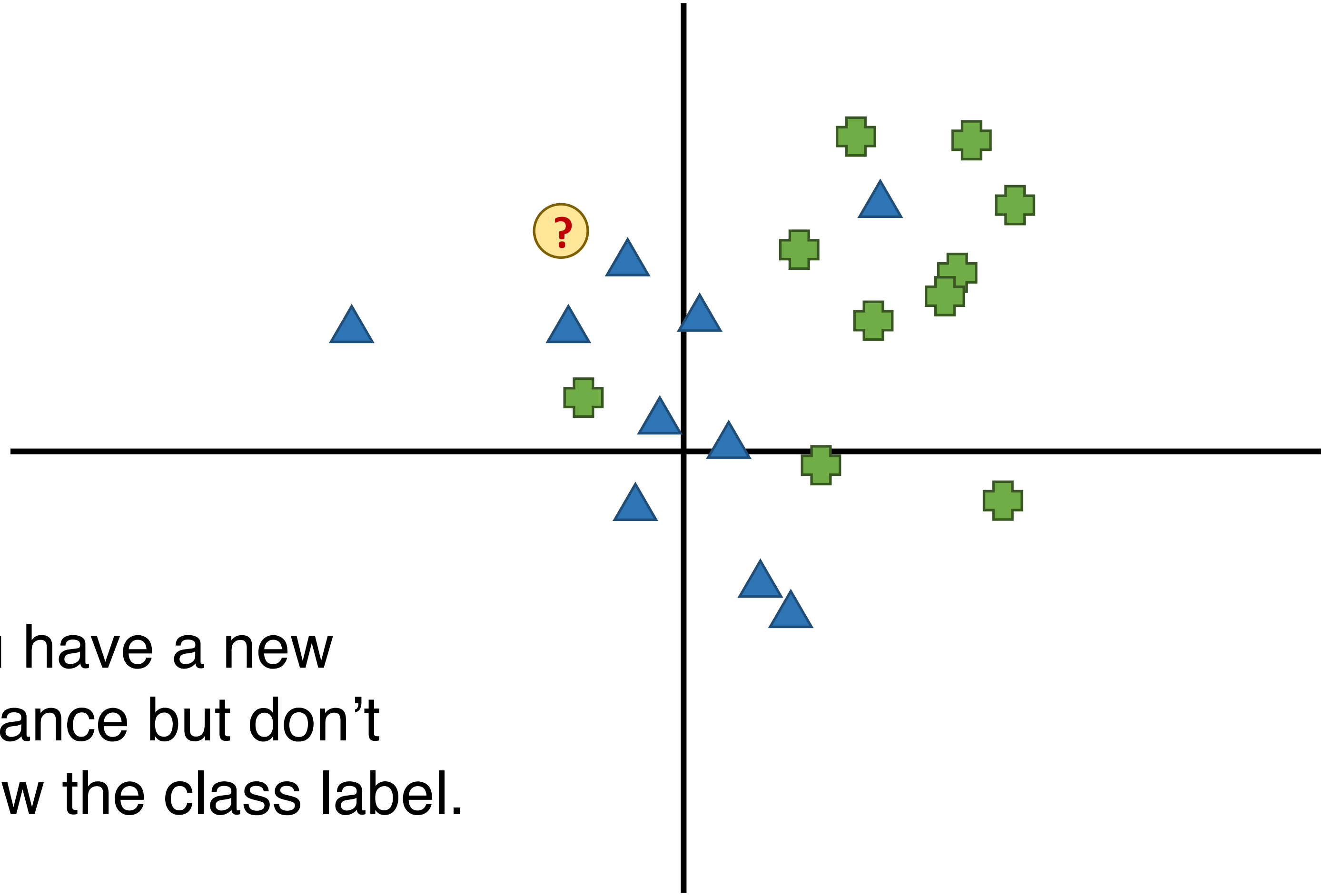
- A norm is a measure of a vector’s length
- The Euclidean norm is also called the **L2 norm**
  - We’ll learn about other norms later

Distance-based Prediction



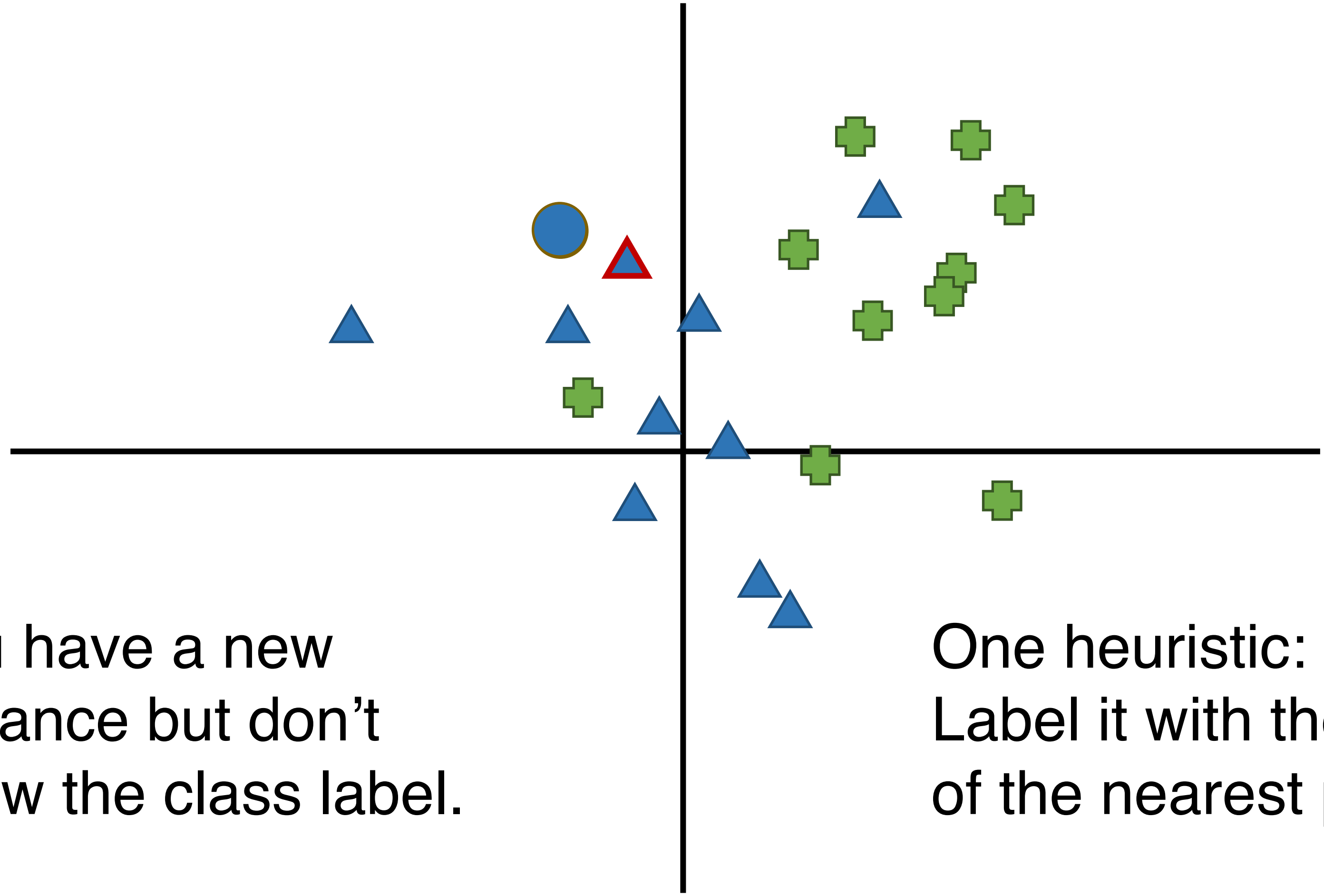
Suppose you have these 20 instances, labeled with one of two classes (blue or green)

Distance-based Prediction



You have a new instance but don't know the class label.

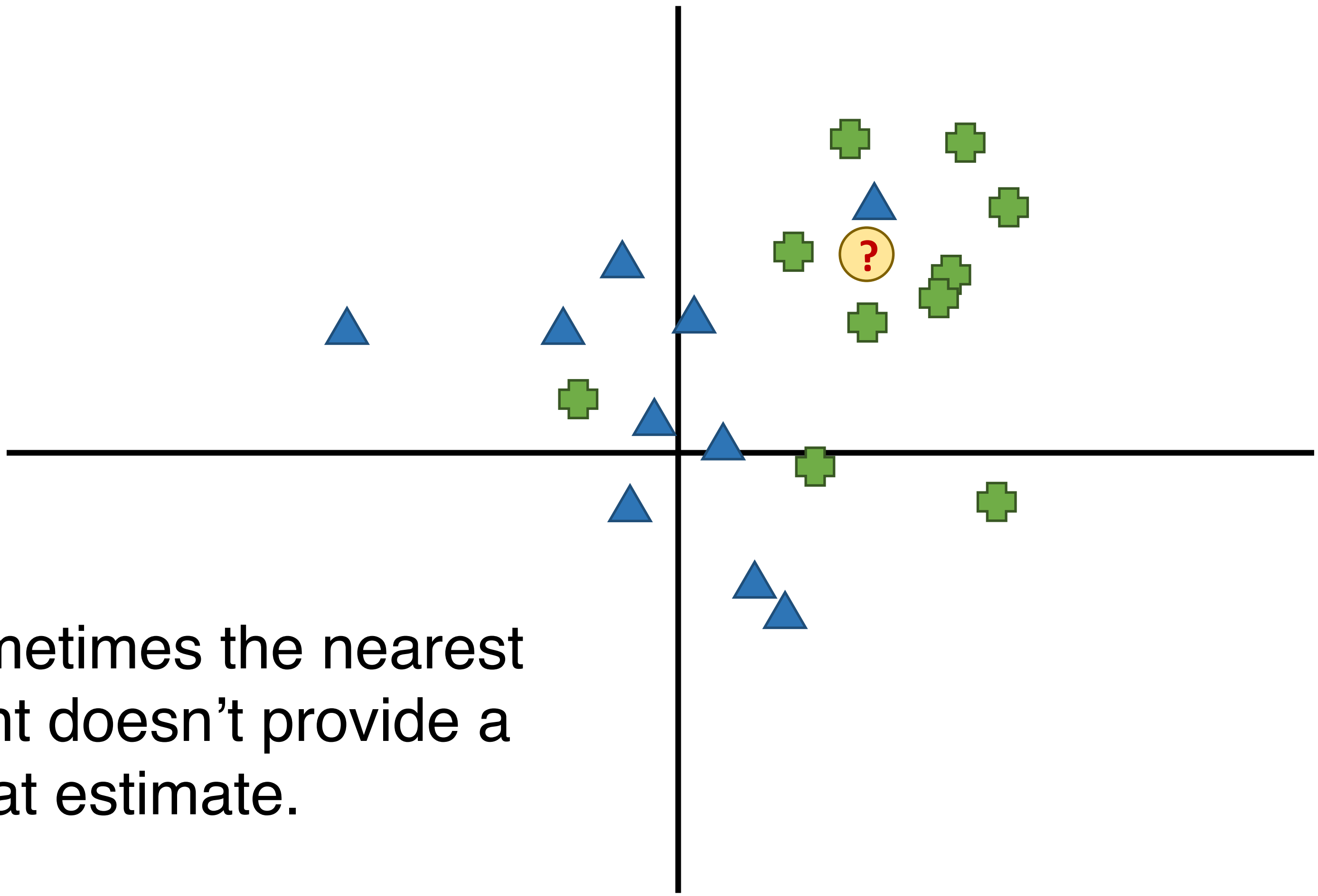
Distance-based Prediction



You have a new instance but don't know the class label.

