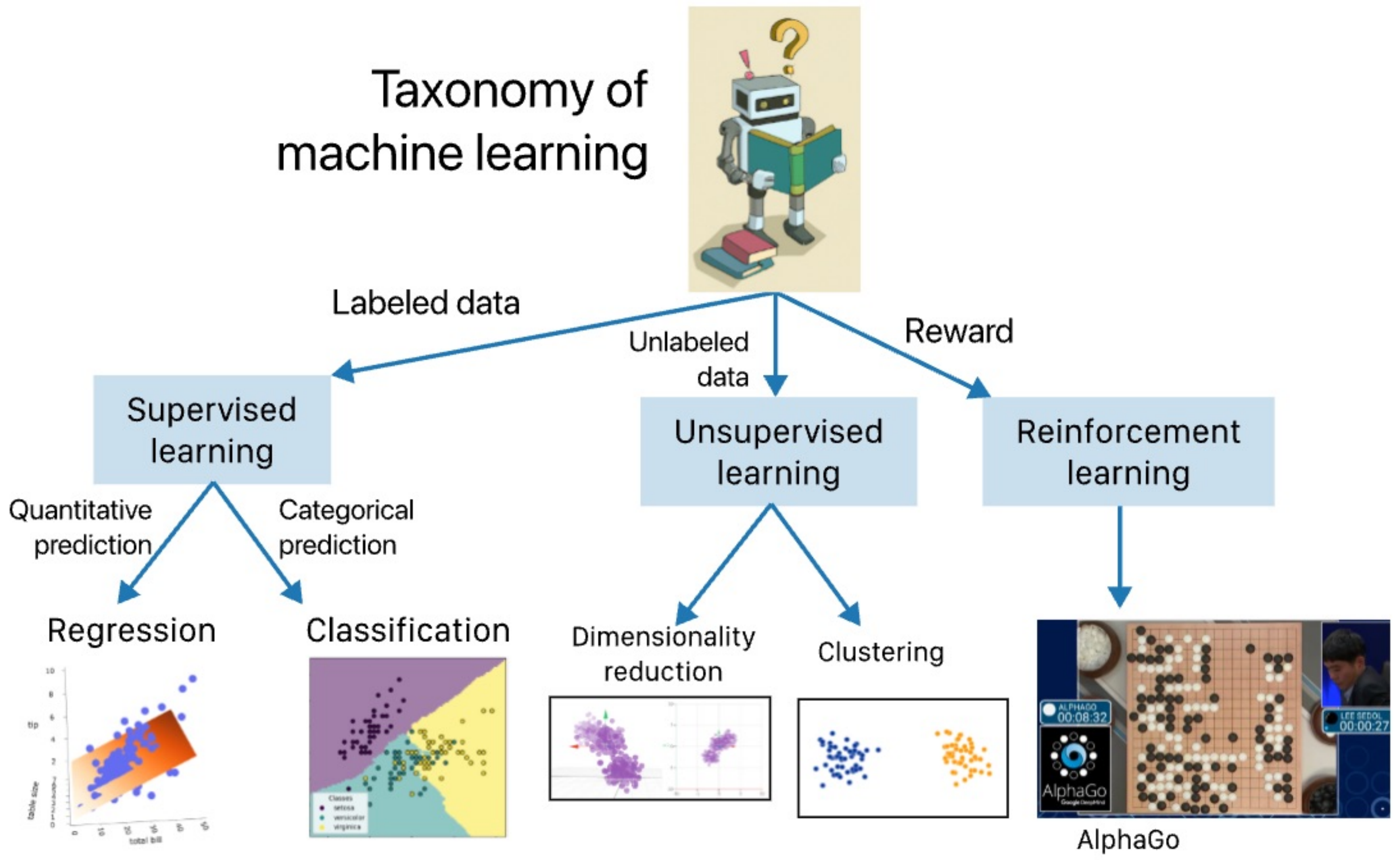
The taxonomy of machine learning



use X to predict y

$\Rightarrow$ supervised learning.

- In Lectures 11-19, we focused on building models for **regression**.

  In regression, we predict a **continuous** target variable, y, using some features, X.

