LECTURE 25

Clustering

Using clustering algorithms to identify patterns in our data.

Overview

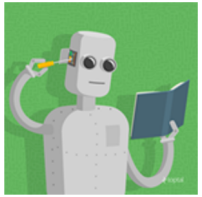
We've completed the core narrative arc of our class

- Sampling
- Problem: **Tabular Data Manipulation**
  - Solutions: ***Pandas***
- Problem: **Regression and Classification**
  - Solution:
    - ***Linear Models, Decision Trees, Boosting, SVM***
- Problem: **Dimensionality Reduction**
  - Solution: ***PCA***

Machine Learning

It turns out that these last two problems are examples of **Machine Learning** algorithms that "learn" from data

Review: Taxonomy of Machine Learning



Labeled Data

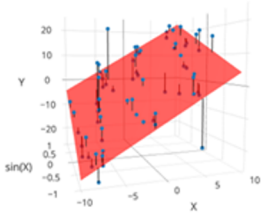
Supervised Learning

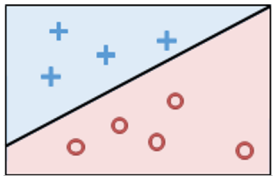
Quantitative Response

Categorical Response

Regression

Classification





In “Supervised Learning”:

- Goal is to create a function that maps inputs to outputs
- Model is learned from example input/output pairs. Each pair consists of:
  - Input vector
  - Output value (label)
- Regression: Output value is quantitative
- Classification: Output value is categorical

Review: Taxonomy of Machine Learning

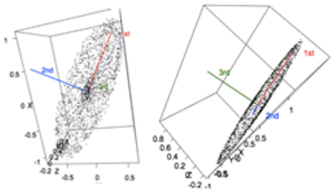
In “Unsupervised Learning”:

- Goal is to identify patterns in **unlabeled** data
  - We do not have input/output pairs

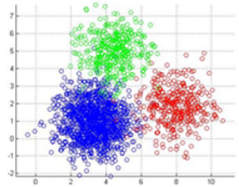
Unlabeled Data

Unsupervised Learning

Dimensionality Reduction



Clustering



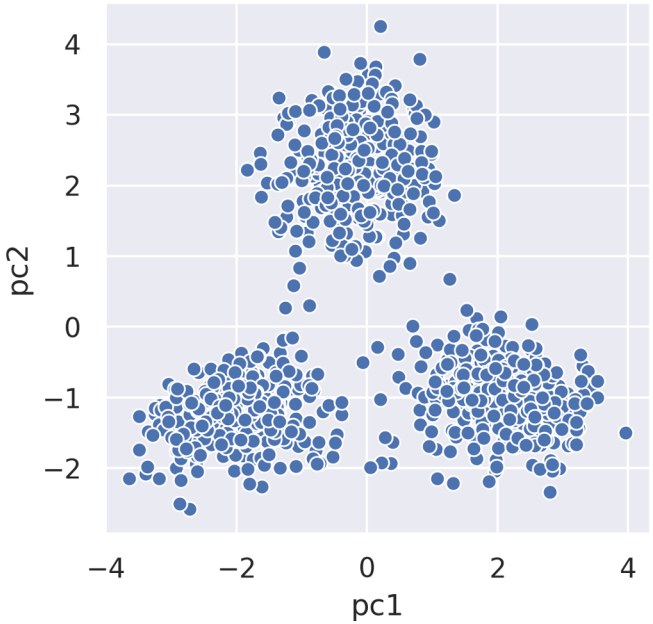
Problem: Dimensionality Reduction

- Solution: PCA

Problem: Clustering

Clustering Example

Consider the figure which represents the 1st and 2nd PC of how much time patrons spent at 8 different zoo exhibits



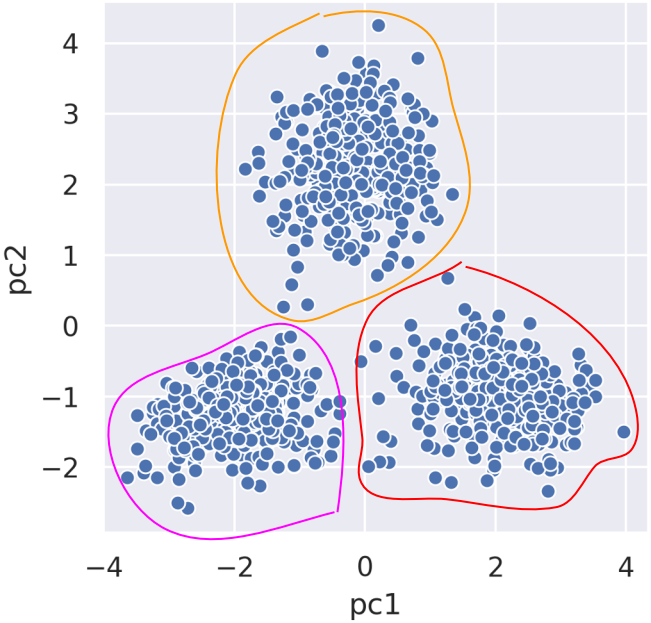
Goal of clustering: Assign each point to a cluster

This is an unsupervised task

- We don’t have labels for each visitor
- Want to infer pattern even without labels

Clustering Example

Consider the figure which represents the 1st and 2nd PC of how much time patrons spent at 8 different zoo exhibits



Goal of clustering: Assign each point to a cluster

This is an unsupervised task

- We don’t have labels for each visitor
- Want to infer pattern even without labels

Clustering Example 1: Netflix

Suppose you're Netflix and have information on customer viewing habits

- Can use clustering to assign each person or show to a "cluster"
- Don't have to define clusters in advance

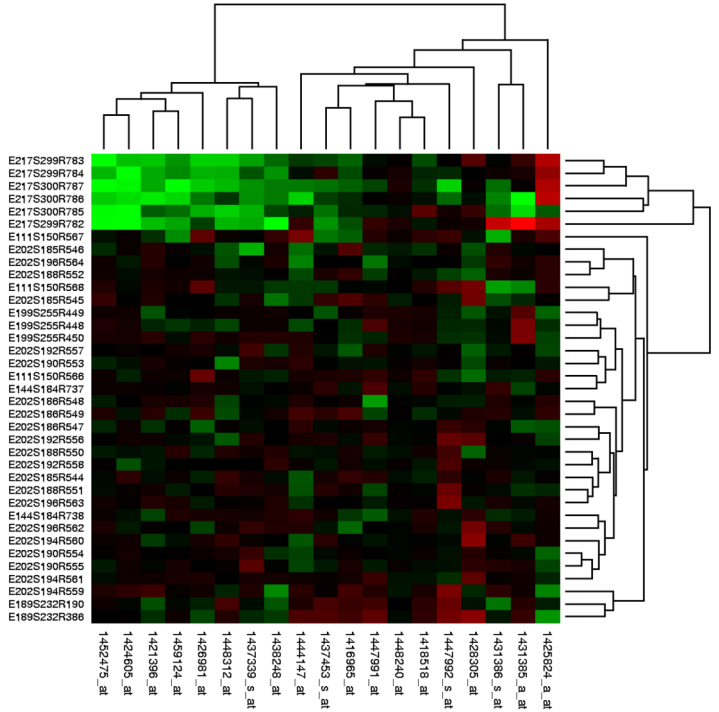
Clustering is different from classification

- With classification, you have to decide on labels in advance
- Clustering discovers groups automatically

Clustering Example 2: Reverse Engineering Biology

In plot to the right:

- Rows are conditions (e.g. a row might be: “poured acid on the cells”)
- Columns are genes



