Carnegie Mellon University Heinz College

Unstructured Data Analytics for Policy

Lecture 6: Dimensionality reduction with images, intro to clustering

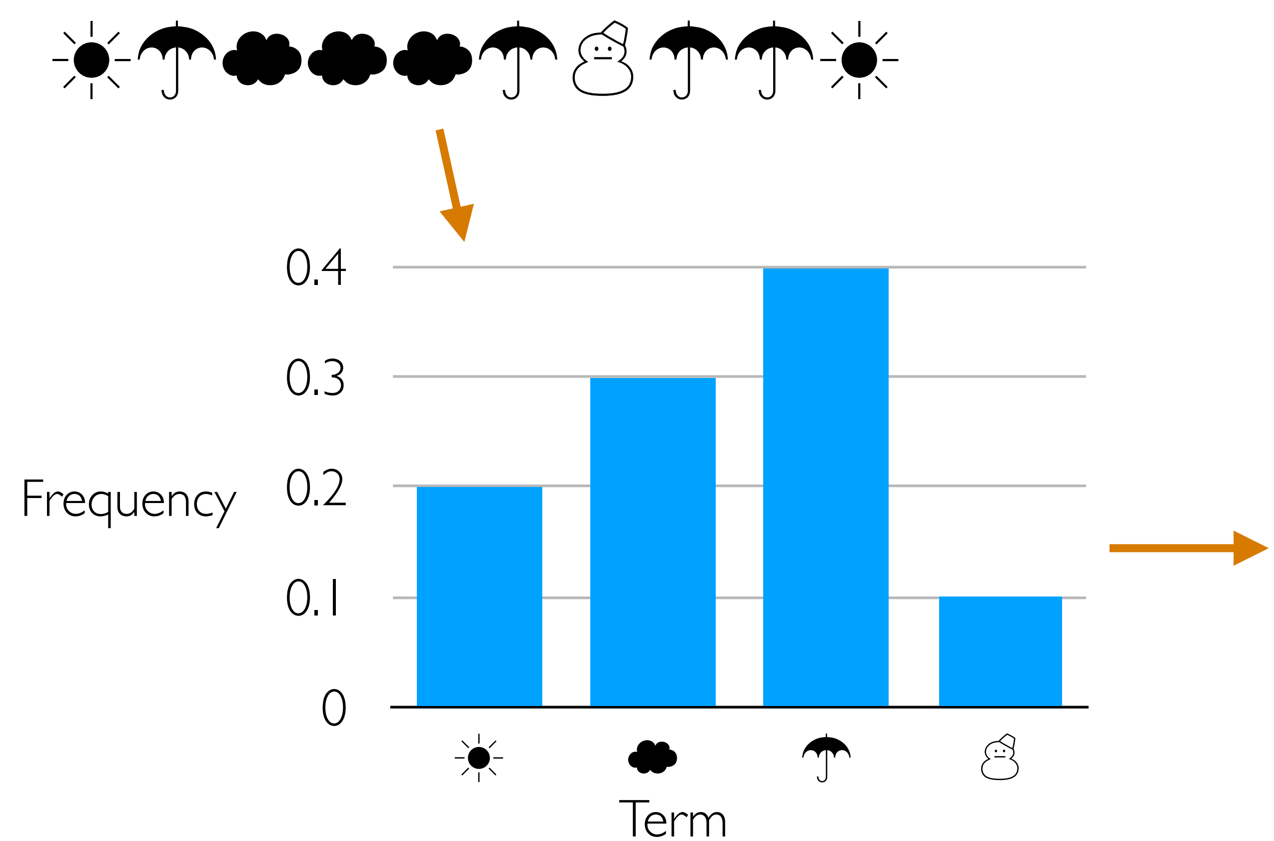
George Chen

Let’s look at images

(Flashback) Recap: Basic Text Analysis

- Represent text in terms of "features"
  (each feature: how often a specific word/phrase appears)
  - Can repeat this for different documents:
    represent each document as a "feature vector"

"Sentence":



$$\begin{bmatrix} 0.2 \\ 0.3 \\ 0.4 \\ 0.1 \end{bmatrix}$$

This is a point in 4-dimensional space, $\mathbb{R}^4$

\# dimensions = number of terms

In general (not just text): first represent data as feature vectors

Example: Representing an Image

0: black
1: white

0

Go row by row and look at pixel values

Image source: The Mandalorian

Example: Representing an Image

0: black
1: white

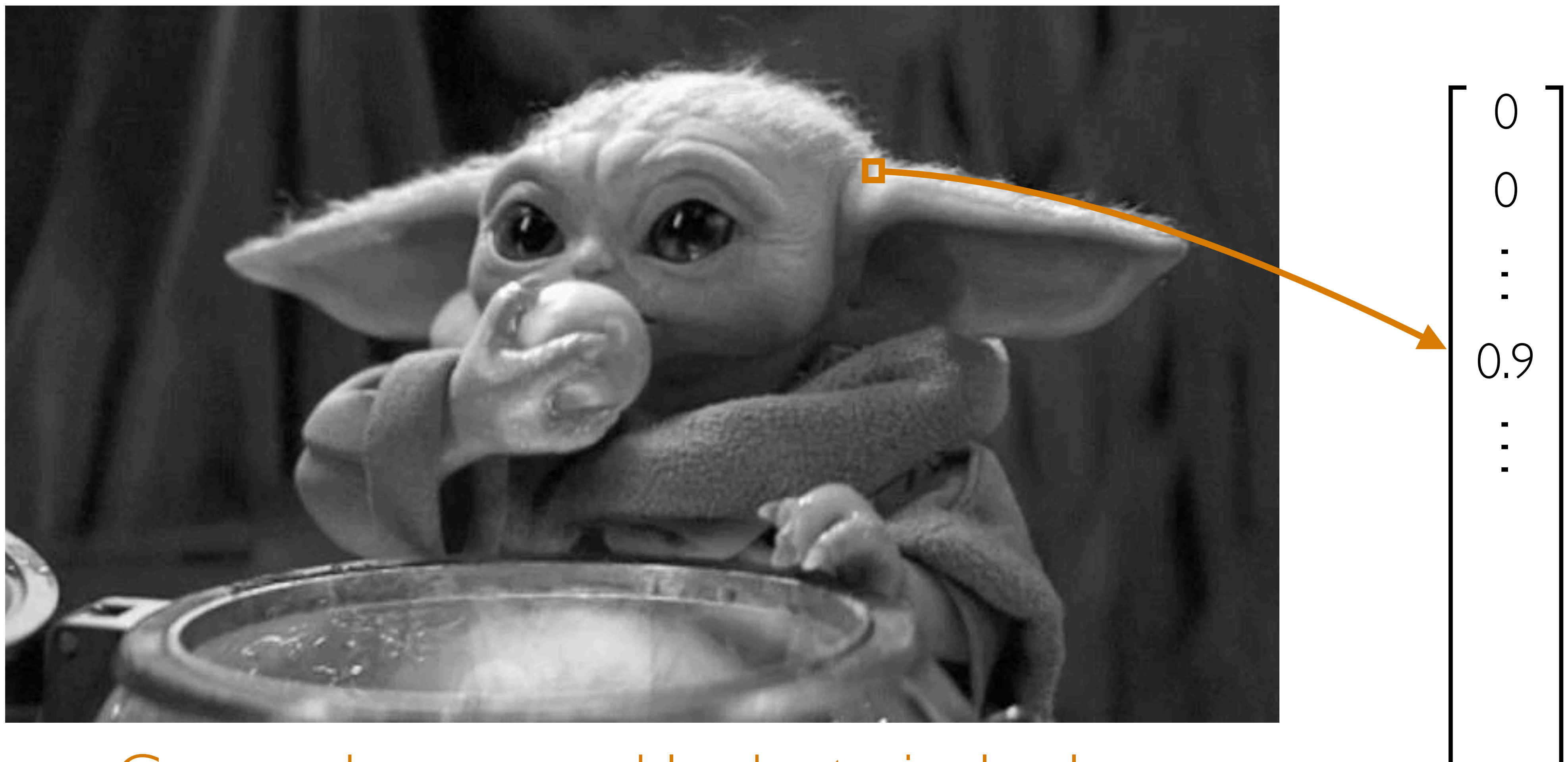
Go row by row and look at pixel values

$$\begin{bmatrix} 0 \\ 0 \\ \vdots \\ \\ \end{bmatrix}$$

Image source: The Mandalorian

Example: Representing an Image

0: black
1: white



Go row by row and look at pixel values

Image source: The Mandalorian

Example: Representing an Image

0: black
1: white

$$\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0.9 \\ \vdots \\ \\ 0.1 \end{bmatrix}$$

Go row by row and look at pixel values

dimensions = image width × image height

Very high dimensional!

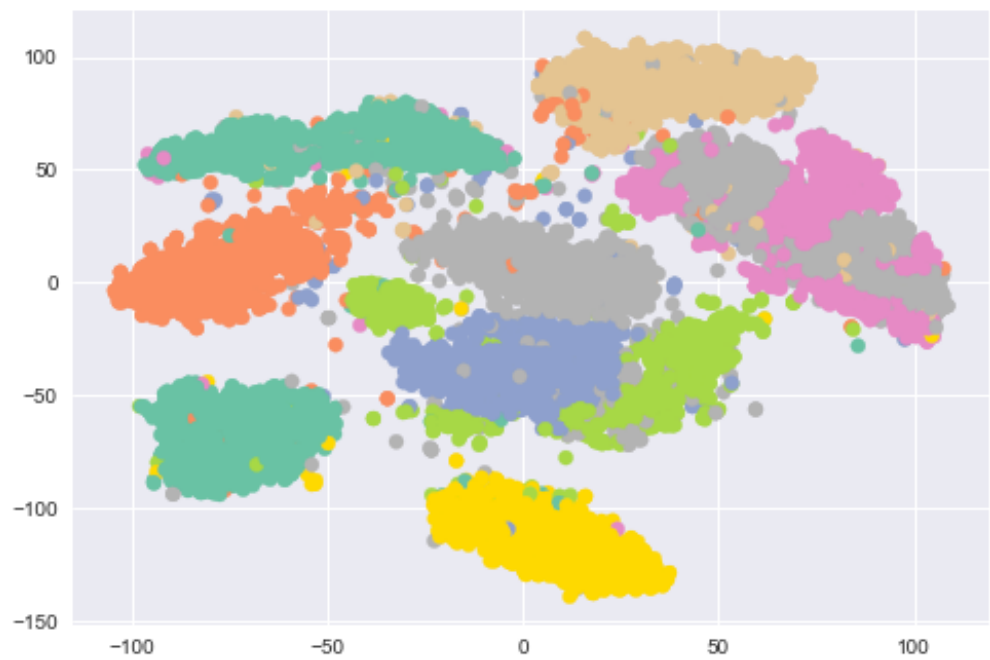
Image source: The Mandalorian

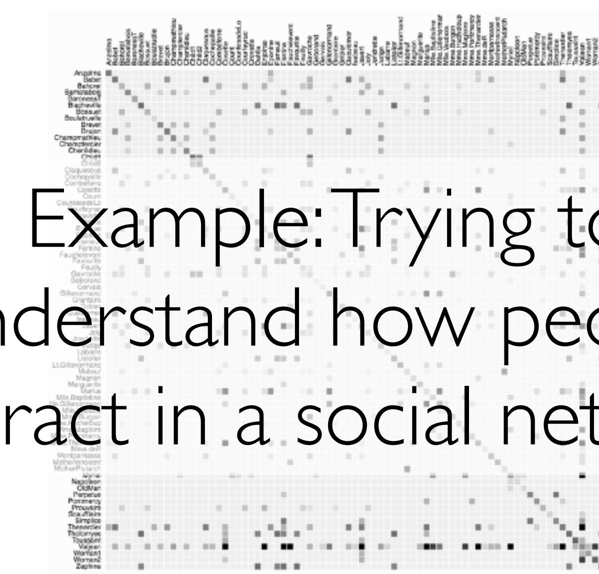
Dimensionality Reduction for Images

Demo

Visualization

is a way of debugging data analysis!





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, and "correct" is ambiguous!