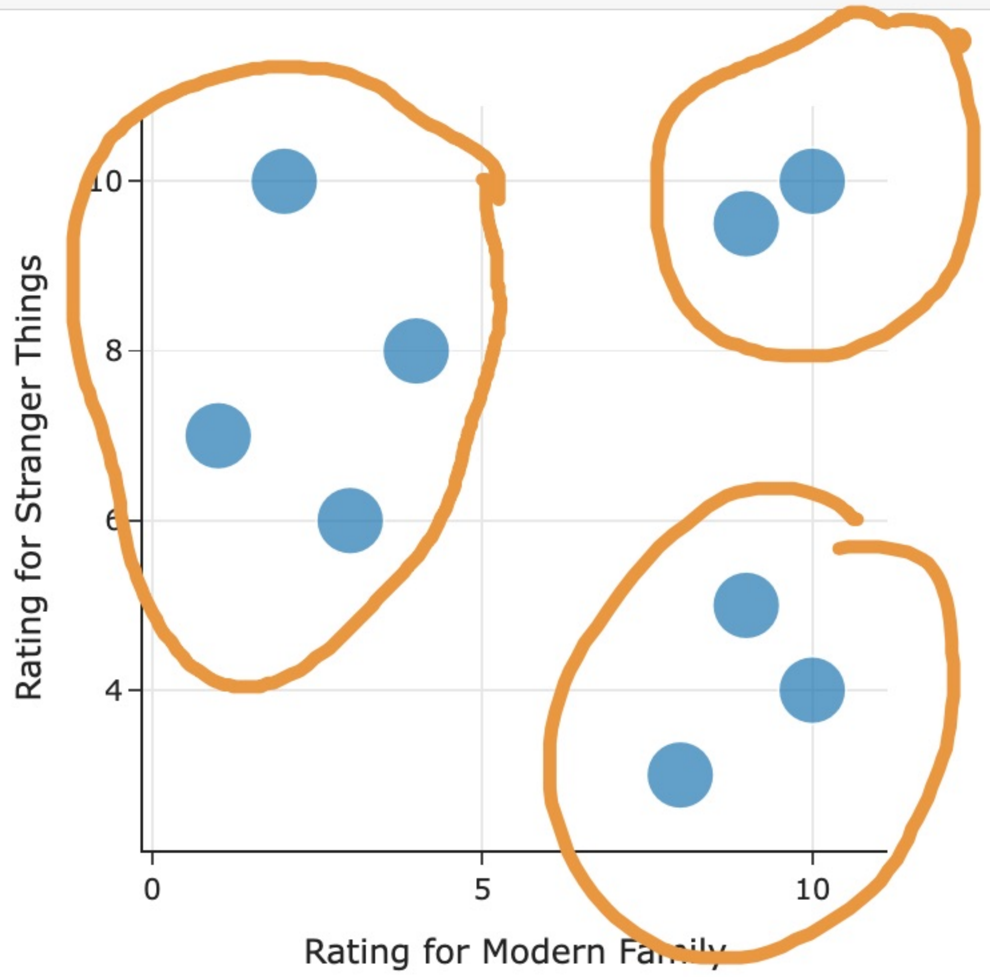
- In the past few lectures, we switched our focus to building models for **classification**.

  In classification, we predict a **categorical** target variable, y, using some features, X.

Example: TV show ratings 📺

- Suppose we have the ratings that several customers of a streaming service gave to two popular TV shows: *Modern Family* and *Stranger Things*.

In [32]:
```
1 util.show_ratings()
```



- The data naturally falls into three groups, or **clusters**, based on users with similar preferences.

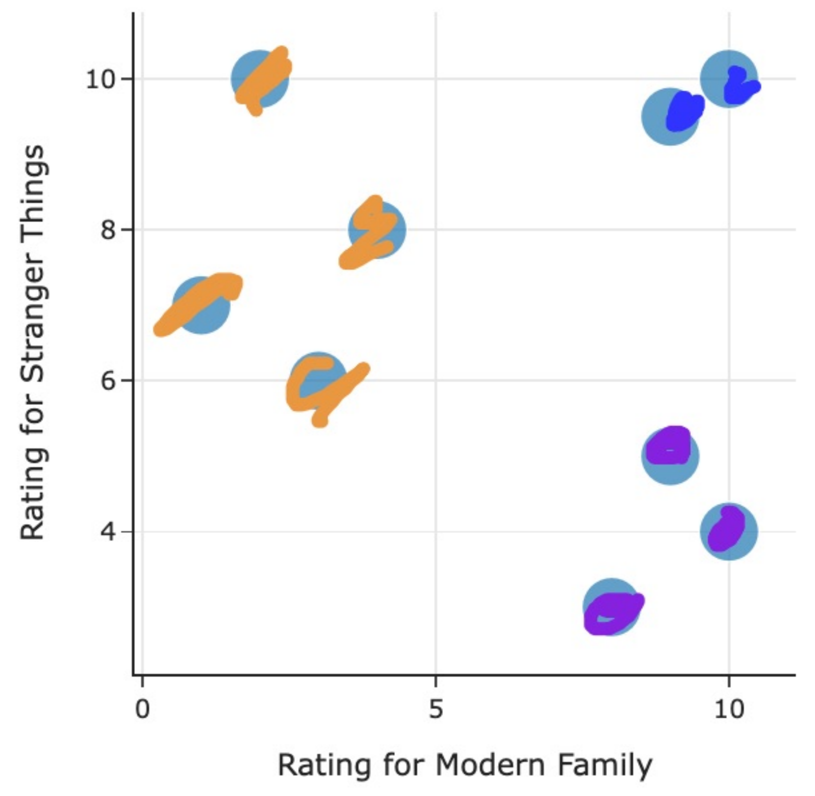
All we're given are the ratings each customer gave to the two shows; the customers aren't already part of any group.