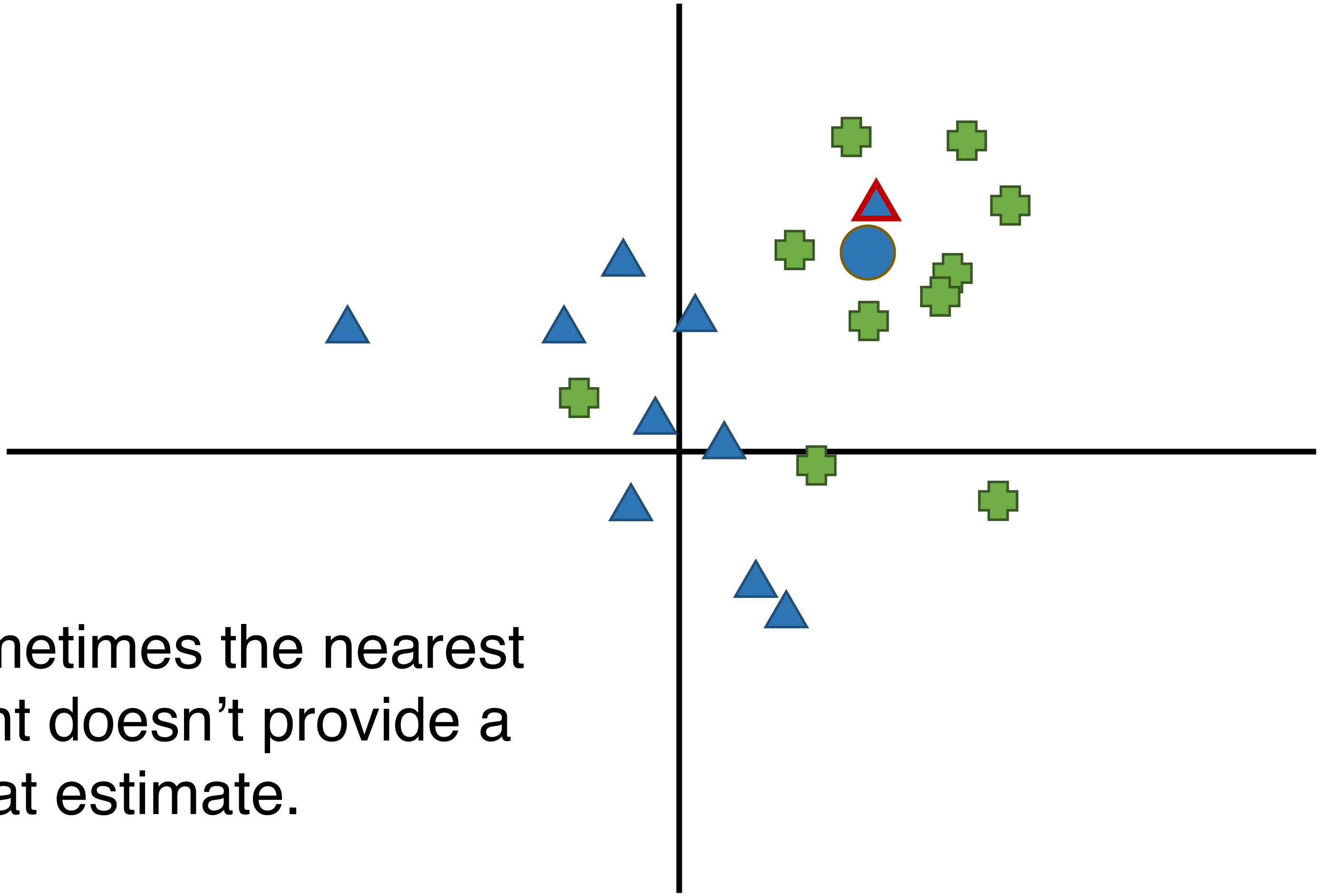
One heuristic:
Label it with the label of the nearest point.

Distance-based Prediction



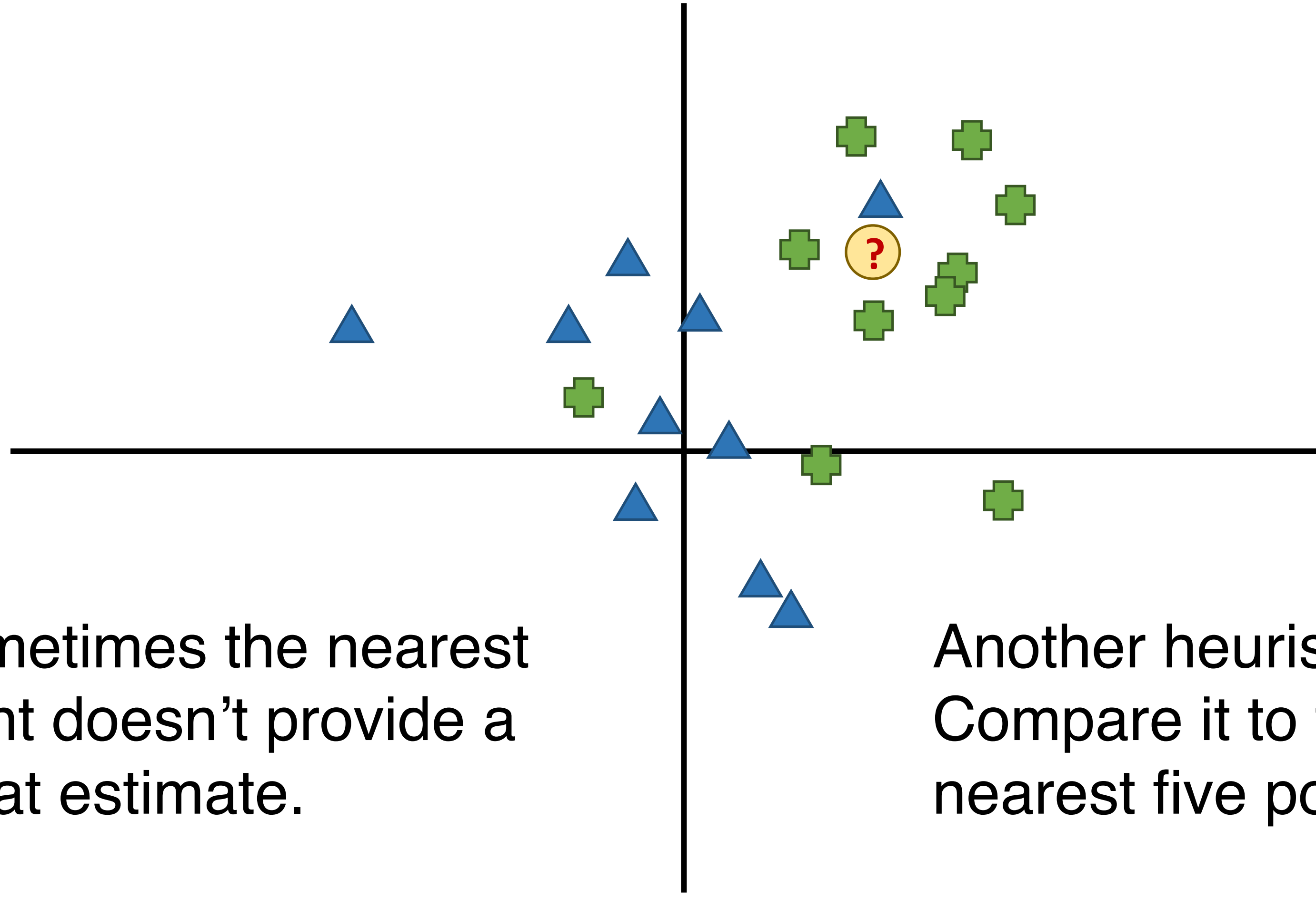
Sometimes the nearest point doesn't provide a great estimate.

Distance-based Prediction



Sometimes the nearest point doesn't provide a great estimate.

