Carnegie Mellon University
Heinz College

95-865 Unstructured Data Analytics

Lecture 6: Wrap up manifold learning, intro to clustering

Slides by George H. Chen

Administrivia

- 😱 Quiz 1 is this Friday during your recitation slot (5pm-6:20pm HBH A301)
    - Material coverage: up to today's lecture (Mar 24 lecture)
- HW1 is due tonight
    - Friendly reminder: please look at the PDF that you're submitting before you submit it **(check that everything displays correctly)**
- You now have access to 13 real past Quiz 1s (see my Canvas announcements)
- There's an optional Quiz 1 review session tomorrow (Tuesday Mar 25) 7pm-8:30pm
    - It's over Zoom (check Canvas -> Zoom for the link)

Reminder of Quiz 1 (and Quiz 2) Format

Format:

- **In-person, on paper**
- Each quiz is **80 minutes**
- No electronics may be used during the exam
  (e.g., do not use a laptop, tablet, phone, calculator)
- Open notes (must be on paper and not electronic)

  There is no limit on how many sheets of notes you bring

Dimensionality Reduction for Visualization

- There are *many* methods (I've posted a link on the course webpage to a scikit-learn example using ~10 methods)
- PCA is very well-understood; the new axes can be interpreted
- Nonlinear dimensionality reduction (manifold learning):
  new axes may not really be all that interpretable
- PCA is good to try first (look at plot & explained variance ratios)
  - If PCA works poorly, then t-SNE could be a good 2nd thing to try
- If you have good reason to believe that only certain features matter, of course you could restrict your analysis to those!
- t-SNE can be annoying to use but is still very popular
  - Promising recently developed alternative: PaCMAP (Wang et al 2021) accounts for local and global structure simultaneously and also uses "mid-near" neighbors of points — link on course webpage

Let’s look at images

(Flashback) Multiple Documents

Choose a common vocabulary to use across all documents

☀ ☁ ☂ ☃

Text doc #1 $\longrightarrow$ [Feature vector #1]

Text doc #2 $\longrightarrow$ [Feature vector #2]

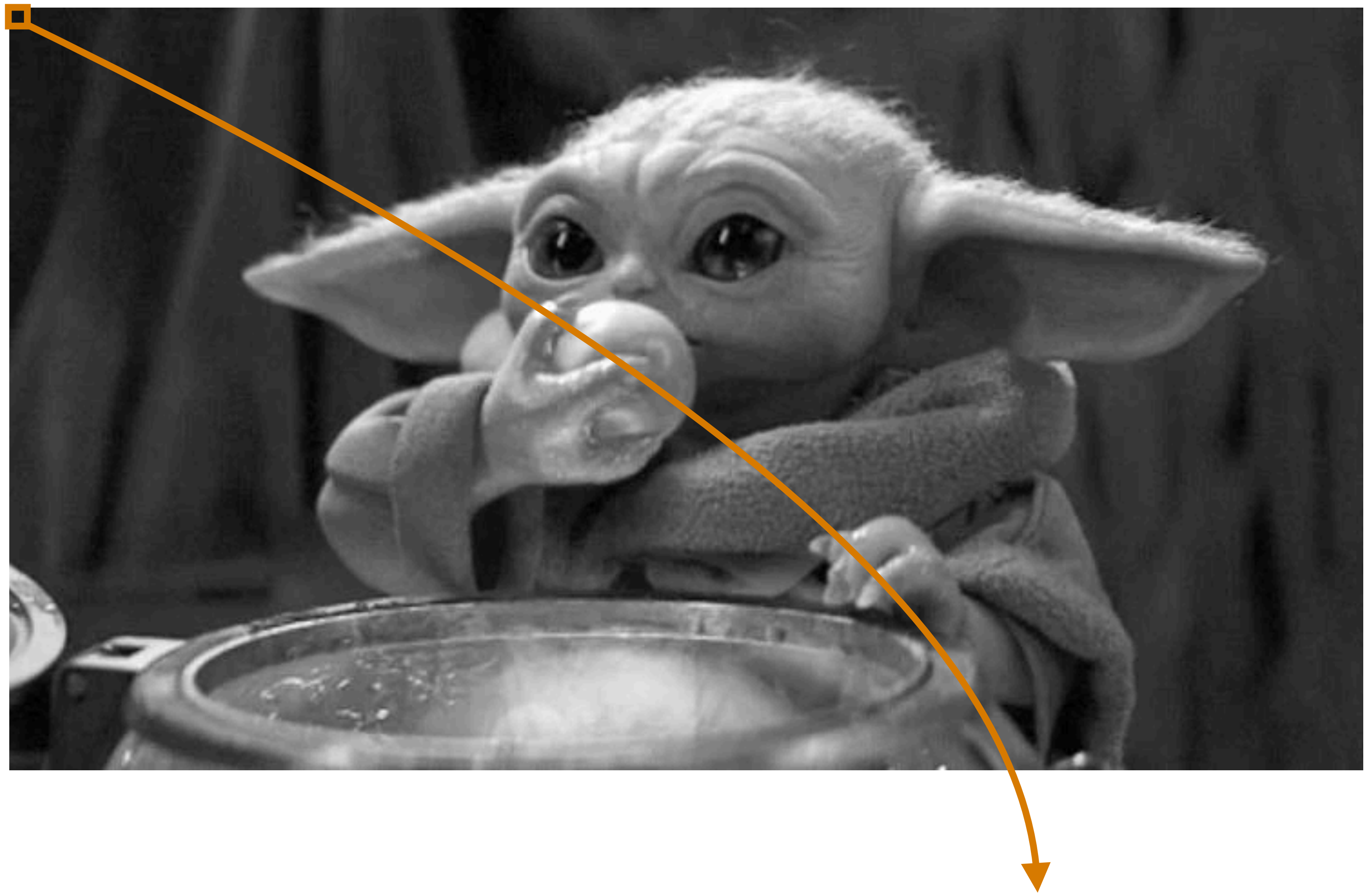
⋮

Text doc #*n* $\longrightarrow$ [Feature vector #*n*]

This idea of representing data as feature vectors is very general — not just for text!

Example: Representing an Image

Go row by row and look at pixel values



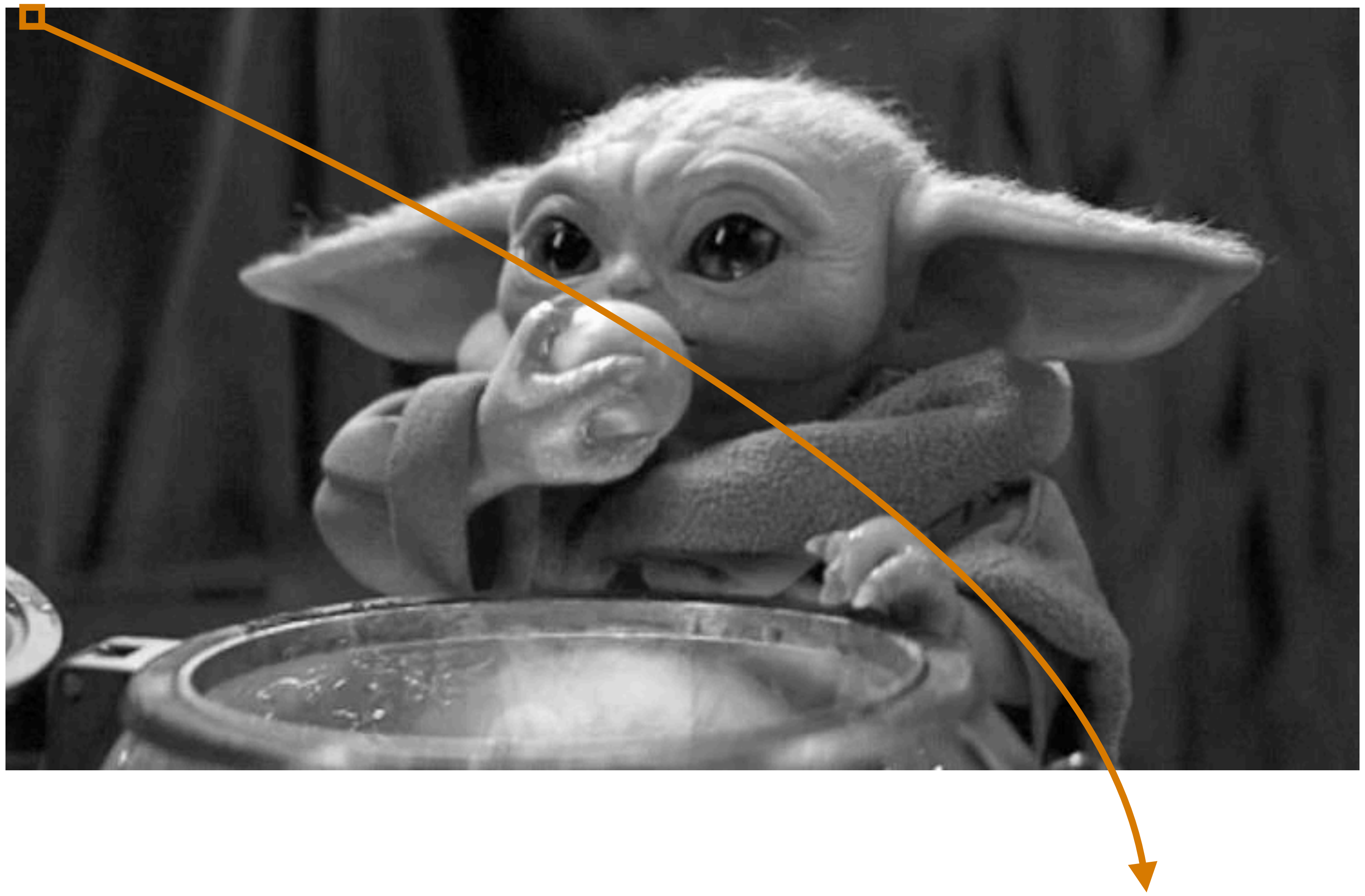
1: black
0: white

[1]