Clustering

- **Goal**: Given a set of n data points stored as vectors in $\mathbb{R}^d$, $\vec x_1, \vec x_2, \ldots, \vec x_n$, and a positive integer k, **place the data points into k clusters of nearby points**.

  In the scatter plot below, $n = 9$ and $d = 2$.

In [33]:
```
util.show_ratings()
```



- Think of clusters as **colors**; in other words, the goal of clustering is to assign each point a color, such that points of the same color are similar to one another.

- Note, unlike with regression or classification, there is no "right answer" that we're trying to predict – there is no y! This is what makes clustering **unsupervised**.

number of clusters

Reflections on choosing a centroid

- Some values of k seemed more intuitive than others; k is a **hyperparameter** that we'll need to tune.

  More on this later.

Reflections on choosing a centroid

- Some values of k seemed more intuitive than others; k is a **hyperparameter** that we'll need to tune.
  More on this later.

- For a fixed k, some clusterings "looked" better than others; we'll need a way to quantify this.

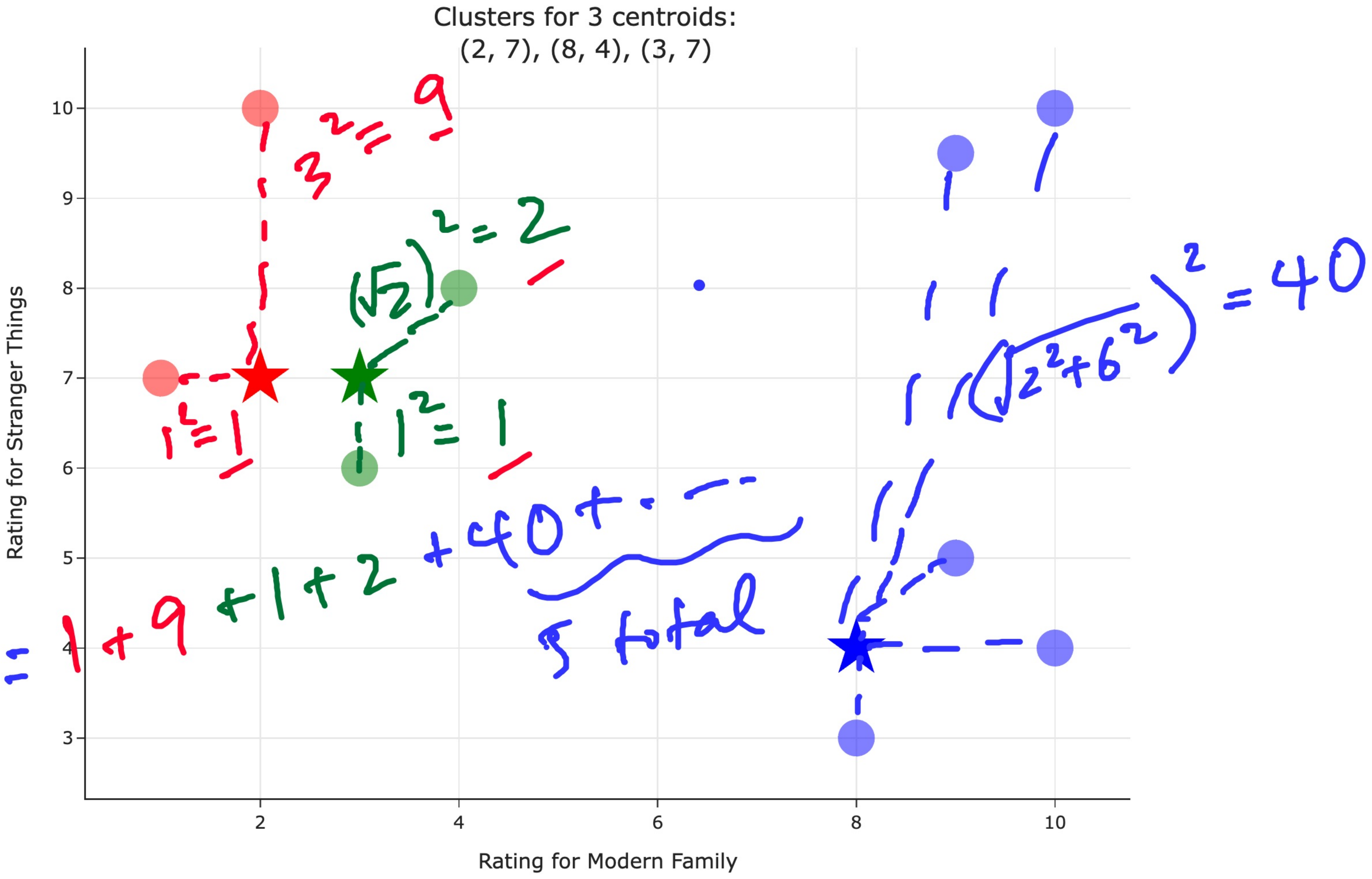
- As we did at the start of the second half of the course, we'll formulate an **objective function** to minimize. Specifically, we'll minimize **inertia**, I:

$$I(\vec{\mu}_1, \vec{\mu}_2, \ldots, \vec{\mu}_k) = \text{total squared distance of each point } \vec{x}_i \text{ to its closest centroid } \vec{\mu}_j$$

inertia

- Lower values of inertia lead to better clusterings; our goal is to find the set of centroids $\vec{\mu}_1, \vec{\mu}_2, \ldots \vec{\mu}_k$ that **minimize inertia**, I.

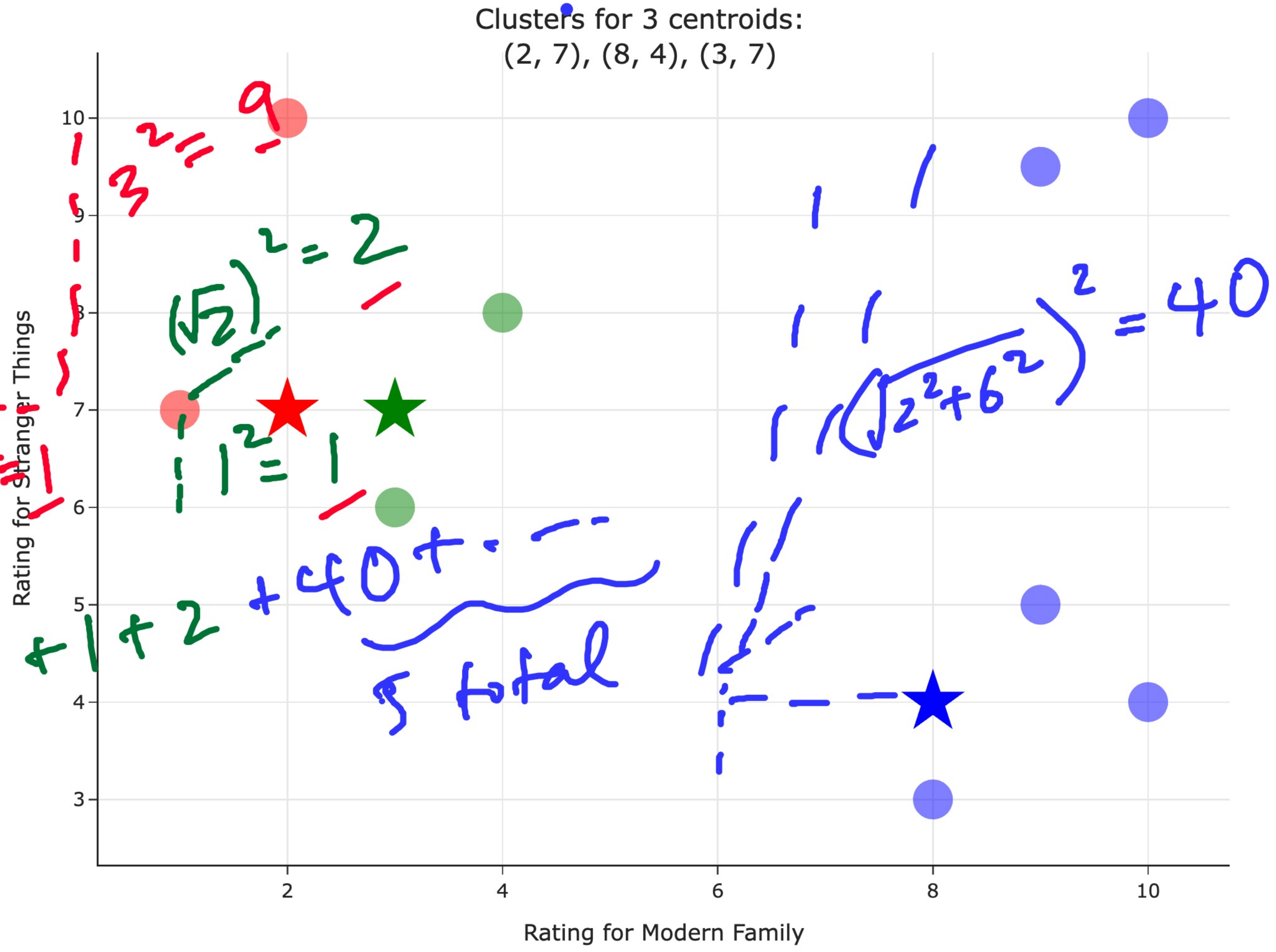
In [3]:
```
util.visualize_centroids([(2, 7), (8, 4), (3, 7)])
```