Green indicates that the gene was ~off

- The ~9 genes on the left all got turned off by the 6 experiments at the top

Clustering brings similar observations together

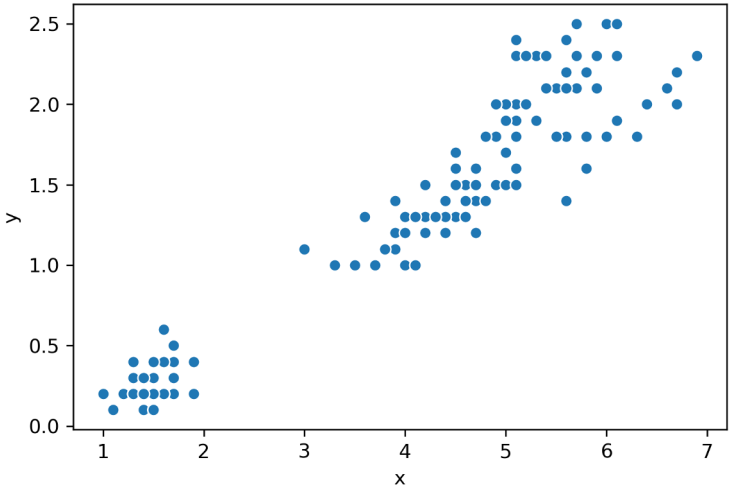
K-Means Clustering Algorithm

K-Means Clustering

Most popular clustering approach: K-Means

- Pick an arbitrary k, and randomly place k “centers”, each a different color
- Repeat until convergence:
  - Color points according to the closest center
  - Move center for each color to center of points with that color

Raw data →

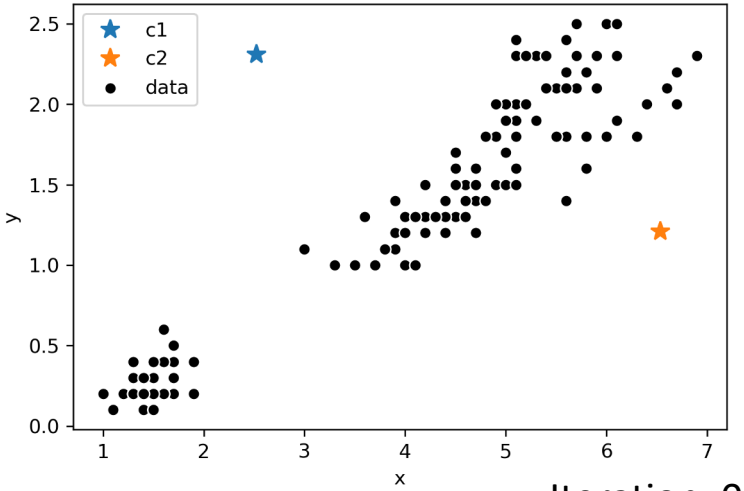


K-Means Clustering

Most popular clustering approach: K-Means.

- Pick an arbitrary k, and **randomly place k "centers",** each a different color
- Repeat until convergence:
  - Color points according to the closest center
  - Move center for each color to center of points with that color

Initial random placement of two centers



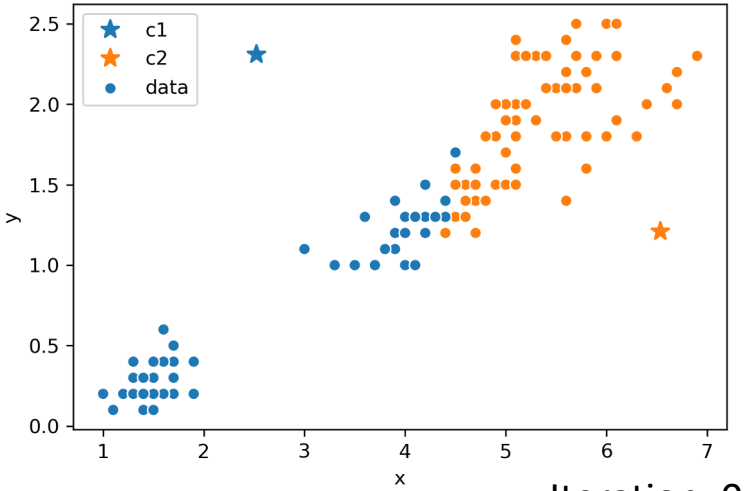
Iteration: 0

K-Means Clustering

Most popular clustering approach: K-Means

- Pick an arbitrary k, and randomly place k “centers”, each a different color
- Repeat until convergence:
  - **Color points according to the closest center**
  - Move center for each color to center of points with that color

Data colored by closest center



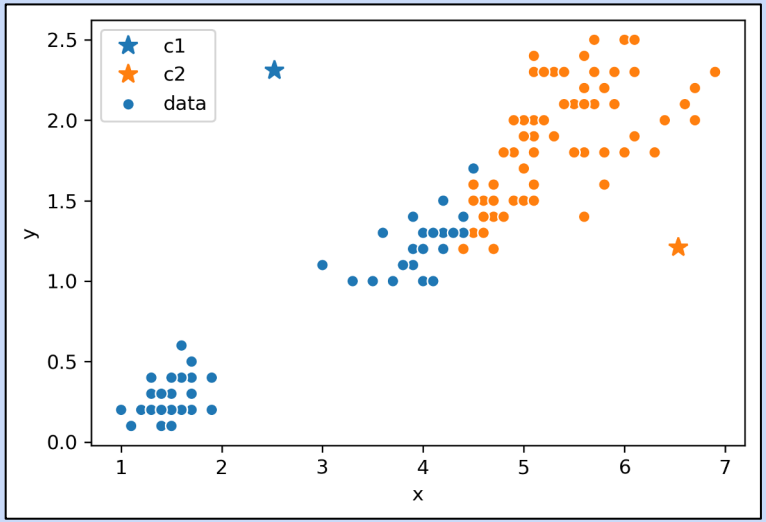
Iteration: 0

K-Means Clustering

Most popular clustering approach: K-Means

- Pick an arbitrary k, and randomly place k “centers”, each a different color
- Repeat until convergence:
  - Color points according to the closest center
  - **Move center for each color to center of points with that color**

Where should the centers go next?

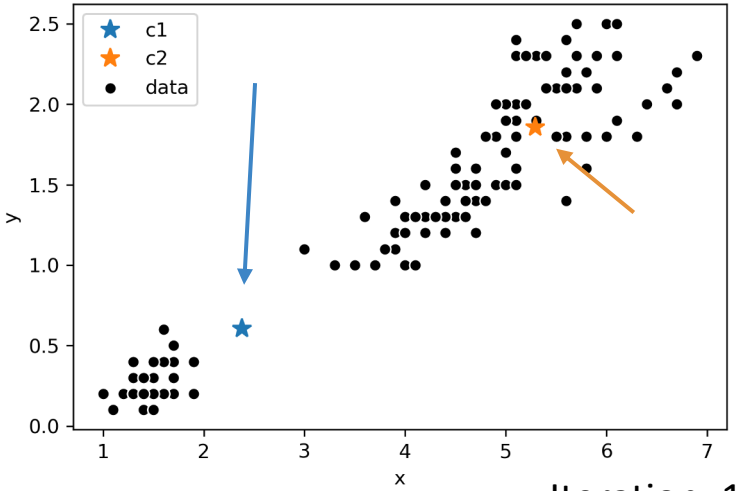


K-Means Clustering

Most popular clustering approach: K-Means

- Pick an arbitrary k, and randomly place k “centers”, each a different color
- Repeat until convergence:
  - Color points according to the closest center
  - **Move center for each color to center of points with that color**