Image source: The Mandalorian

Example: Representing an Image

Go row by row and look at pixel values



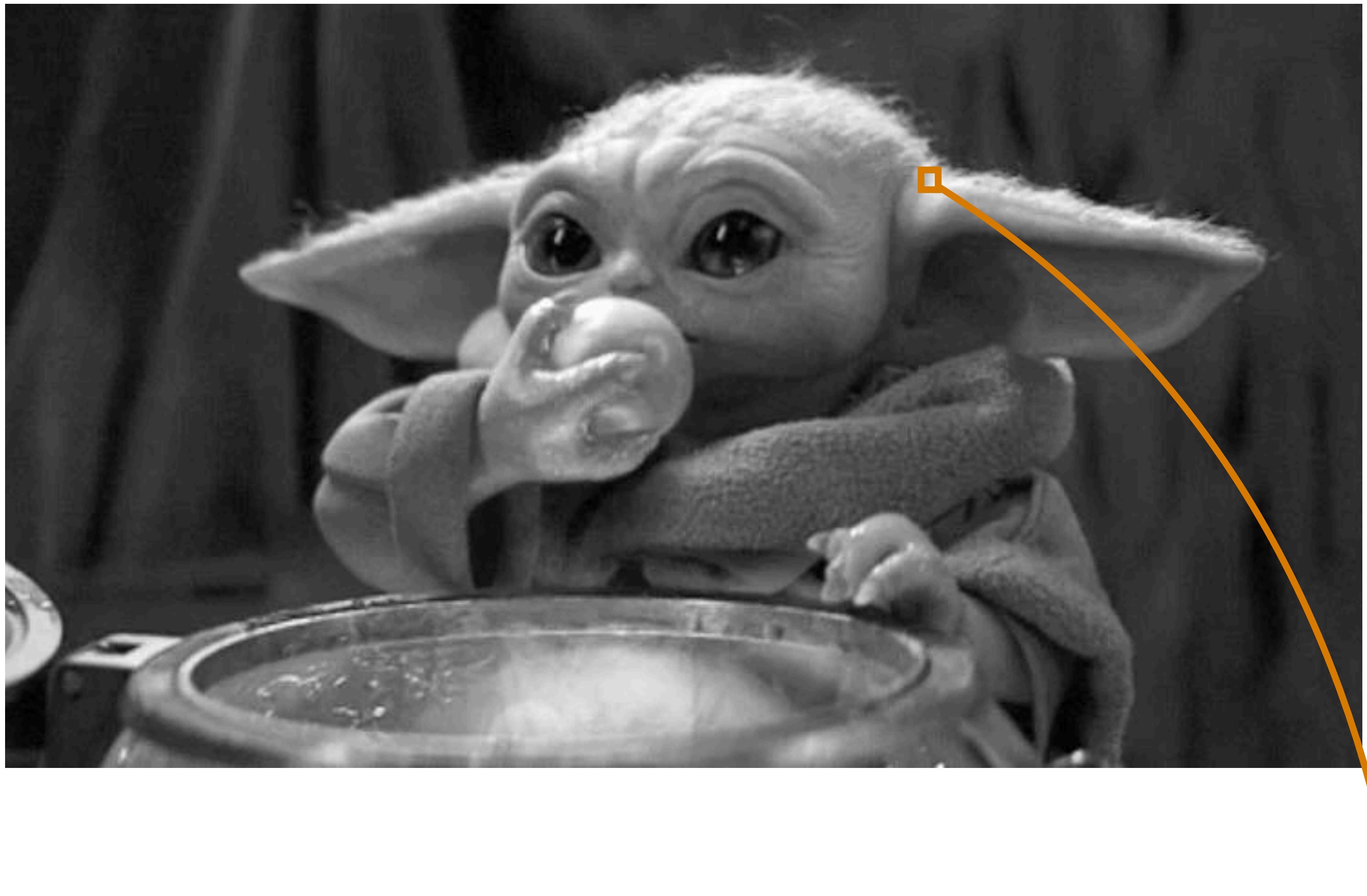
1: black
0: white

[1   0.9   ]

Image source: The Mandalorian

Example: Representing an Image

Go row by row and look at pixel values



1: black
0: white

$[\ 1 \quad 0.9 \quad \cdots \quad 0.1 \qquad\qquad]$

Image source: The Mandalorian

Example: Representing an Image

Go row by row and look at pixel values

1: black
0: white

$$\left[\ 1 \quad 0.9 \quad \cdots \quad 0.1 \quad \cdots \quad 0.9\ \right]$$

# dimensions = image height × image width

Image source: The Mandalorian

Very high dimensional!

Terminology Remark

$$[\ 1 \quad 0.9 \quad \cdots \quad 0.1 \quad \cdots \quad 0.9\ ]$$

⚠️ We use "dimension" to means two different things:

- number of axes we can index into for a table/array (e.g., 2D means there are rows & columns)