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

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

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

Let's look at a *structured* dataset (easier to explain clustering): drug consumption data

Drug Consumption Data

Demo

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
(more similar means closer in distance)

The Art of

Defining Similarity/Distance

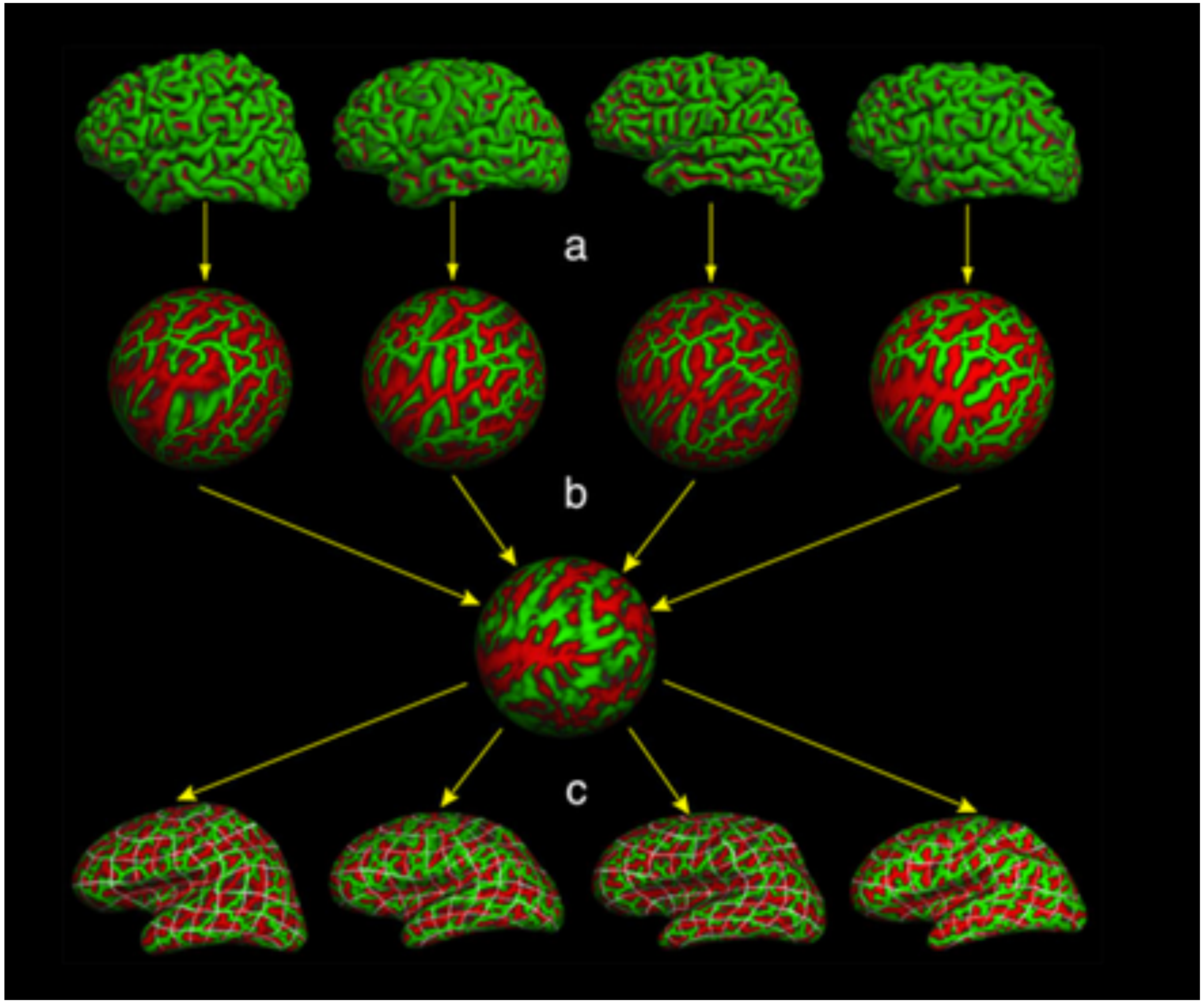
By far the most popular approach: if your data are represented as feature vectors, use Euclidean distance between feature vectors

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

Is a Distance/Similarity Function Any Good?

Easy thing to try:

- Pick a data point (for example, randomly)
- Compute its similarity (distance) to all the other data points, and sort them from most similar to least similar (or smallest distance to largest)
- Manually examine the most similar (closest) data points by looking at the raw data

If the most similar/closest points are not interpretable, it's quite likely that your distance/similarity function isn't very good ☹️

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

Density-based clustering

Based on finding parts of the data with higher density

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

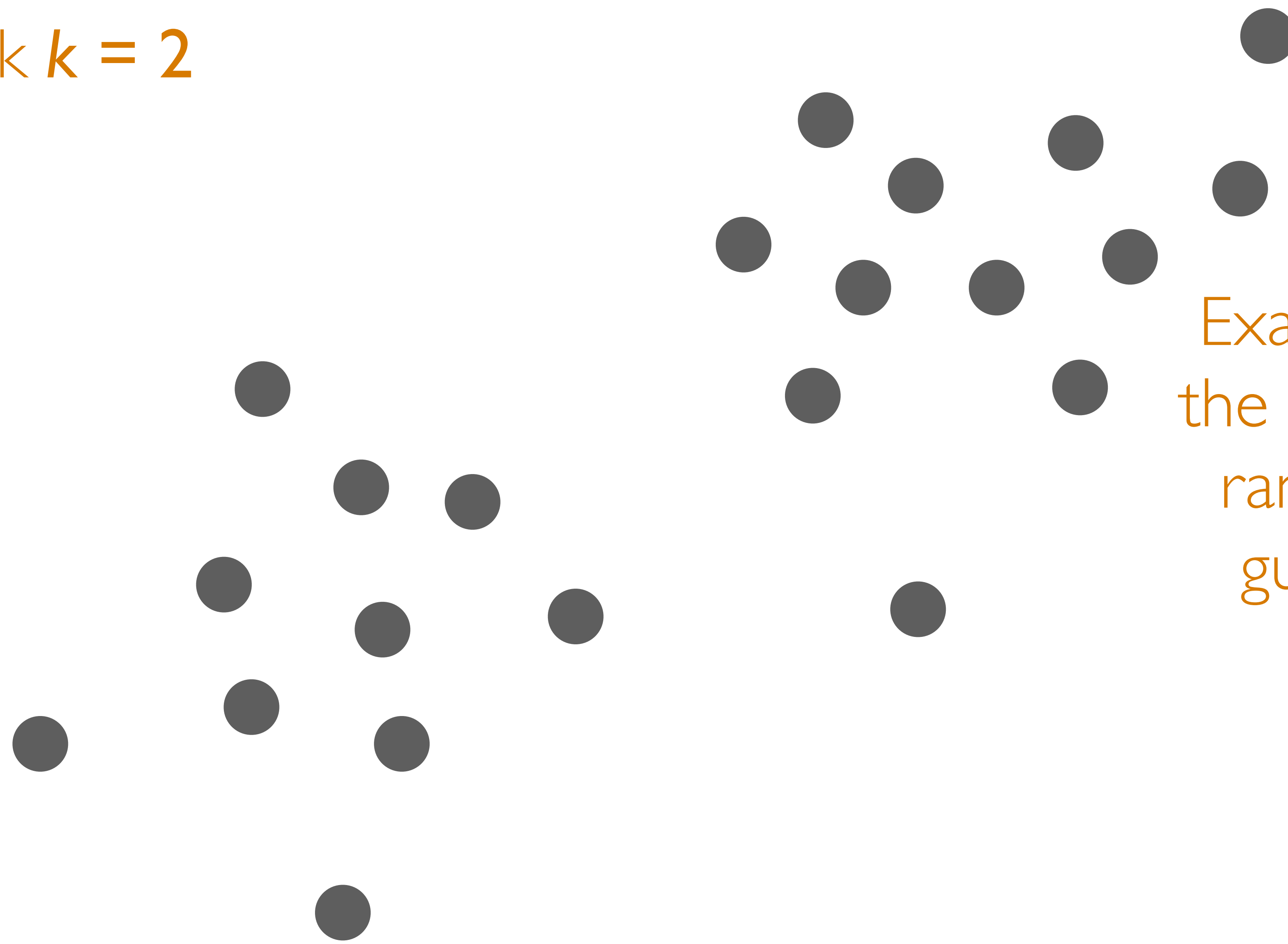
It won't yet be apparent what this method has
to do with generative models

k-means

Step 0: Pick ***k***

We'll pick ***k* = 2**

Step 1: Pick <u>guesses</u> for where cluster centers are



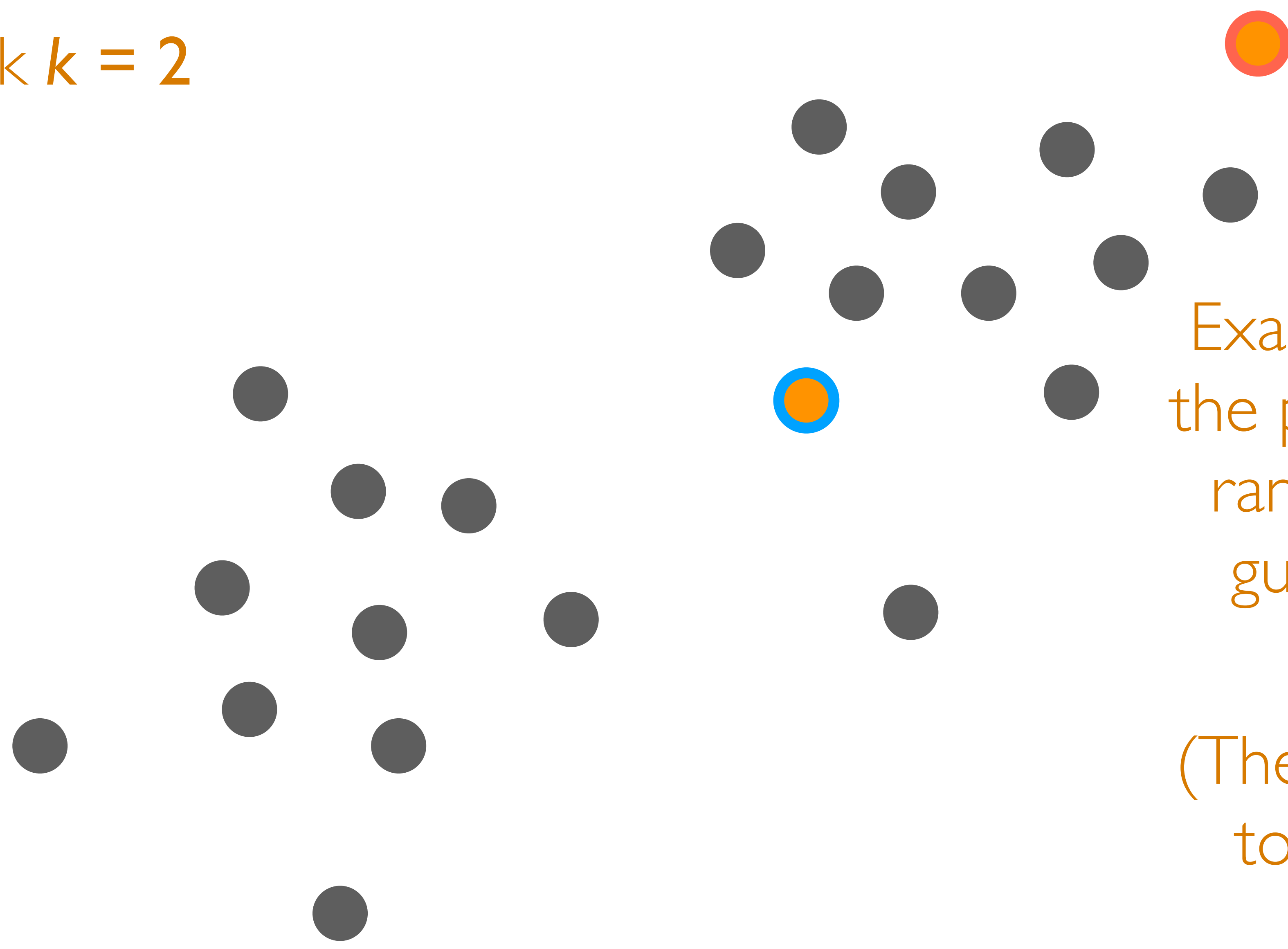
Example: choose ***k*** of the points uniformly at random to be initial guesses for cluster centers

k-means

Step 0: Pick ***k***

We'll pick ***k* = 2**

Step 1: Pick <u>guesses</u> for where cluster centers are



Example: choose ***k*** of the points uniformly at random to be initial guesses for cluster centers