Centers moved to their new homes



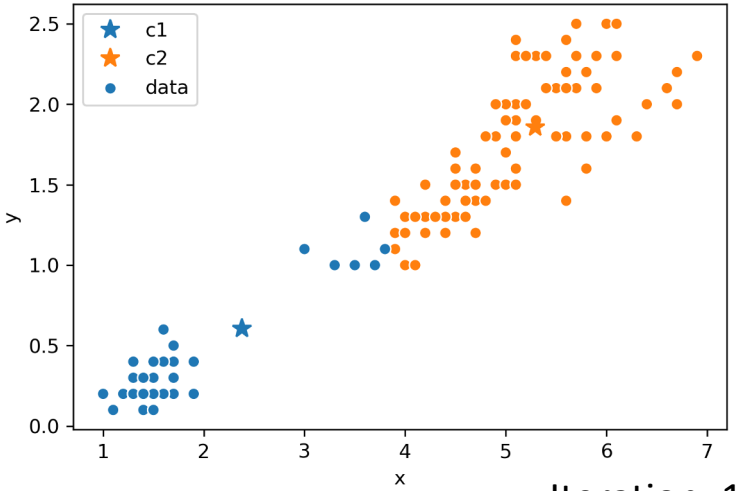
Iteration: 1

K-Means Clustering

Most popular clustering approach: K-Means

- Pick an arbitrary k, and randomly place k “centers”, each a different color
- Repeat until convergence:
  - **Color points according to the closest center**
  - Move center for each color to center of points with that color

Data colored by closest center (in new position)



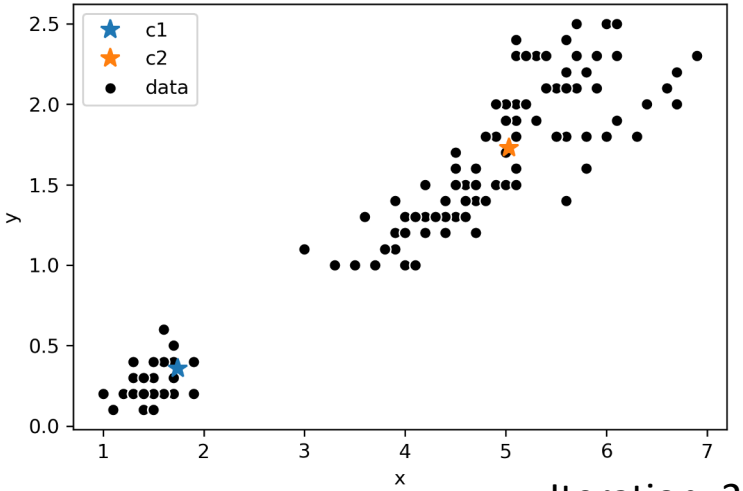
Iteration: 1

K-Means Clustering

Most popular clustering approach: K-Means

- Pick an arbitrary k, and randomly place k “centers”, each a different color
- Repeat until convergence:
  - Color points according to the closest center
  - **Move center for each color to center of points with that color**

Centers moved to new position



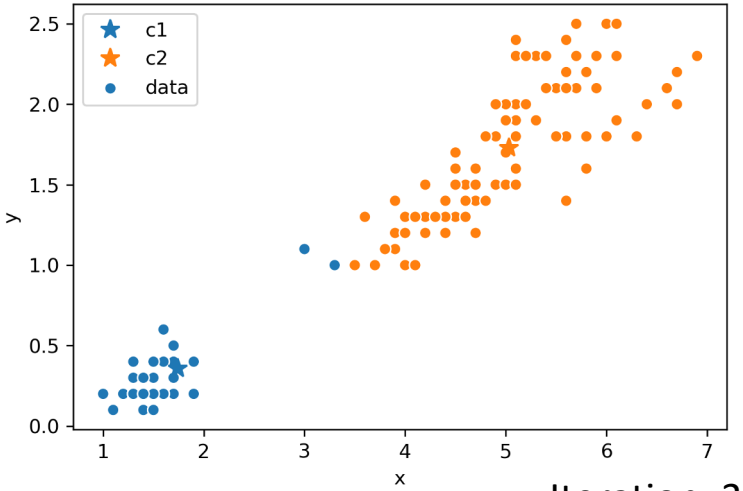
Iteration: 2

K-Means Clustering

Most popular clustering approach: K-Means.

- Pick an arbitrary k, and randomly place k “centers”, each a different color.
- Repeat until convergence:
  - **Color points according to the closest center.**
  - Move center for each color to center of points with that color

Data colored by closest center (in new position)



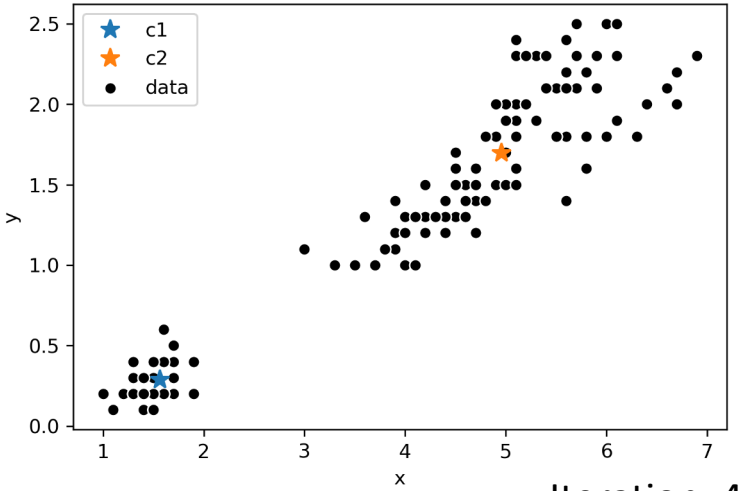
Iteration: 3

K-Means Clustering

Most popular clustering approach: K-Means

- Pick an arbitrary k, and randomly place k “centers”, each a different color
- Repeat until convergence:
  - Color points according to the closest center
  - **Move center for each color to center of points with that color**

Centers moved to new position



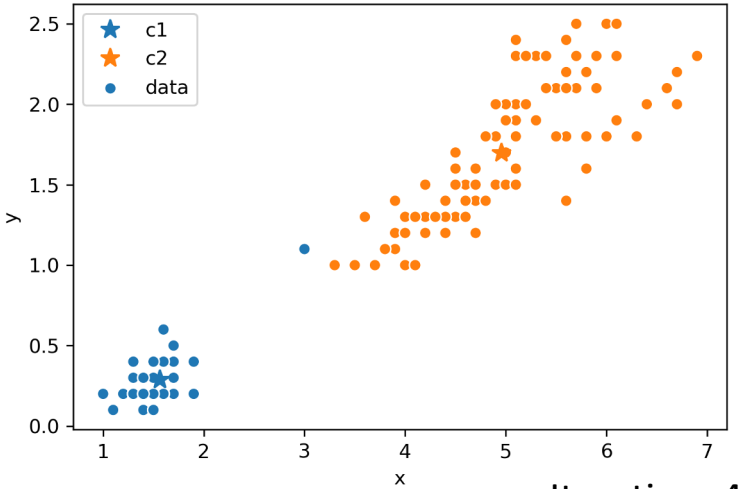
Iteration: 4

K-Means Clustering

Most popular clustering approach: K-Means

- Pick an arbitrary k, and randomly place k “centers”, each a different color
- Repeat until convergence:
  - **Color points according to the closest center**
  - Move center for each color to center of points with that color

Data colored by closest center (in new position)



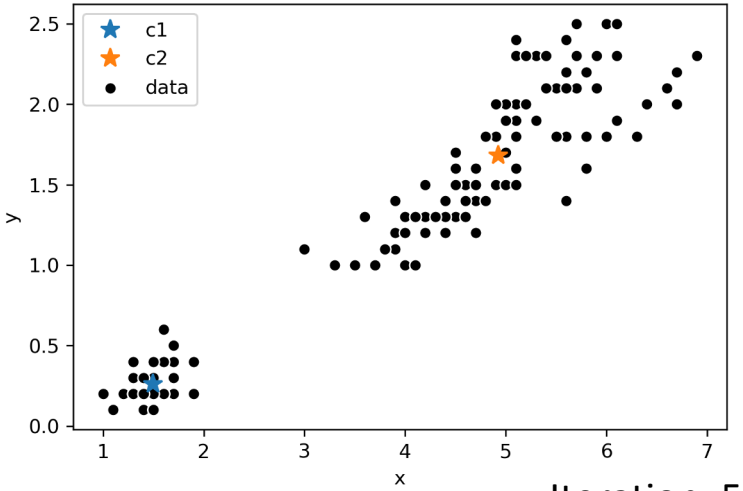
Iteration: 4

K-Means Clustering

Most popular clustering approach: K-Means

- Pick an arbitrary k, and randomly place k “centers”, each a different color
- Repeat until convergence:
  - Color points according to the closest center
  - **Move center for each color to center of points with that color**