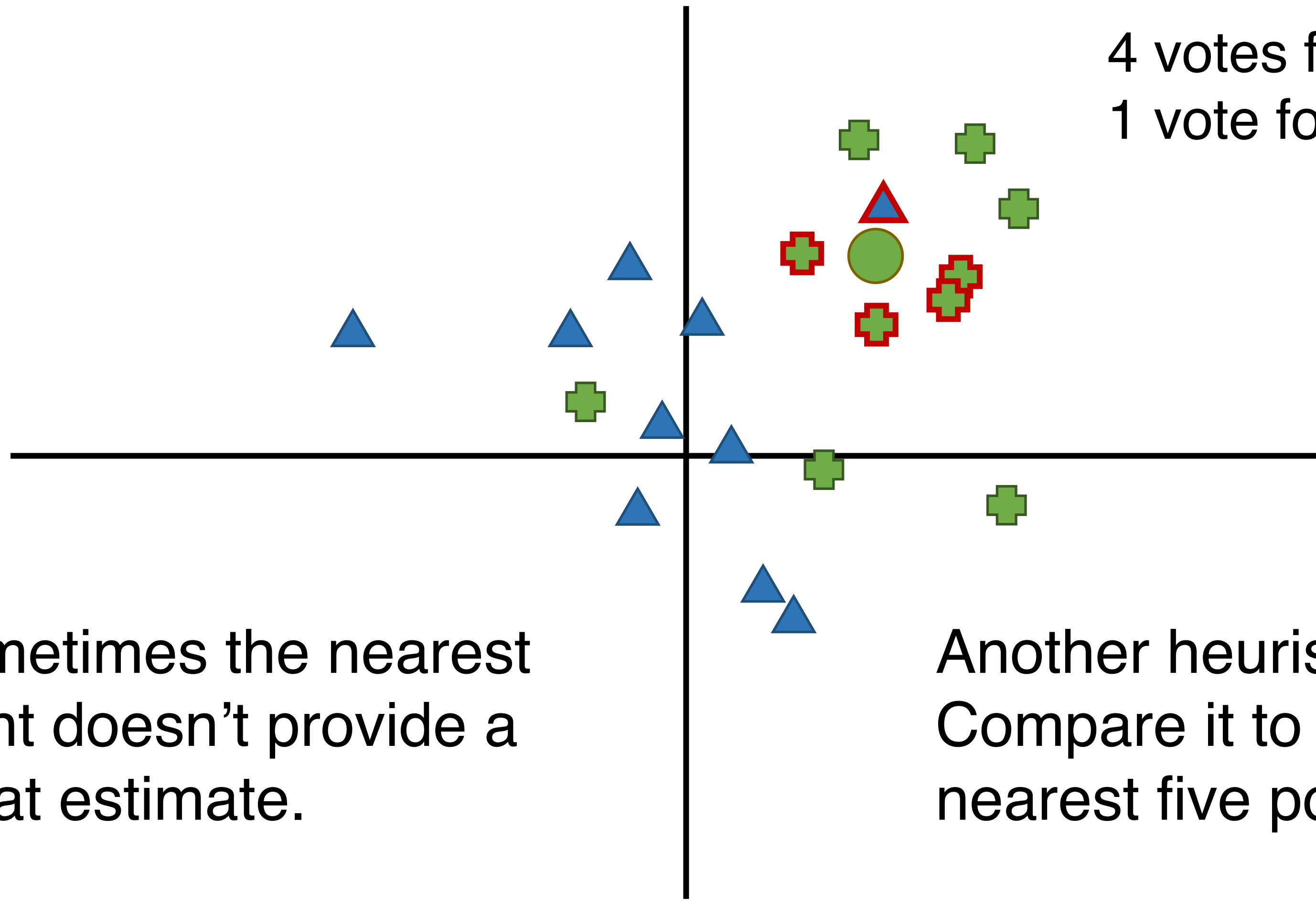
Distance-based Prediction



Sometimes the nearest point doesn't provide a great estimate.

Another heuristic: Compare it to the nearest five points.

Distance-based Prediction



4 votes for green
1 vote for blue

Sometimes the nearest point doesn't provide a great estimate.

Another heuristic: Compare it to the nearest five points.

Distance-based Prediction

The **k-nearest neighbors** (kNN) algorithm classifies an instance as follows:

1. Find the *k* labeled instances that have the lowest distance to the unlabeled instance
2. Return the majority class (most common label) in the set of *k* nearest instances

Can also be used for regression instead of classification (but less common)

- Replace “majority class” in step 2 above with “average value”

Distance-based Prediction

When you run the kNN algorithm, you have to decide what *k* should be.

Mostly an empirical question; trial and error experimentally.

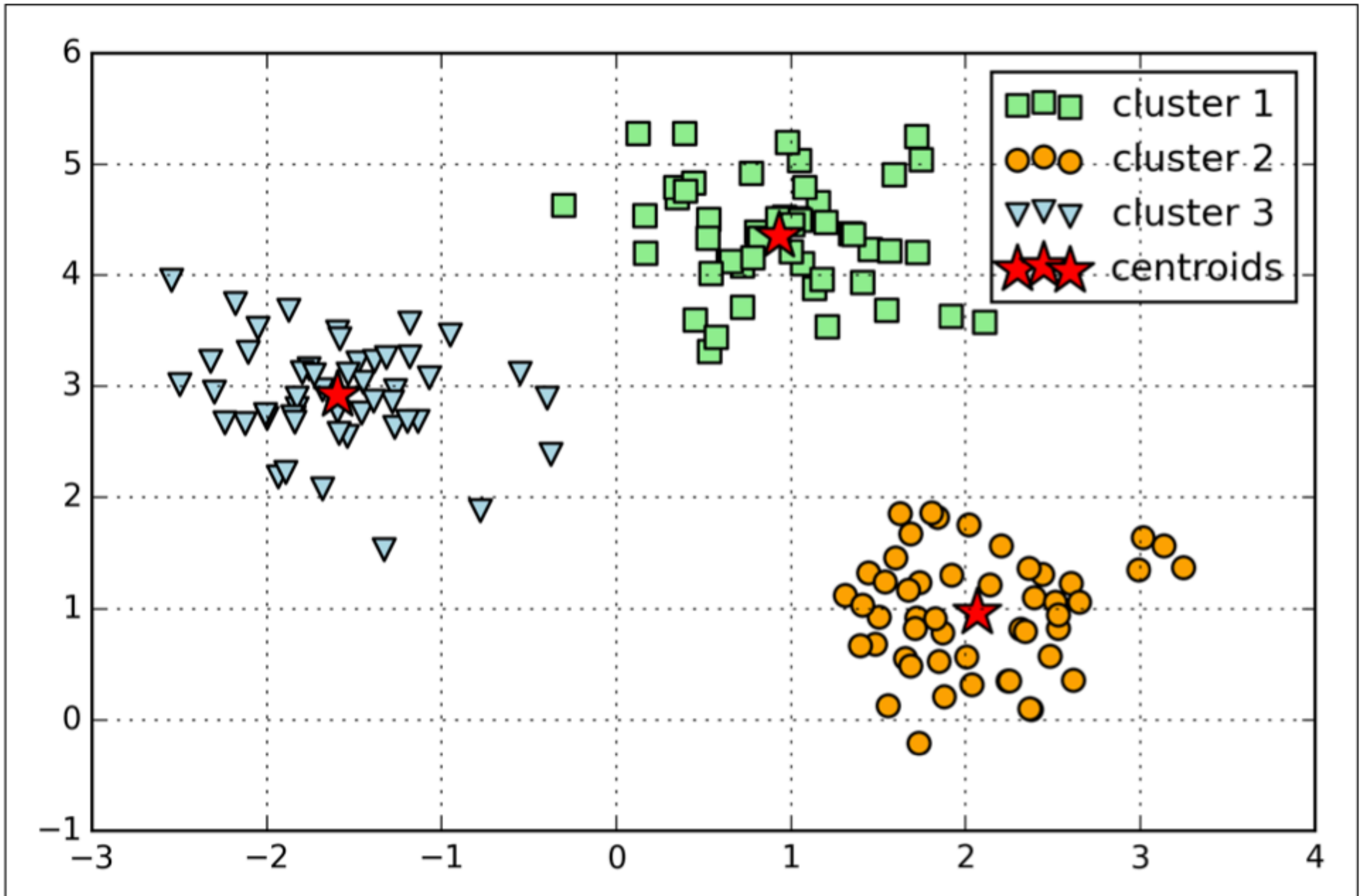
- If k is too small, prediction will sensitive to noise.
- If k is too large, algorithm loses the local context that makes it work.

Distance-based Prediction

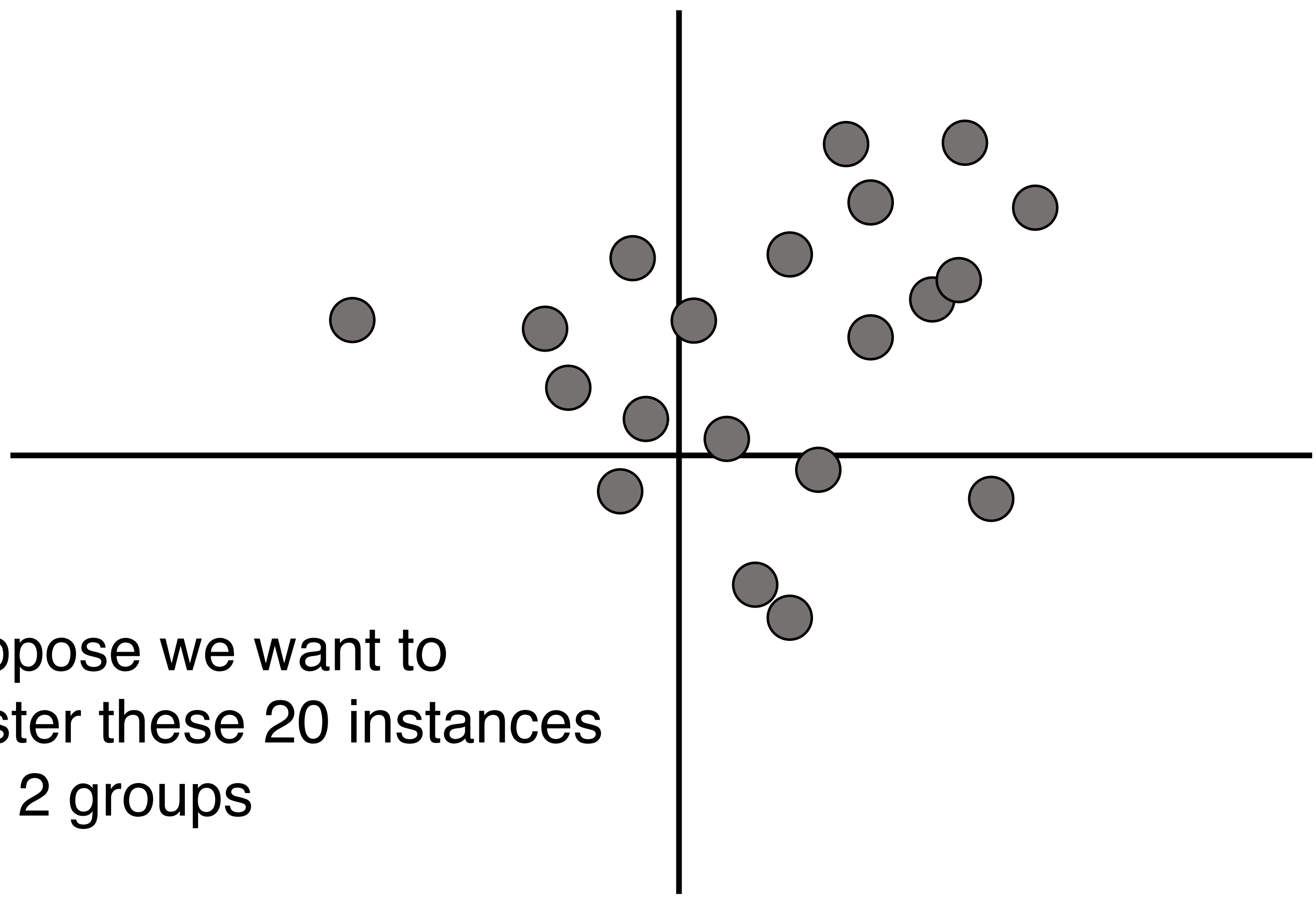
Common variant of kNN:
weigh the nearest neighbors by their distance