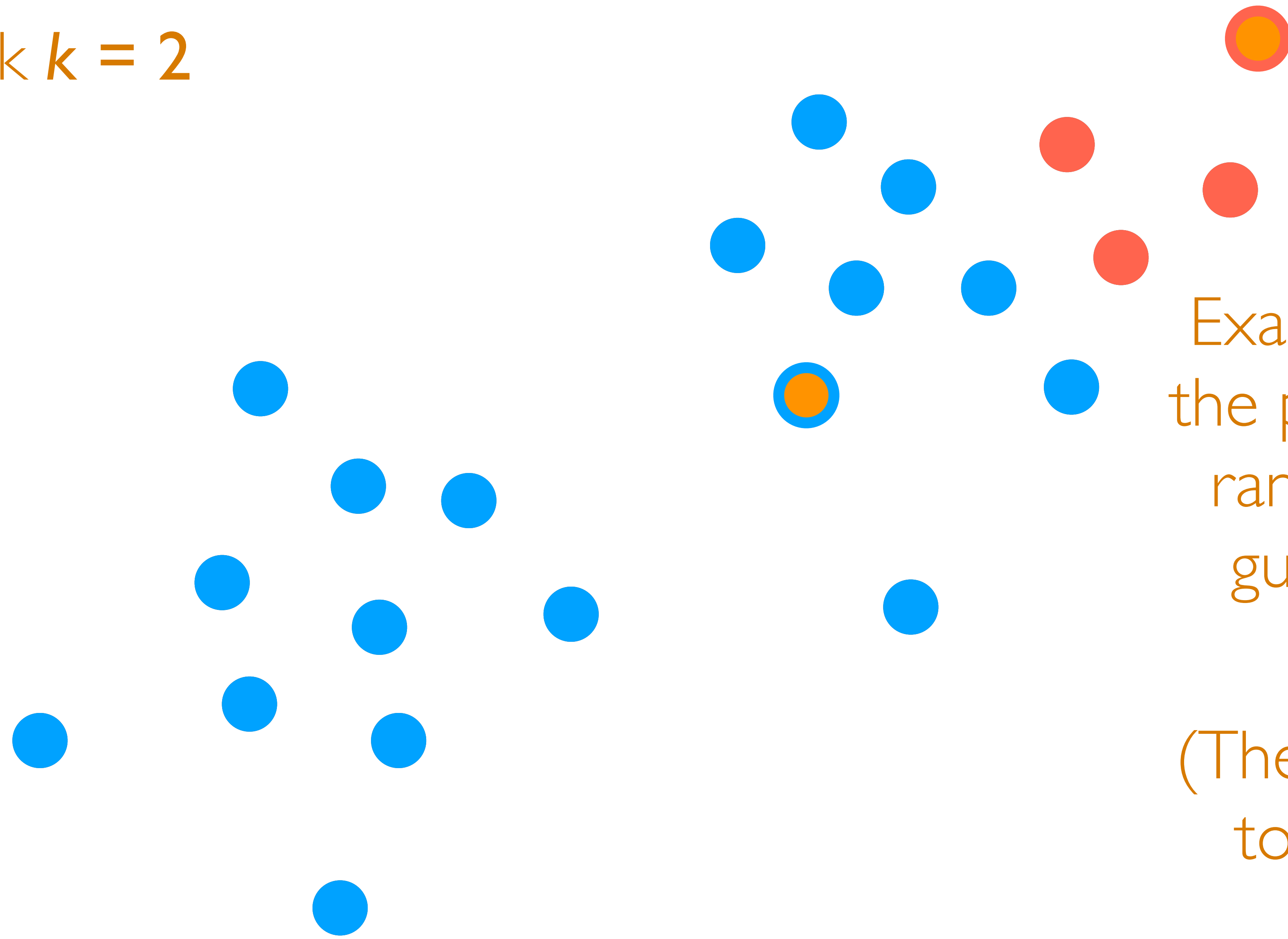
(There are many ways to make the initial guesses)

k-means

Step 0: Pick ***k***

We'll pick ***k* = 2**

Step 1: Pick <u>guesses</u> for where cluster centers are



Example: choose ***k*** of the points uniformly at random to be initial guesses for cluster centers

(There are many ways to make the initial guesses)

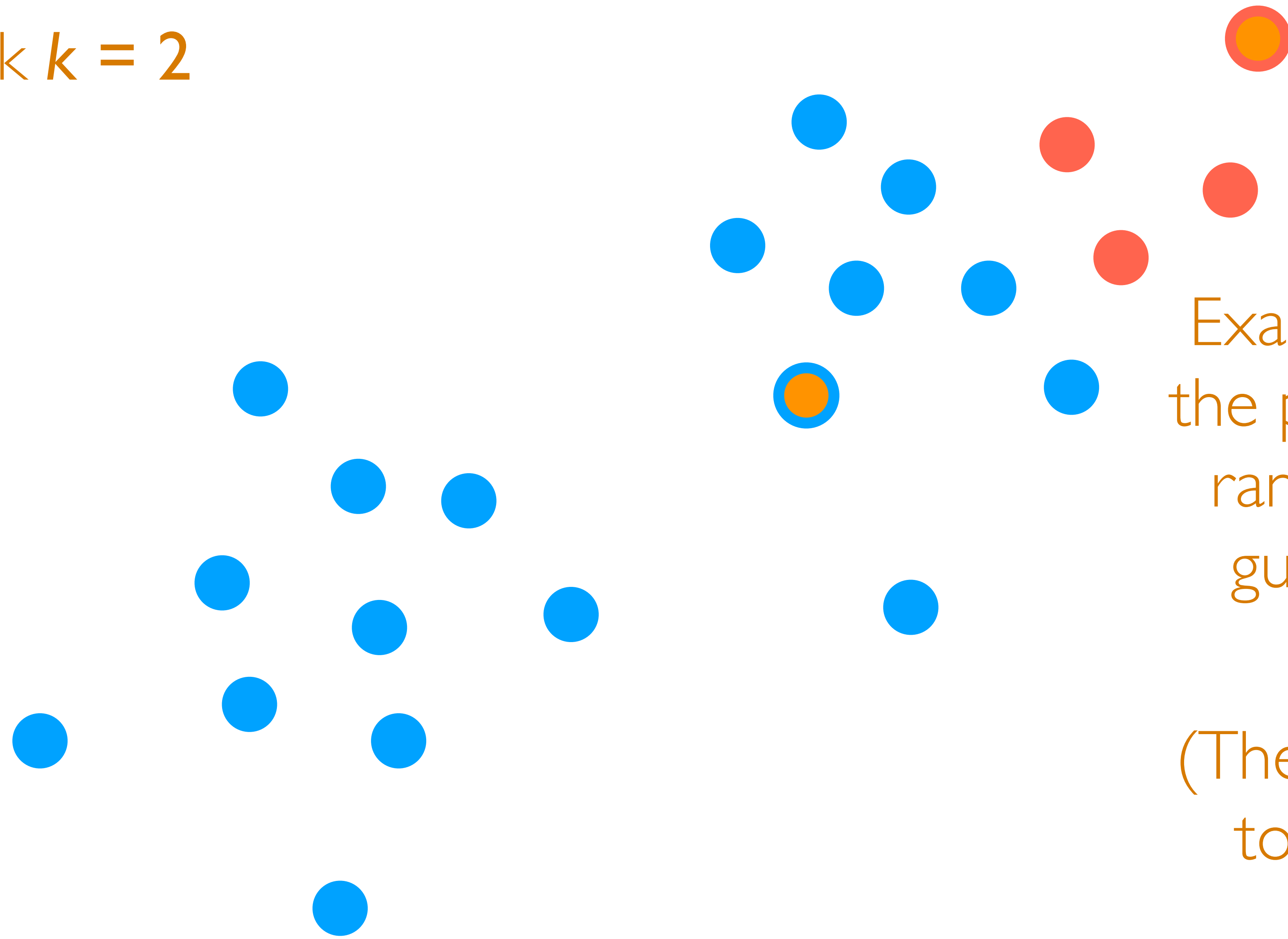
Step 2: Assign each point to belong to the closest cluster

k-means

Step 0: Pick ***k***

We'll pick ***k* = 2**

Step 1: Pick <u>guesses</u> for where cluster centers are



Example: choose ***k*** of the points uniformly at random to be initial guesses for cluster centers

(There are many ways to make the initial guesses)

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

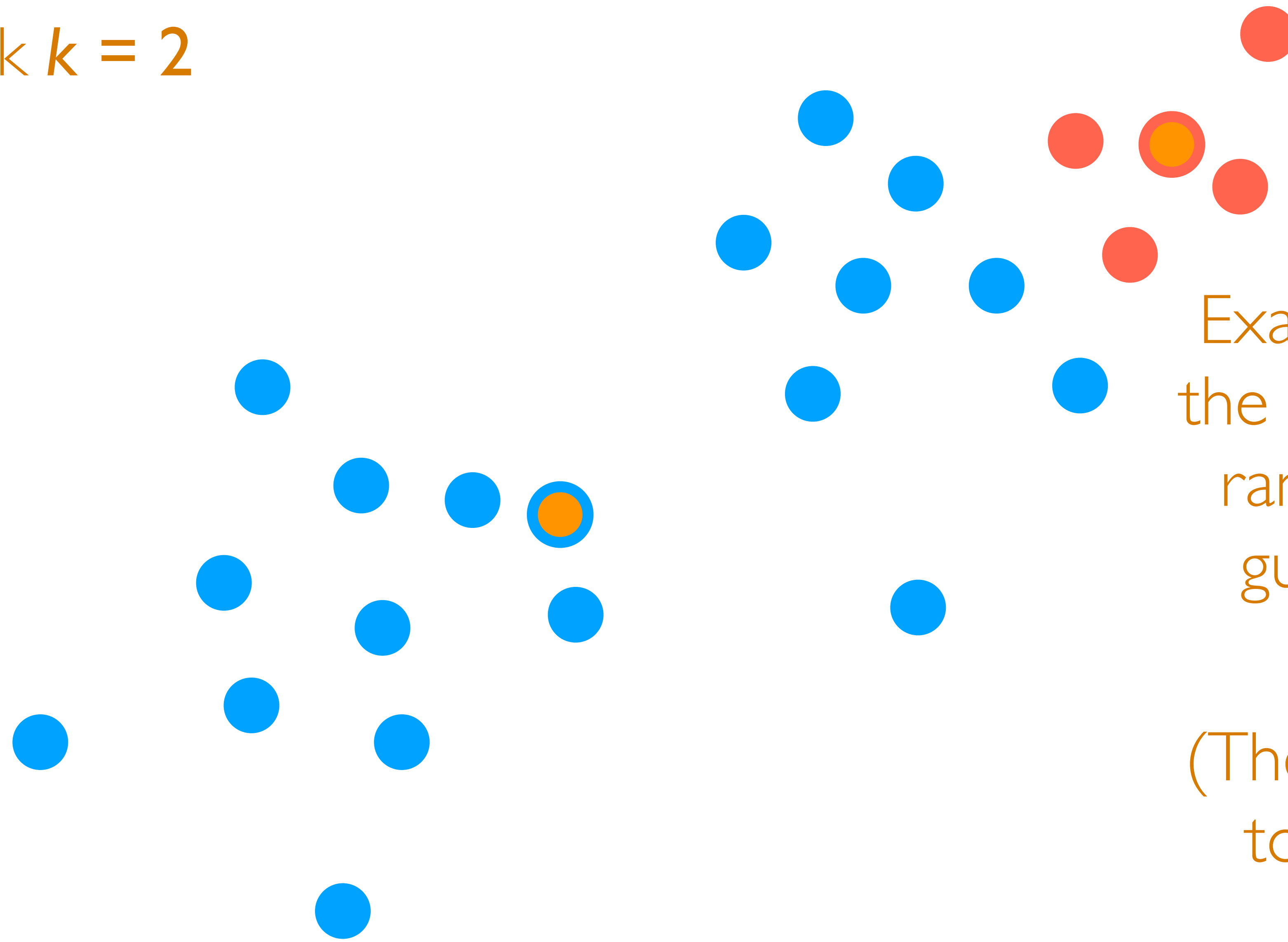
Step 3: Update cluster means (to be the center of mass per cluster)

k-means

Step 0: Pick ***k***

We'll pick ***k* = 2**

Step 1: Pick <u>guesses</u> for where cluster centers are



Example: choose ***k*** of the points uniformly at random to be initial guesses for cluster centers

(There are many ways to make the initial guesses)