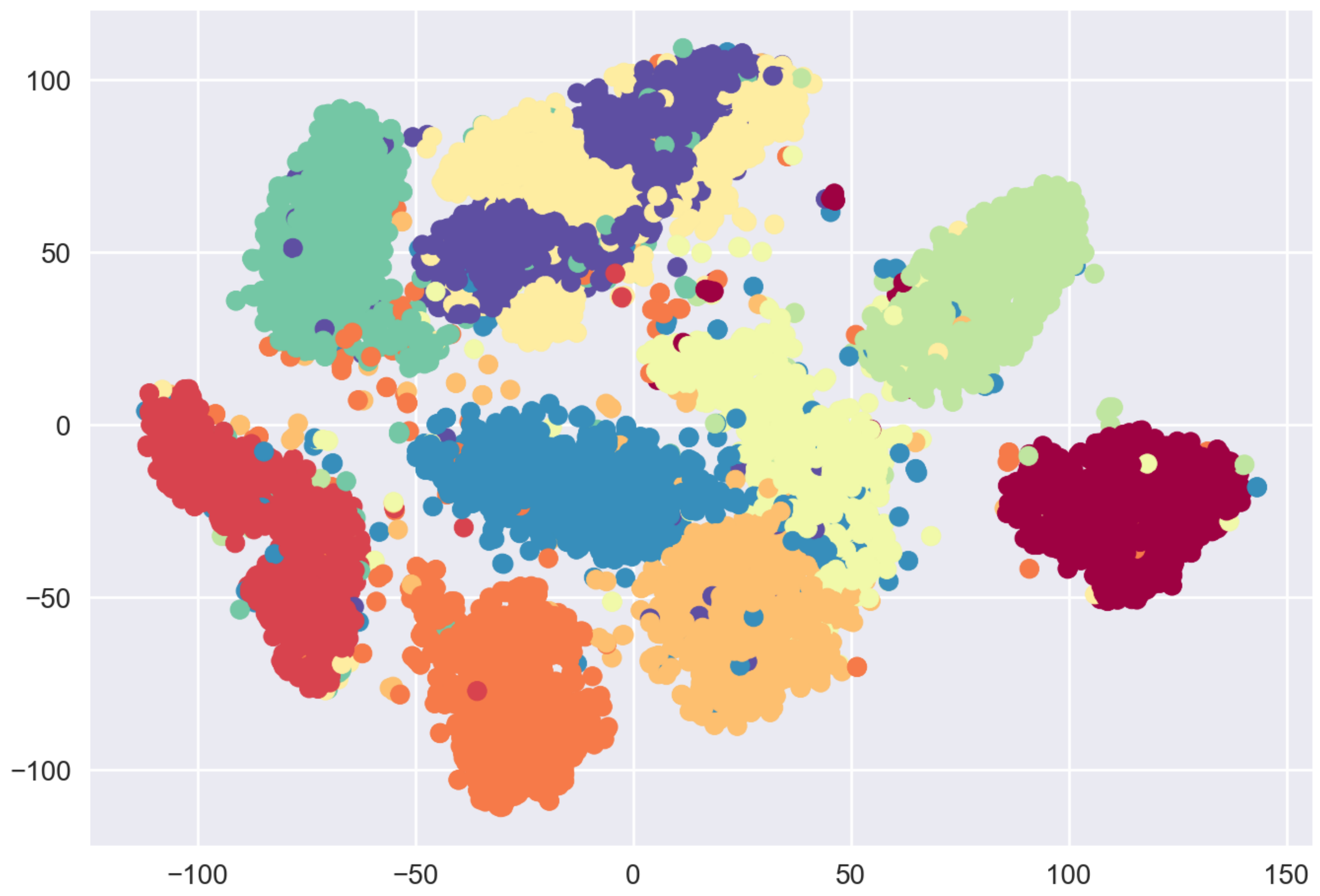
  # dimensions = 1

- total number of entries in the table/array

  # dimensions = image height × image width

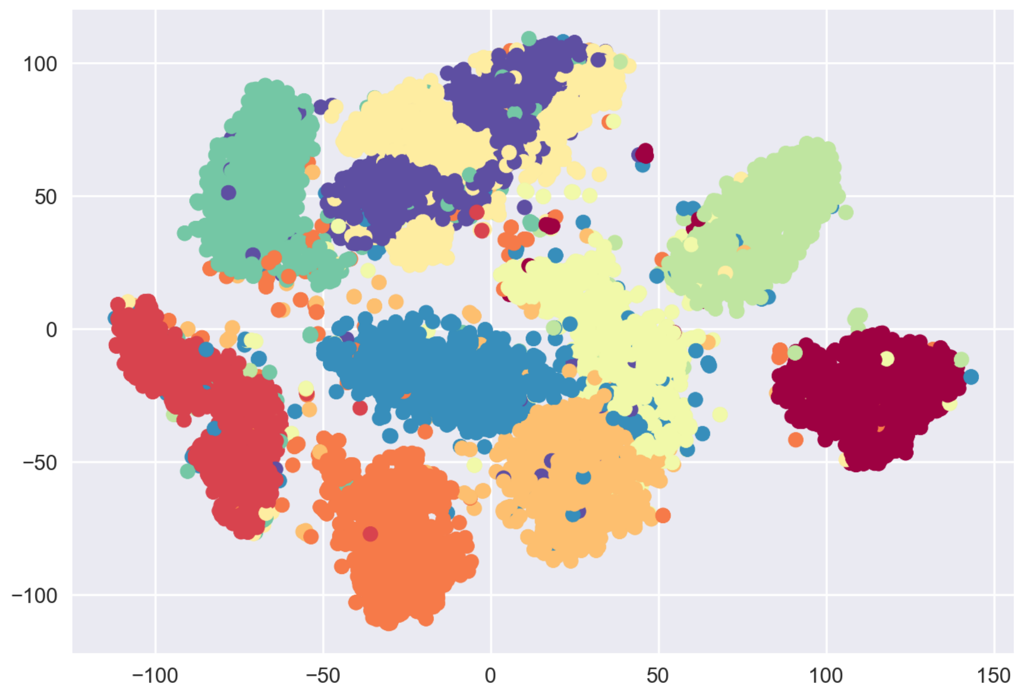
Dimensionality Reduction for Images

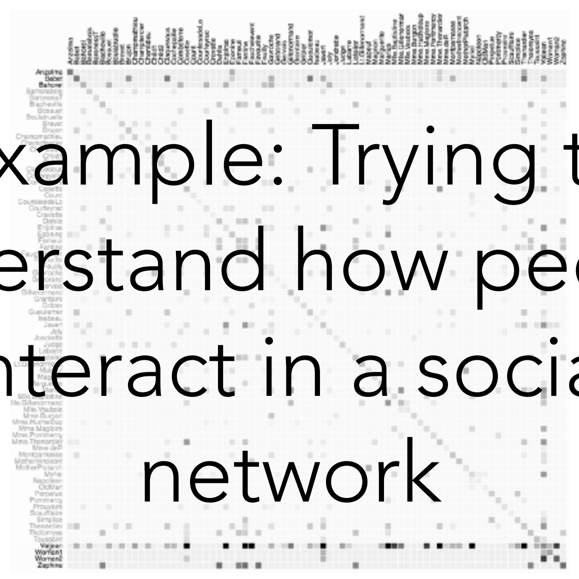
Demo



2D t-SNE plot of handwritten digit images shows clumps that correspond to real digits — this is an example of **clustering structure** showing up in real data

Remark on "Label Information"





Example: Trying to understand how people interact in a social network

Important:
Handwritten digit demo is a toy example where we know which images correspond to digits 0, 1, …, 9

Many real UDA problems:
The data are messy and it's not obvious what the "correct" labels/answers look like — what "correct" means can be ambiguous!

Later on in the course (when we cover predictive analytics), we look at how to take advantage of knowing the "correct" answers

Top right image source: https://bost.ocks.org/mike/miserables/

Clustering Structure Often Occurs!

Lots of real examples, such as:

- Crime might happen more often in specific hot spots
- People applying for micro loans have a few specific uses in mind (education, electricity, healthcare, etc)
- Users in a recommendation system can share similar taste in products

Clustering methods aim to group together data points that are "similar" into "clusters", while having different clusters be "dissimilar"

But what does "similar" or "dissimilar" mean?

Clustering methods will either directly assume a specific meaning of "similarity", or some allow you to specify a similarity/distance function

Note: distance is inversely related to similarity

(two points being more similar $\Longleftrightarrow$ they are closer in distance)

The Art of Defining Similarity/Distance

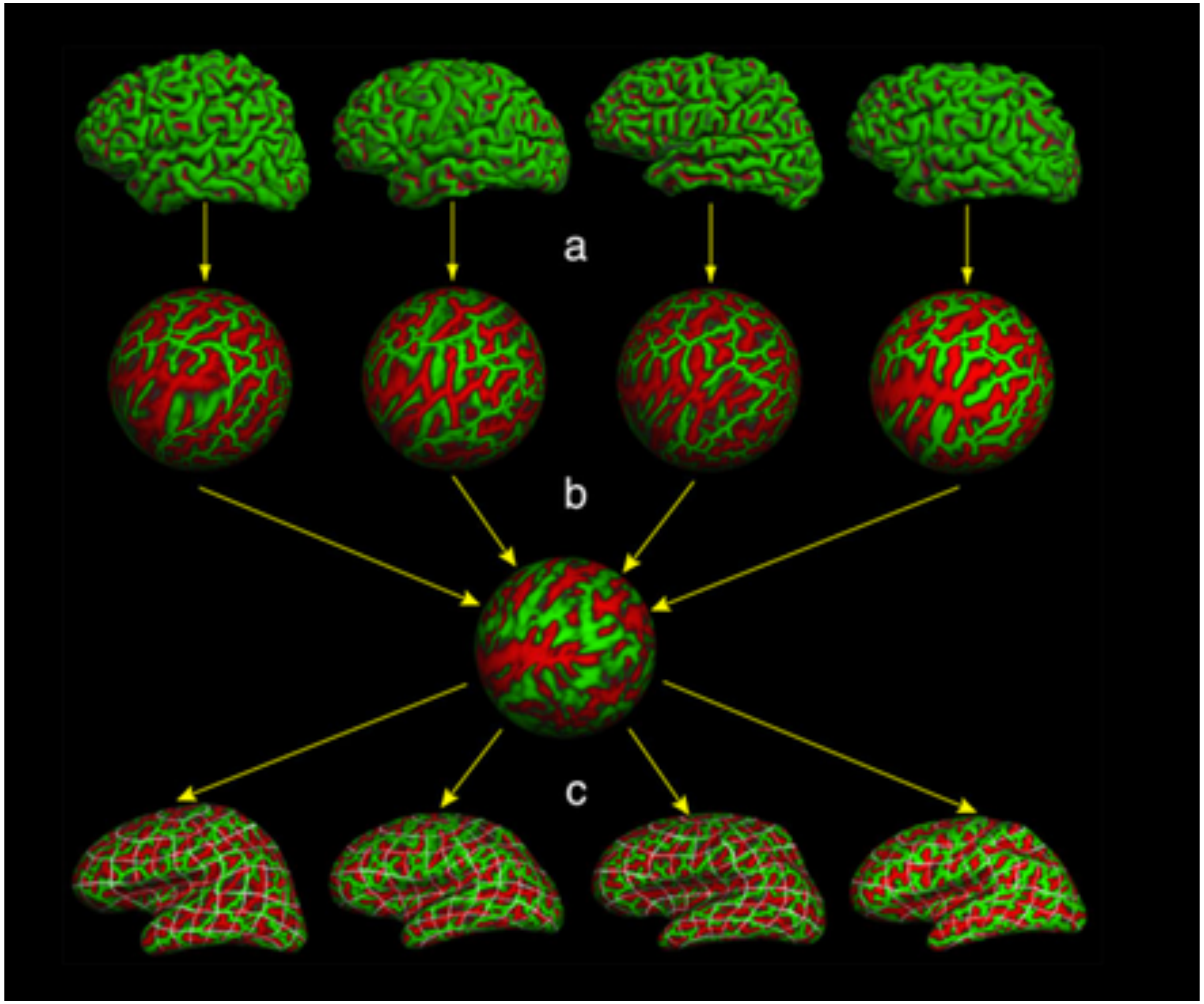
By far the most popular approach: if your data are represented as feature vectors, use Euclidean distance between feature vectors

Example: Spell Check

Distance between “apple” and “ap;ple”?

One way to compute: find minimum number of single-letter insertions/deletions/substitutions to convert one to the other (called the Levenshtein distance)

Example: Comparing Different People's Brain Scans



FreeSurfer software: "align" different people's brain scans by mapping them to a common coordinate system on a sphere

Clustering Methods

There's a whole zoo of clustering methods

Several main categories (although there are other categories!):

Generative models

1. Pretend data generated by specific model with parameters
2. Learn the parameters ("fit model to data")
3. Use fitted model to determine cluster assignments

We mainly focus on this

Hierarchical clustering

Top-down: Start with everything in 1 cluster and decide on how to recursively split

Bottom-up: Start with everything in its own cluster and decide on how to iteratively merge clusters

