Carnegie Mellon University
Heinz College

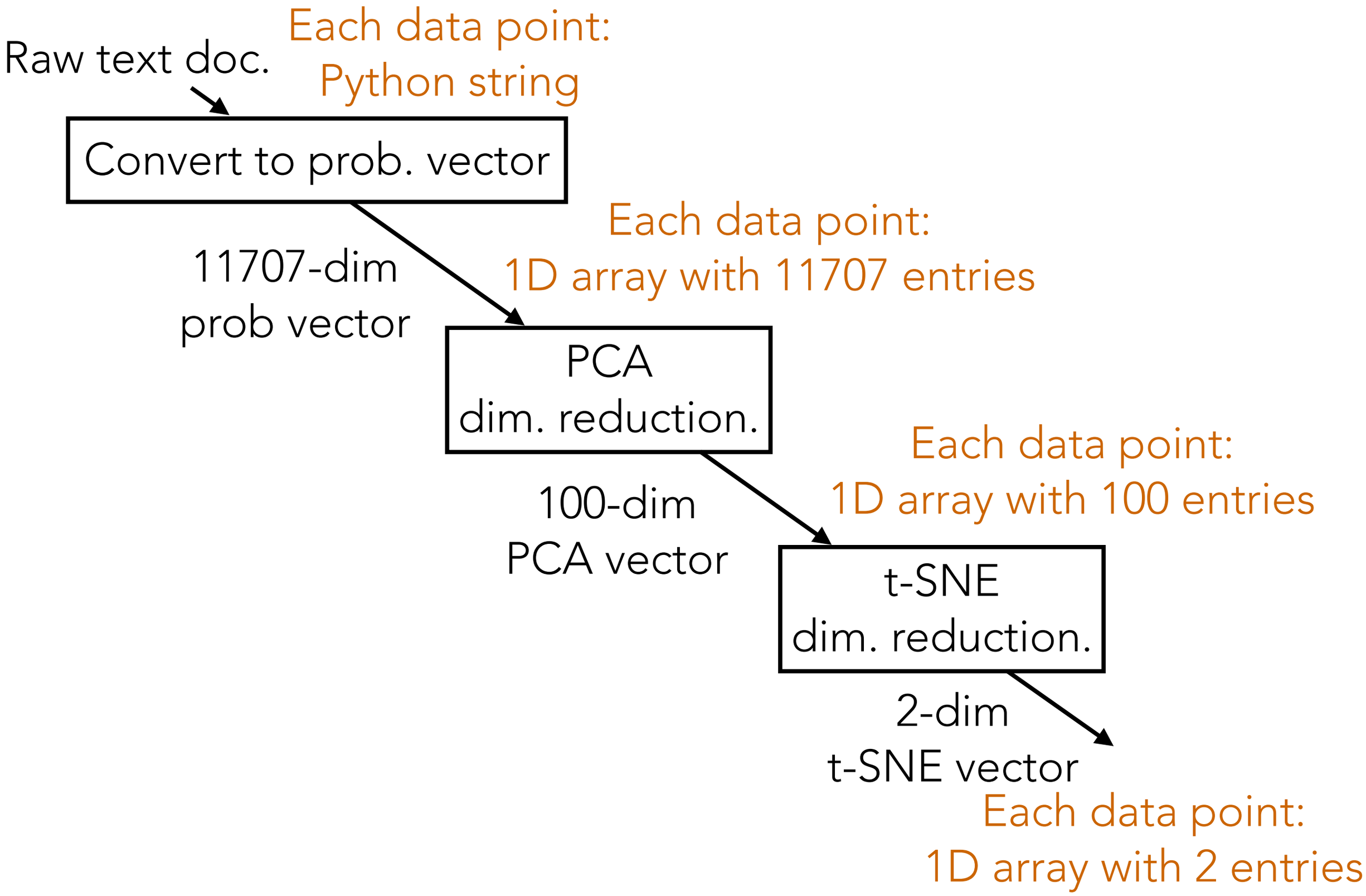
94-775 Unstructured Data Analytics for Policy

Lecture 8: Clustering (cont'd)

Slides by George H. Chen

(Flashback) Recap of Start of Clustering on Text Demo

- We're clustering on the 20 Newsgroups dataset (preprocessed by lemmatizing every token)
- We filtered out some documents that are likely not in English
- We filtered out vocab words that showed up in too many or too few documents
    - Resulting 2D table of feature vectors: `filtered_tf`
    - Resulting 1D table of vocabulary words: `filtered_vocab`
- After filtering, text documents still varied wildly in length
    - Convert each row of `filtered_tf` into a probability vector
        - Resulting 2D table of probability vectors: `prob_vectors`


Raw text doc.
Each data point:
Python string
Convert to prob. vector
11707-dim
prob vector
Each data point:
1D array with 11707 entries
PCA
dim. reduction.
100-dim
PCA vector
Each data point:
1D array with 100 entries
t-SNE
dim. reduction.
2-dim
t-SNE vector
Each data point:
1D array with 2 entries

"learn"

Convert to prob. vector

11707-dim prob vector

[0, …, 0, 1, 0, …, 0]

at index of vocabulary word “learn” (which happens to be index 6269)

"car"

Convert to prob. vector

11707-dim prob vector

[0, …, 0, 1, 0, …, 0]

index 2134

"study"

Convert to prob. vector

11707-dim prob vector

[0, …, 0, 1, 0, …, 0]

index 10117

Intuitively, "learn" & "study" should be more semantically similar so we would like them to be closer to each other compared to "learn" & "car"

What happens when we compute these pairwise distances in the 11707-dim prob vector space?

"learn"

Convert to prob. vector

11707-dim
prob vector

PCA
dim. reduction.

100-dim
PCA vector

"car"

Convert to prob. vector

11707-dim
prob vector

PCA
dim. reduction.

100-dim
PCA vector

"study"

Convert to prob. vector

11707-dim
prob vector

PCA
dim. reduction.

100-dim
PCA vector

Let's see what happens when we compute pairwise distances using the 100-dim PCA vectors instead

Clustering on Text

Resuming the demo…

Automatically Choosing the Number of Clusters *k*

For *k* = 2, 3, … up to some user-specified max value:

Fit model (*k*-means) using *k*

Compute a score for the model

But what score function should we use?