- (e.g., when calculating the majority class, give more votes to the instances that are closest)

k-means Clustering

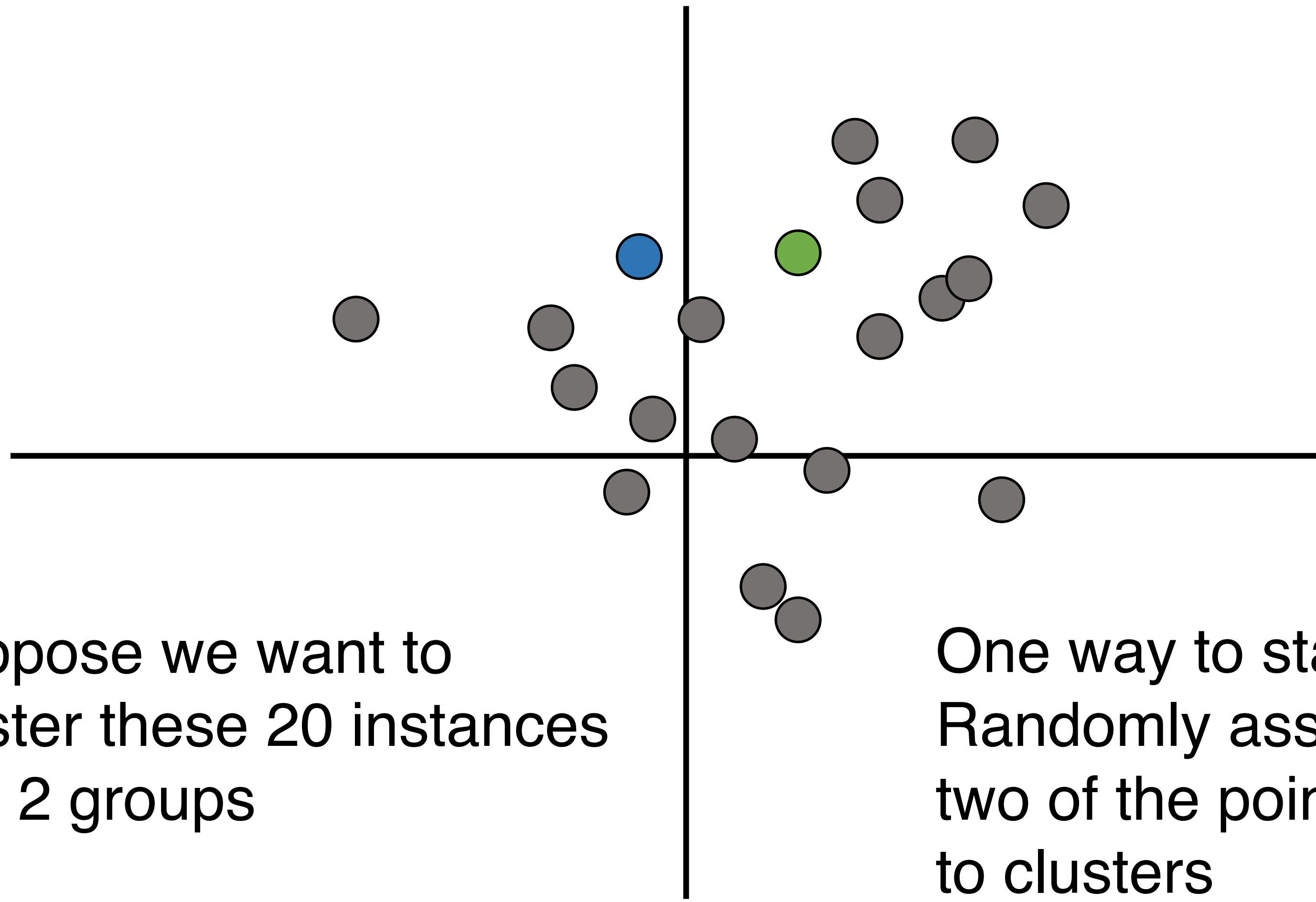


k-means Clustering



Suppose we want to cluster these 20 instances into 2 groups

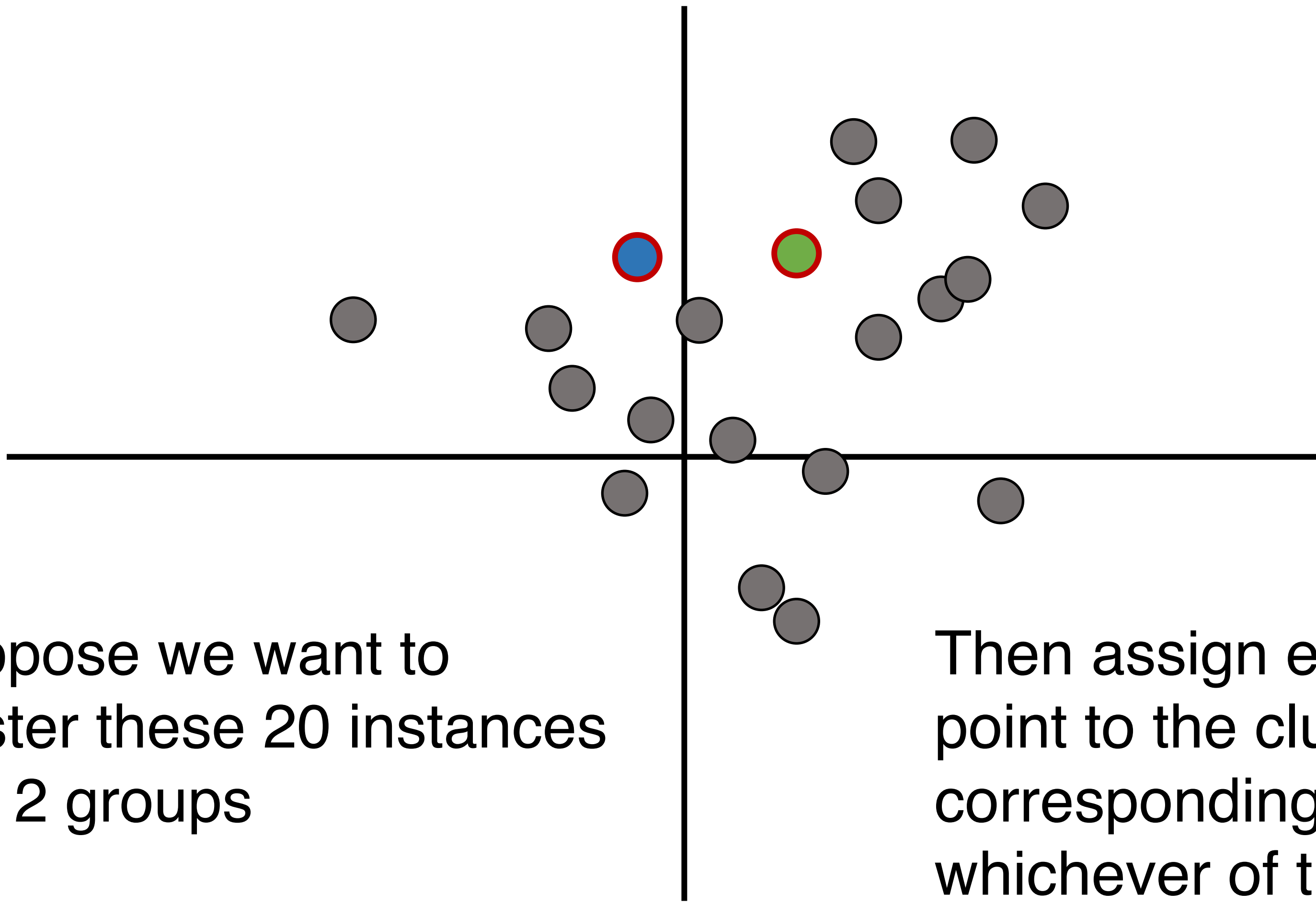
k-means Clustering



Suppose we want to cluster these 20 instances into 2 groups

One way to start: Randomly assign two of the points to clusters

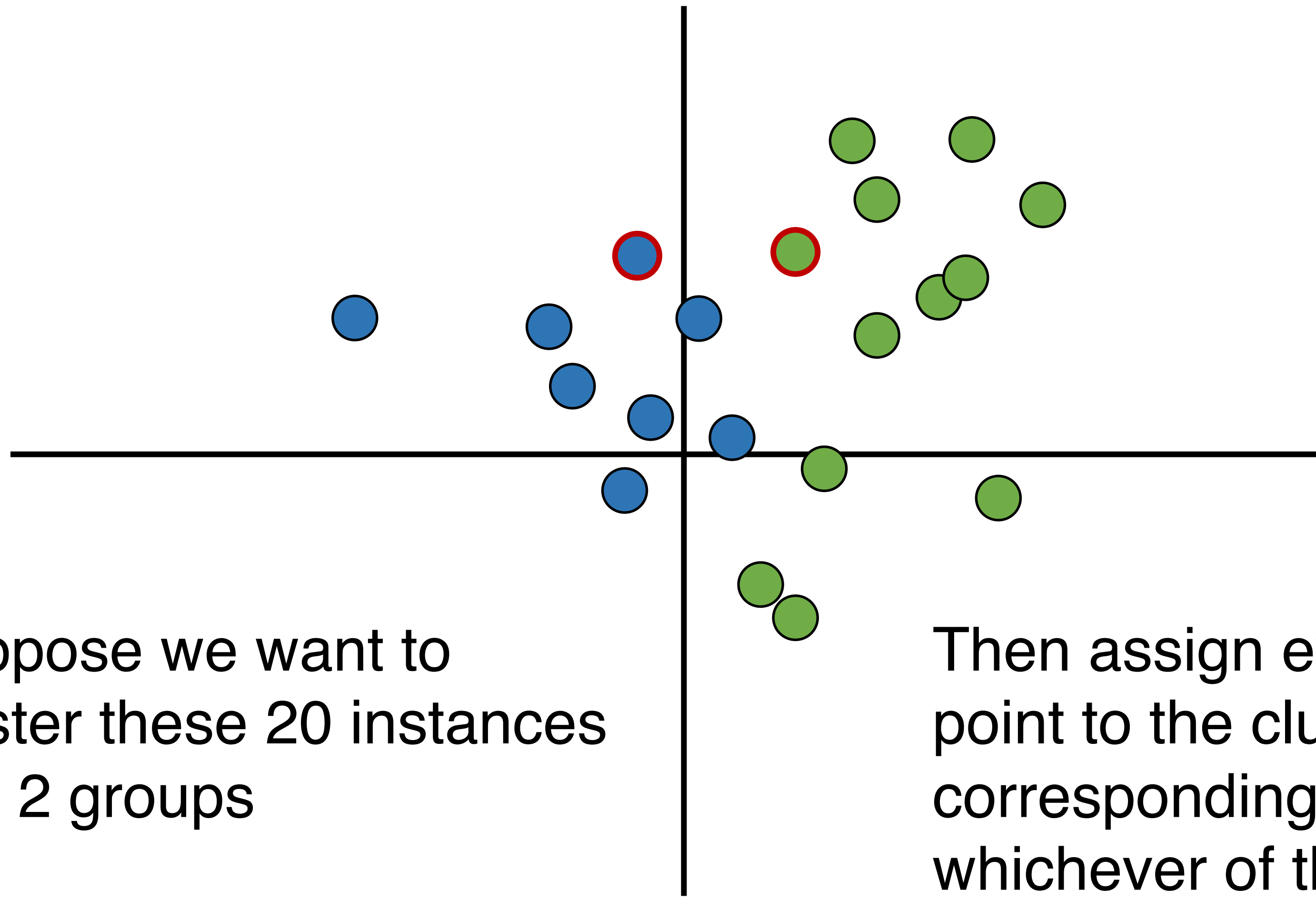
k-means Clustering