We're going to start with perhaps the most famous of clustering methods

It won't yet be apparent what this method has to do with generative models

k-means

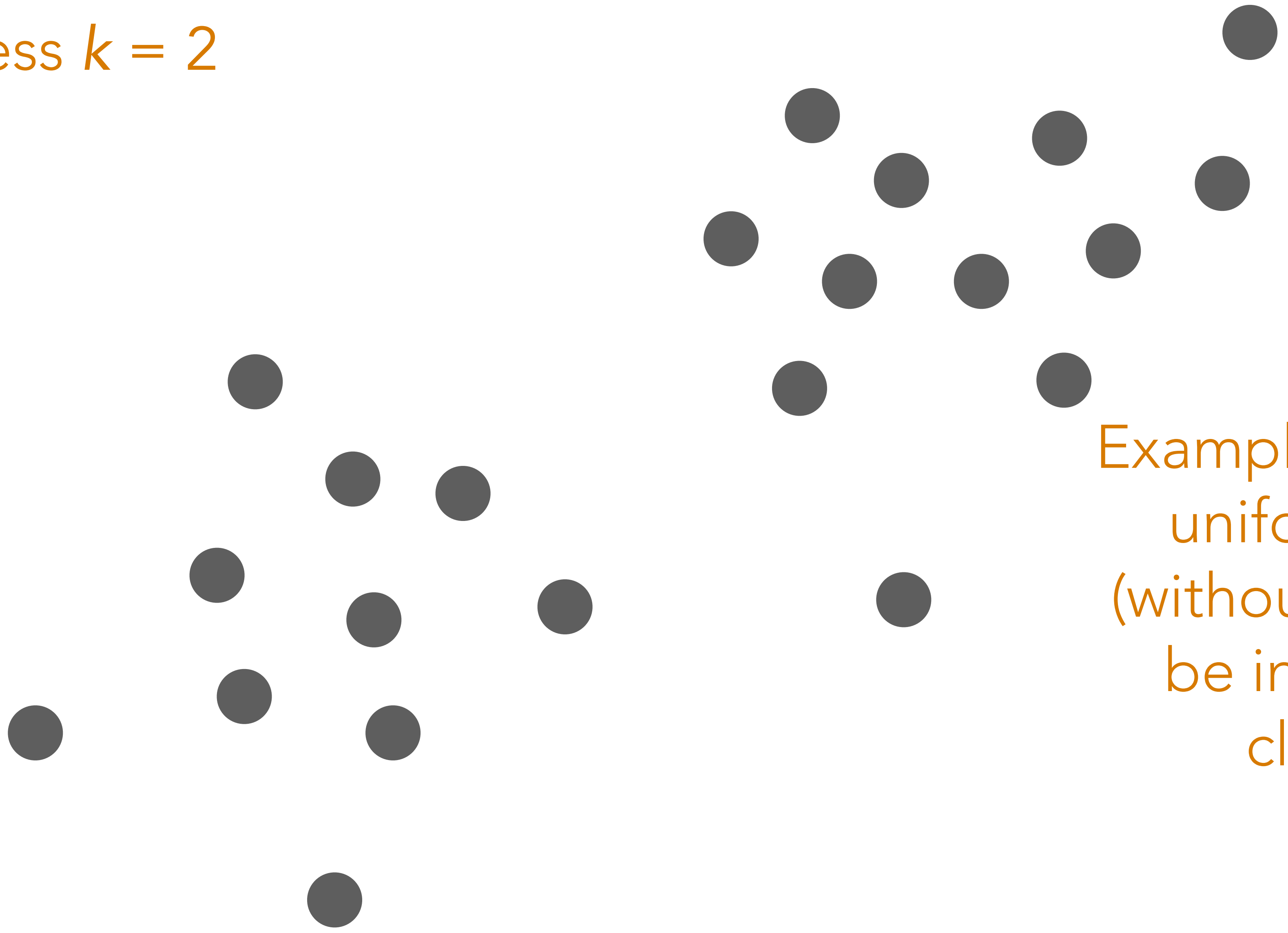
$\# \text{ of clusters} \leftarrow \boxed{k}$

Distance function: Euclidean

Step 0: Guess *k*

We'll guess *k* = 2

Step 1: Guess where cluster centers are



Example: choose *k* points uniformly at random (without replacement) to be initial guesses for cluster centers

k-means

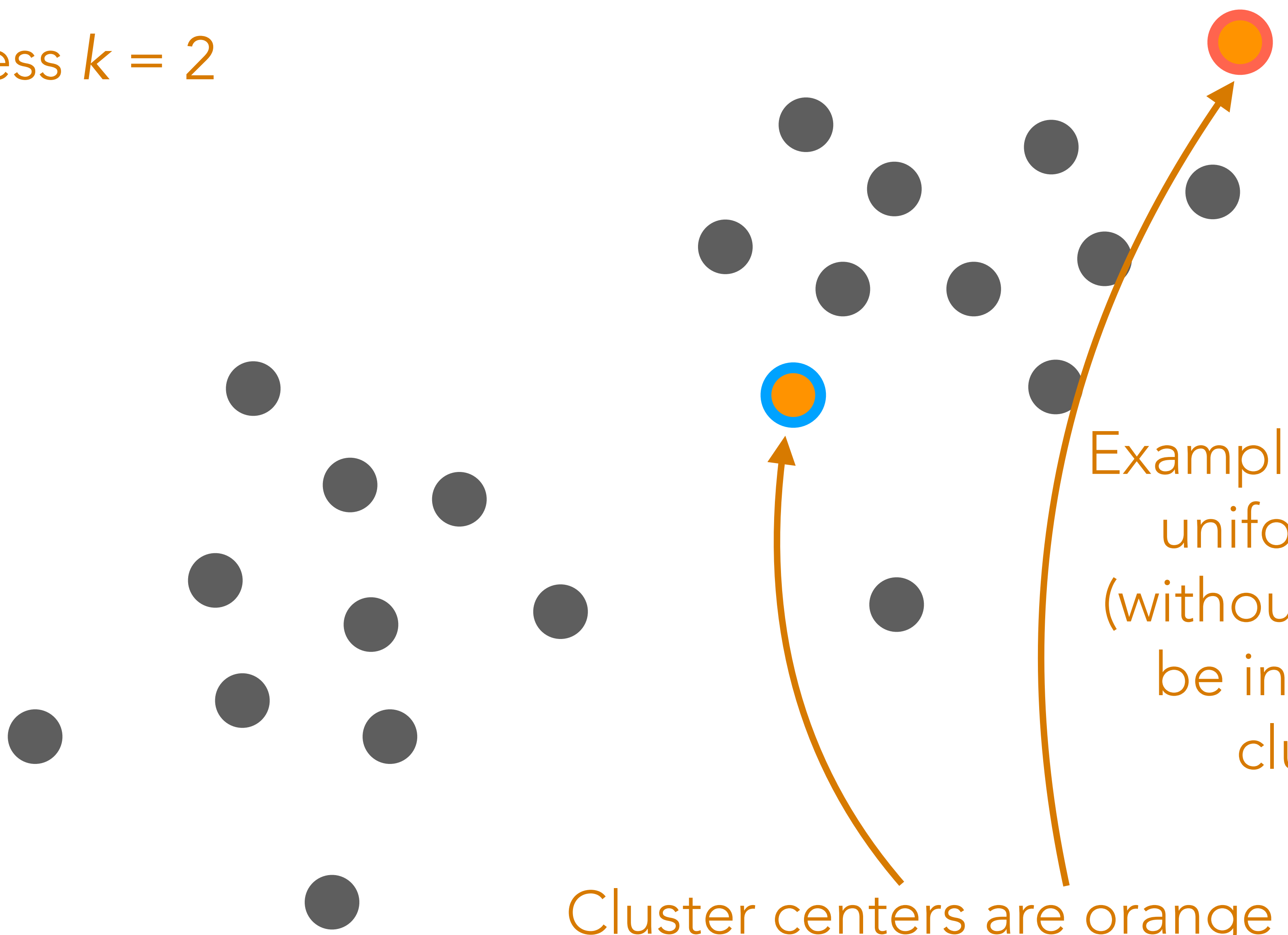
# of clusters

Distance function: Euclidean

Step 0: Guess *k*

We'll guess *k* = 2

Step 1: Guess where cluster centers are



Example: choose *k* points uniformly at random (without replacement) to be initial guesses for cluster centers

Cluster centers are orange points (outline says whether it is the blue or red cluster)

k-means

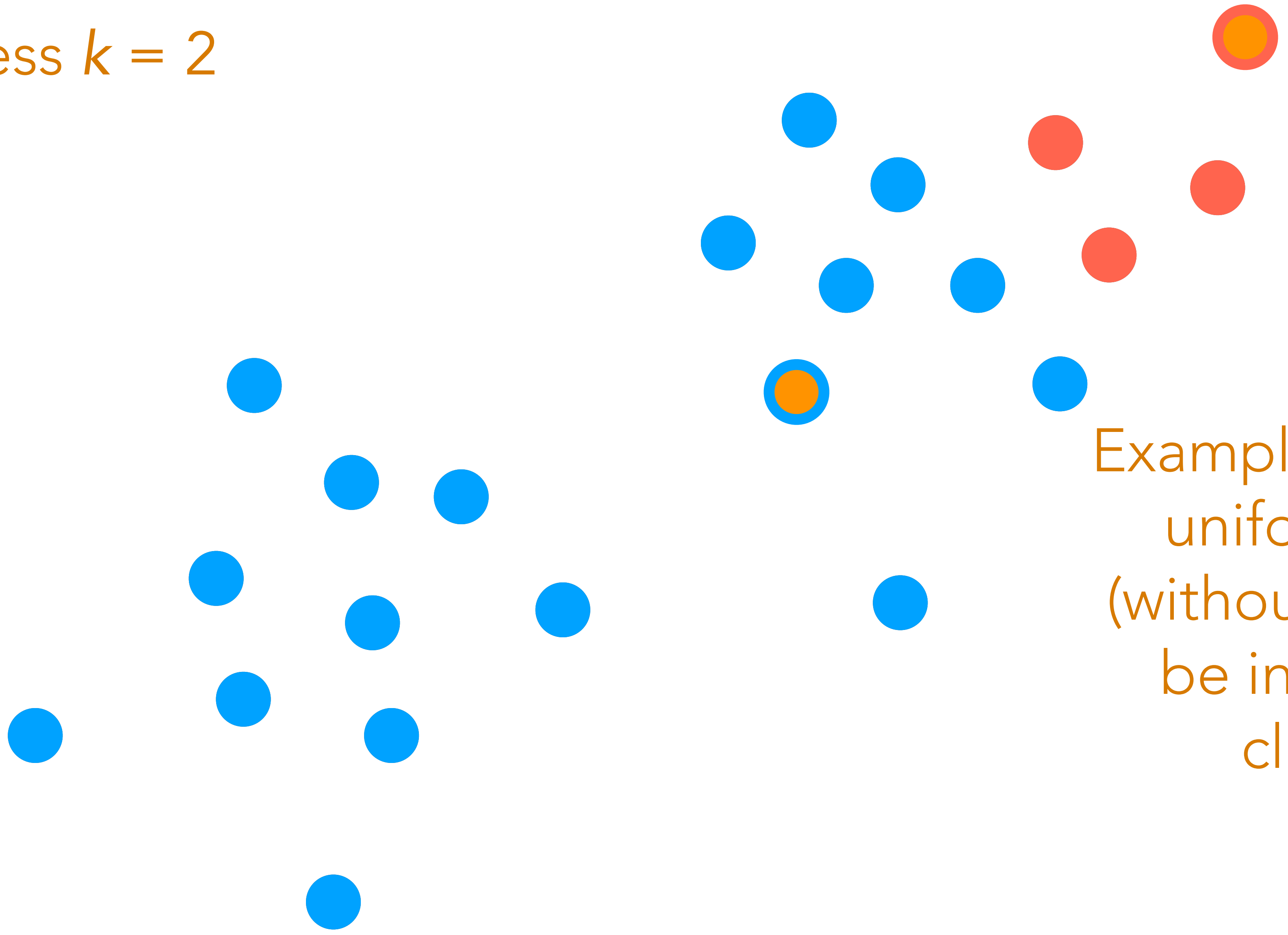
# of clusters

Distance function: Euclidean

Step 0: Guess *k*

We'll guess *k* = 2

Step 1: Guess where cluster centers are



Example: choose *k* points uniformly at random (without replacement) to be initial guesses for cluster centers