inertia = $1 + 9 + 1 + 2 + \underbrace{40 + \ldots}_{5\ \text{total}}$

In [3]:

```python
util.visualize_centroids([(2, 7), (8, 4), (3, 7)])
```



$$\text{inertia} = 1 + 9 + 1 + 2 + \underbrace{40 + \ldots}_{5 \text{ total}}$$

Reflections on choosing a centroid

- Some values of k seemed more intuitive than others; k is a **hyperparameter** that we'll need to tune.

  More on this later.

- For a fixed k, some clusterings "looked" better than others; we'll need a way to quantify this.

- As we did at the start of the second half of the course, we'll formulate an **objective function** to minimize. Specifically, we'll minimize **inertia**, I:

$$I(\vec{\mu}_1, \vec{\mu}_2, \ldots, \vec{\mu}_k) = \text{total squared distance of each point } \vec{x}_i \text{ to its closest centroid } \vec{\mu}_j$$

L_2 / Pythagorean / Euclidean

$$\sum \|\vec{x}_i - \vec{\mu}_j\|_2^2$$

- Lower values of inertia lead to better clusterings; our goal is to find the set of centroids $\vec{\mu}_1, \vec{\mu}_2, \ldots \vec{\mu}_k$ that **minimize inertia**, I.

$$d = \sqrt{a^2 + b^2}$$

Audio Show meeting More

You are screen sharing Stop share

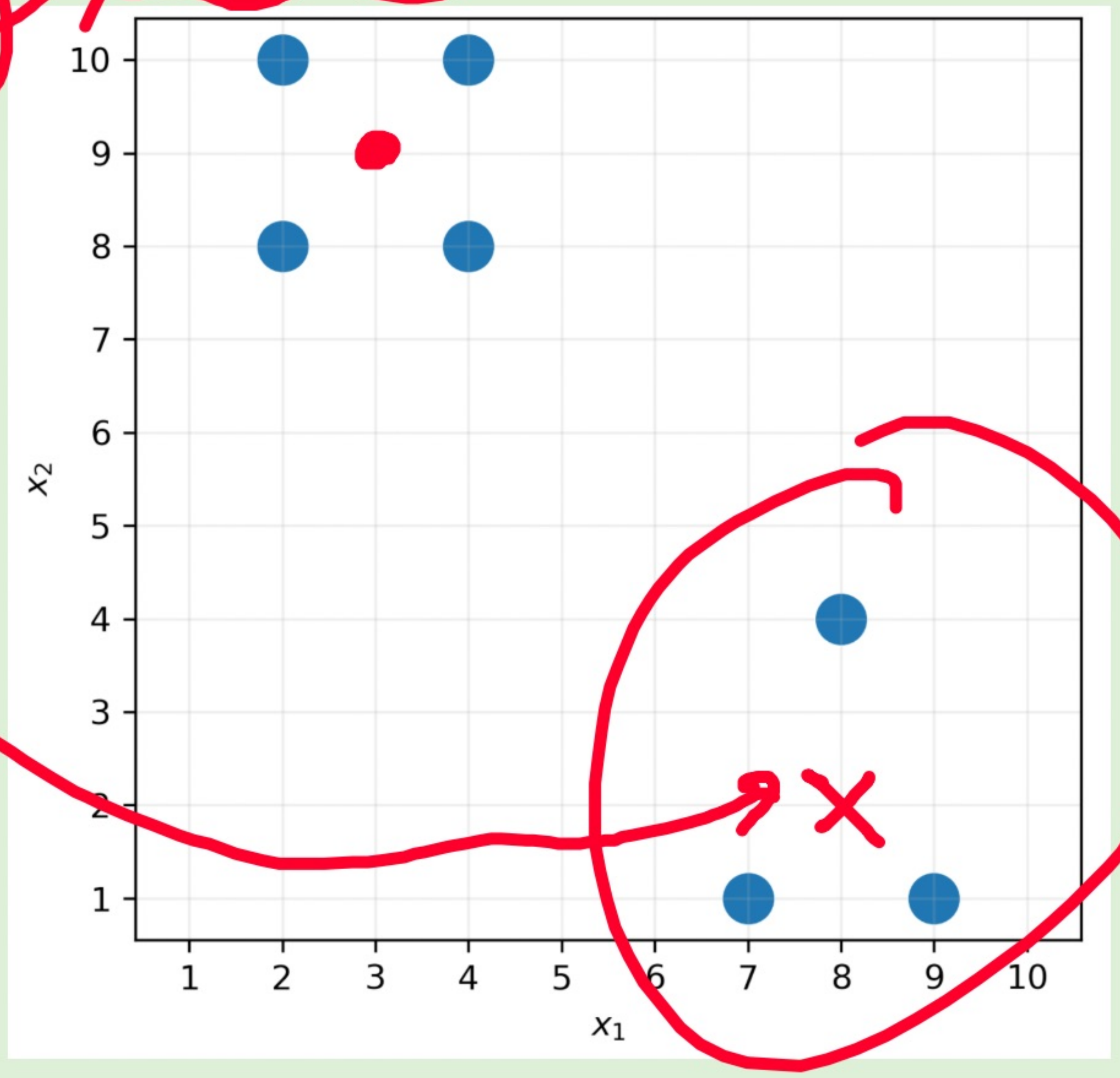
Talking:

Activity

Recall, inertia is defined as follows:

$$I(\vec{\mu}_1, \vec{\mu}_2, \ldots, \vec{\mu}_k) = \text{total squared distance of each point } \vec{x}_i \text{ to its closest centroid } \vec{\mu}_j$$

Suppose we arrange the dataset below into $k = 2$ clusters. What is the **minimum possible inertia**?



(mean of x, mean of y)

remember, mean minimizes MSE!

where do we place centroid?

how do I define the center?

Meeting chat

EECS 398 Winter 2025 Remote Office Hours

Who can see your messages? Recording on

To: Waiting room participants

Type message here...

Minimizing inertia

- **Goal**: Find the centroids $\vec{\mu}_1, \vec{\mu}_2, \ldots, \vec{\mu}_k$ that minimize inertia:

$$I(\vec{\mu}_1, \vec{\mu}_2, \ldots, \vec{\mu}_k) = \text{total squared distance of each point } \vec{x}_i \text{ to its closest centroid } \vec{\mu}_j$$

- **Issue**: There is no efficient way to find the centroids that minimize inertia!

- There are k^n possible assignments of points to clusters; it would be computationally infeasible to try them all.

  It can be shown that finding the optimal centroid locations is NP-hard.

$$\underbrace{3 \times 3 \times 3 \times \cdots \times 3}_{10 \text{ points}} = 3^{10} \text{ possible colorings}$$

Minimizing inertia

- **Goal**: Find the centroids $\vec{\mu}_1, \vec{\mu}_2, \ldots, \vec{\mu}_k$ that minimize inertia:

$$I(\vec{\mu}_1, \vec{\mu}_2, \ldots, \vec{\mu}_k) = \text{total squared distance}$$
$$\text{of each point } \vec{x}_i$$
$$\text{to its closest centroid } \vec{\mu}_j$$

- **Issue**: There is no efficient way to find the centroids that minimize inertia!

- There are k^n possible assignments of points to clusters; it would be computationally infeasible to try them all.
  It can be shown that finding the optimal centroid locations is NP-hard.

- We can't use calculus to minimize I, either – we use calculus to minimize continuous functions, but the assignment of a point $\vec{x}_i$ to a centroid $\vec{\mu}_j$ is a discrete operation.

k-means clustering (i.e. Lloyd's algorithm)

- Fortunately, there's an efficient algorithm that (tries to) find the centroid locations that minimize inertia. The resulting clustering technique is called **k-means clustering**.
  Note that this has no relation to k-nearest neighbors, which we used for both regression and classification. Remember that clustering is an unsupervised technique!

0. **Randomly** initialize k centroids.
   There are other ways of initializing the centroids as well.

1. Assign each point to the nearest centroid.

color each point based on closest ☆

2. Move each centroid to the **center** of its group.
   We compute the center of a group by taking the mean of the group's coordinates.

move the ☆ to the center of its group.

3. **Repeat** steps 1 and 2 until the centroids stop changing!
   This is an iterative algorithm!

Why does k-means work?

- On each iteration, **inertia can only stay the same or decrease** – it cannot increase.