Centers moved to new position



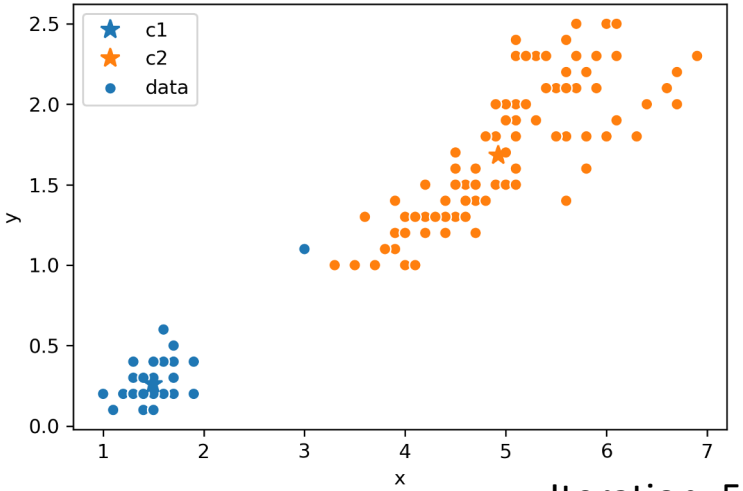
Iteration: 5

K-Means Clustering

Most popular clustering approach: K-Means

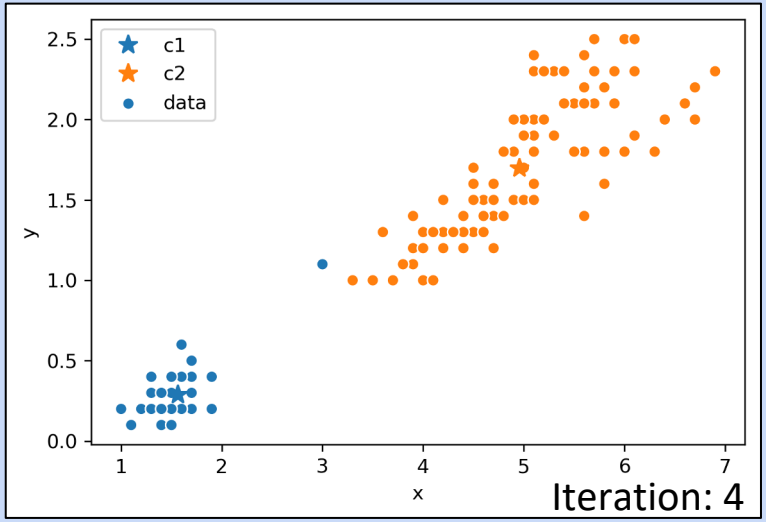
- Pick an arbitrary k, and randomly place k “centers”, each a different color
- Repeat until convergence:
  - **Color points according to the closest center**
  - Move center for each color to center of points with that color

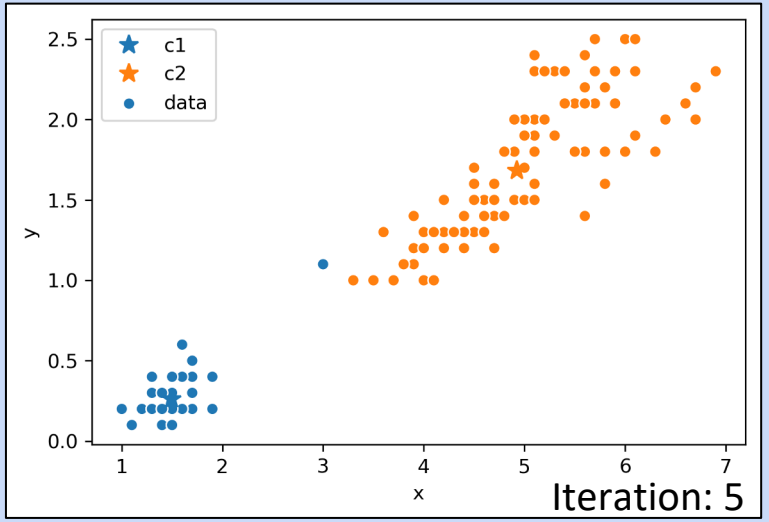
Data colored by closest center (in new position)



Iteration: 5

K-Means Clustering

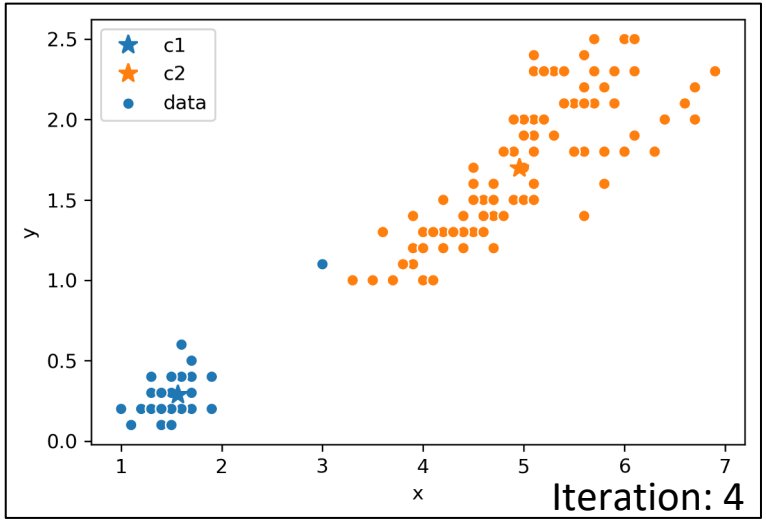


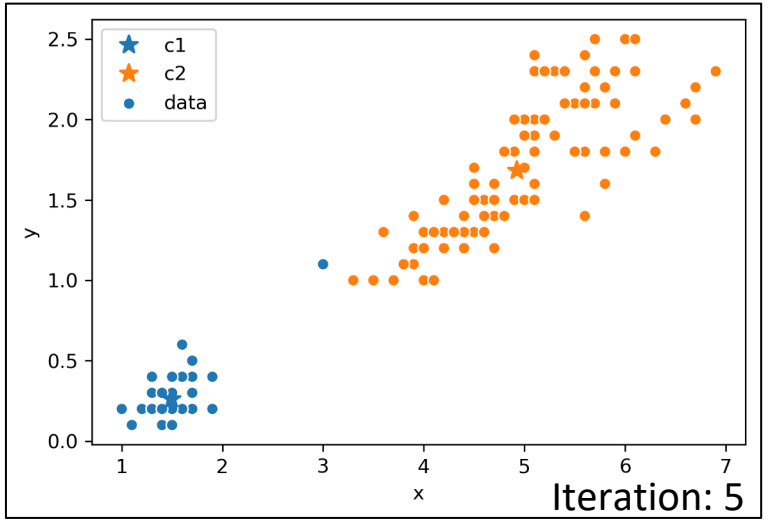


Above we see the results after iteration 4 and 5

- Centers moved slightly between iteration 4 and 5
- But no points changed color
- Are we done?