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

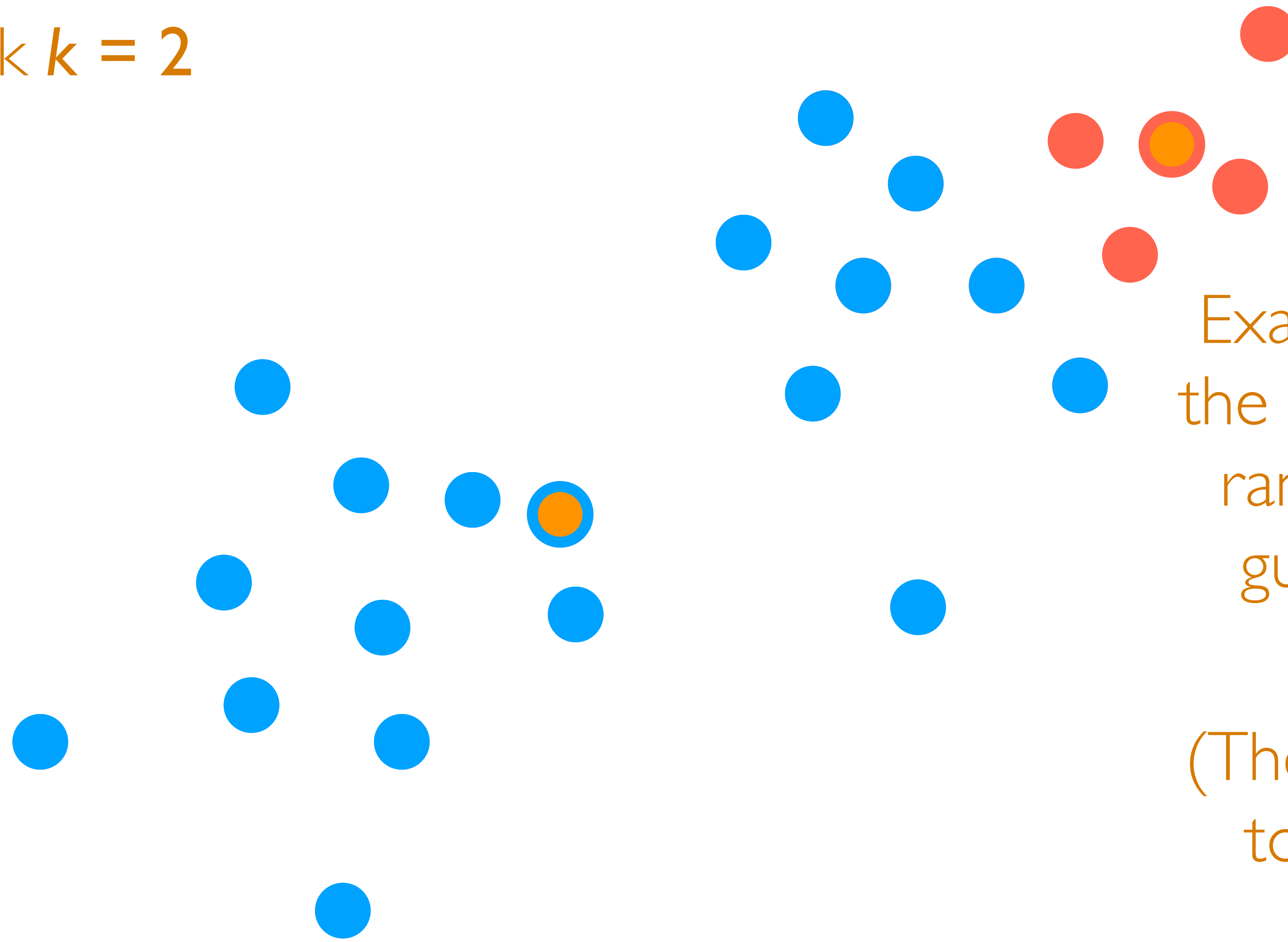
Step 3: Update cluster means (to be the center of mass per cluster)

k-means

Step 0: Pick ***k***

We'll pick ***k* = 2**

Step 1: Pick <u>guesses</u> for where cluster centers are



Example: choose ***k*** of the points uniformly at random to be initial guesses for cluster centers

(There are many ways to make the initial guesses)

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

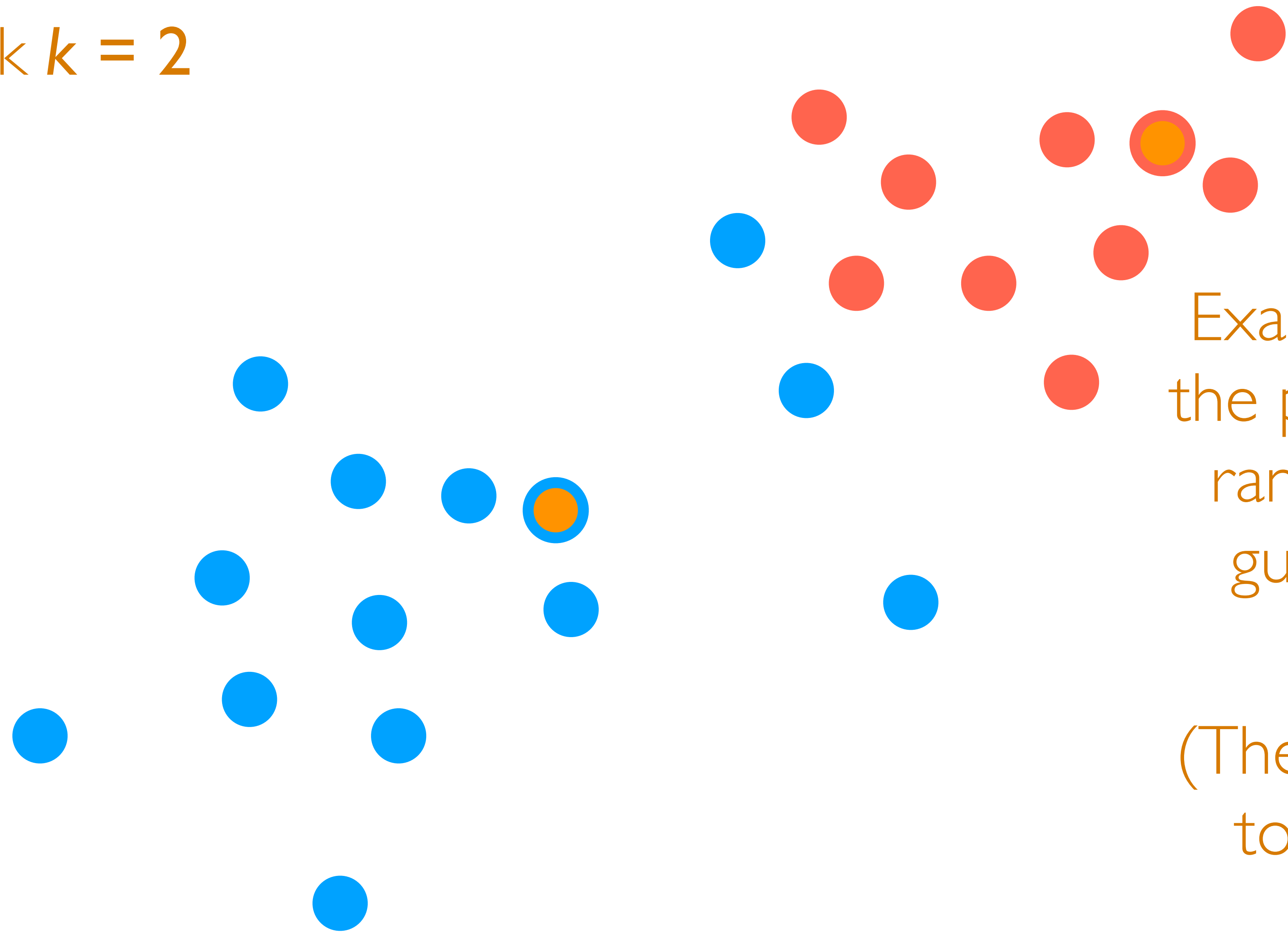
Step 3: Update cluster means (to be the center of mass per cluster)

k-means

Step 0: Pick ***k***

We'll pick ***k* = 2**

Step 1: Pick <u>guesses</u> for where cluster centers are



Example: choose ***k*** of the points uniformly at random to be initial guesses for cluster centers

(There are many ways to make the initial guesses)

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

Repeat

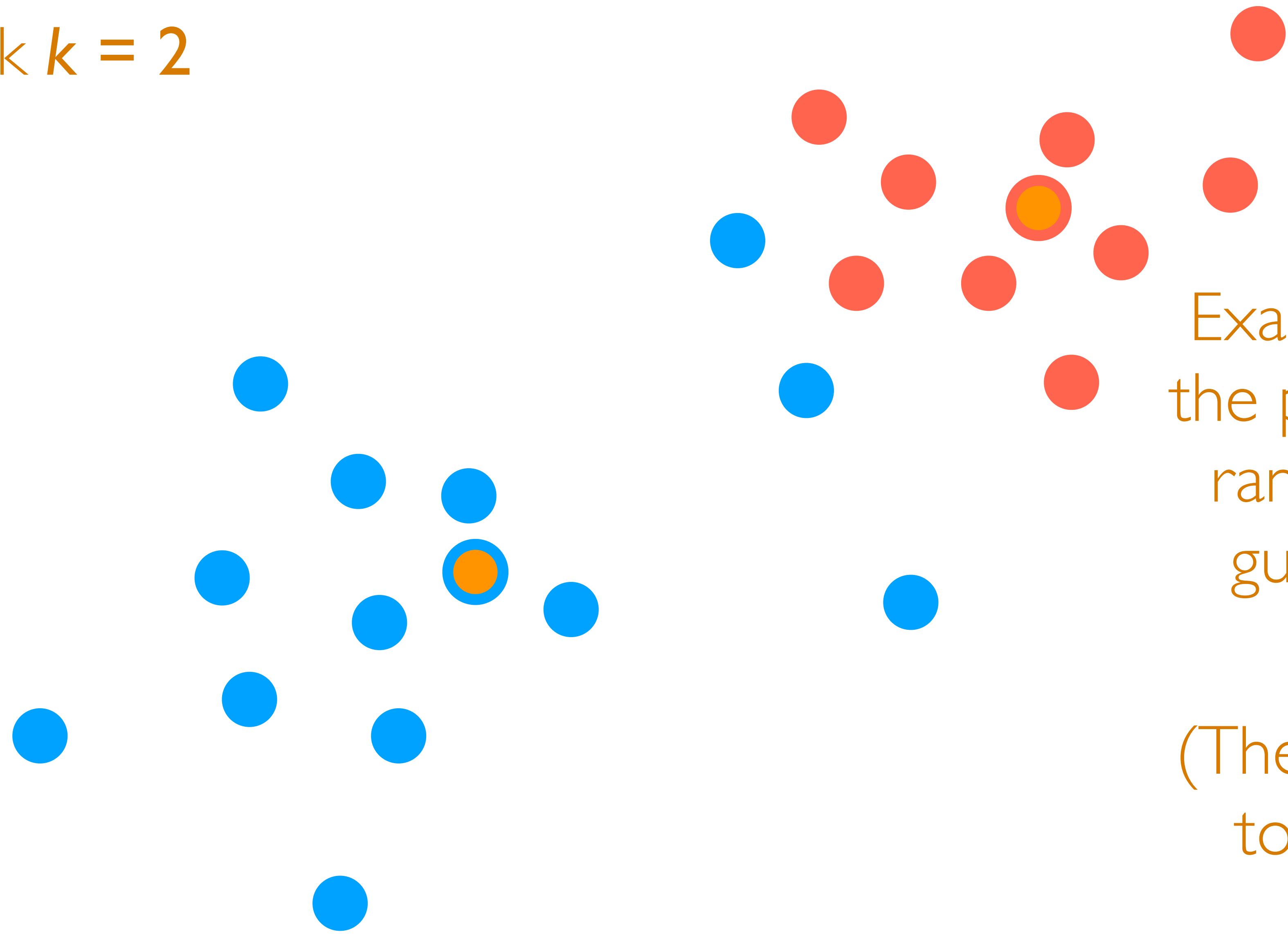
Step 3: Update cluster means (to be the center of mass per cluster)

k-means

Step 0: Pick ***k***

We'll pick ***k* = 2**

Step 1: Pick <u>guesses</u> for where cluster centers are



Example: choose ***k*** of the points uniformly at random to be initial guesses for cluster centers

(There are many ways to make the initial guesses)