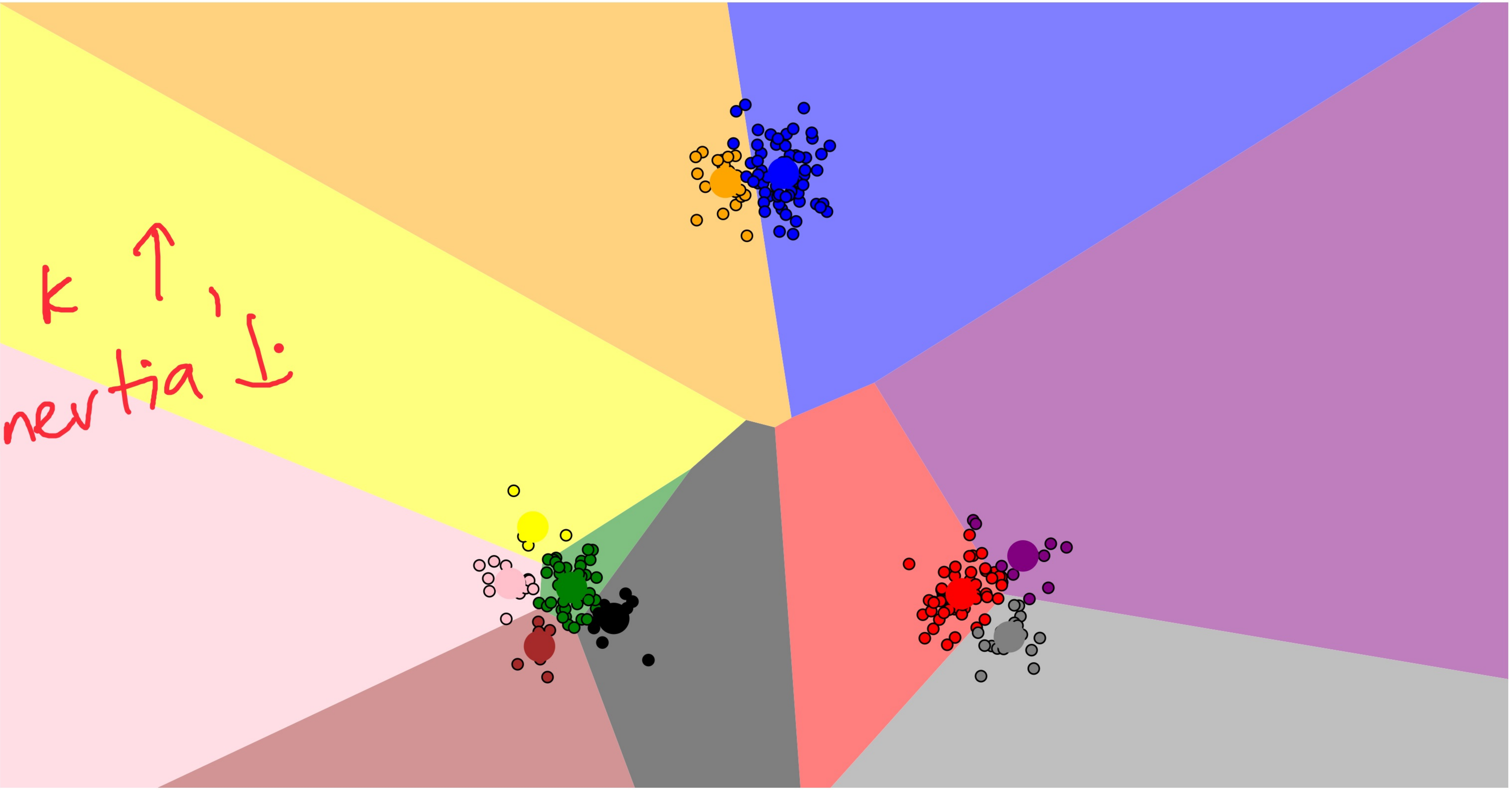
$$I(\vec{\mu}_1, \vec{\mu}_2, \ldots, \vec{\mu}_k) = \text{total squared distance of each point } \vec{x}_i \text{ to its closest centroid } \vec{\mu}_j$$

color move

- Why? Step 1 and step 2 **alternate** minimizing inertia in different ways:
  - In Step 1, we assign each point to the nearest centroid; this reduces the squared distance of each point to its closest centroid.
  - In Step 2, we move the centroids to the **center (mean position)** of their groups; this reduces the total **squared** distance from a centroid to the points assigned to it.

techniques work (and don't w... three fundamentally different approaches. This post, the fir... the k-means algorithm. To begin, click an initialization strategy below:

as $k \uparrow$, inertia $\downarrow$



Restart Reassign Points

K-Means Algorithm

The elbow method

- For several different values of k, let's compute the inertia of the resulting clustering, using the scatter plot from the previous slide.