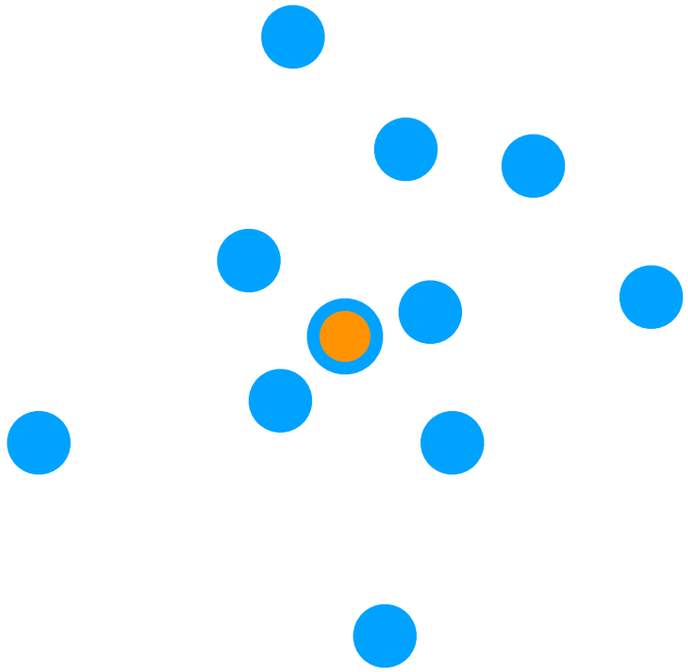
Use whichever *k* has the best score

No single way of choosing *k* is the "best" way

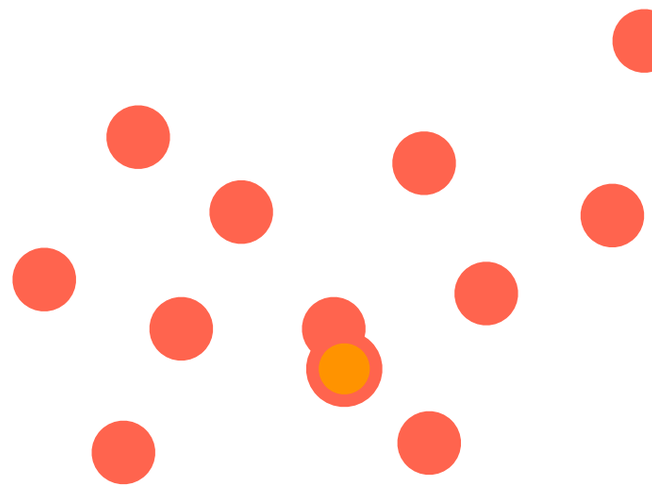
Here’s an example of a score function

Residual Sum of Squares

Look at one cluster at a time



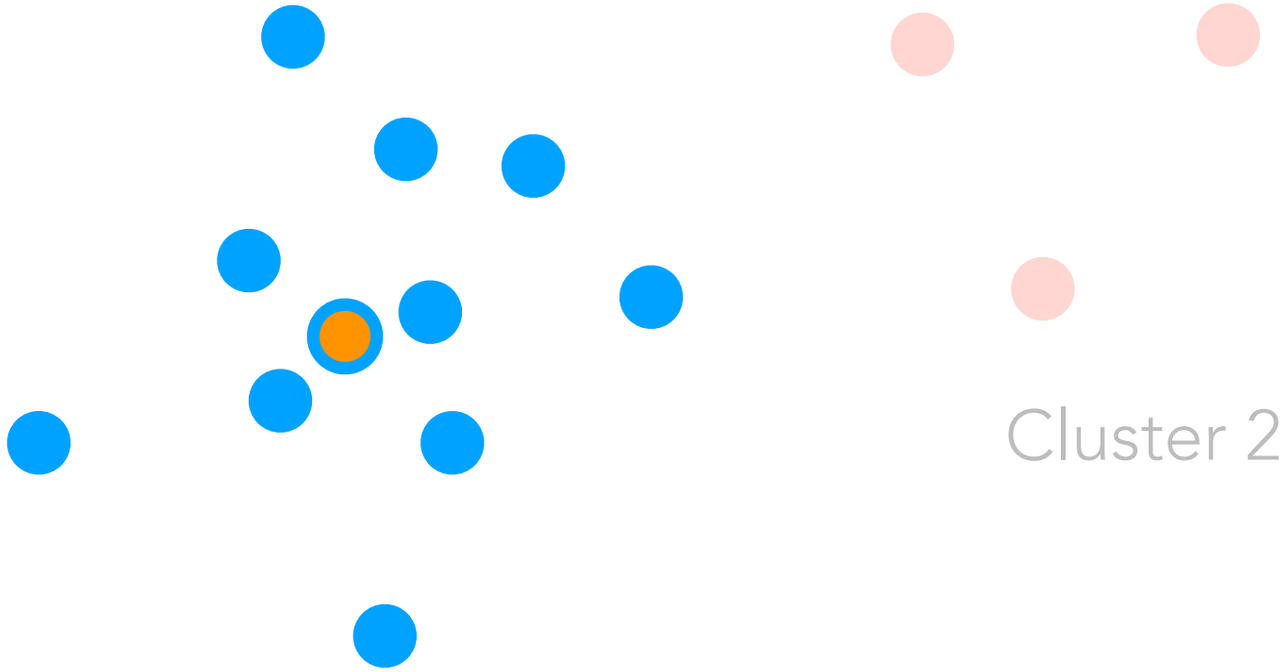
Cluster 1



Cluster 2

Residual Sum of Squares

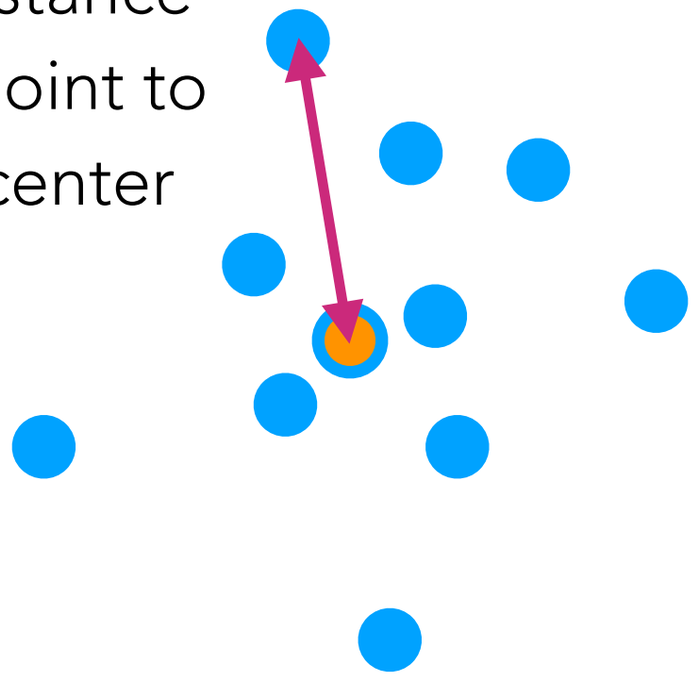
Look at one cluster at a time



Residual Sum of Squares

Look at one cluster at a time