Suppose we want to cluster these 20 instances into 2 groups

Then assign every point to the cluster corresponding to whichever of the two points it is closer to

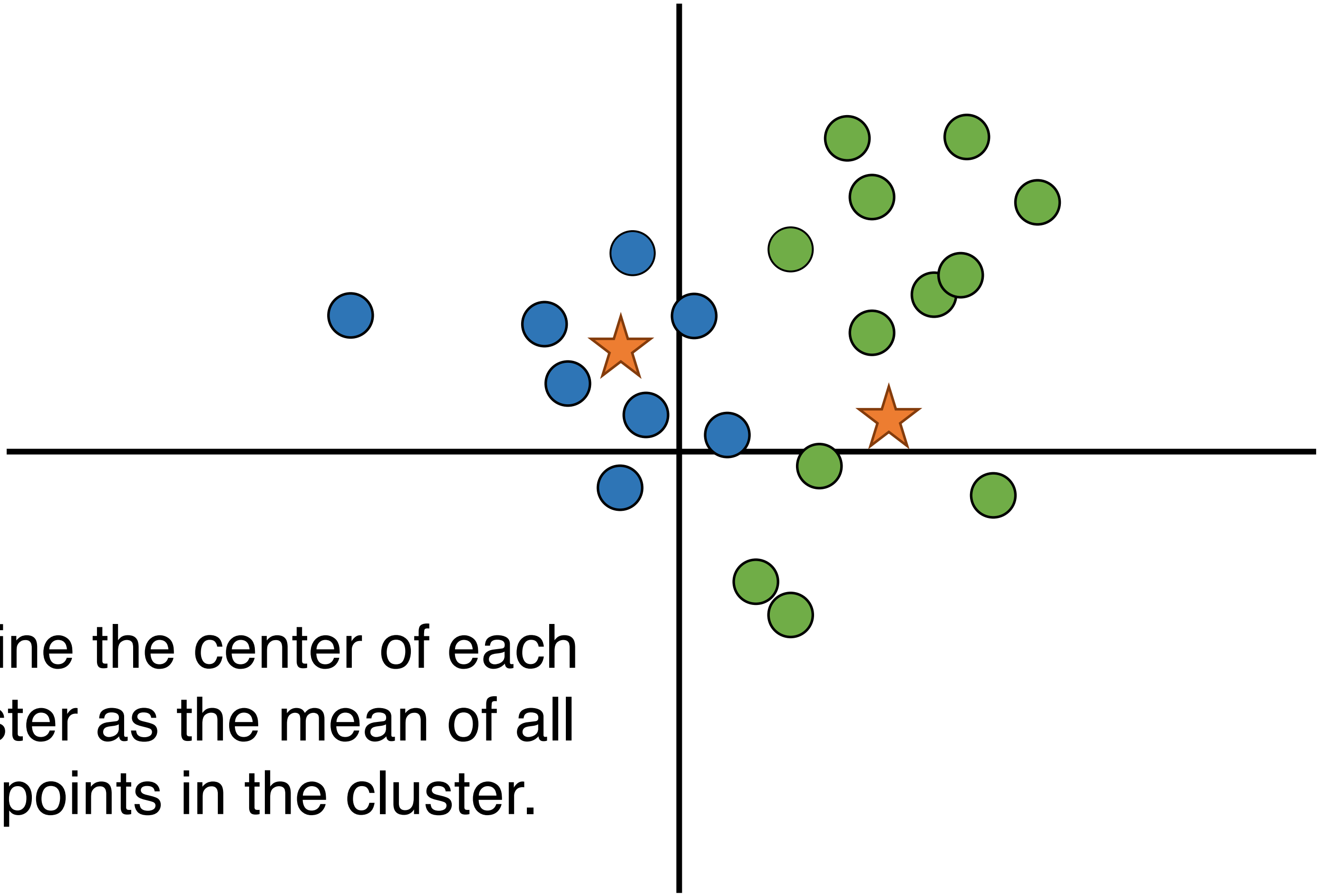
k-means Clustering



Suppose we want to cluster these 20 instances into 2 groups

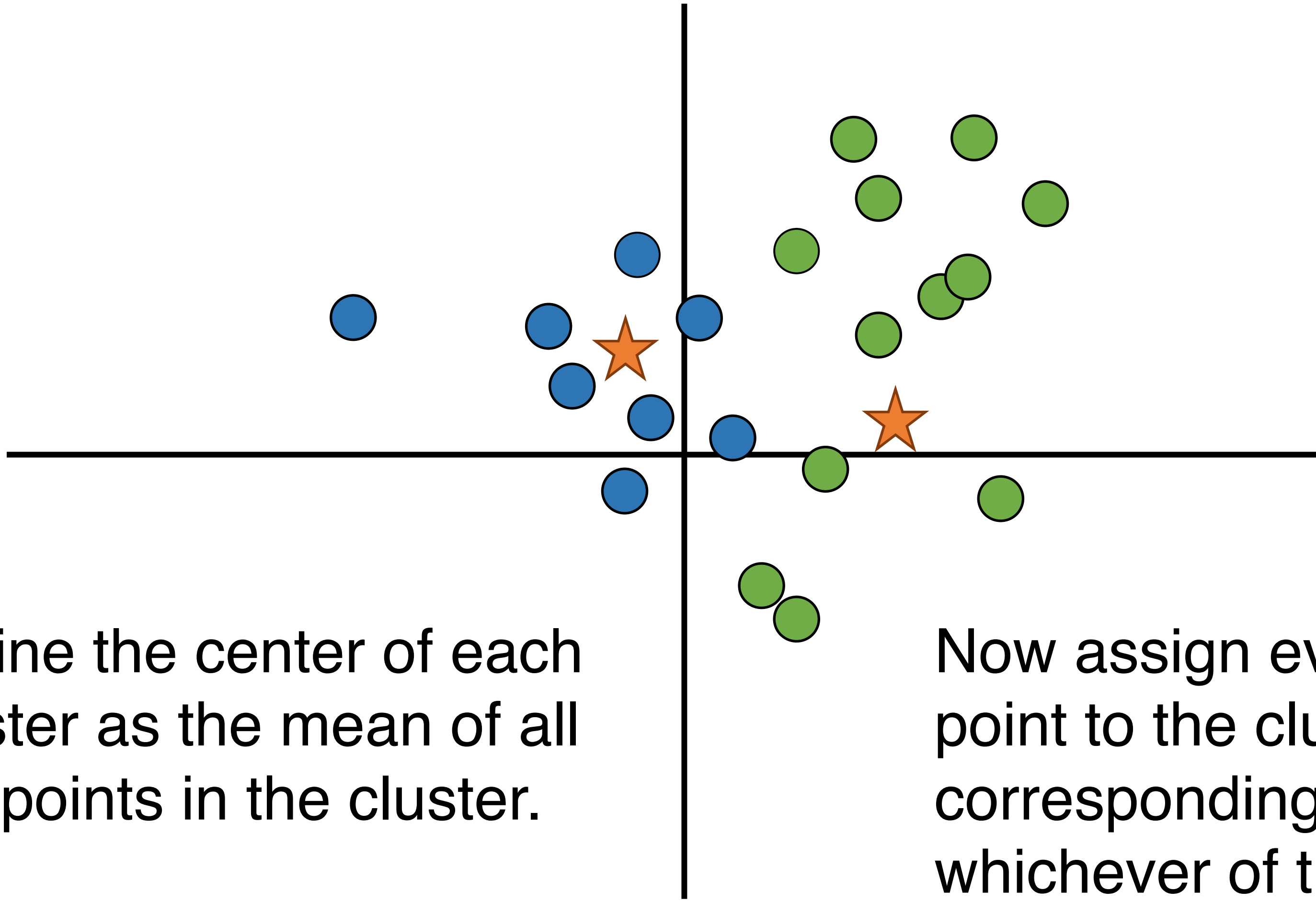
Then assign every point to the cluster corresponding to whichever of the two points it is closer to

k-means Clustering



Define the center of each cluster as the mean of all the points in the cluster.