K-Means Clustering





Above we see the results after iteration 4 and 5

- Centers moved slightly between iteration 4 and 5
- But no points changed color
- Are we done?
  - Yes! If we tried iteration 6, we'd see that centers don't move at all

K-Means vs. K-Nearest Neighbors

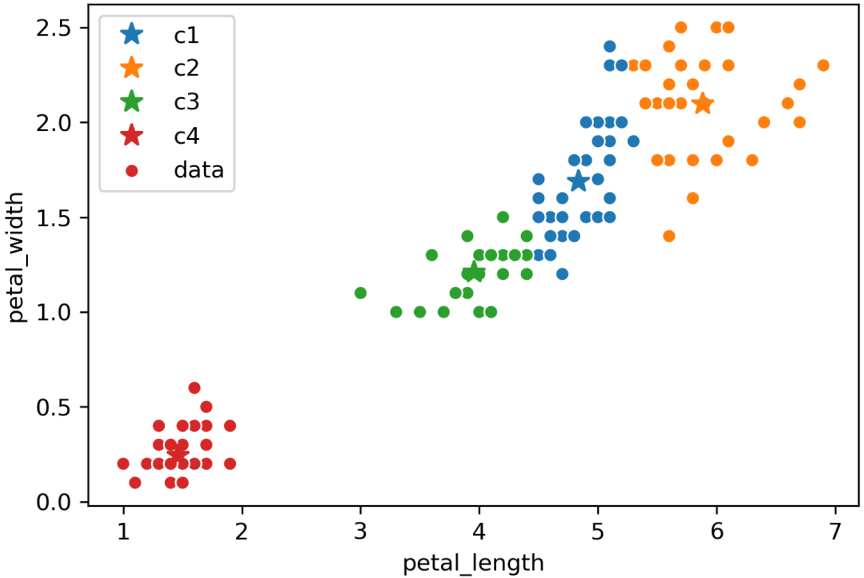
Quick note: K-Means is a totally different algorithm than “K-Nearest Neighbors”

- K-Means: For clustering
  - Assigns each point to one of K clusters
- K-Nearest Neighbors: For Classification (or less often, Regression)
  - Prediction is the most common class among the k-nearest data points the training set
  - Won’t discuss in our class

Minimizing Inertia

K-Means Clustering for K = 4

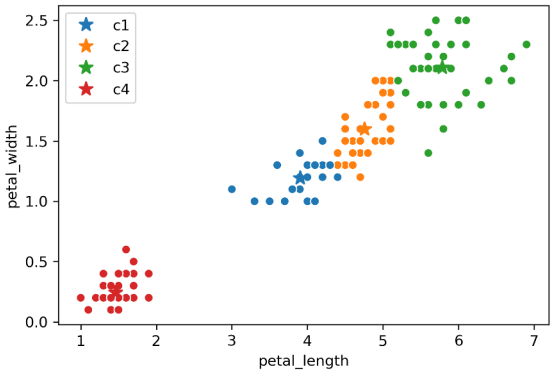
Below is an example of an output for K=4:



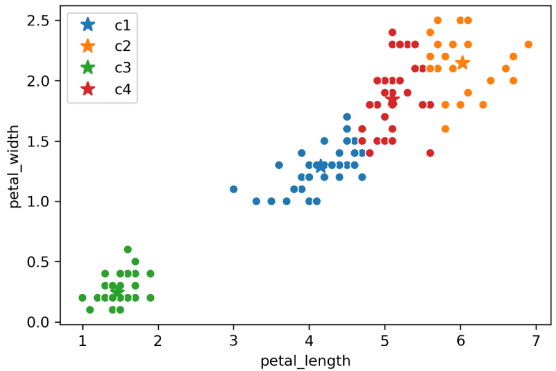
K-Means Clustering for K = 4

Each time you run K-Means, you get a different output

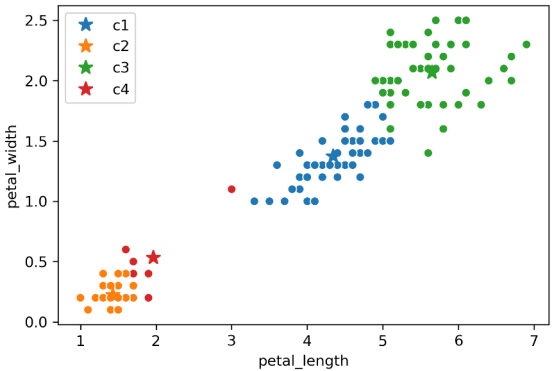
`random.seed(25)`



`random.seed(29)`



`random.seed(40)`



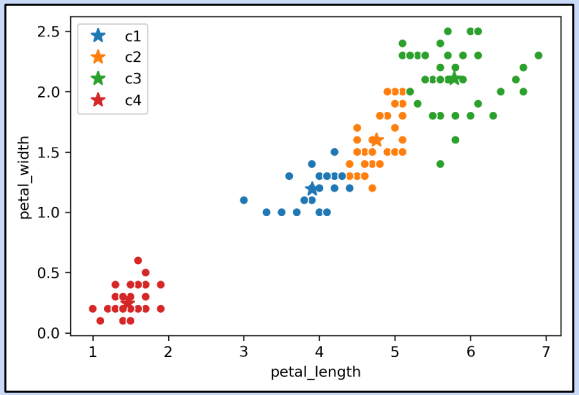
Which is best?

- One approach: Define some sort of loss function

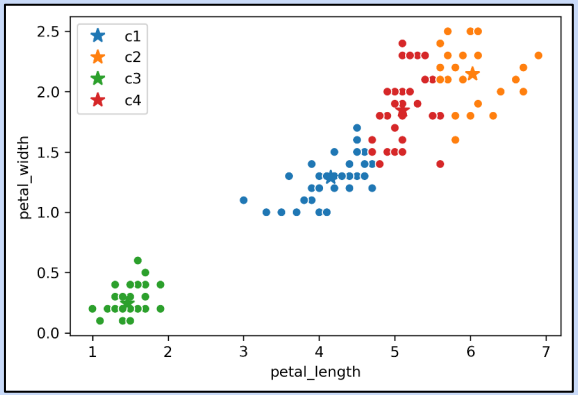
K-Means Clustering for K = 4

Each time you run K-Means, you get a different output

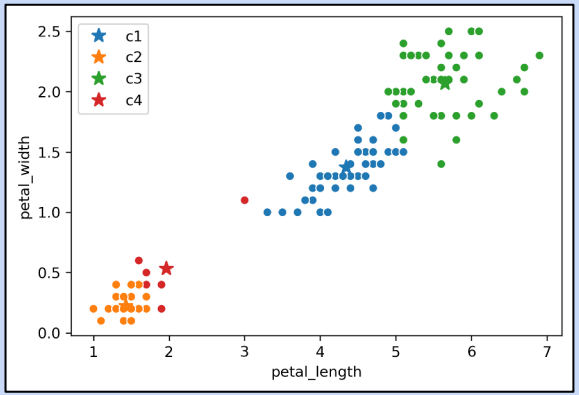
`random.seed(25)`



`random.seed(29)`



`random.seed(40)`

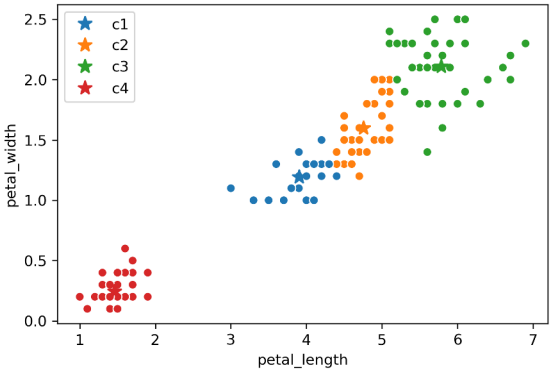


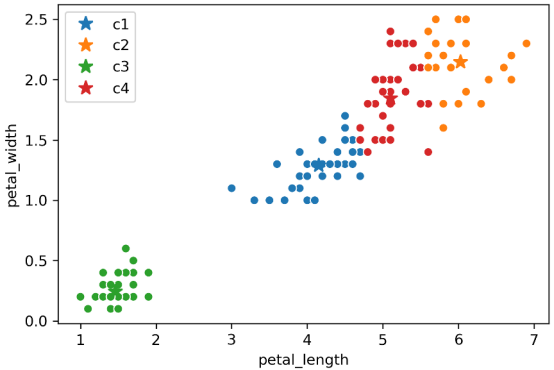
Which is best?