Measure distance from each point to its cluster center



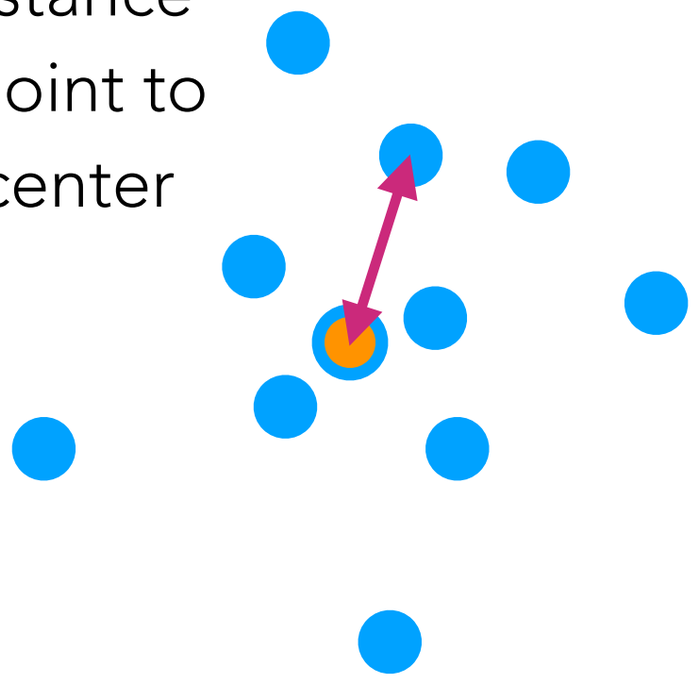
Cluster 1

Cluster 2

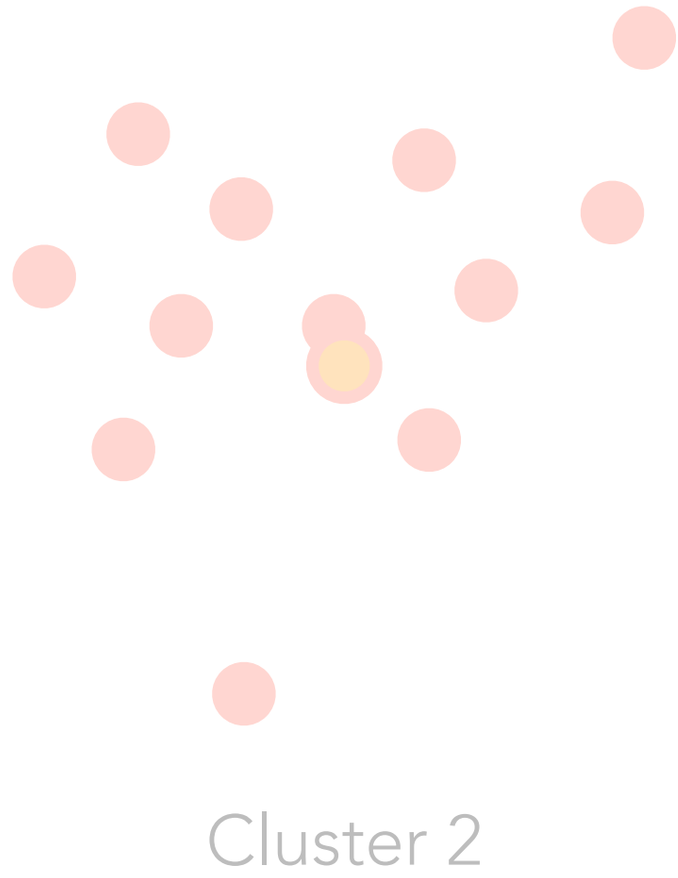
Residual Sum of Squares

Look at one cluster at a time

Measure distance from each point to its cluster center



Cluster 1

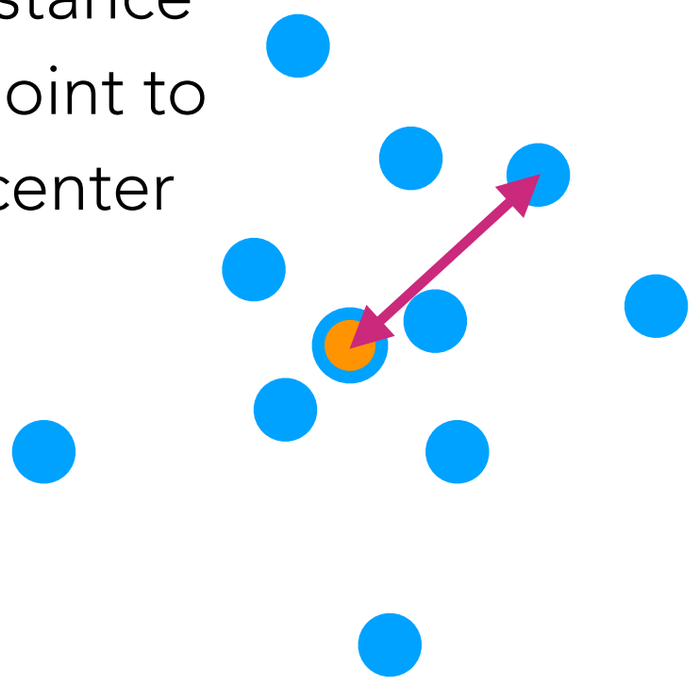


Cluster 2

Residual Sum of Squares

Look at one cluster at a time

Measure distance from each point to its cluster center