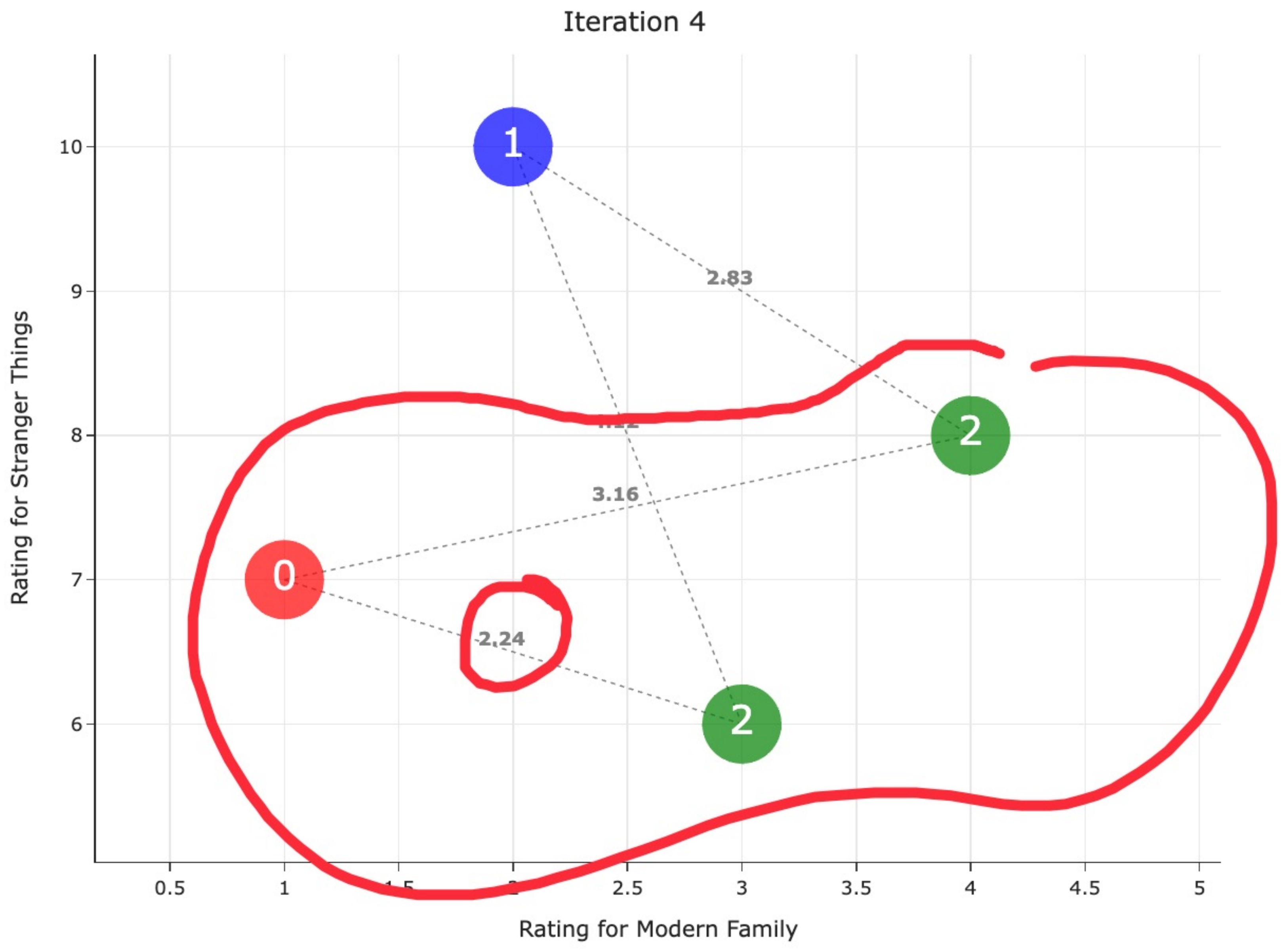
In [15]:
```
util.show_elbow()
```

- The **elbow method** says to choose the k that appears at the elbow of the plot of inertia vs. k, since there are diminishing returns for using more than k clusters.

Above, we see an elbow at $k = 3$, which gives us the k that matches our natural intuition in this example.

In [27]:

```
util.color_ratings(title='Iteration 4', show_distances=[(0, 2), (1, 2)], labels=[0, 1, 2, 2, 5, 5, 5, 7, 7])
```



min dist (2, 0) = 2.24

min dist (2, 1) = 2.83