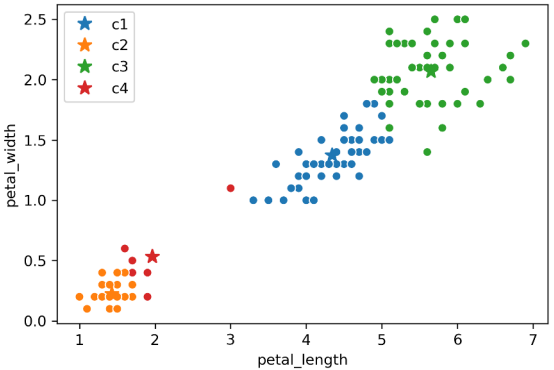
- One approach: Define some sort of loss function
- Come up with a loss function for clustering

K-Means Clustering for K = 4

Each time you run K-Means, you get a different output -- can define a loss to decide which is best







Come up with a loss function for clustering -- your ideas:

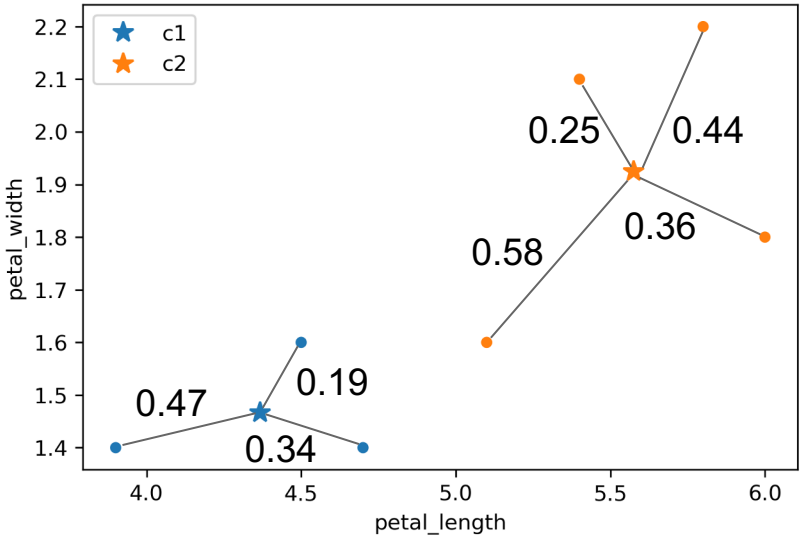
- The sum of distances from each point to its center
- Could take into account balance of number of points per cluster

K-Means Clustering for K = 4

To evaluate different clustering results, we need a loss function

Two common loss functions:

- Inertia: Sum of squared distances from each data point to its center
- Distortion: Weighted sum of squared distances from each data point to its center



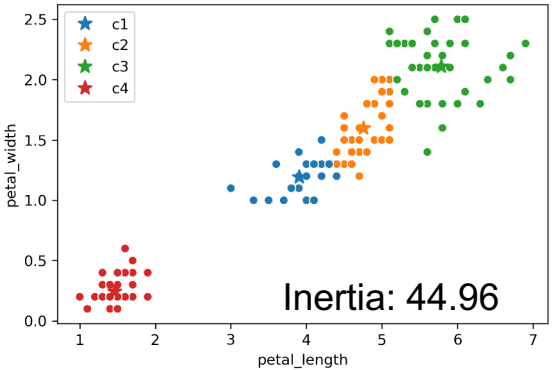
Example:

- Inertia: $0.47^2 + 0.19^2 + 0.34^2 + 0.25^2 + 0.58^2 + 0.36^2 + 0.44^2$
- Distortion: $(0.47^2 + 0.19^2 + 0.34^2)/3 + (0.25^2 + 0.58^2 + 0.36^2 + 0.44^2)/4$

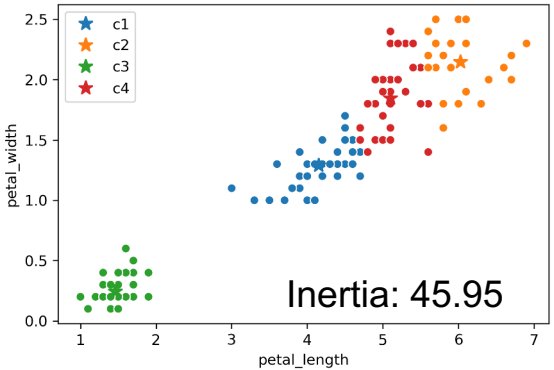
K-Means Clustering for K = 4

Each time you run K-Means, you get a different output

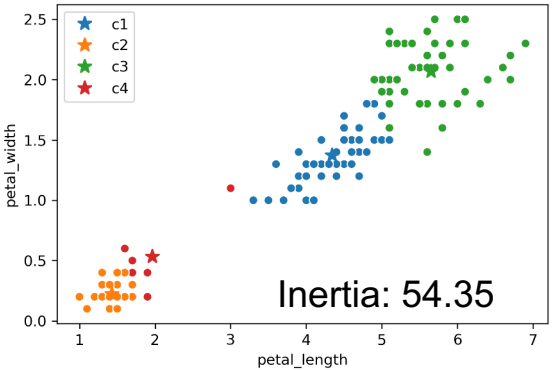
`random.seed(25)`



`random.seed(29)`



`random.seed(40)`



Our loss function says that the leftmost clustering is best (inertia: 44.96) and rightmost clustering (inertia: 54.35) is worst

K-Means and Inertia

It turns out that the function K-Means is trying to minimize is inertia

… but often fails to find global optimum. Why? Sketch below

Can think of K-means as a pair of optimizers that take turns

- First optimizer:
  - Holds center positions constant
  - Optimizes data colors
- Second optimizer:
  - Holds data colors constant
  - Optimizes center positions
- Neither gets total control

Optimizing Inertia

Hard problem: Give an algorithm that optimizes inertia

- Your algorithm should return the EXACT best centers and colors
- Don't worry about runtime

A "coloring" is just a choice of color for every point, e.g. point 1 = red, point 2 = green, point 3 = red, point 4 = blue

Algorithm:

- For all possible k^n colorings:
  - Compute the k centers for that coloring
  - Compute the inertia for the k centers
    - If current inertia is better than best known, write down the current centers and coloring and call that the new best known