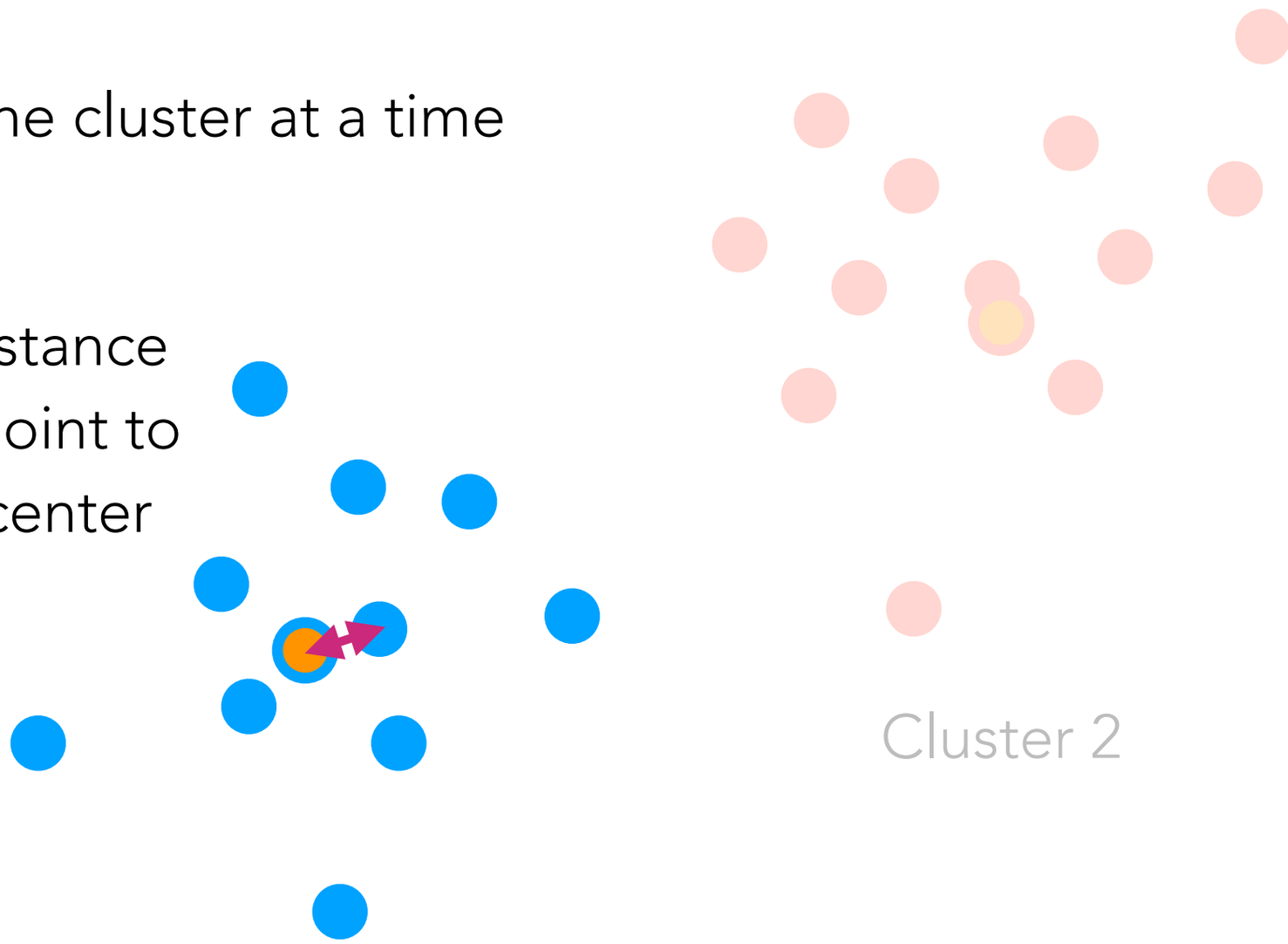
Cluster 1

Cluster 2

Residual Sum of Squares

Look at one cluster at a time

Measure distance from each point to its cluster center



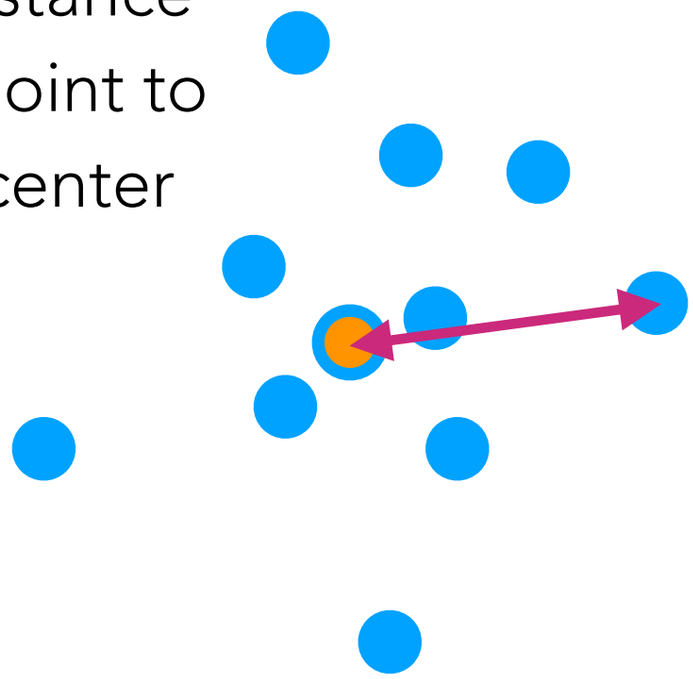
Cluster 1

Cluster 2

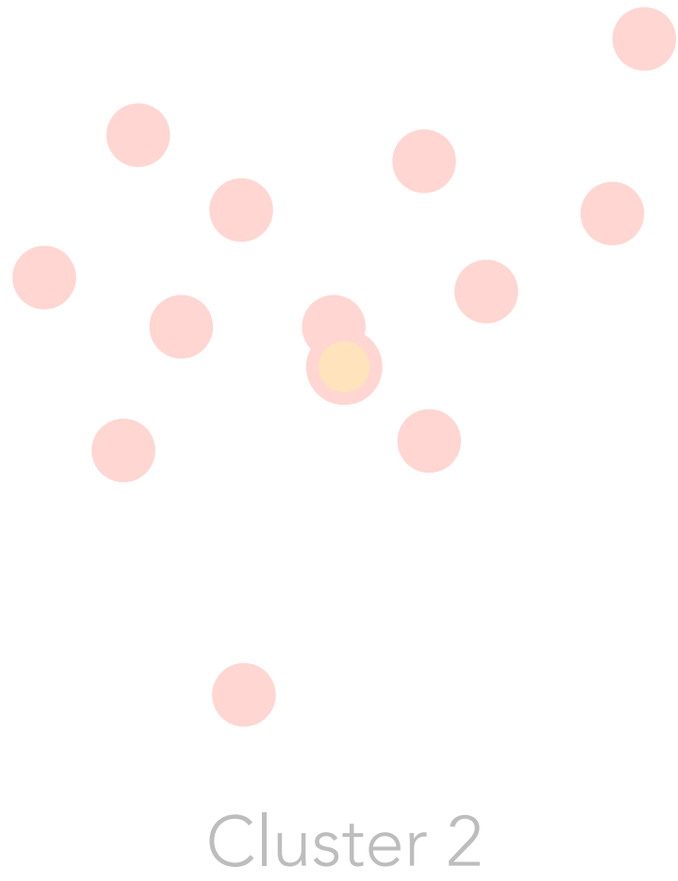
Residual Sum of Squares

Look at one cluster at a time

Measure distance from each point to its cluster center



Cluster 1

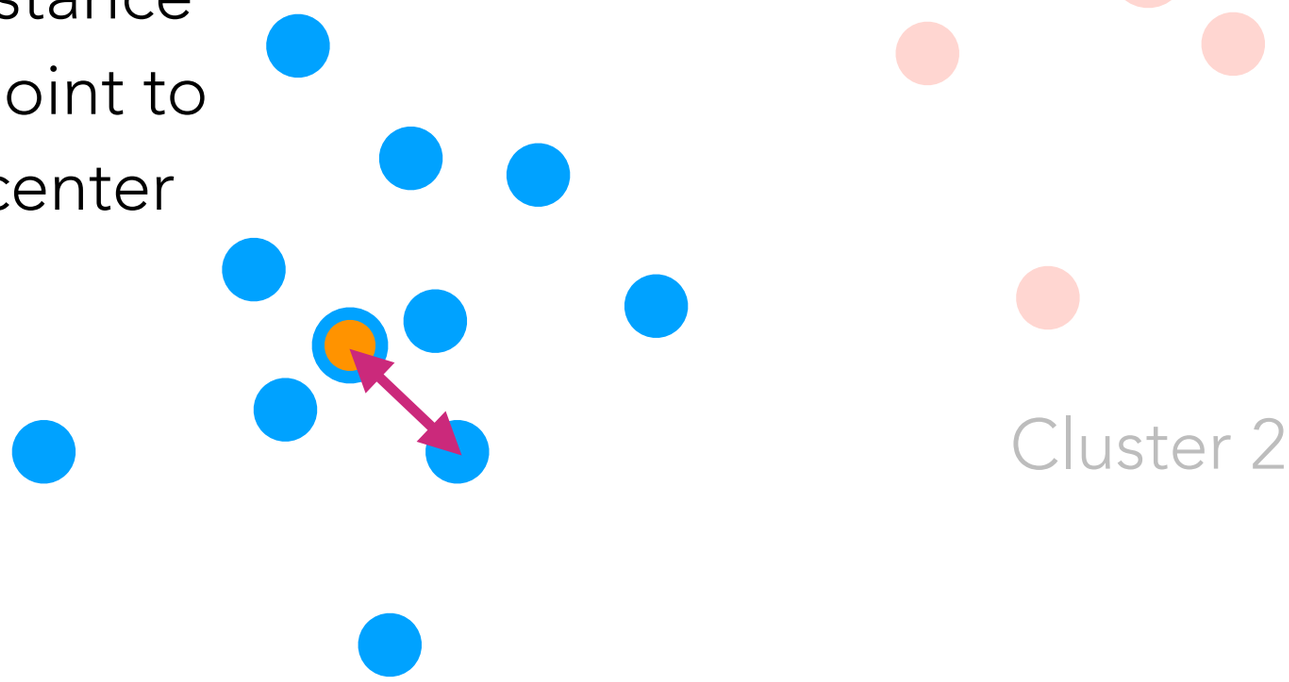


Cluster 2

Residual Sum of Squares