Step 2: Assign each point to belong to the closest cluster

k-means

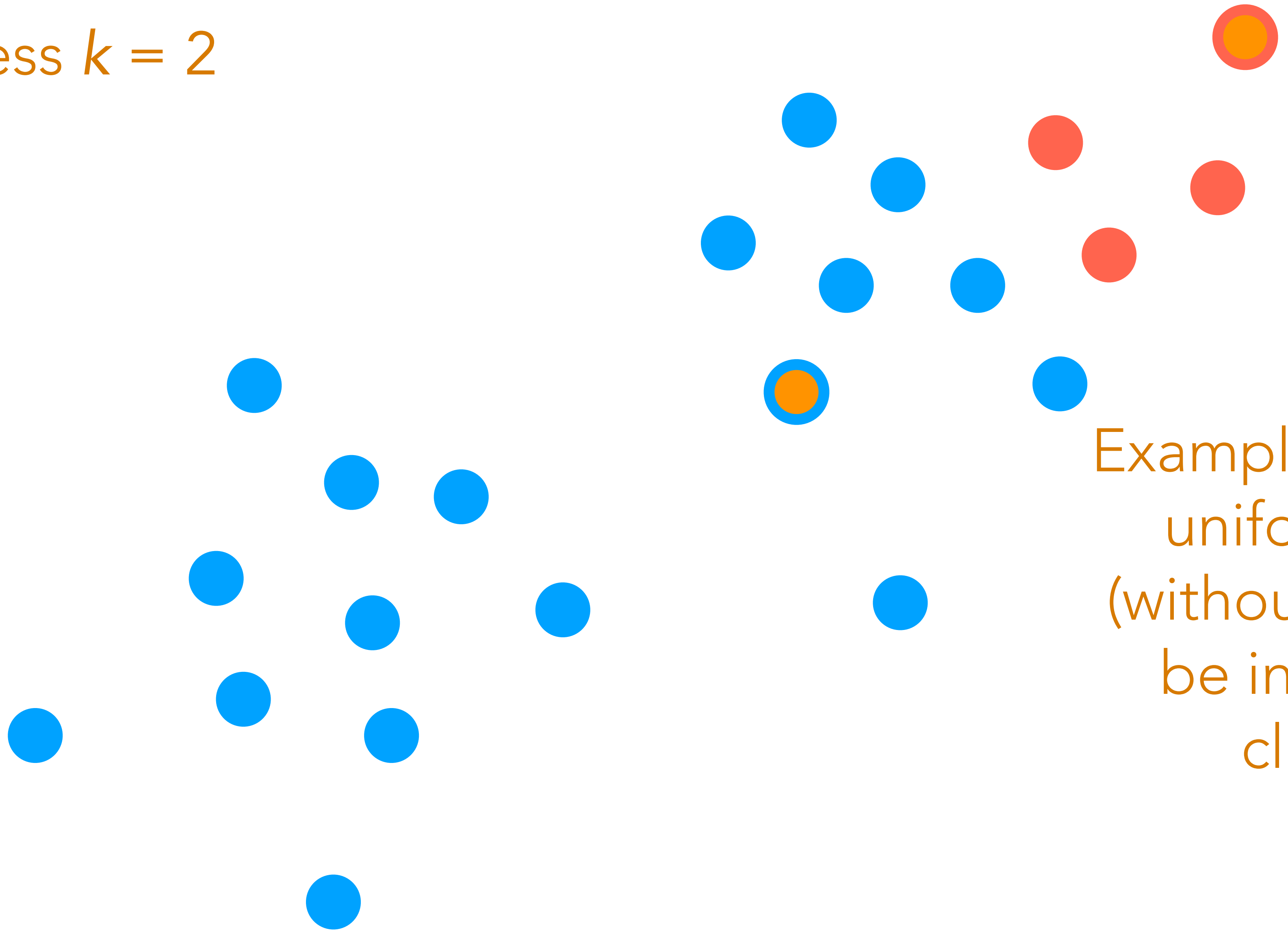
# of clusters

Distance function: Euclidean

Step 0: Guess *k*

We'll guess *k* = 2

Step 1: Guess where cluster centers are



Example: choose *k* points uniformly at random (without replacement) to be initial guesses for cluster centers

Step 2: Assign each point to belong to the closest cluster

Step 3: Update cluster means (to be the center of mass per cluster)

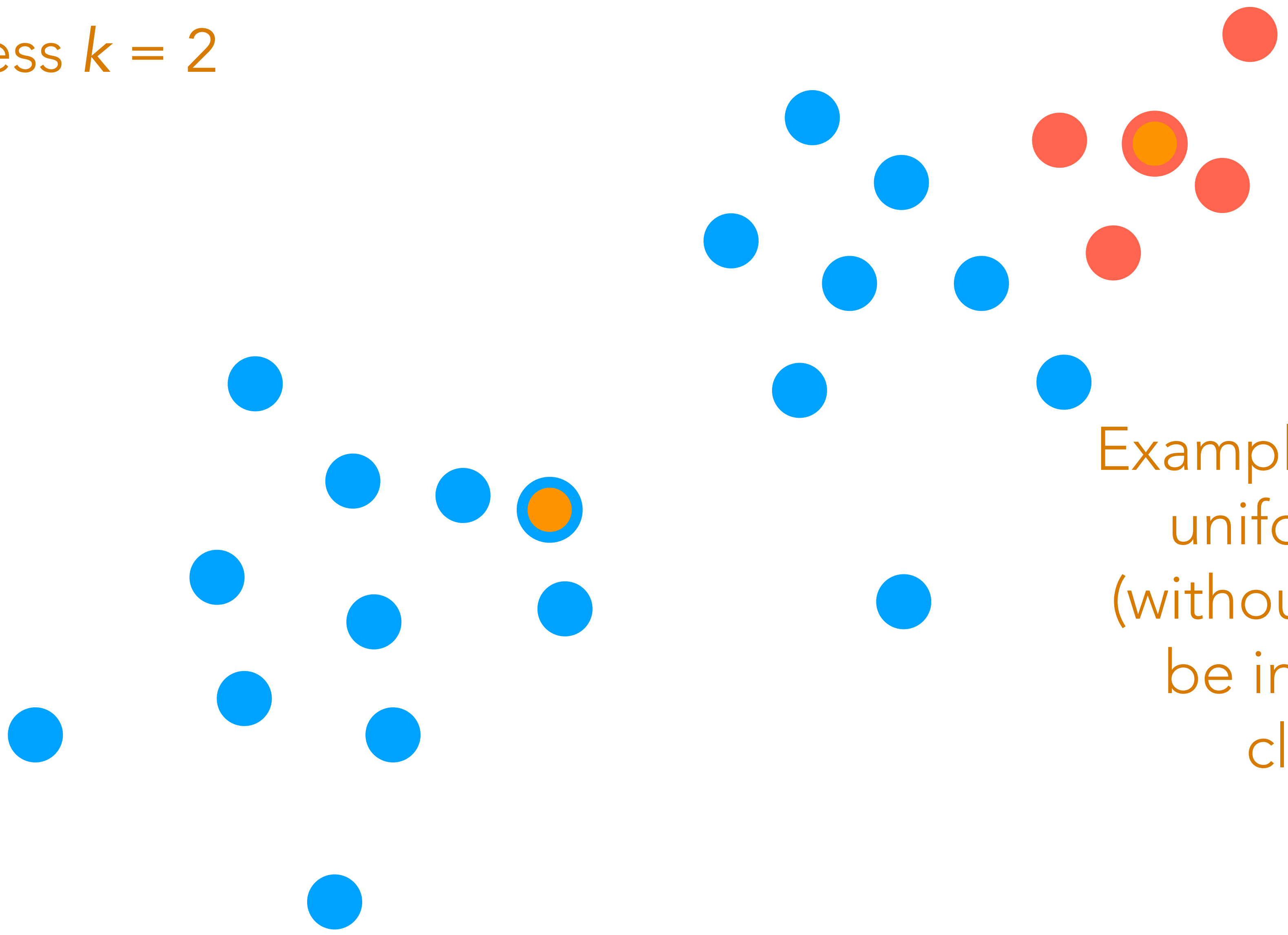
# of clusters k-means

Distance function: Euclidean

Step 0: Guess k

We'll guess k = 2

Step 1: Guess where cluster centers are



Example: choose k points uniformly at random (without replacement) to be initial guesses for cluster centers