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

Repeat

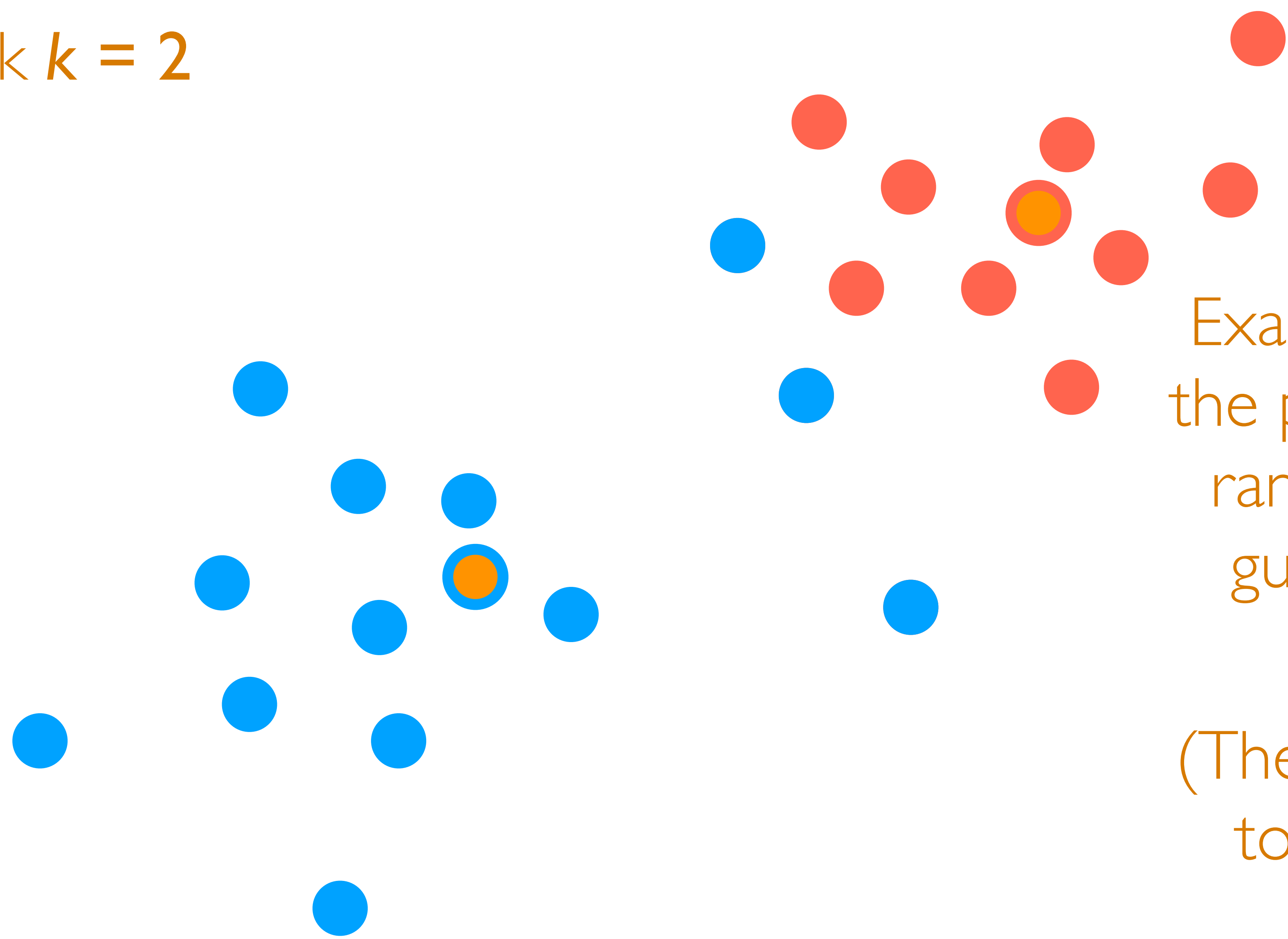
Step 3: Update cluster means (to be the center of mass per cluster)

k-means

Step 0: Pick ***k***

We'll pick ***k* = 2**

Step 1: Pick guesses for where cluster centers are



Example: choose ***k*** of the points uniformly at random to be initial guesses for cluster centers

(There are many ways to make the initial guesses)

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

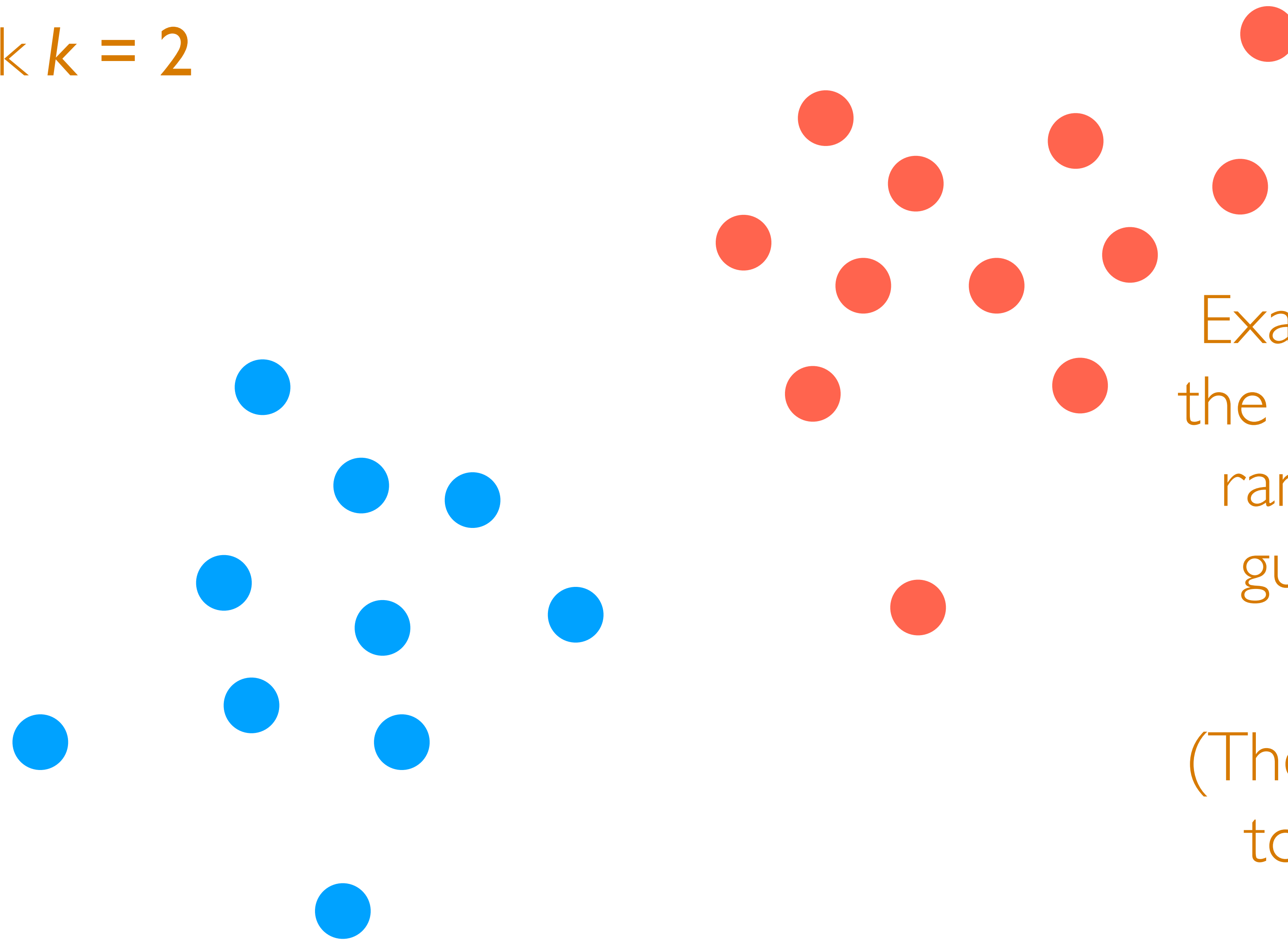
Step 3: Update cluster means (to be the center of mass per cluster)

k-means

Step 0: Pick ***k***

We'll pick ***k* = 2**

Step 1: Pick <u>guesses</u> for where cluster centers are



Example: choose ***k*** of the points uniformly at random to be initial guesses for cluster centers

(There are many ways to make the initial guesses)

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

Repeat

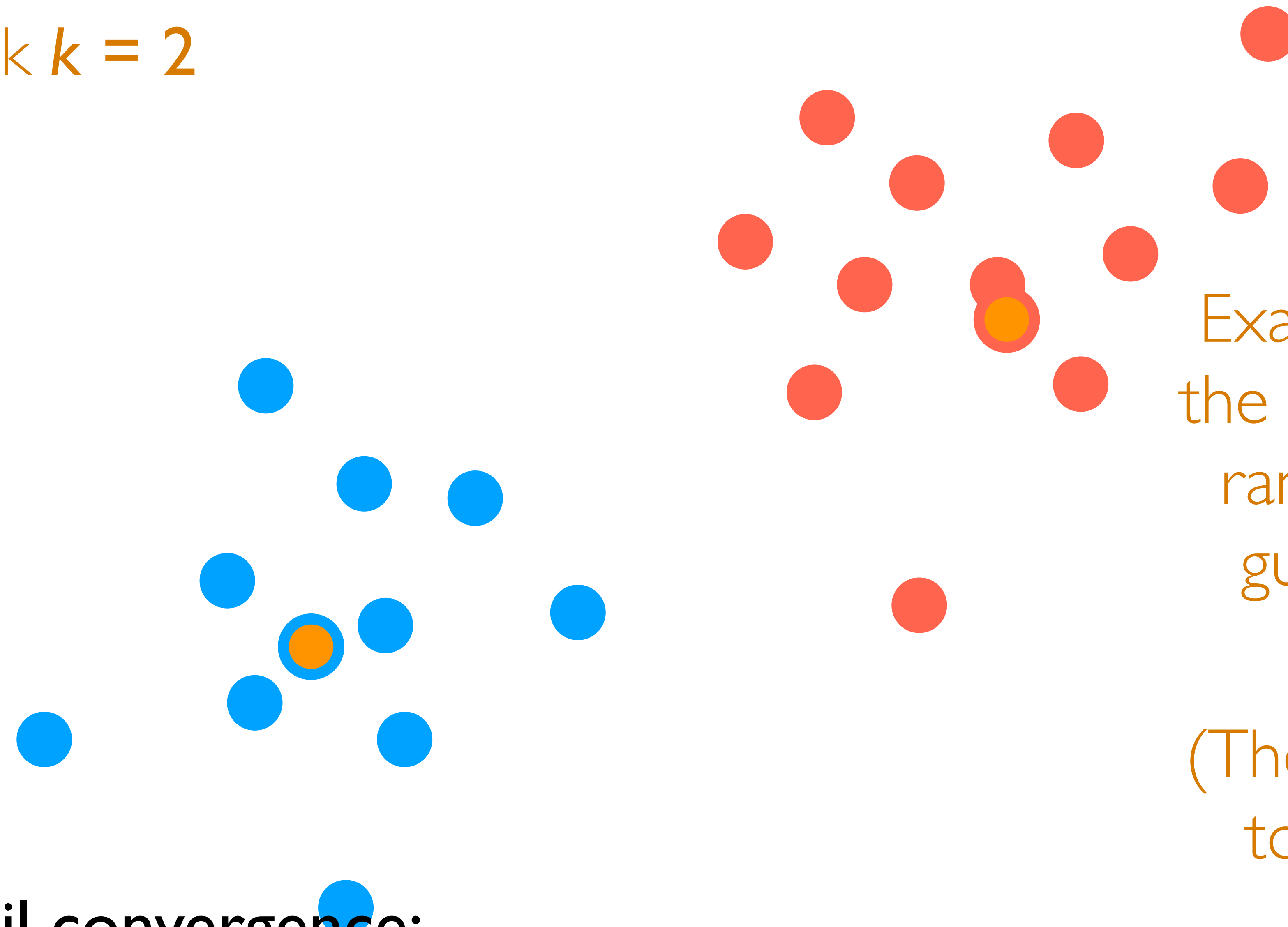
Step 3: Update cluster means (to be the center of mass per cluster)

k-means

Step 0: Pick ***k***

We'll pick ***k* = 2**

Step 1: Pick <u>guesses</u> for where cluster centers are



Example: choose ***k*** of the points uniformly at random to be initial guesses for cluster centers

(There are many ways to make the initial guesses)

Repeat until convergence:

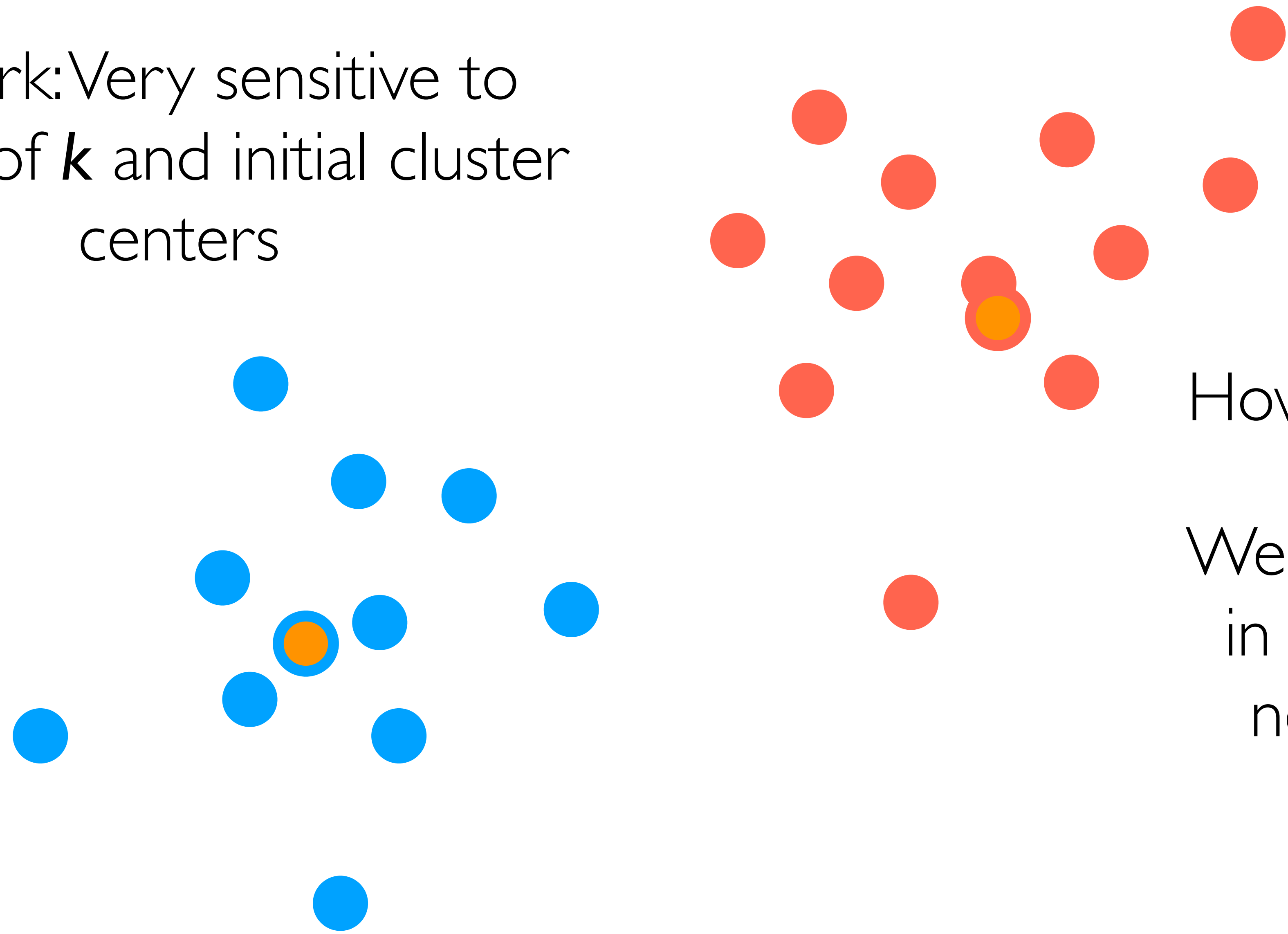
Step 2: Assign each point to belong to the closest cluster

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 pick ***k***?

We'll discuss this in more detail next lecture

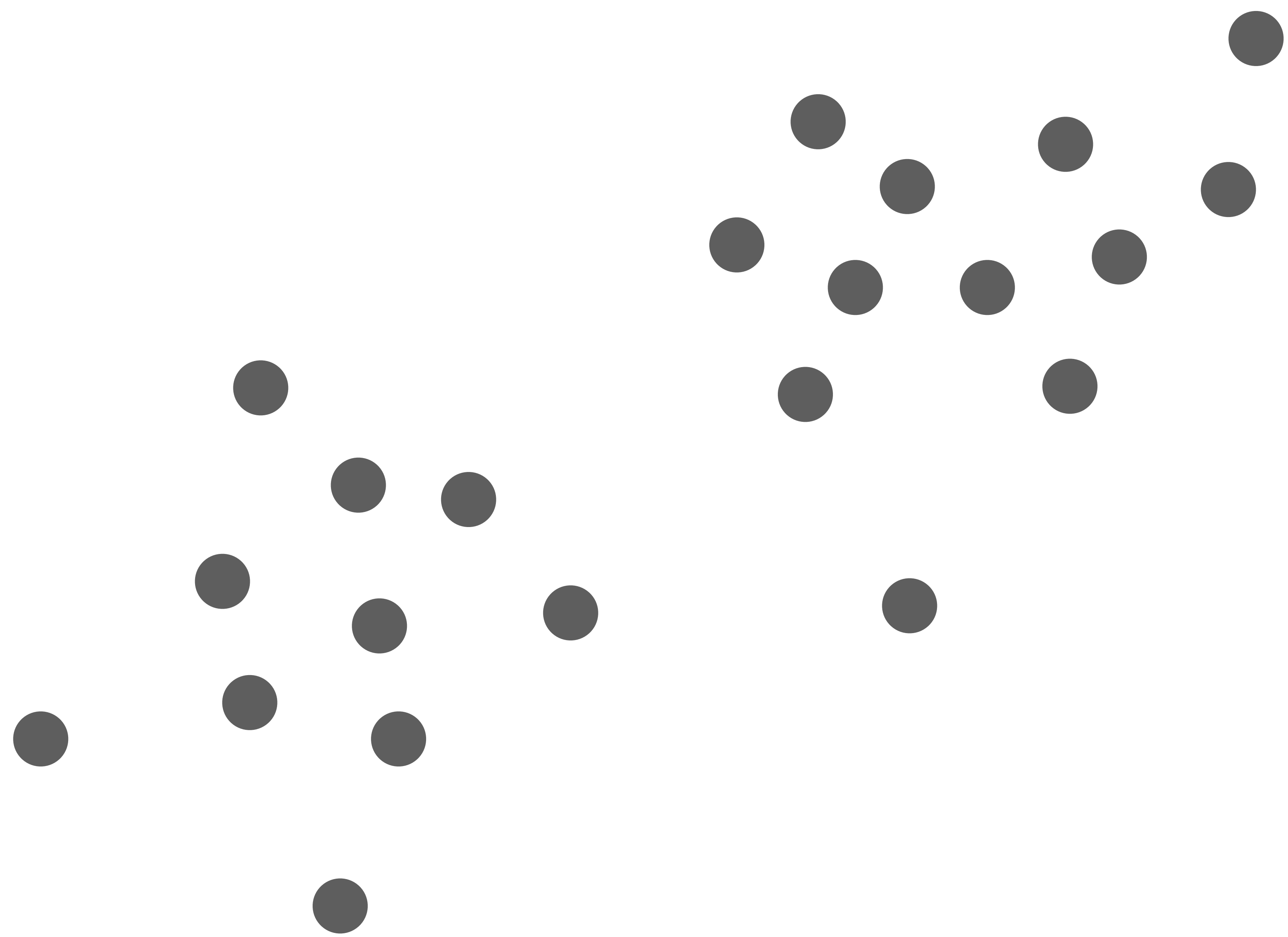
Suggested way to pick initial cluster centers: "*k*-means++" method

(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 more general model, which will help us understand *k*-means

Gaussian Mixture Model (GMM)



What random process could have generated these points?

Generative Process

Think of flipping a coin

each outcome: heads or tails

Each flip doesn't depend on any of the previous flips

Generative Process

Think of flipping a coin

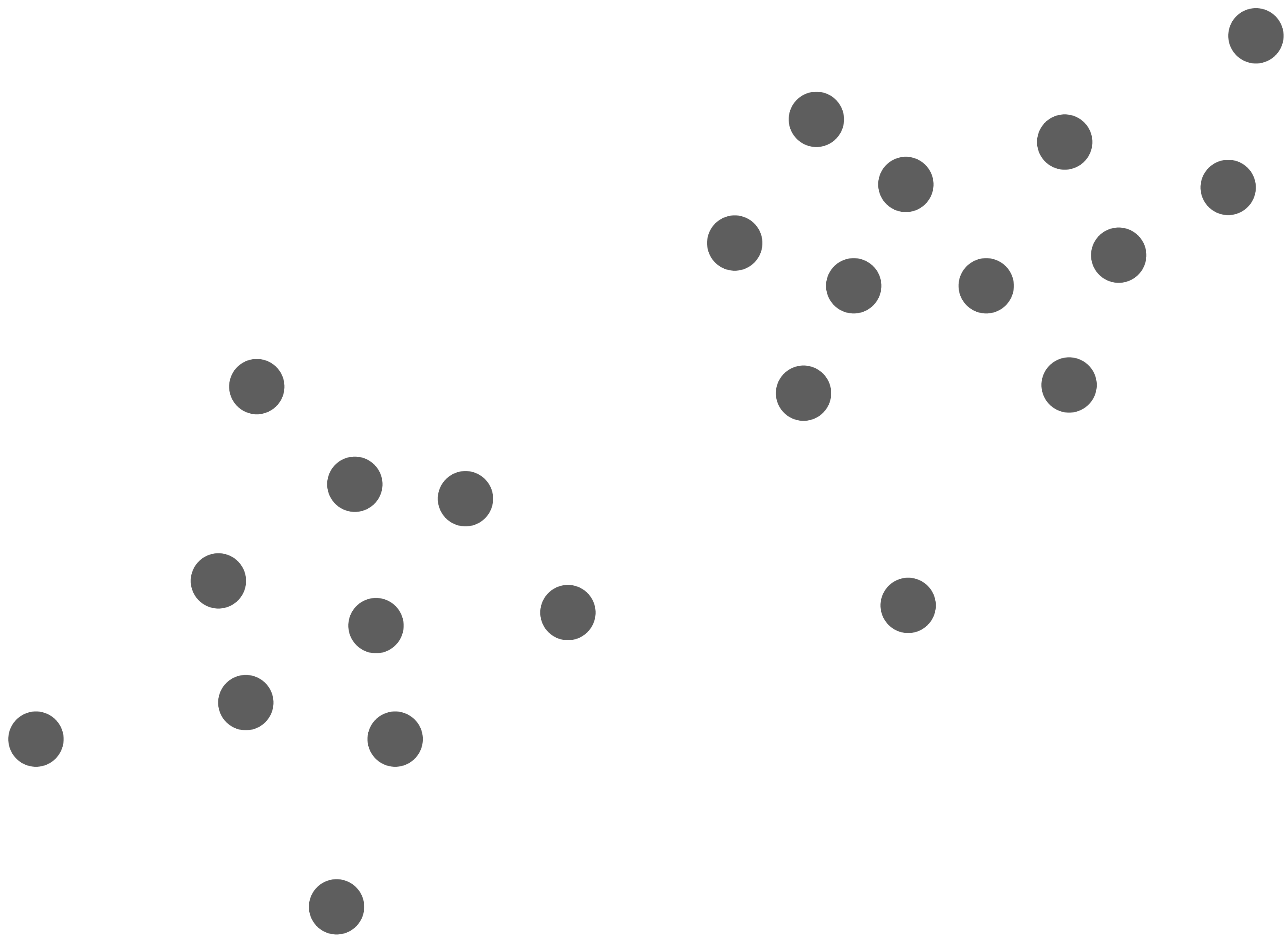
each outcome: 2D point

Each flip doesn't depend on any of the previous flips

Okay, maybe it's bizarre to think of it as a coin…

If it helps, just think of it as you pushing a button and a random 2D point appears…

Gaussian Mixture Model (GMM)