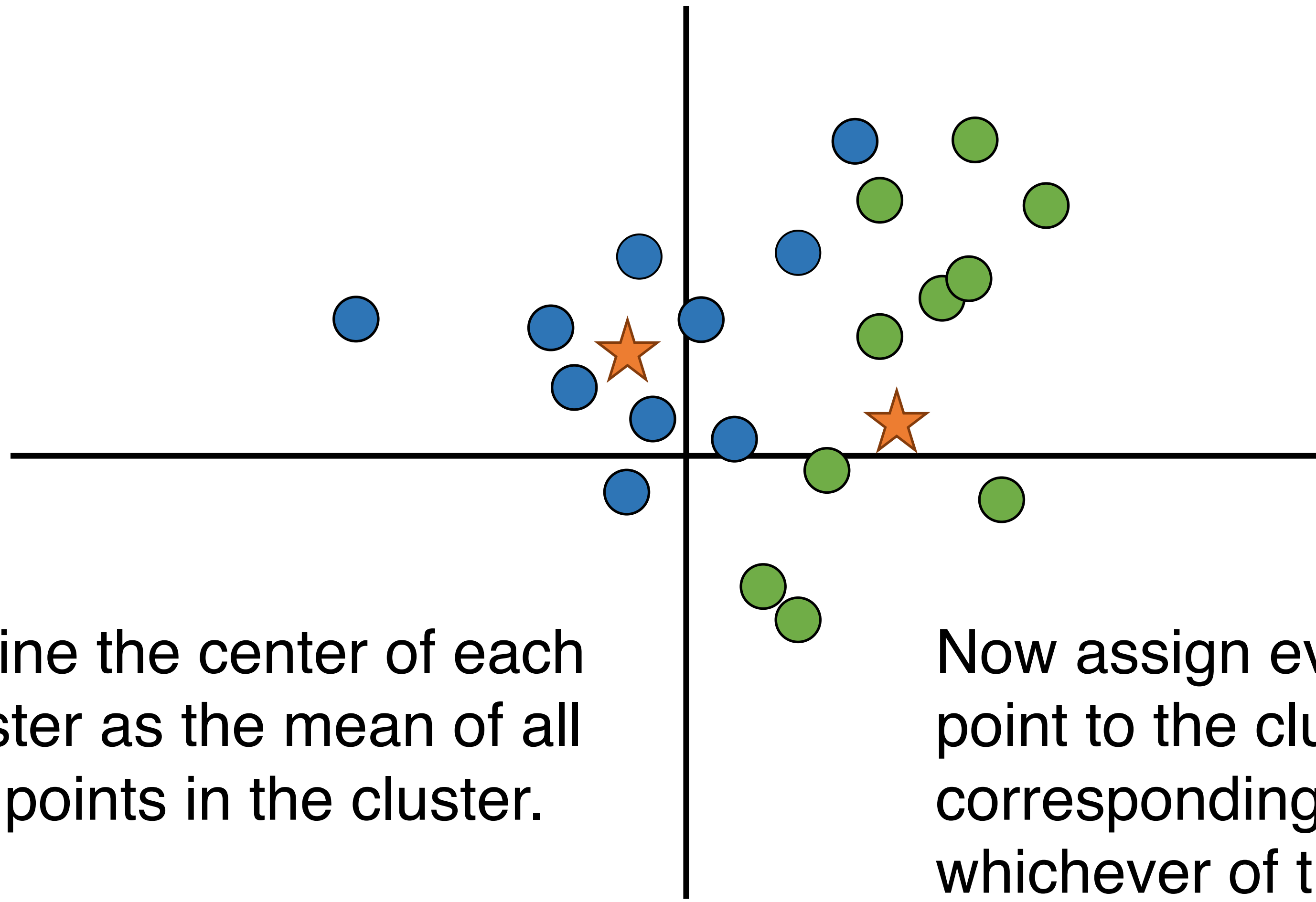
k-means Clustering



Define the center of each cluster as the mean of all the points in the cluster.

Now assign every point to the cluster corresponding to whichever of the two centers it is closer to

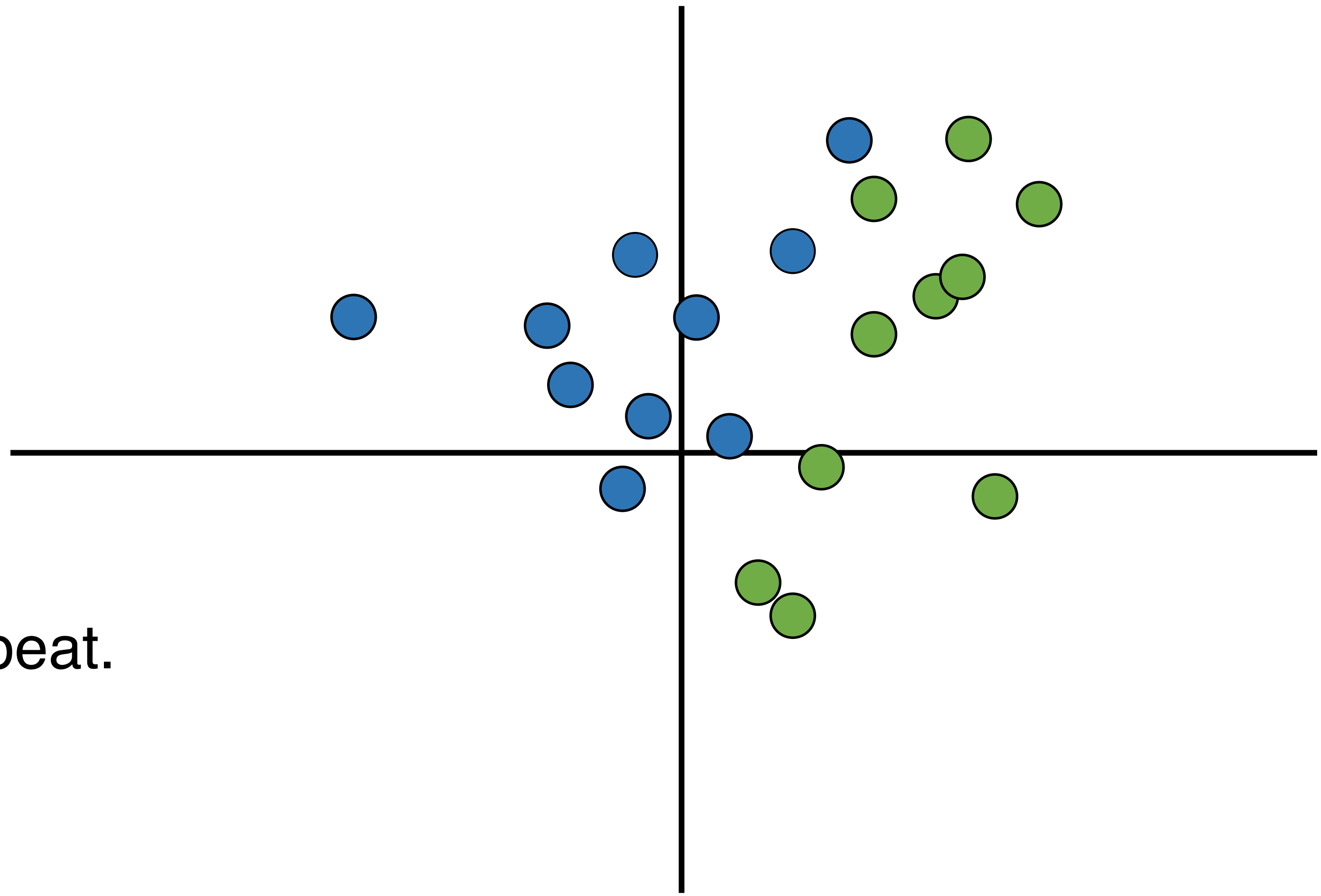
k-means Clustering



Define the center of each cluster as the mean of all the points in the cluster.

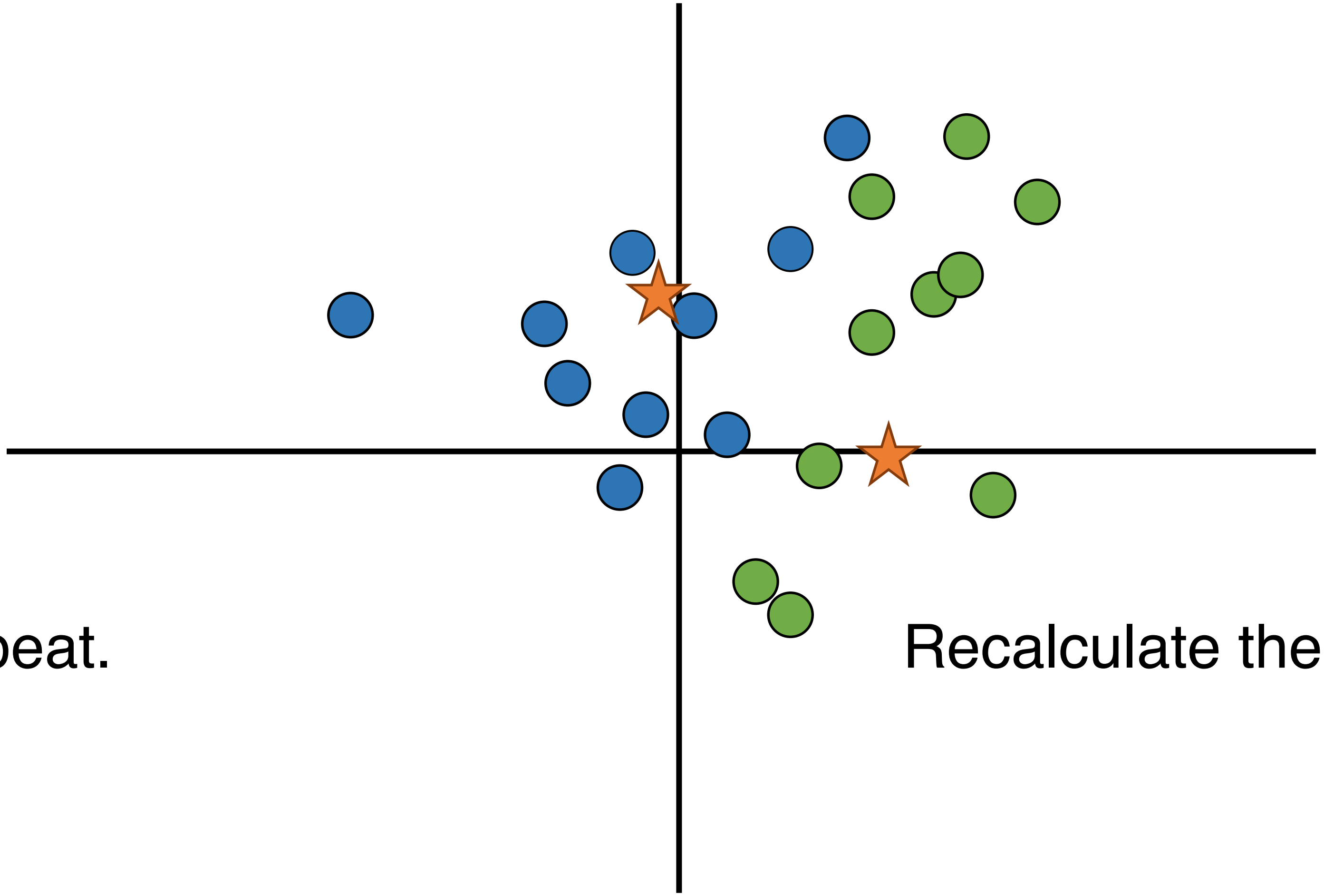
Now assign every point to the cluster corresponding to whichever of the two centers it is closer to

k-means Clustering



Repeat.