Look at one cluster at a time

Measure distance from each point to its cluster center



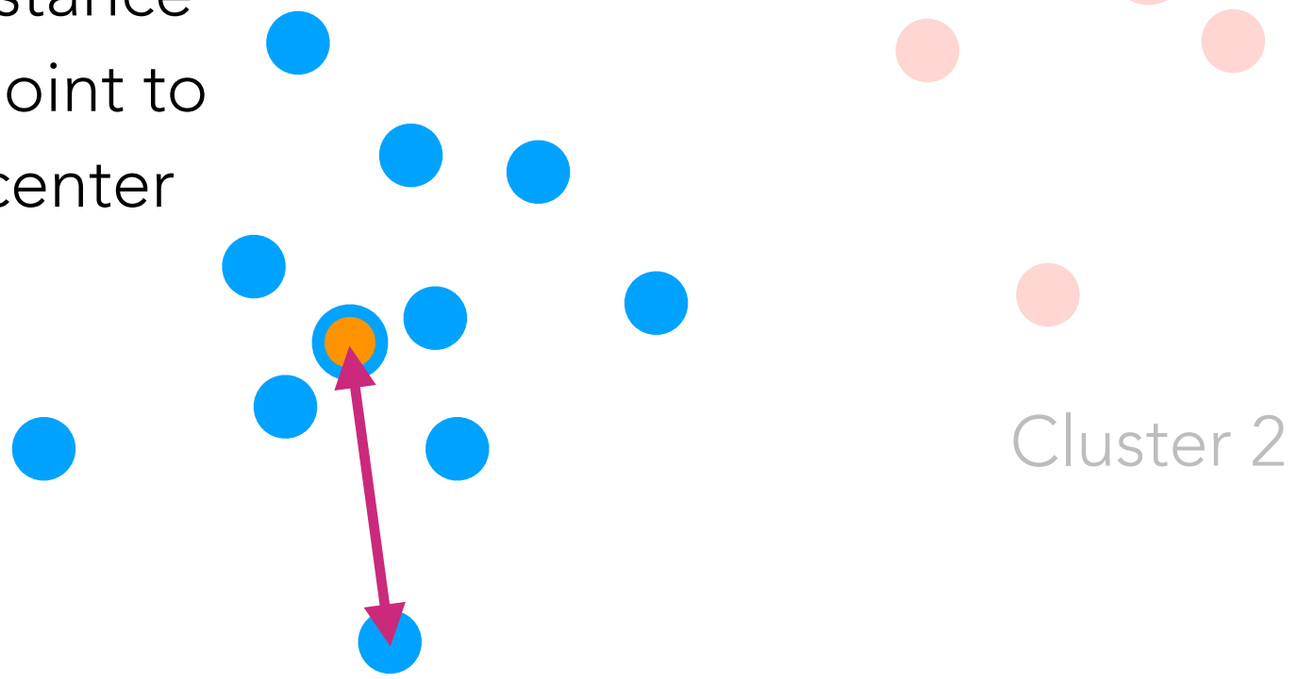
Cluster 1

Cluster 2

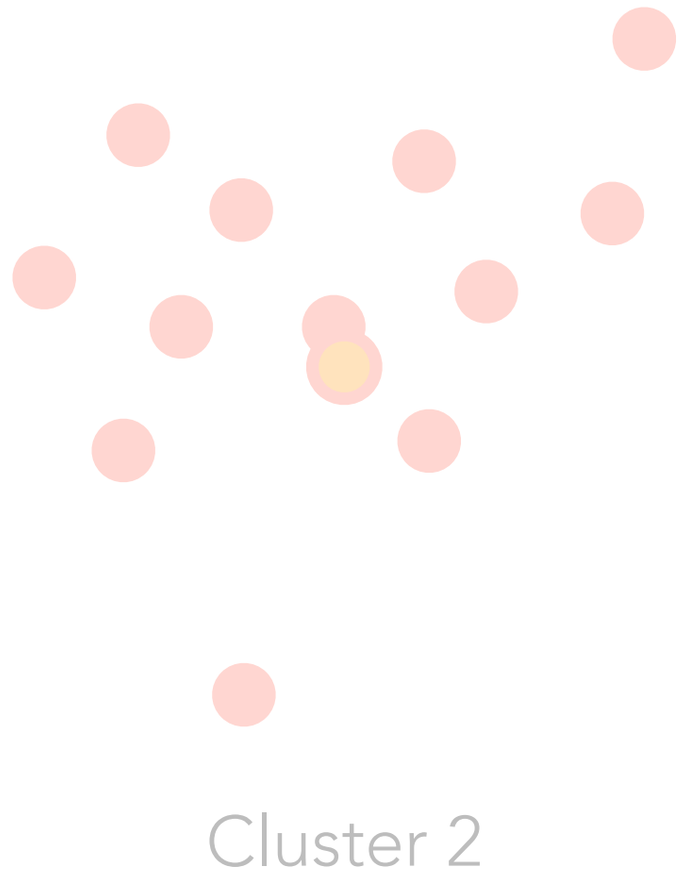
Residual Sum of Squares

Look at one cluster at a time

Measure distance from each point to its cluster center



Cluster 1

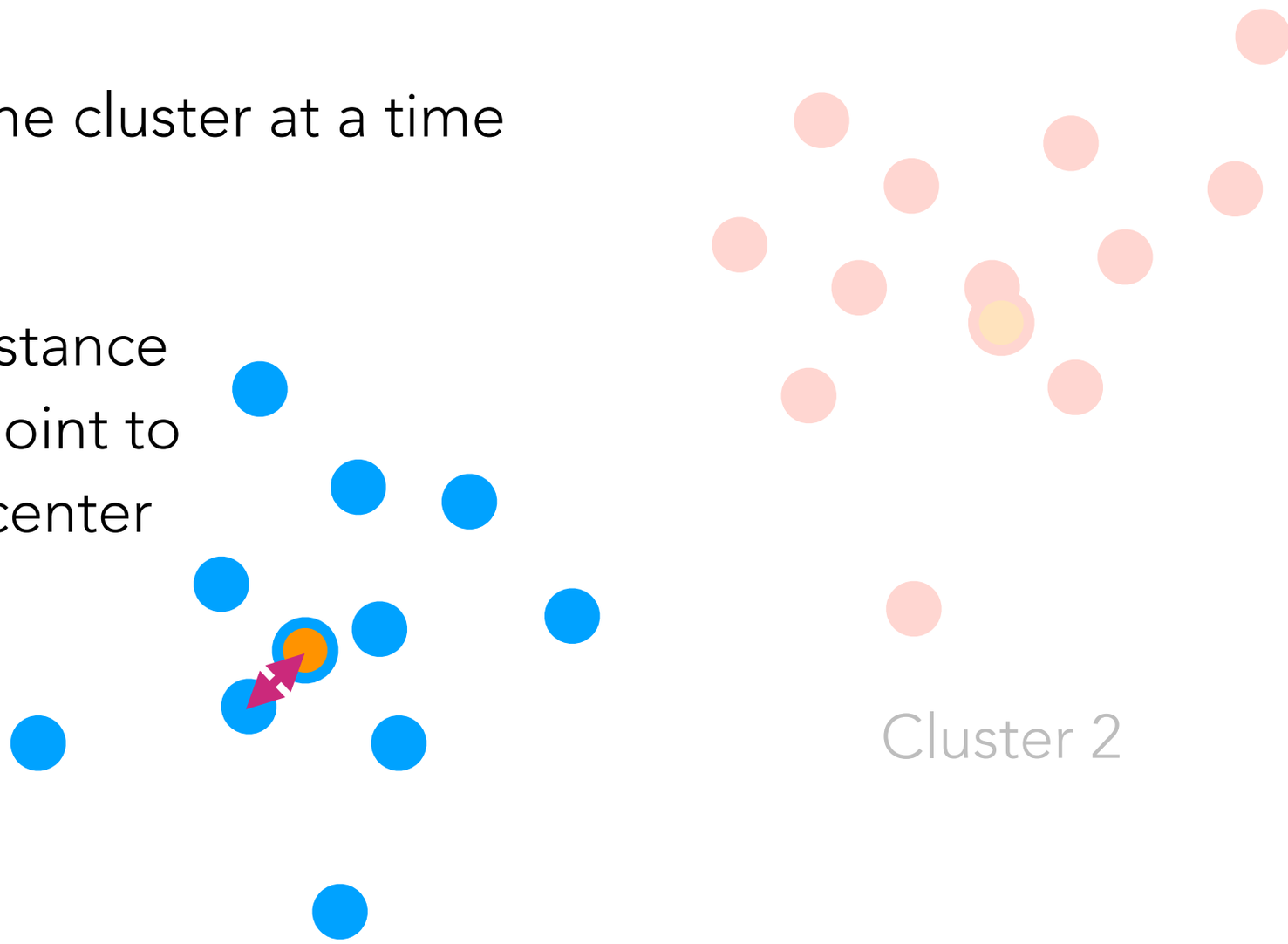


Cluster 2

Residual Sum of Squares

Look at one cluster at a time

Measure distance from each point to its cluster center