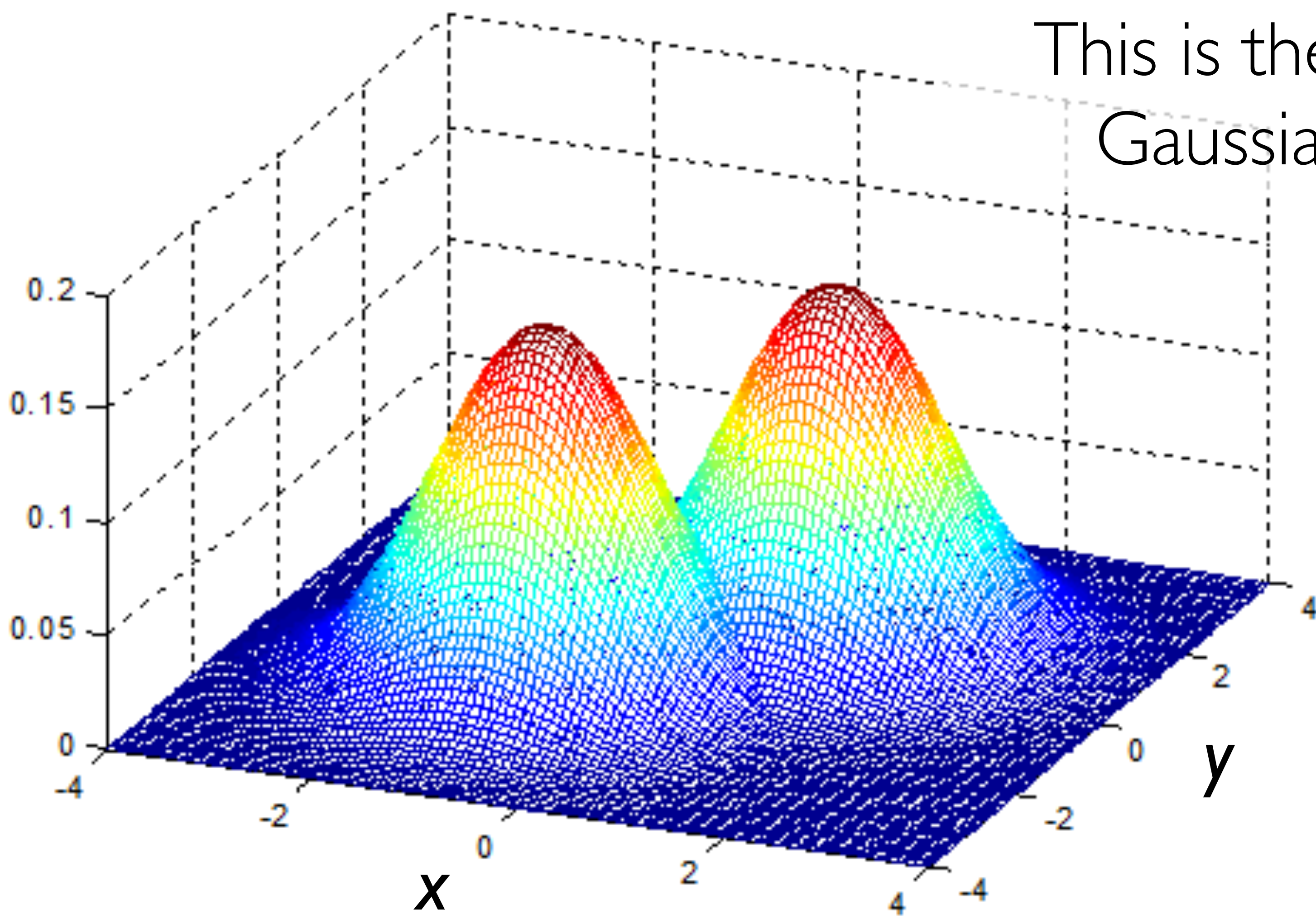


We now discuss a way to generate points in this manner

Gaussian Mixture Model (GMM)

Assume: points sampled independently from a probability distribution

how probable point generated at **(*x*, *y*)** is



This is the sum of two 2D Gaussian distributions!

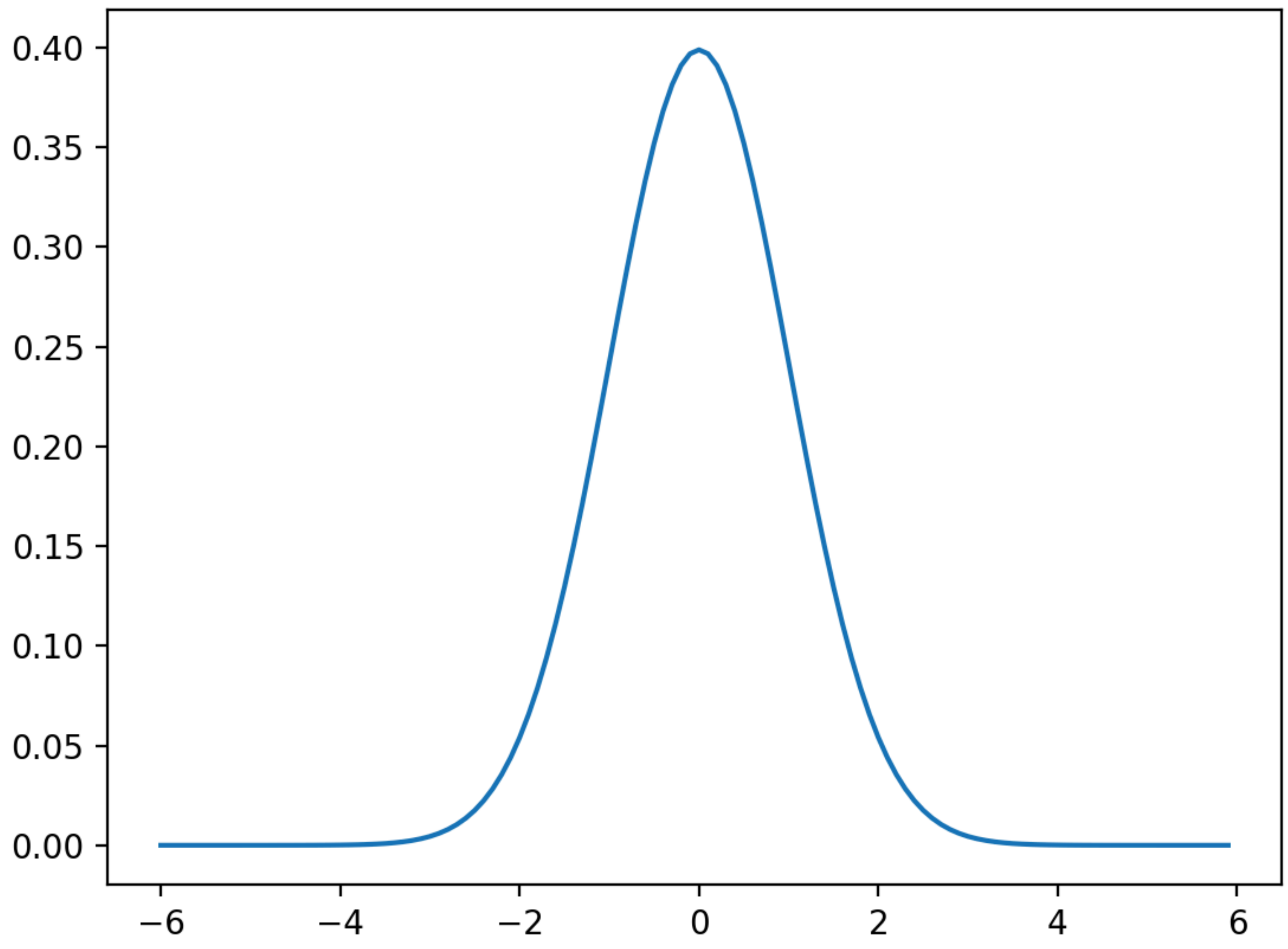
Red = more likely

Blue = less likely

Example of a 2D probability distribution

Image source: https://www.intechopen.com/source/html/17742/media/image25.png

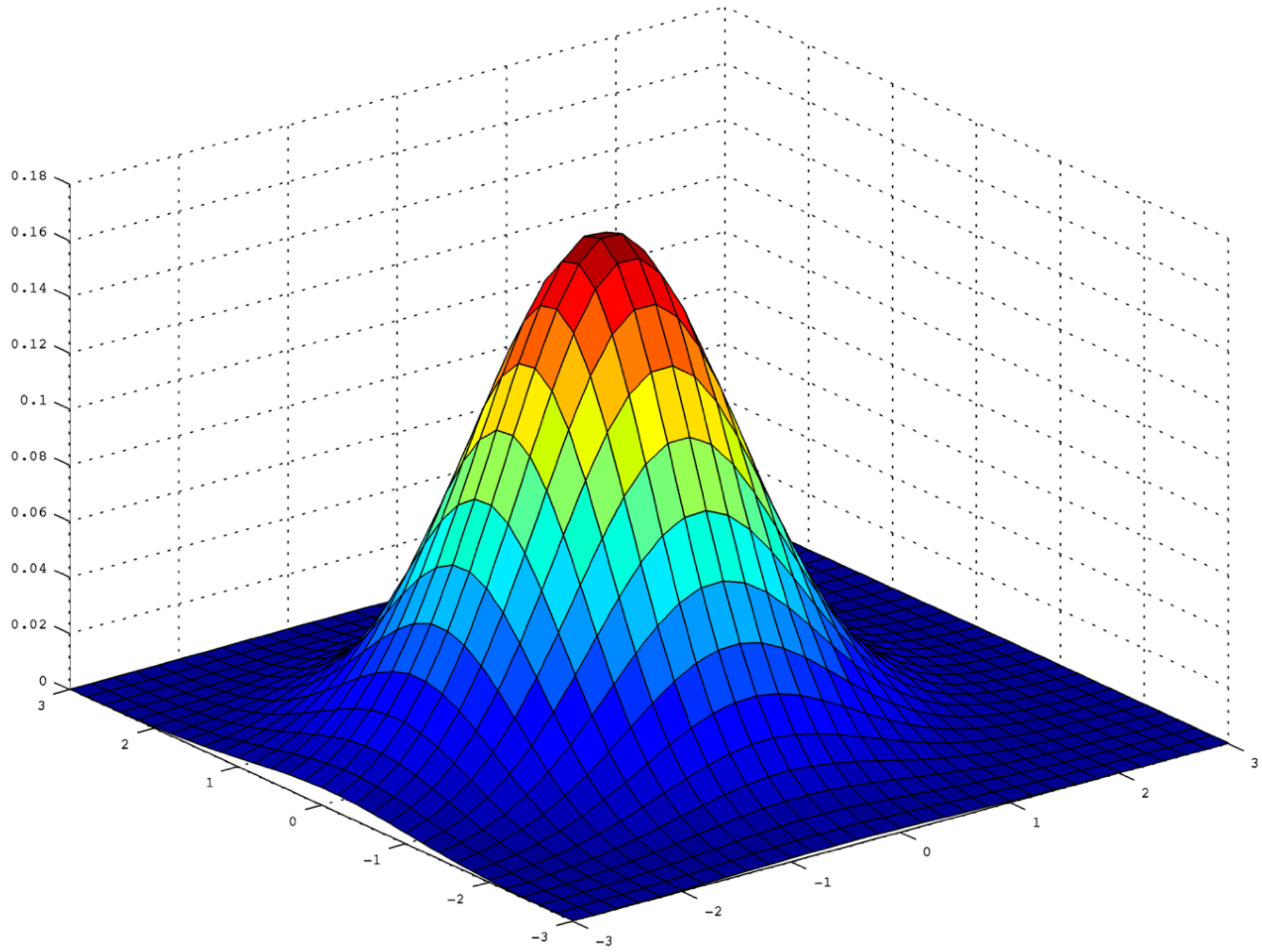
Quick Reminder: 1D Gaussian



This is a 1D Gaussian distribution

Image source: https://matthew-brett.github.io/teaching//smoothing_intro-3.hires.png