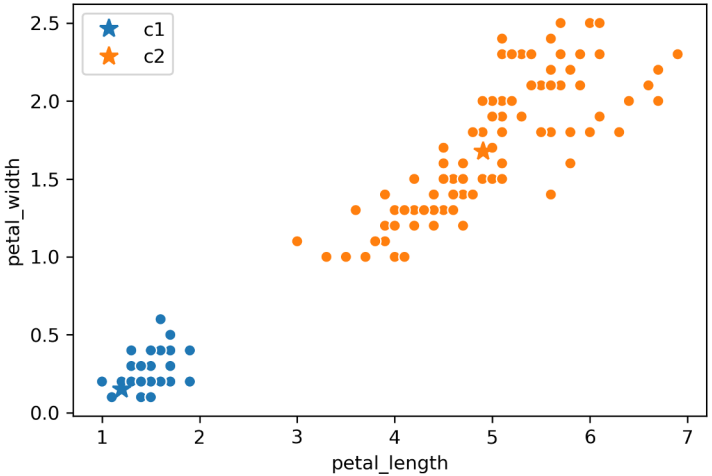
No better algorithm has been found

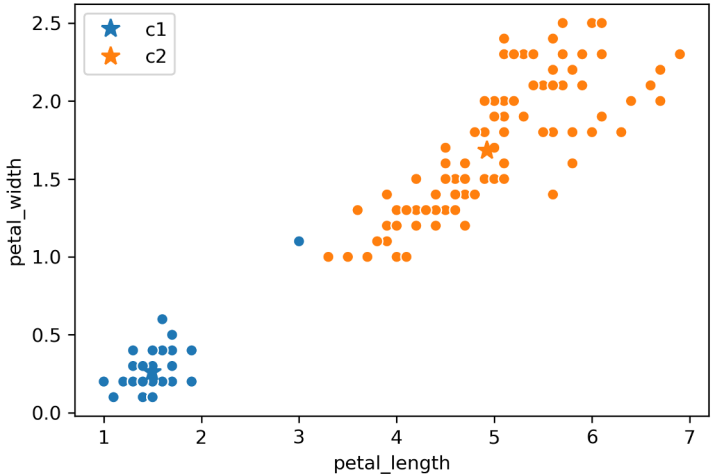
For those who know what this means:
K-Means is known to be an NP-hard problem

Agglomerative Clustering

K-Means

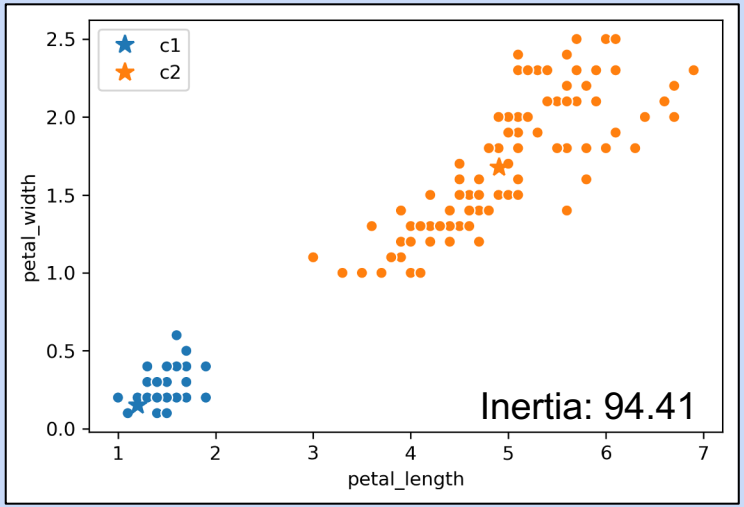
Which clustering result do you like better?

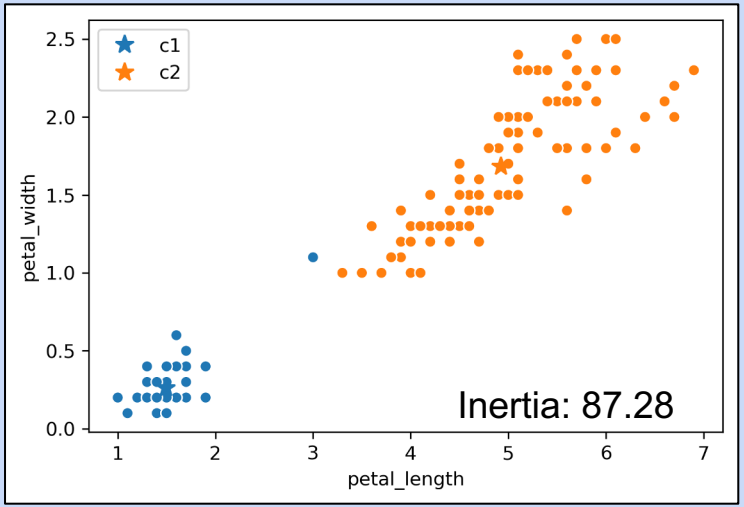




K-Means

Which clustering result do you like better?



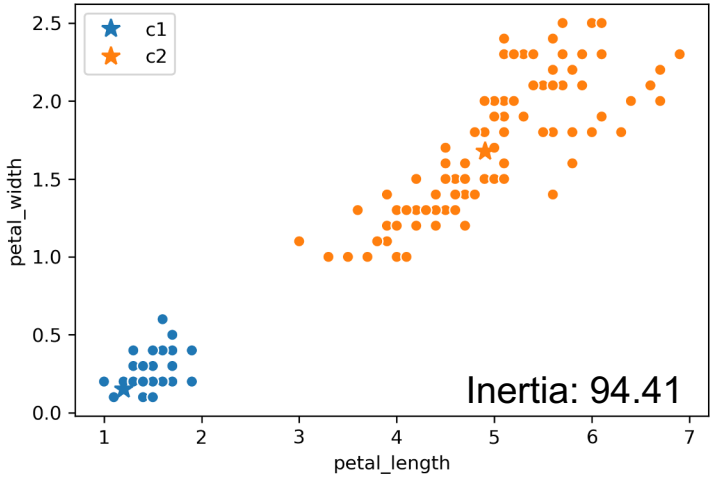


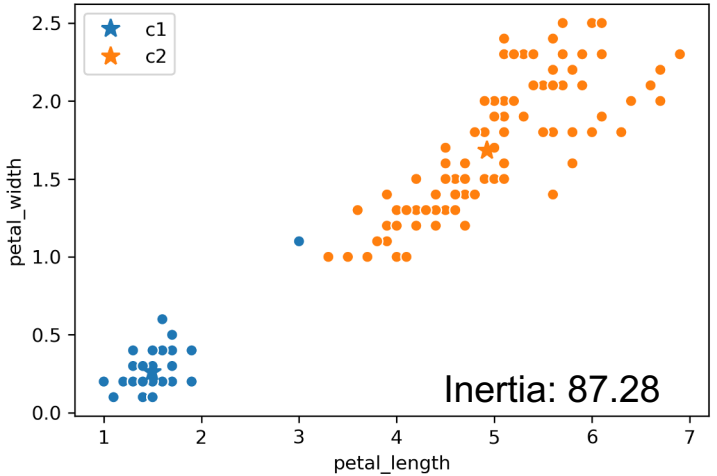
K-Means likes the one on the right better. It has lower inertia

- Why is the inertia on the right lower?
- Is clustering on the right “wrong”?

K-Means

Which clustering result do you like better?





K-Means likes the one on the right better because it has lower inertia: sum of squared distances from each data point to its center

- Why is the inertia lower? K-Means optimizes for distance, not “blobbiness”
- Is clustering on the right “wrong”? Good question!