Step 2: Assign each point to belong to the closest cluster

Step 3: Update cluster means (to be the center of mass per cluster)

k-means

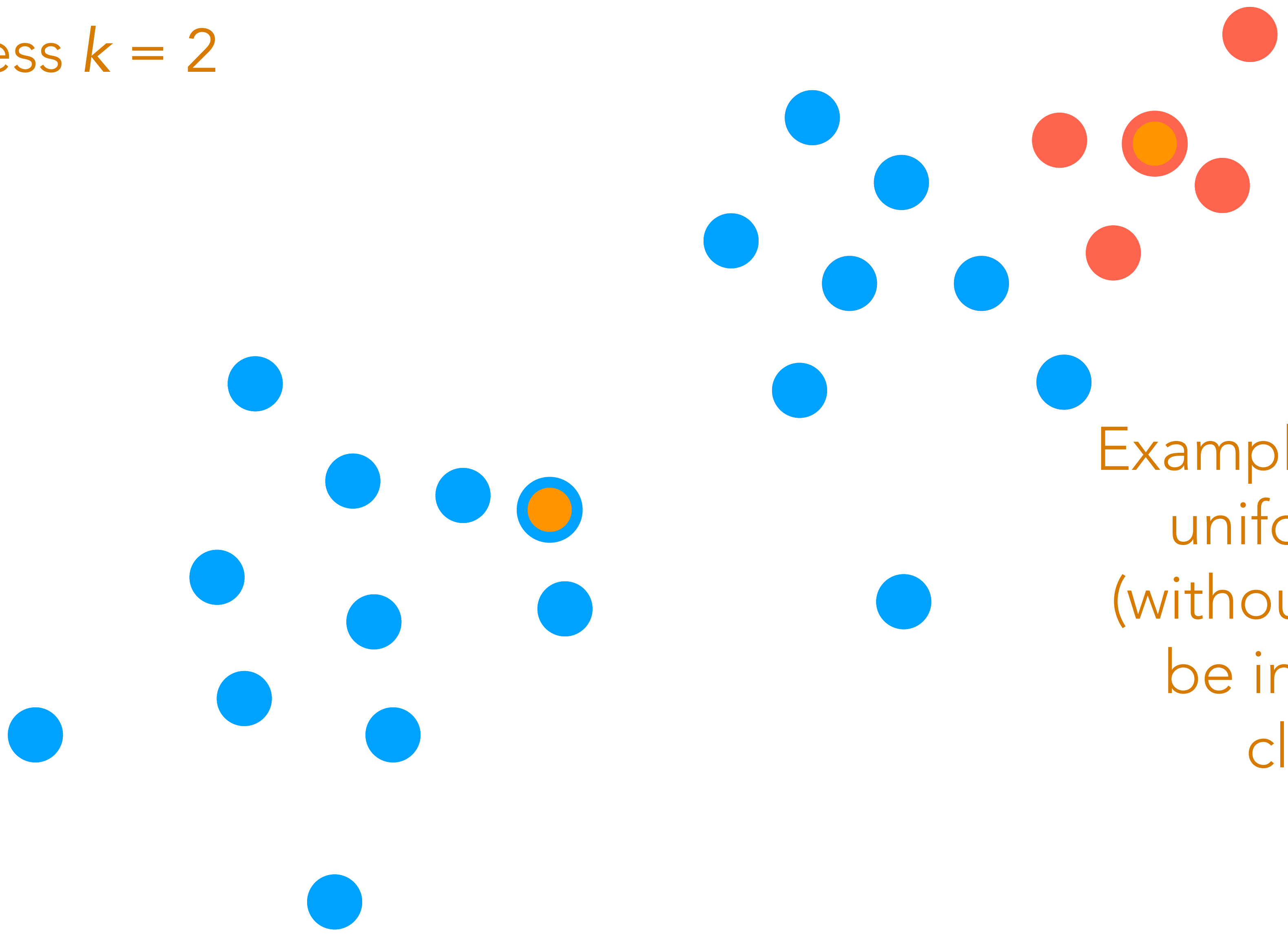
# of clusters

Distance function: Euclidean

Step 0: Guess *k*

We'll guess *k* = 2

Step 1: Guess where cluster centers are



Example: choose *k* points uniformly at random (without replacement) to be initial guesses for cluster centers

Repeat Step 2: Assign each point to belong to the closest cluster

Step 3: Update cluster means (to be the center of mass per cluster)

k-means

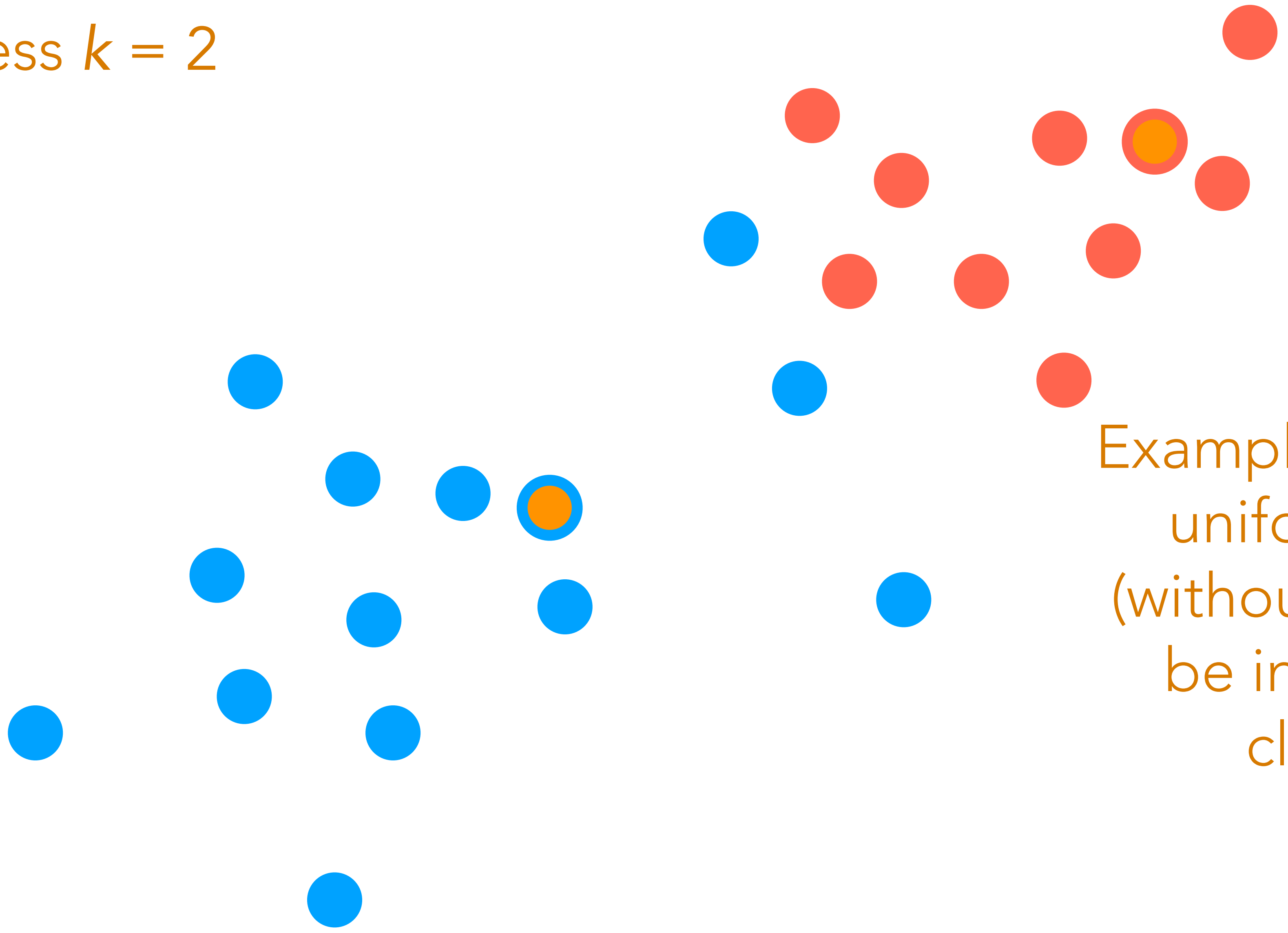
# of clusters

Distance function: Euclidean

Step 0: Guess *k*

We'll guess *k* = 2

Step 1: Guess where cluster centers are



Example: choose *k* points uniformly at random (without replacement) to be initial guesses for cluster centers

Step 2: Assign each point to belong to the closest cluster

Repeat

Step 3: Update cluster means (to be the center of mass per cluster)