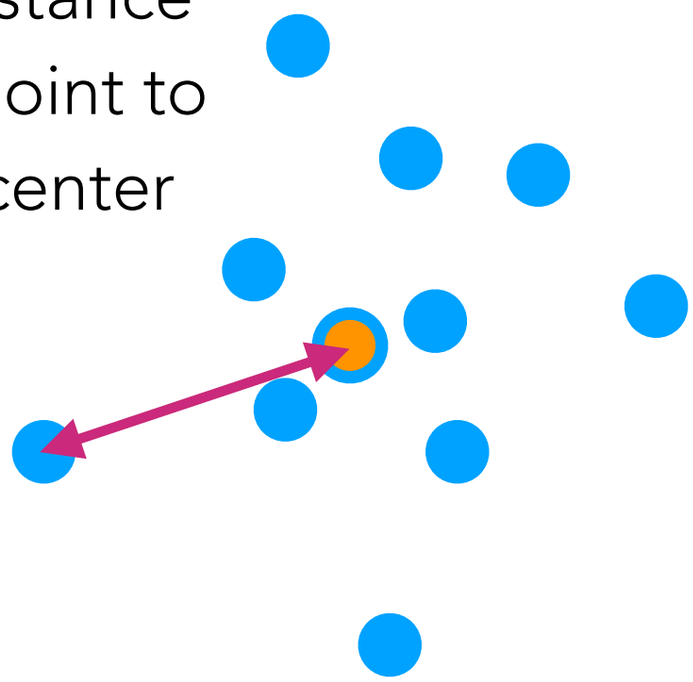
Cluster 1

Cluster 2

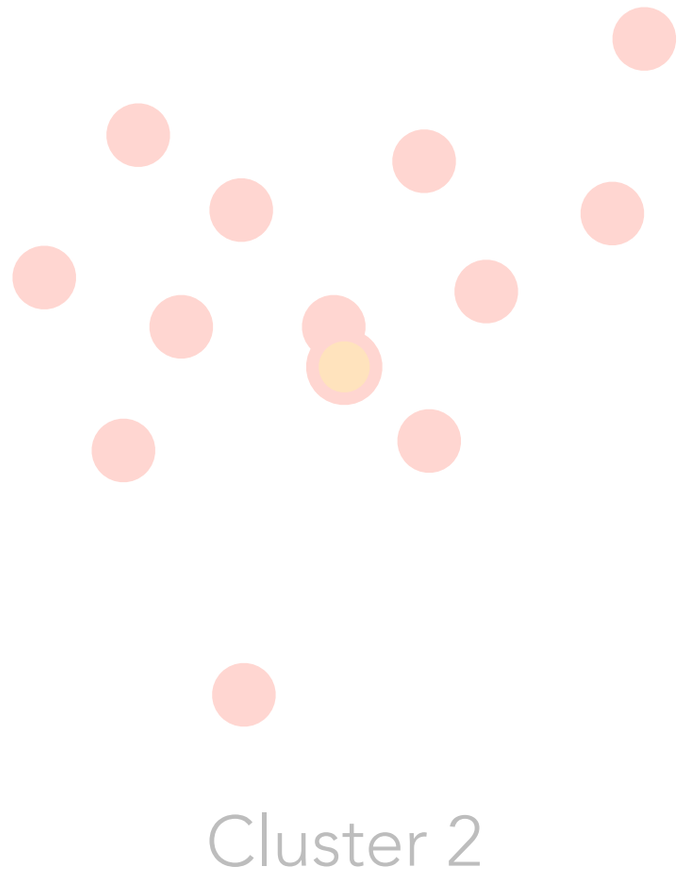
Residual Sum of Squares

Look at one cluster at a time

Measure distance from each point to its cluster center



Cluster 1

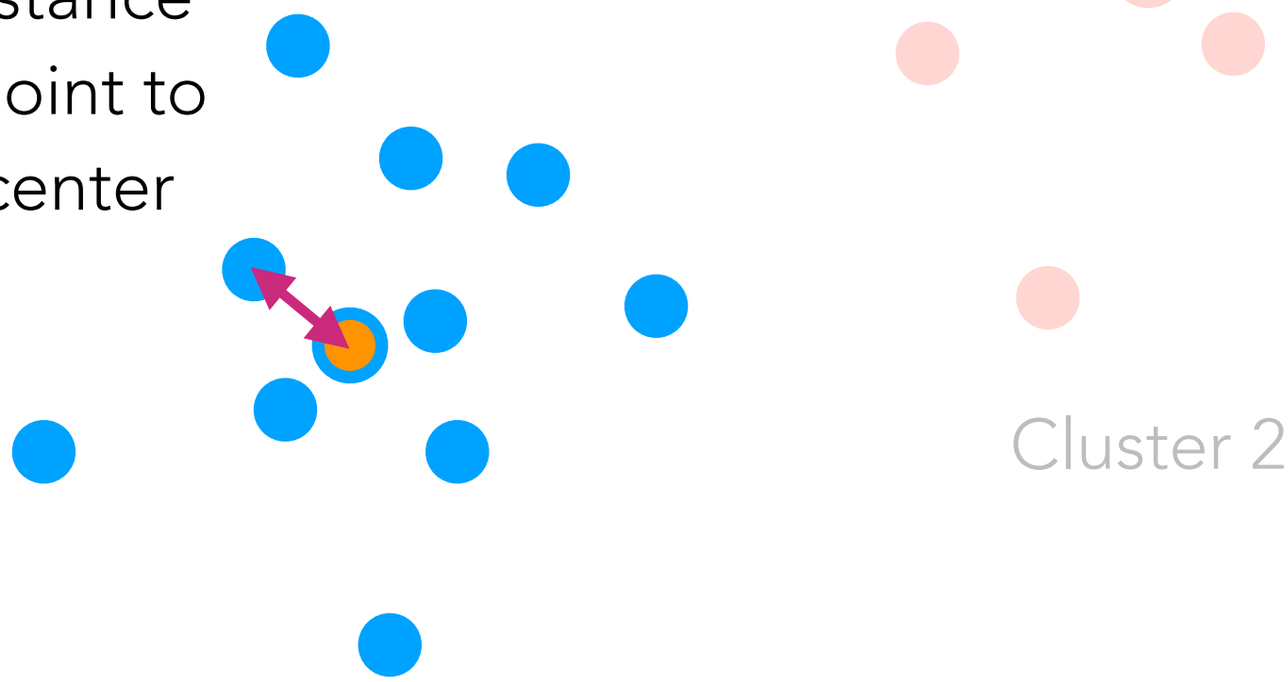


Cluster 2

Residual Sum of Squares

Look at one cluster at a time

Measure distance from each point to its cluster center



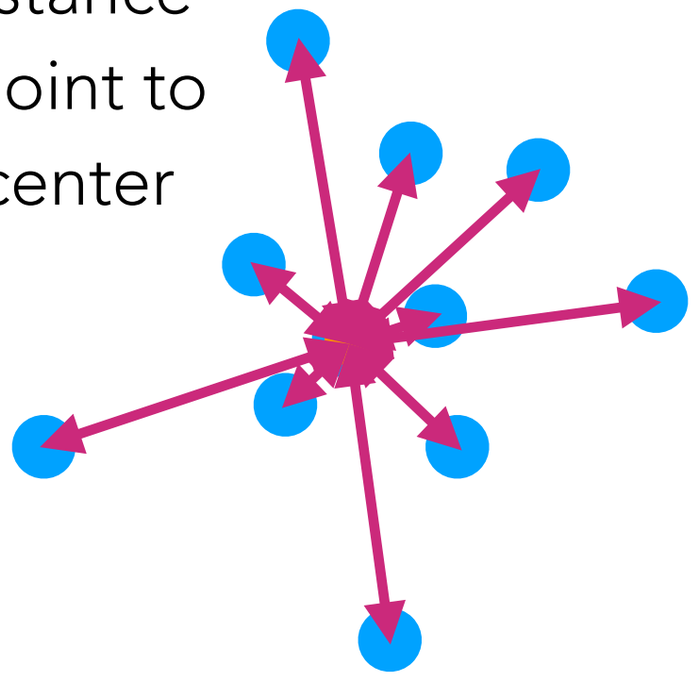
Cluster 1