2D Gaussian



This is a 2D Gaussian distribution

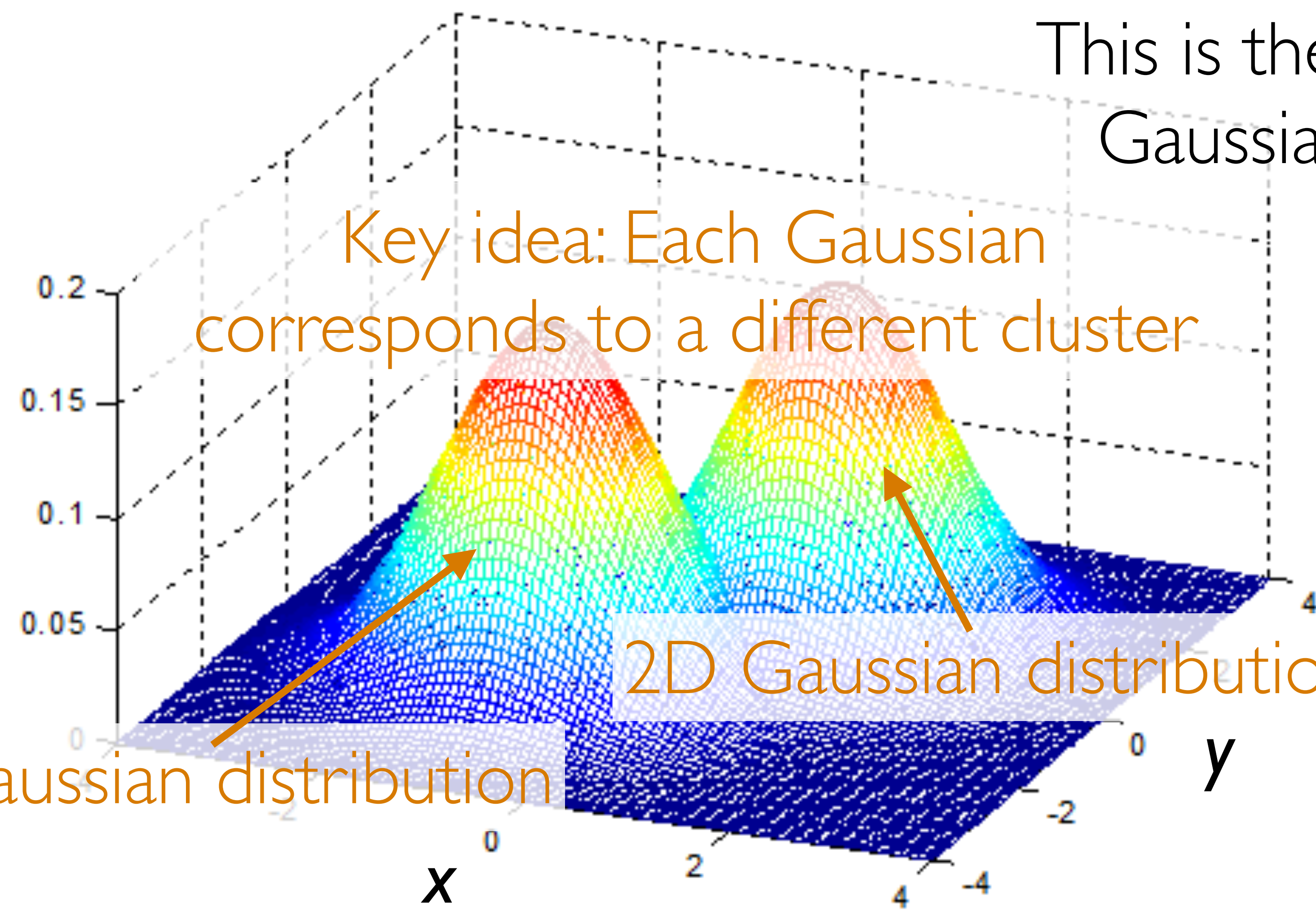
Image source: https://i.stack.imgur.com/OIWce.png

Gaussian Mixture Model (GMM)

Assume: points sampled independently from a probability distribution

This is the sum of two 2D Gaussian distributions!

how probable point generated at **(*x*, *y*)** is



Red = more likely

Blue = less likely

Example of a 2D probability distribution

Image source: https://www.intechopen.com/source/html/17742/media/image25.png

Gaussian Mixture Model (GMM)

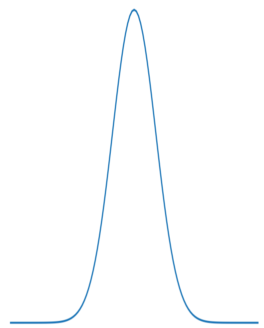
- For a fixed value ***k*** and dimension ***d***, a GMM is the sum of ***k*** ***d***-dimensional Gaussian distributions so that the overall probability distribution looks like ***k*** mountains

(We've been looking at ***d* = 2**)

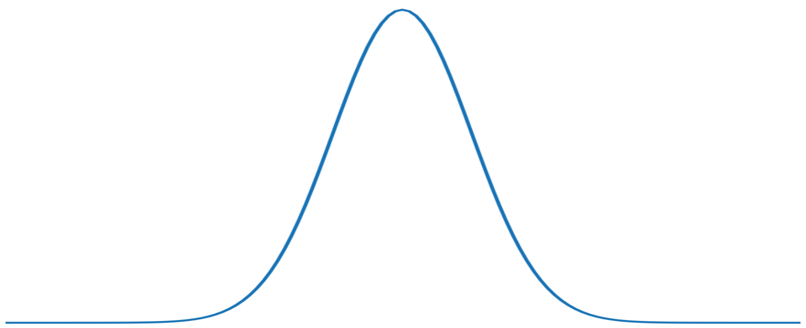
  - Each mountain corresponds to a different cluster
  - Different mountains can have different peak heights
  - One missing thing we haven't discussed yet:
    different mountains can have different shapes

2D Gaussian Shape

In 1D, you can have a skinny Gaussian or a wide Gaussian



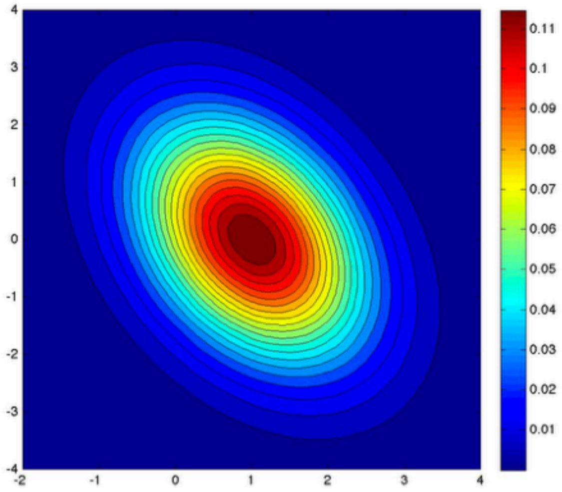
Less uncertainty



More uncertainty

In 2D, you can more generally have ellipse-shaped Gaussians

Ellipse enables encoding relationship between variables



Can't have arbitrary shapes

Top-down view of an example 2D Gaussian distribution

Image source: https://www.cs.colorado.edu/~mozer/Teaching/syllabi/ProbabilisticModels2013/homework/assign5/a52dgauss.jpg

Gaussian Mixture Model (GMM)

- For a fixed value ***k*** and dimension ***d***, a GMM is the sum of ***k*** ***d***-dimensional Gaussian distributions so that the overall probability distribution looks like ***k*** mountains

(We've been looking at ***d* = 2**)

  - Each mountain corresponds to a different cluster
  - Different mountains can have different peak heights
  - Different mountains can have different ellipse shapes (captures "covariance" information)

Example: 1D GMM with 2 Clusters

<u>Cluster 1</u>

Probability of generating a point from cluster 1 = 0.5

Gaussian mean = −5

Gaussian std dev = 1

<u>Cluster 2</u>

Probability of generating a point from cluster 2 = 0.5

Gaussian mean = 5

Gaussian std dev = 1

What do you think this looks like?

Example: 1D GMM with 2 Clusters

Cluster 1

Probability of generating a point from cluster 1 = 0.5

Gaussian mean = −5

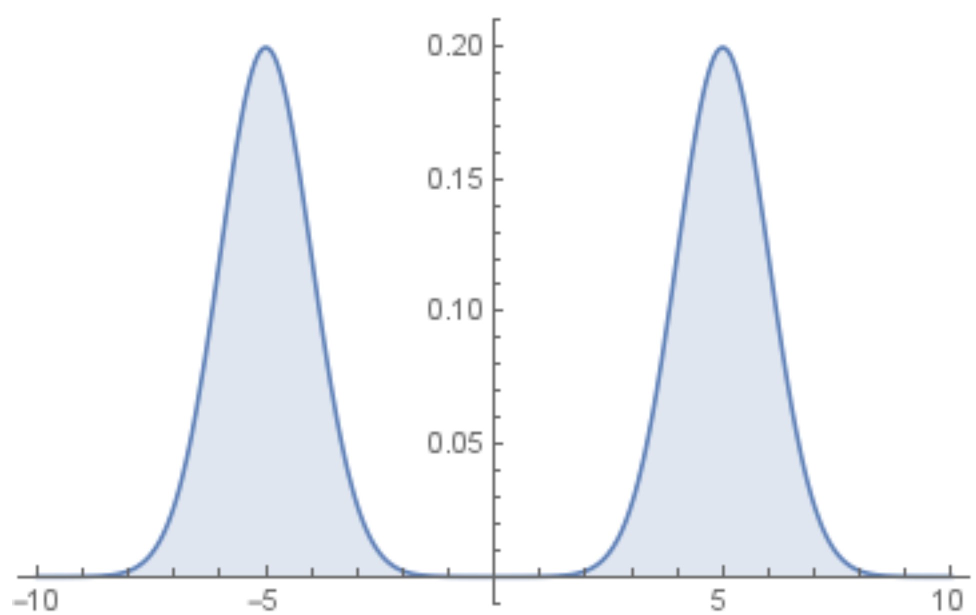
Gaussian std dev = 1

Cluster 2

Probability of generating a point from cluster 2 = 0.5

Gaussian mean = 5

Gaussian std dev = 1



Example: 1D GMM with 2 Clusters

Cluster 1

Probability of generating a point from cluster 1 = **0.7**

Gaussian mean = −5

Gaussian std dev = 1

Cluster 2

Probability of generating a point from cluster 2 = **0.3**

Gaussian mean = 5

Gaussian std dev = 1

What do you think this looks like?

Example: 1D GMM with 2 Clusters

<u>Cluster 1</u>

Probability of generating a point from cluster 1 = 0.7

Gaussian mean = −5

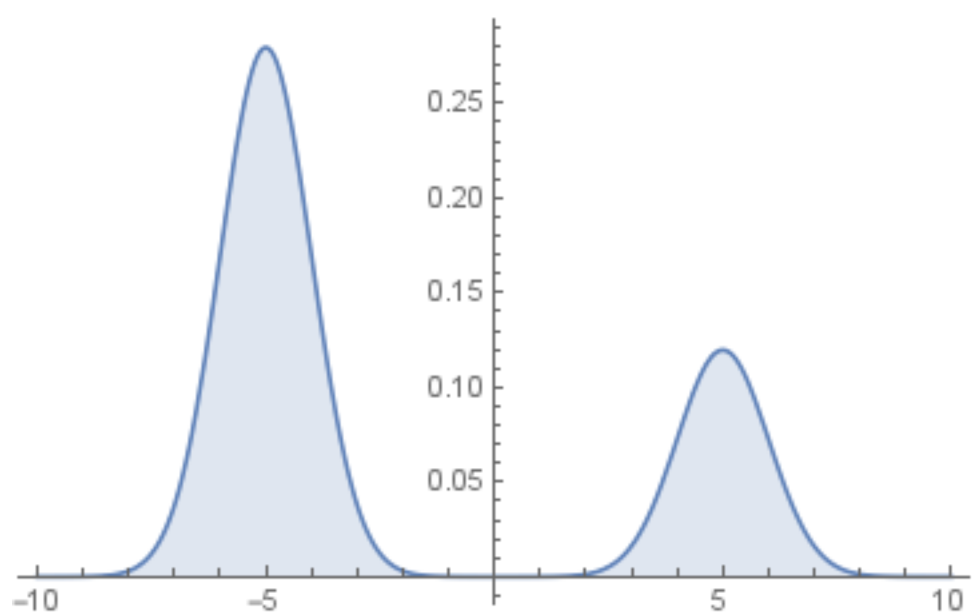
Gaussian std dev = 1

<u>Cluster 2</u>

Probability of generating a point from cluster 2 = 0.3

Gaussian mean = 5

Gaussian std dev = 1



Example: 1D GMM with 2 Clusters

<u>Cluster 1</u>

Probability of generating a point from cluster 1 = 0.7

Gaussian mean = −5

Gaussian std dev = 1

<u>Cluster 2</u>

Probability of generating a point from cluster 2 = 0.3

Gaussian mean = 5

Gaussian std dev = 1

How to generate 1D points from this GMM:

1. Flip biased coin (with probability of heads 0.7)
2. If heads: sample 1 point from Gaussian mean -5, std dev 1

   If tails: sample 1 point from Gaussian mean 5, std dev 1

Example: 1D GMM with 2 Clusters

Cluster 1

Probability of generating a point from cluster 1 = π_1

Gaussian mean = μ_1

Gaussian std dev = σ_1

Cluster 2

Probability of generating a point from cluster 2 = π_2

Gaussian mean = μ_2

Gaussian std dev = σ_2

How to generate 1D points from this GMM:

1. Flip biased coin (with probability of heads π_1)
2. If heads: sample 1 point from Gaussian mean μ_1, std dev σ_1

   If tails: sample 1 point from Gaussian mean μ_2, std dev σ_2

Example: 1D GMM with *k* Clusters

Cluster 1

Probability of generating a point from cluster 1 $= \pi_1$

Gaussian mean $= \mu_1$

Gaussian std dev $= \sigma_1$

…

Cluster ***k***

Probability of generating a point from cluster ***k*** $= \pi_k$

Gaussian mean $= \mu_k$

Gaussian std dev $= \sigma_k$

How to generate 1D points from this GMM:

1. Flip biased ***k***-sided coin (the sides have probabilities $\pi_1, \dots, \pi_k$)
2. Let ***Z*** be the side that we got (it is some value 1, …, ***k***)
3. Sample 1 point from the Gaussian for cluster ***Z***