# of clusters

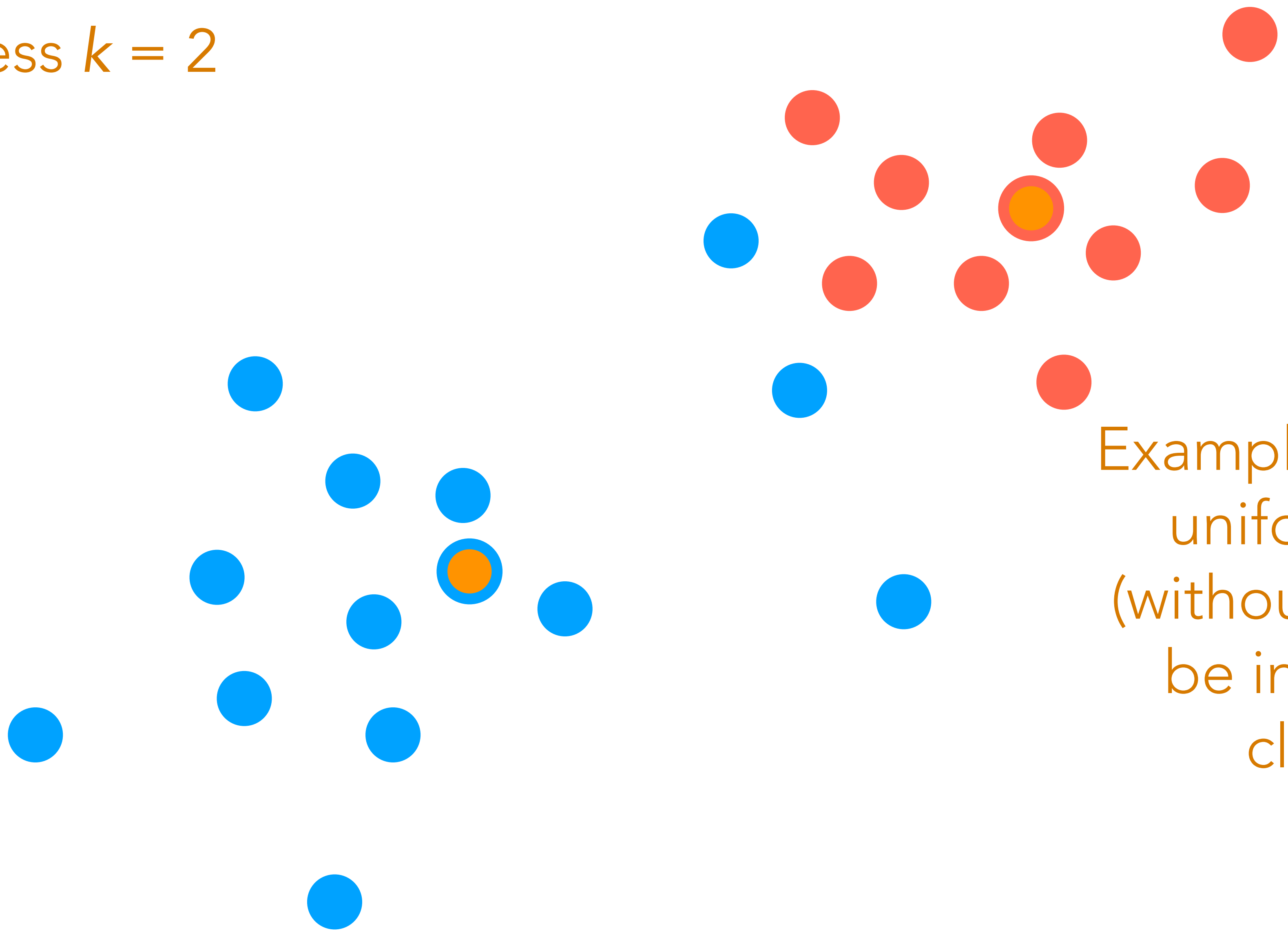
k-means

Distance function: Euclidean

Step 0: Guess *k*

We'll guess *k* = 2

Step 1: Guess where cluster centers are



Example: choose *k* points uniformly at random (without replacement) to be initial guesses for cluster centers

Step 2: Assign each point to belong to the closest cluster

Repeat

Step 3: Update cluster means (to be the center of mass per cluster)

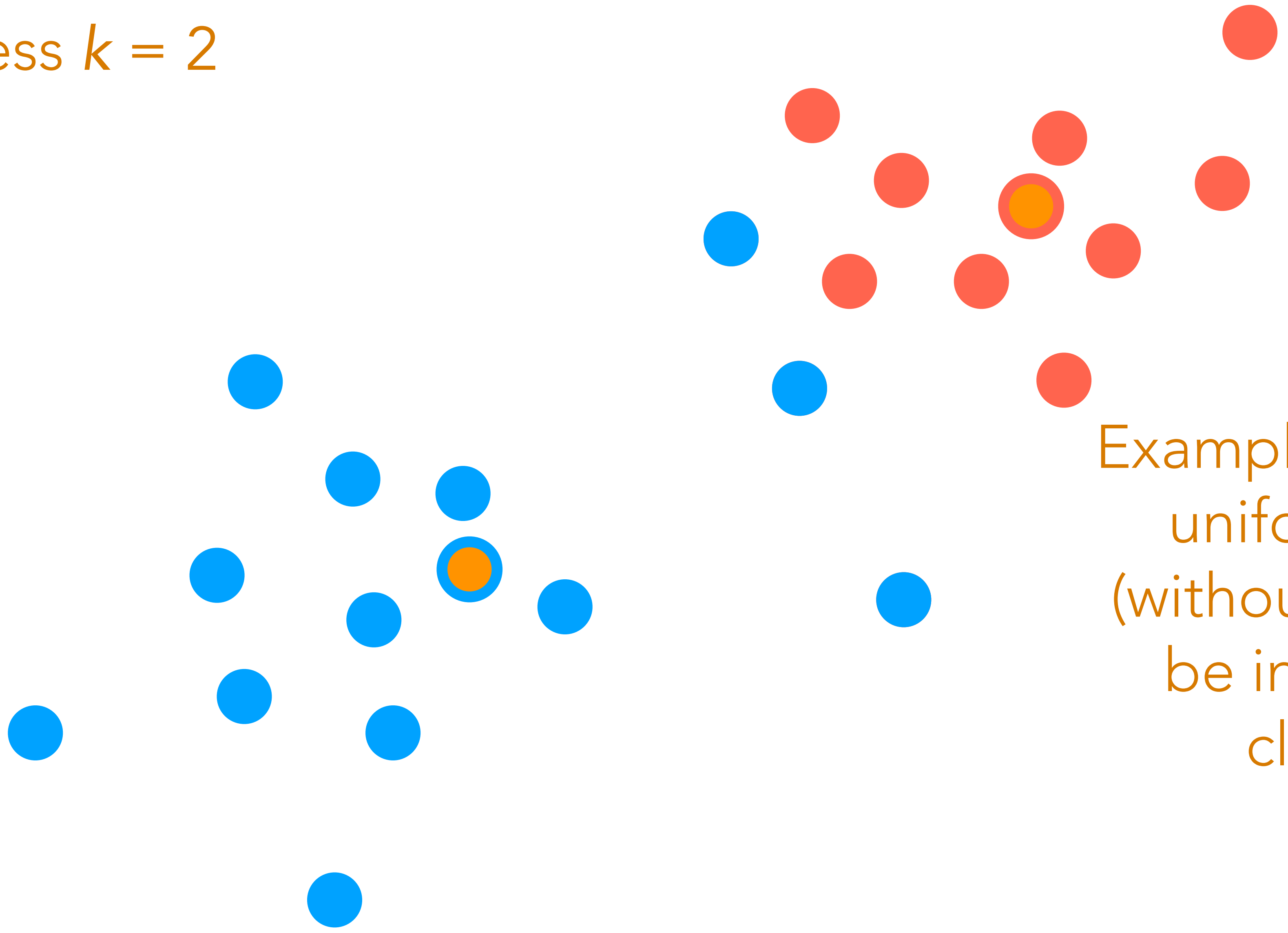
of clusters *k*-means

Distance function: Euclidean

Step 0: Guess *k*

We'll guess *k* = 2

Step 1: Guess where cluster centers are



Example: choose *k* points uniformly at random (without replacement) to be initial guesses for cluster centers

Repeat Step 2: Assign each point to belong to the closest cluster

Step 3: Update cluster means (to be the center of mass per cluster)

k-means

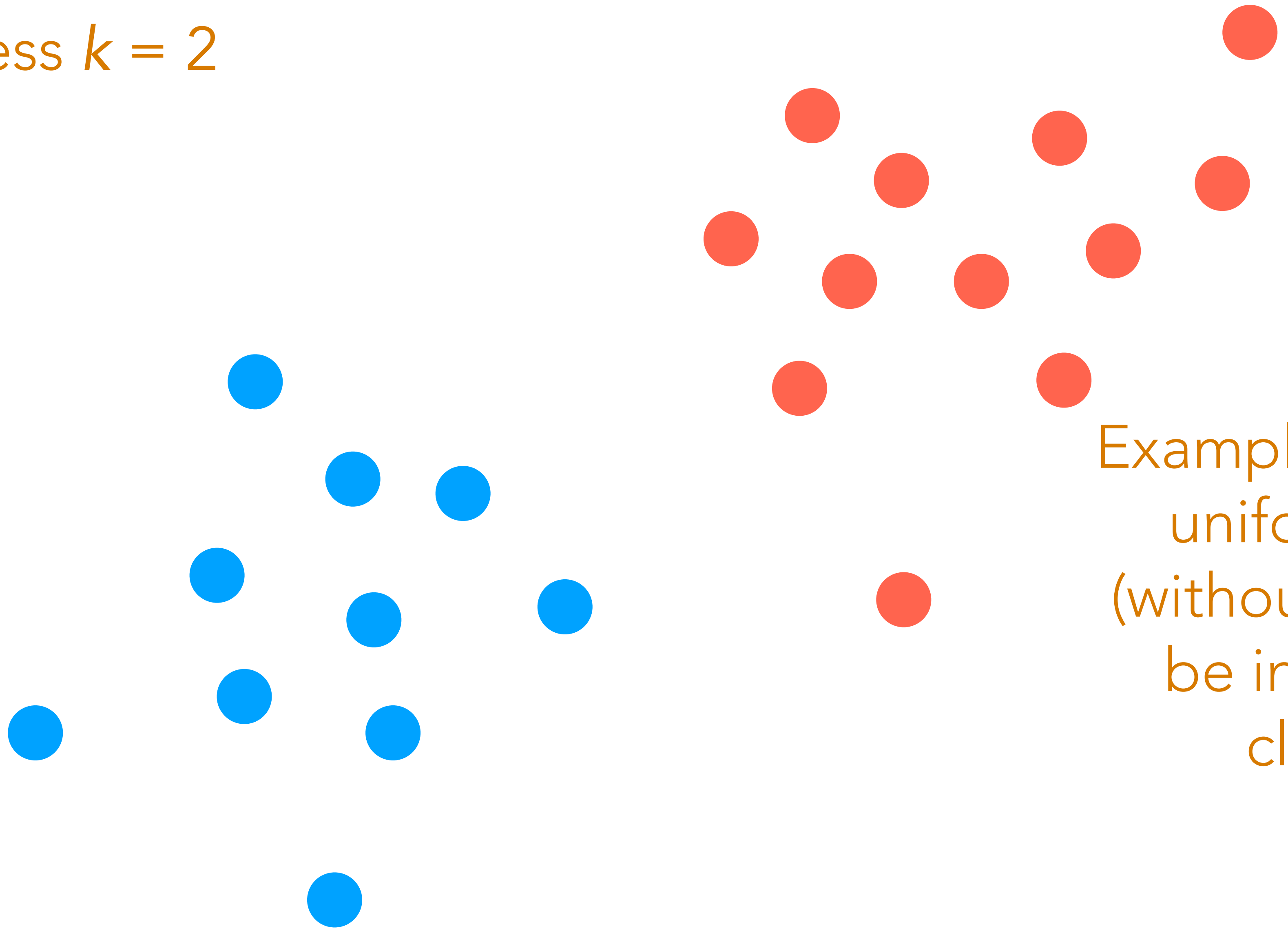
# of clusters

Distance function: Euclidean

Step 0: Guess *k*

We'll guess *k* = 2

Step 1: Guess where cluster centers are



Example: choose *k* points uniformly at random (without replacement) to be initial guesses for cluster centers