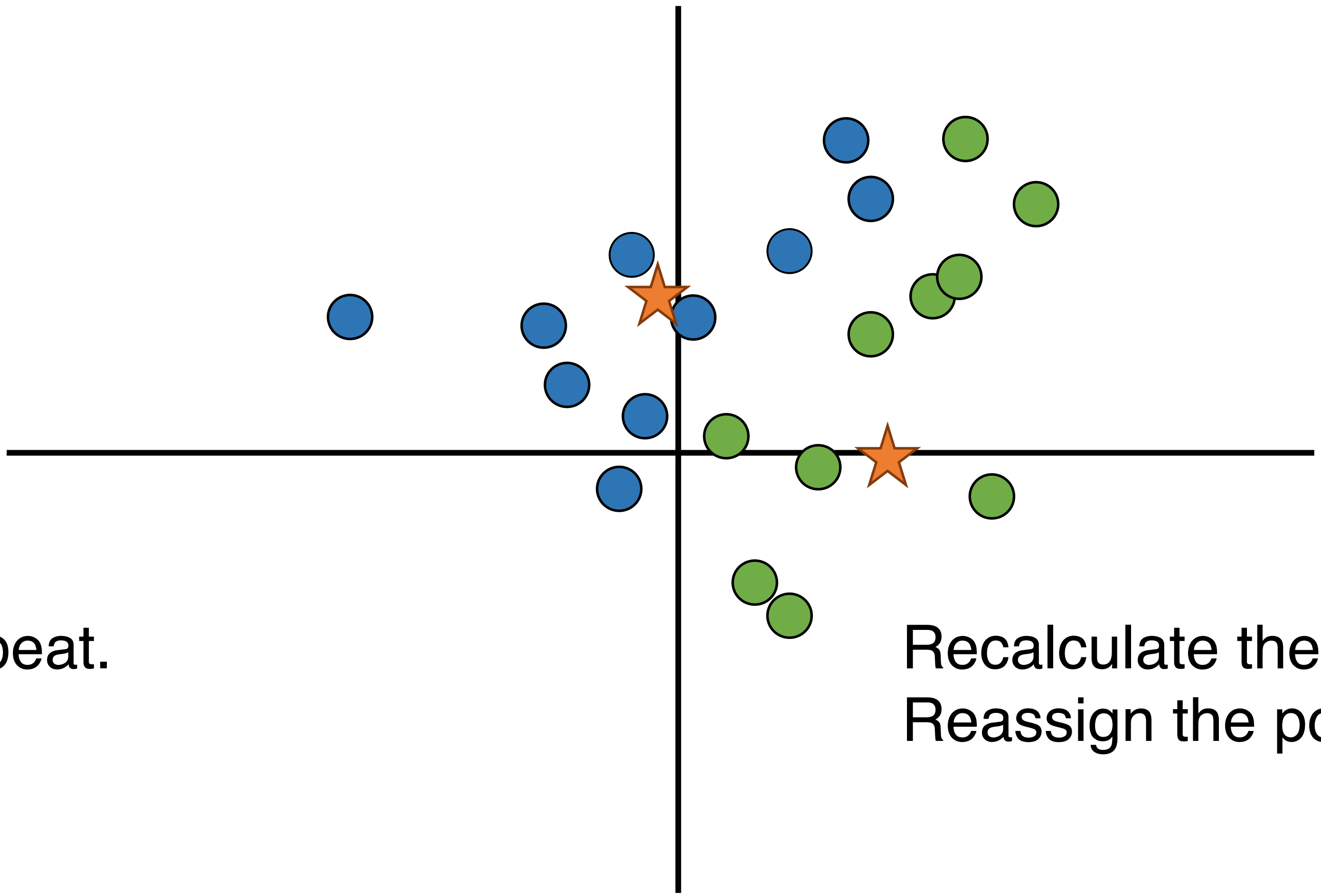
k-means Clustering



Repeat.

Recalculate the means.

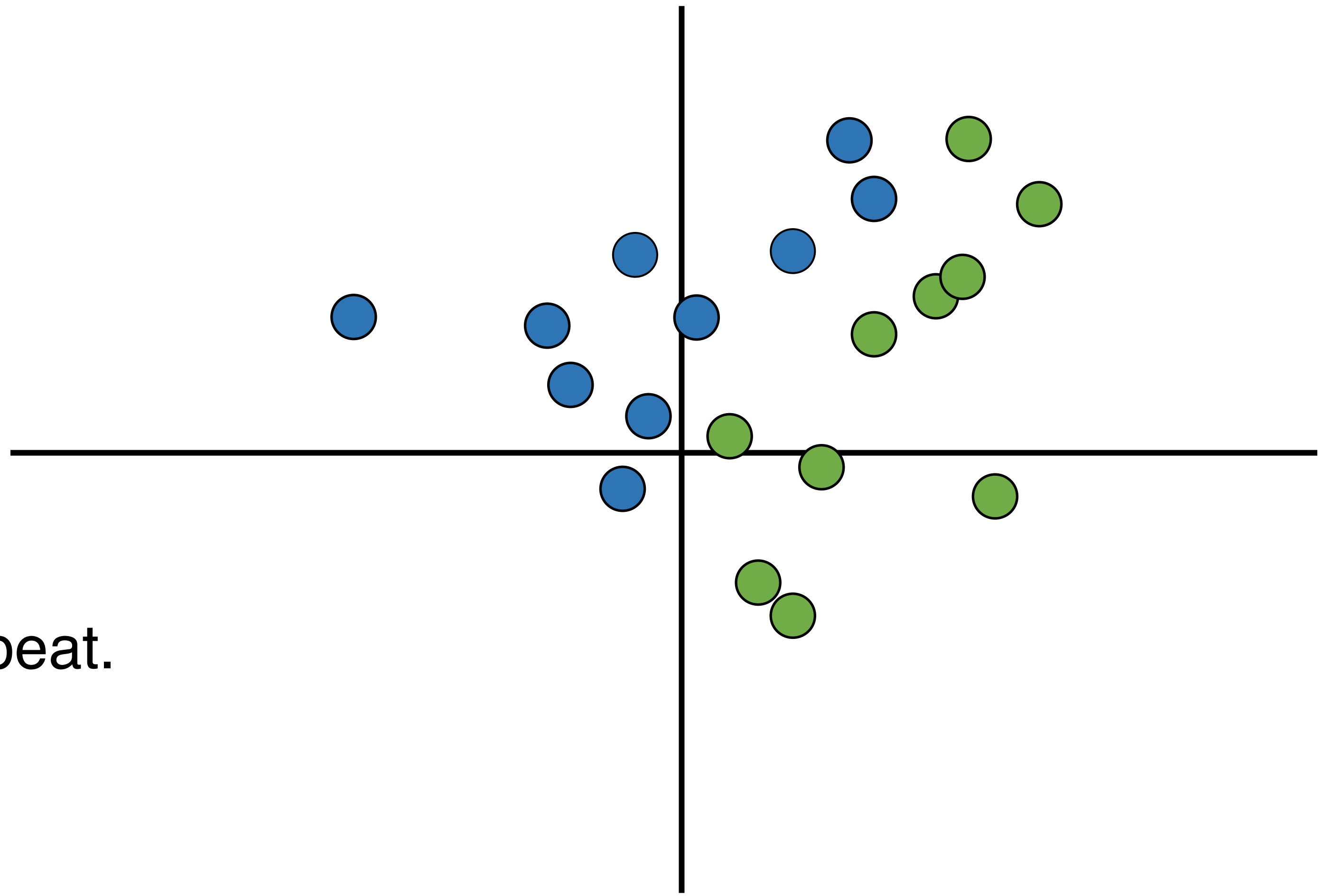
k-means Clustering



Repeat.

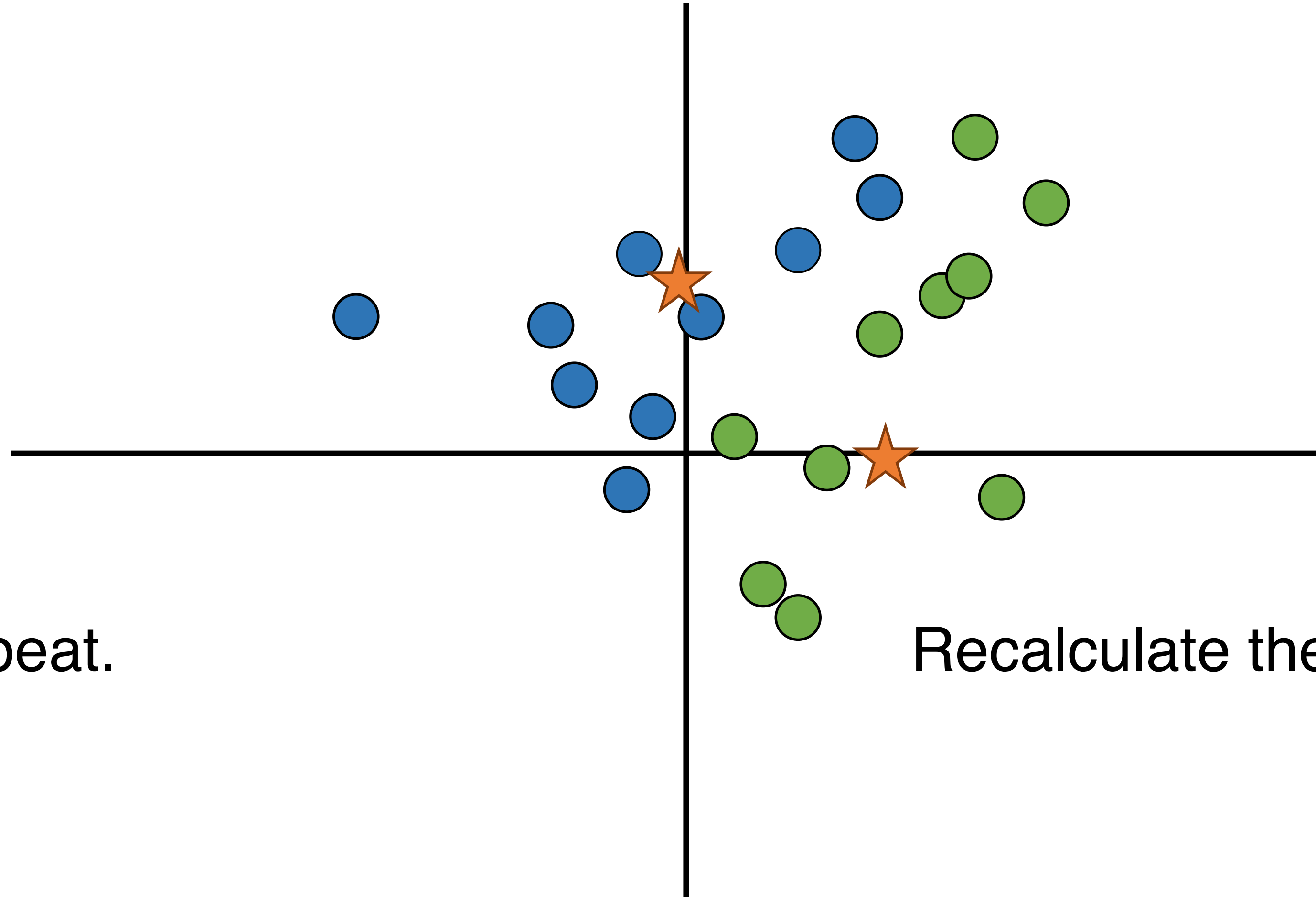
Recalculate the means.
Reassign the points.

k-means Clustering



Repeat.