Cluster 2

Residual Sum of Squares

Look at one cluster at a time

Measure distance from each point to its cluster center



Cluster 1

Cluster 2

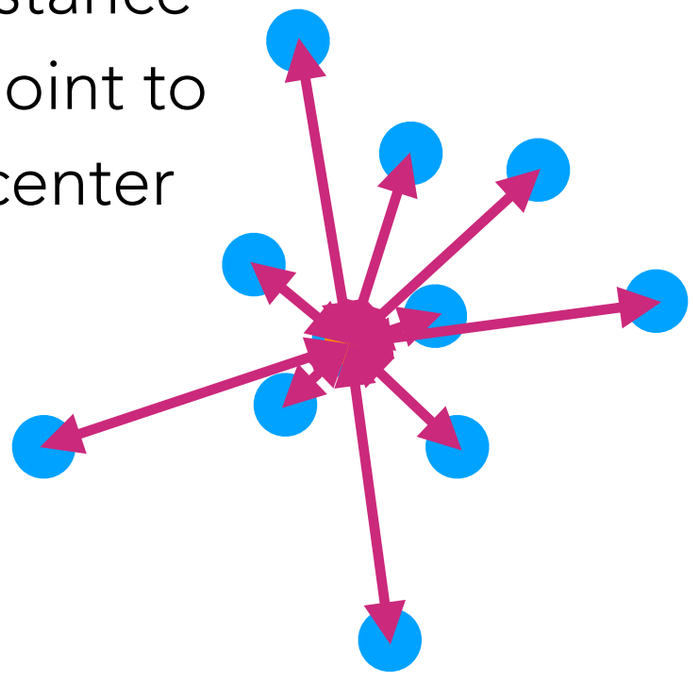
Residual sum of squares for cluster 1:

sum of *squared* purple lengths

Residual Sum of Squares

Look at one cluster at a time

Measure distance from each point to its cluster center



Cluster 1

Cluster 2

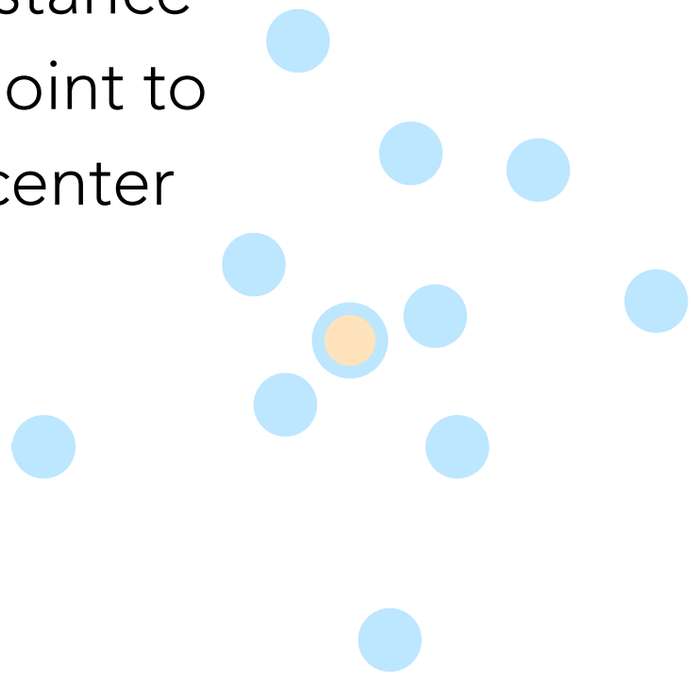
Residual sum of squares for cluster 1:

$$\mathrm{RSS}_1 = \sum_{x \in \text{cluster } 1} \|x - \mu_1\|^2$$

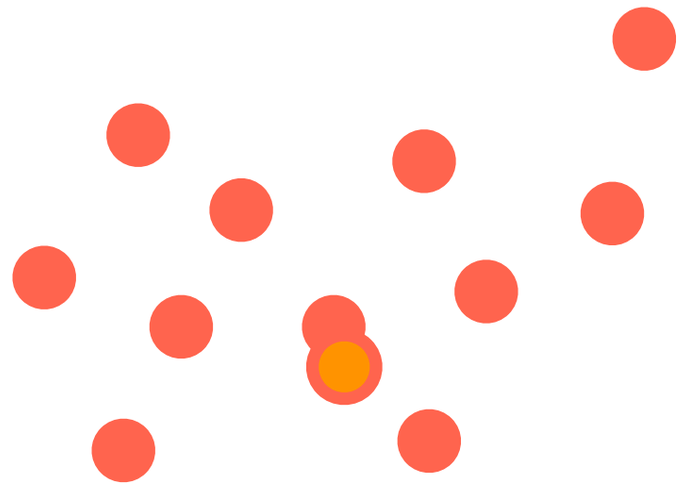
Residual Sum of Squares

Look at one cluster at a time

Measure distance from each point to its cluster center



Cluster 1



Repeat similar calculation for other cluster

Cluster 2

Residual sum of squares for cluster 2:

$$\mathrm{RSS}_2 = \sum_{x \in \text{cluster } 2} \|x - \mu_2\|^2$$

Residual Sum of Squares

$$\mathrm{RSS} = \mathrm{RSS}_1 + \mathrm{RSS}_2 = \sum_{x \in \text{cluster } 1} \|x - \mu_1\|^2 + \sum_{x \in \text{cluster } 2} \|x - \mu_2\|^2$$

In general if there are *k* clusters:

$$\mathrm{RSS} = \sum_{g=1}^{k} \mathrm{RSS}_g = \sum_{g=1}^{k} \sum_{x \in \text{cluster } g} \|x - \mu_g\|^2$$

Remark: *k*-means *tries* to minimize RSS for a fixed value of *k*
(it does so *approximately*, with no guarantee of optimality)

RSS does not account for clusters having, for instance, ellipse shapes

Why is minimizing RSS a bad way to choose *k*?

What happens when *k* is equal to the number of data points?

Example of a Different Way to Choose *k*

RSS measures *within-cluster variation*

$$W = \mathrm{RSS} = \sum_{g=1}^{k} \mathrm{RSS}_g = \sum_{g=1}^{k} \sum_{x \in \text{cluster } g} \|x - \mu_g\|^2$$

Want to also measure *between-cluster variation*

$$B = \sum_{g=1}^{k} (\#\text{ points in cluster } g)\|\mu_g - \bar{\mu}\|^2$$

mean of *all* points

Called the CH index
[Calinski and Harabasz 1974]

A score function to use for choosing *k*:

$$\mathrm{CH}(k) = \frac{B \cdot (n-k)}{W \cdot (k-1)}$$

n = total # points

Pick k with highest $\mathrm{CH}(k)$

(Choose k among 2, 3, … up to pre-specified max)

Another Example of Clustering Metric: Silhouette Score

For a single data point, calculate:

$$s = \frac{b - a}{\max\{a, b\}}$$

$a =$ average distance to points in the **same cluster**

$b =$ average distance to points in the **nearest other cluster**

s takes on a value from -1 (bad) to 1 (good)

Compute this value for every data point and then take the average to get the clustering result's **overall silhouette score**

Silhouette Score: Example of Calculation for a Single Point

$$s = \frac{b - a}{\max\{a, b\}}$$

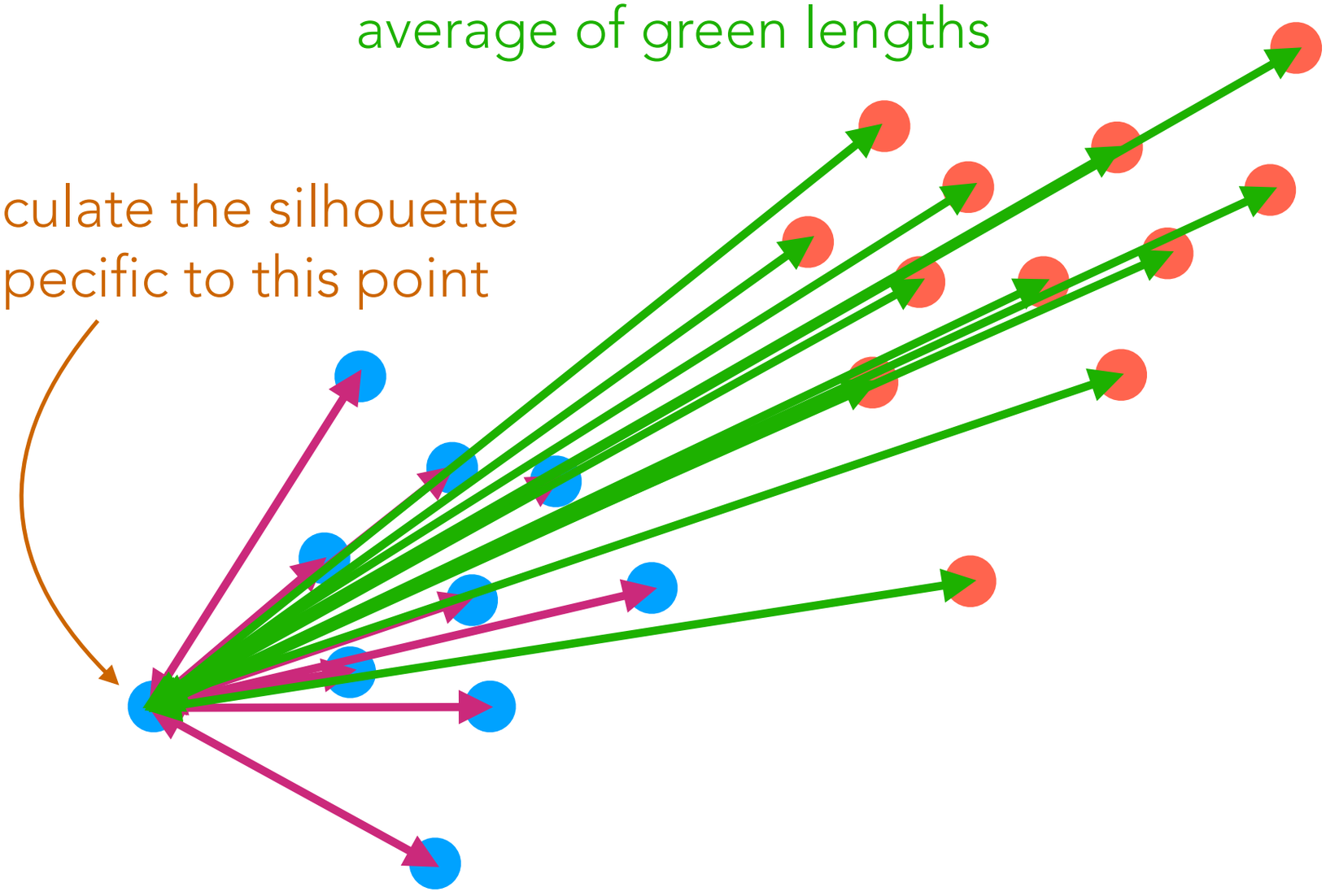
average of purple lengths

$a =$ average distance to points in the **same cluster**

$b =$ average distance to points in the **nearest other cluster**

average of green lengths

Let's calculate the silhouette score specific to this point



Automatically Choosing the Number of Clusters *k*

Demo