Repeat

Step 2: Assign each point to belong to the closest cluster

Step 3: Update cluster means (to be the center of mass per cluster)

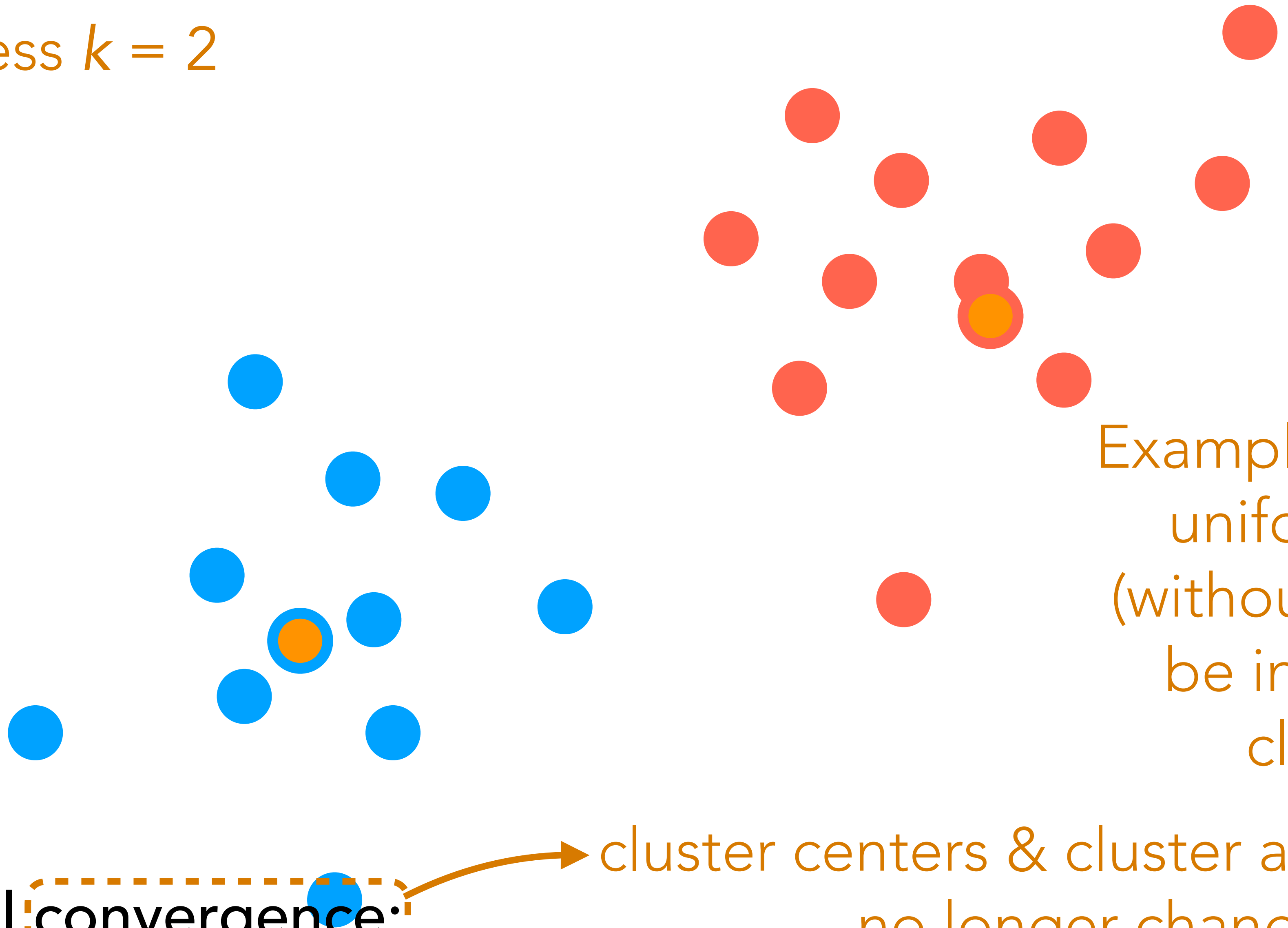
# of clusters ← k-means

Distance function: Euclidean

Step 0: Guess k

We'll guess k = 2

Step 1: Guess where cluster centers are



Example: choose k points uniformly at random (without replacement) to be initial guesses for cluster centers

Repeat until convergence: → cluster centers & cluster assignments no longer change

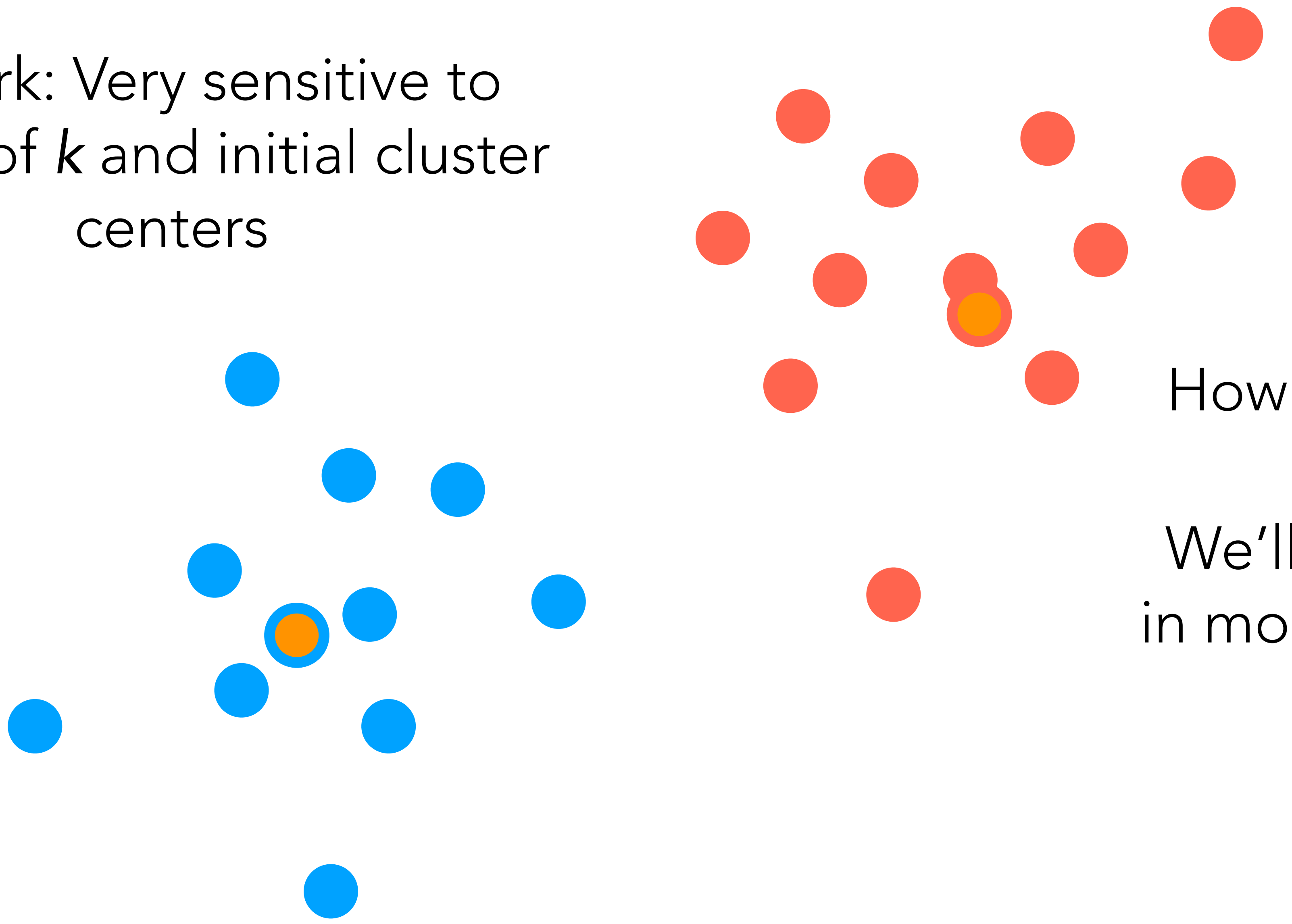
Step 2: Assign each point to belong to the closest cluster

Step 3: Update cluster means (to be the center of mass per cluster)

k-means

Final output: cluster centers, cluster assignment for every point

Remark: Very sensitive to choice of *k* and initial cluster centers



How to guess *k*?

We'll discuss this in more detail next lecture

Suggested way to pick initial cluster centers: "*k*-means++" method

(rough intuition: incrementally add centers; favor adding center far away from centers chosen so far)

When does *k*-means work well?

k-means is related to a generative model, which will help us understand when *k*-means is expected to work well