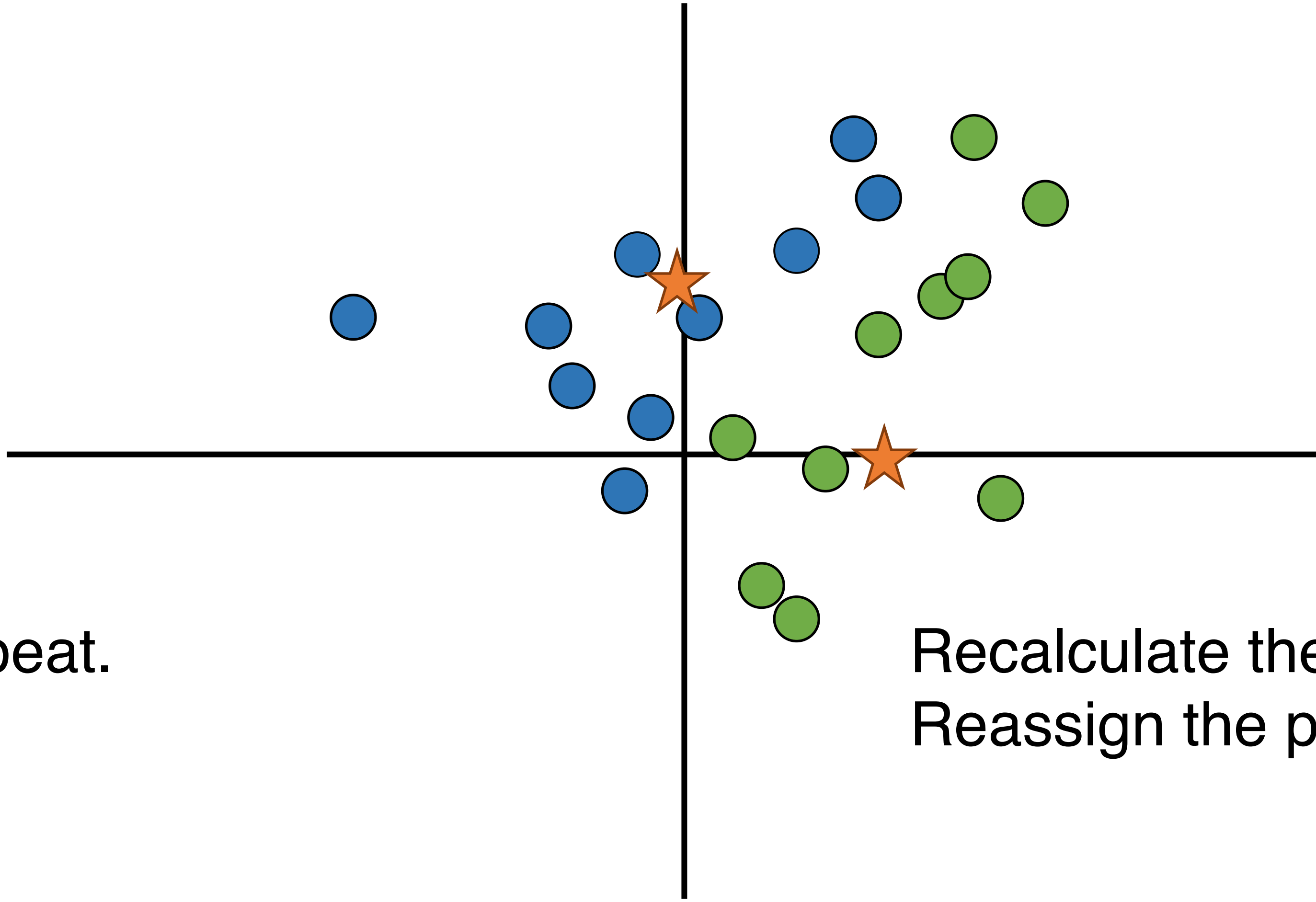
k-means Clustering



Repeat.

Recalculate the means.

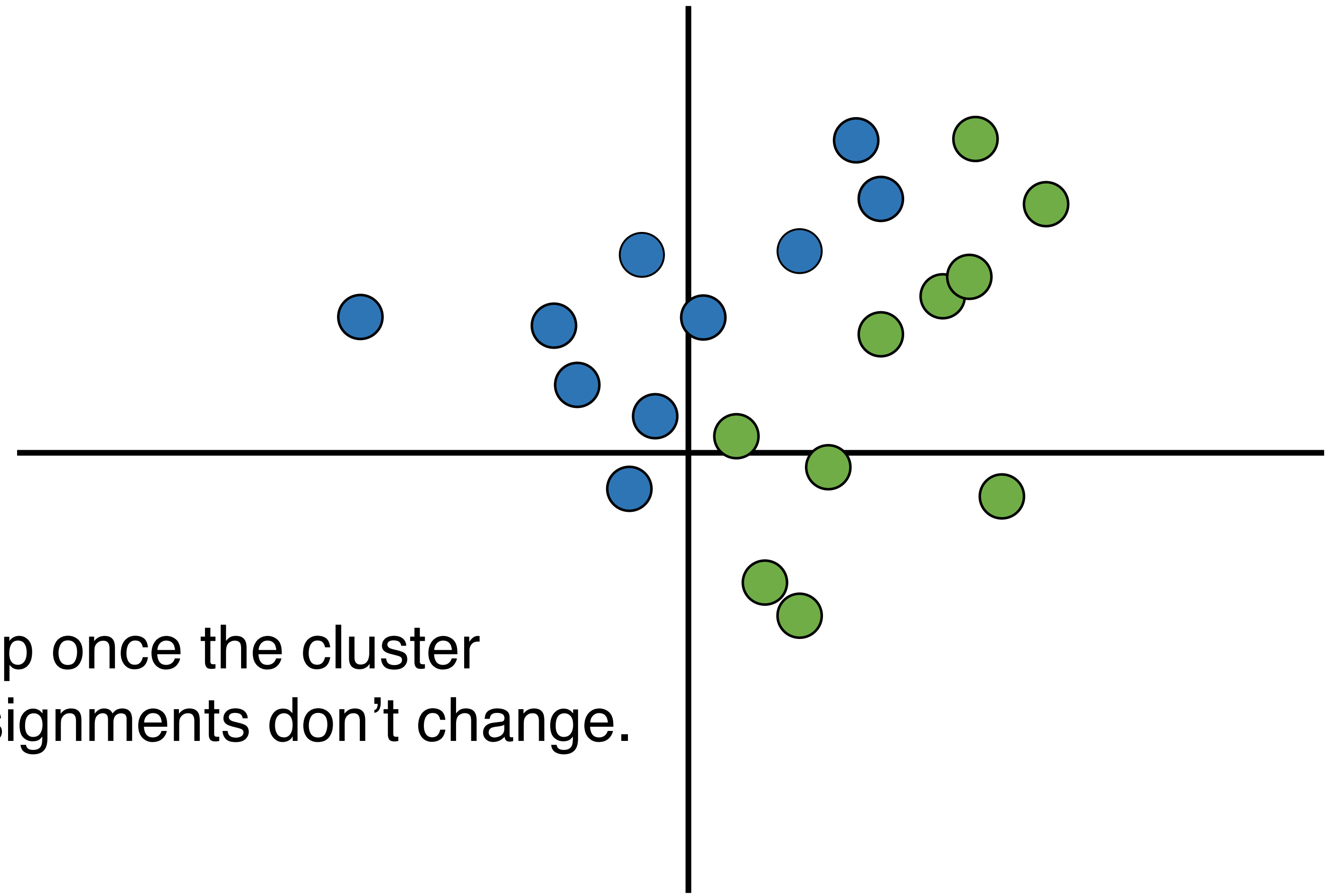
k-means Clustering



Repeat.

Recalculate the means.
Reassign the points.

k-means Clustering



Stop once the cluster assignments don’t change.

k-means Clustering

1. Initialize the cluster means
2. Repeat until assignments stop changing:
   a) Assign each instance to the cluster whose mean is nearest to the instance
   b) Update the cluster means based on the new cluster assignments:

$$\frac{1}{|S_i|} \sum_{x_j \in S_i} x_j$$

where S_i is the set of instances in cluster i, and $|S_i|$ is the number of instances in the cluster.

k-means Clustering

How to initialize? Two common approaches:

- Randomly assign each instance to a cluster and calculate the means.
- Pick *k* points at random and treat them as the cluster means.
    - This is the approach used in the illustration in the previous slides.
    - This approach generally works better than the previous approach (leads to initial cluster means that are more spread out)

Note that both of these approaches involve randomness and will not always lead to the same solution each time!

k-means Clustering

How to choose *k*? Similar challenge as in k-NN.

Usually trial-and-error + some intuition about what the dataset looks like.

Some clustering algorithms can automatically figure out the number of clusters.

- Also based on distance.
- General idea: if points within a cluster are still far apart, the cluster should probably be split into more clusters.

Recap

Both k-NN and k-means require some definition of distance between points.

- Euclidean distance most common.
- There are lots of others
  (many implemented in *sklearn*)

While the illustrations had only two dimensions, the algorithms apply to any number of dimensions, using the definition of Euclidean distance that we learned today.