Agglomerative Clustering

As with regression and classification, there are many ways to do clustering

So far we've seen K-Means, which attempts to minimize inertia

- Results not guaranteed to optimize inertia
- Even global optimum may not match our intuition of the best result

Let's discuss an alternate idea known as "agglomerative clustering"

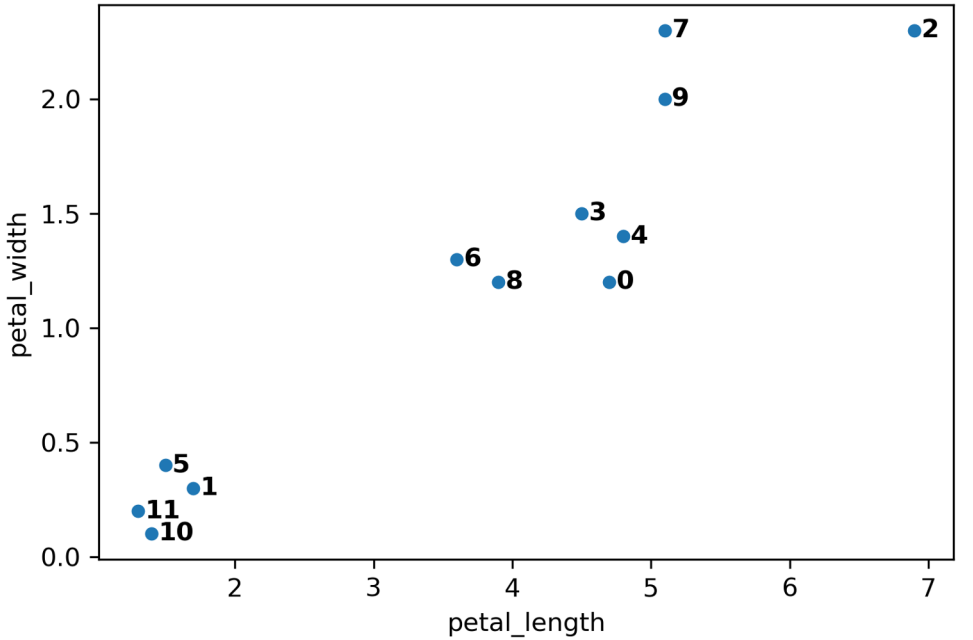
- Basic idea:
  - Every data point starts out as its own cluster
  - Join clusters with neighbors until we have only K clusters left

Let's see an example for K = 2

Agglomerative Clustering Example

When the algorithm starts, every data point is in its own cluster

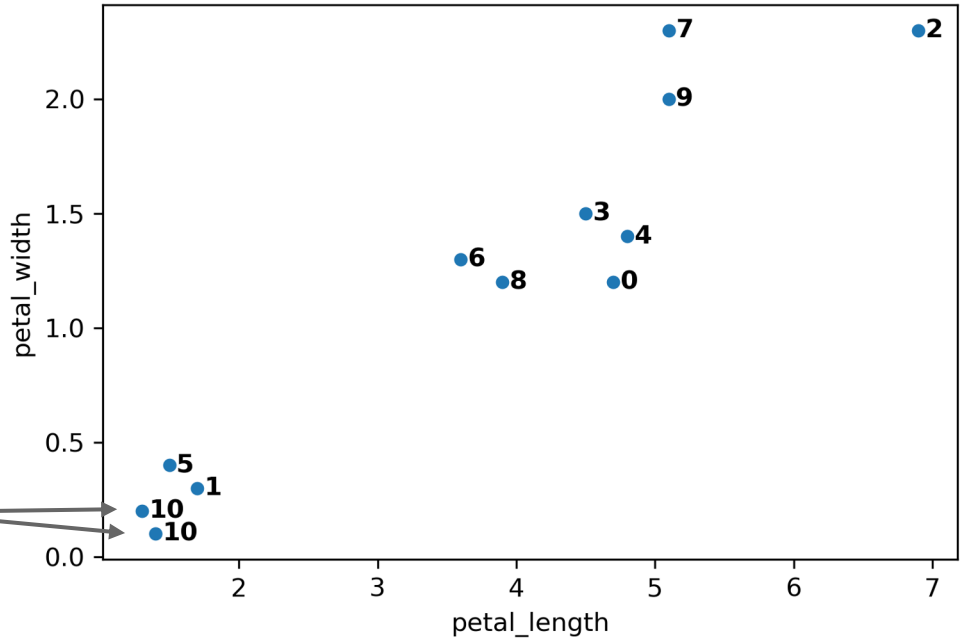
- Below, 12 data points, so 12 clusters
- Closest clusters are 10 and 11, so merge them



Agglomerative Clustering Example

When the algorithm starts, every data point is in its own cluster

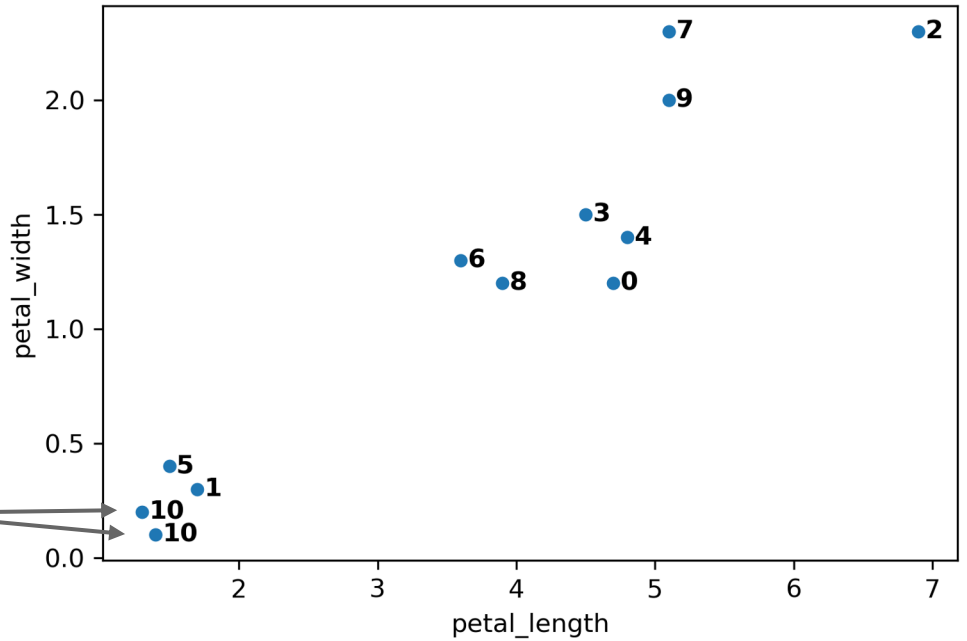
- Below, 12 data points, so 12 clusters
- Closest clusters are 10 and 11, so merge them



Two points in the same cluster

Agglomerative Clustering Example

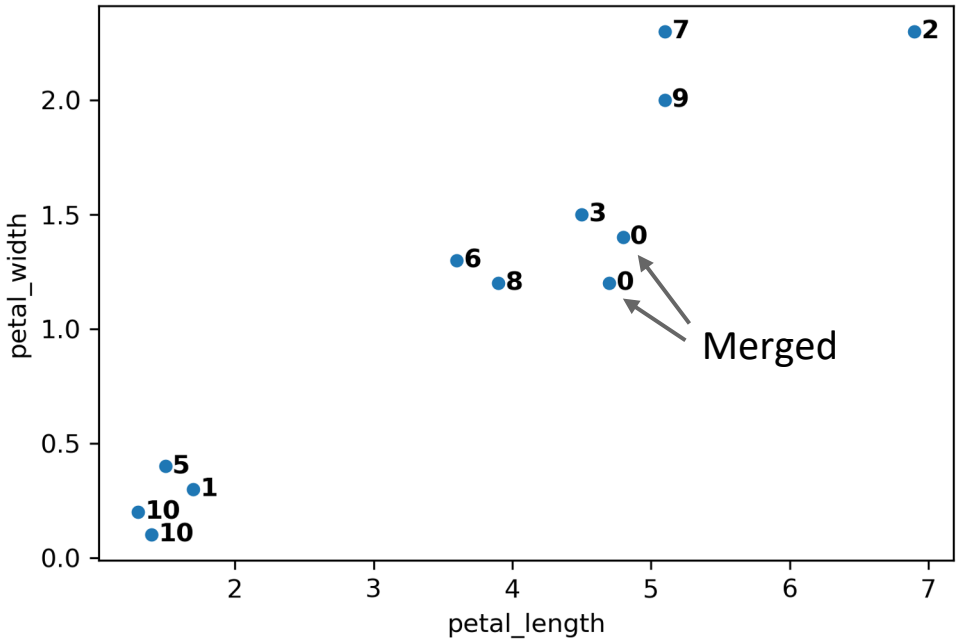
Next two closest are 0 and 4, so merge them



Two points in the same cluster

Agglomerative Clustering Example

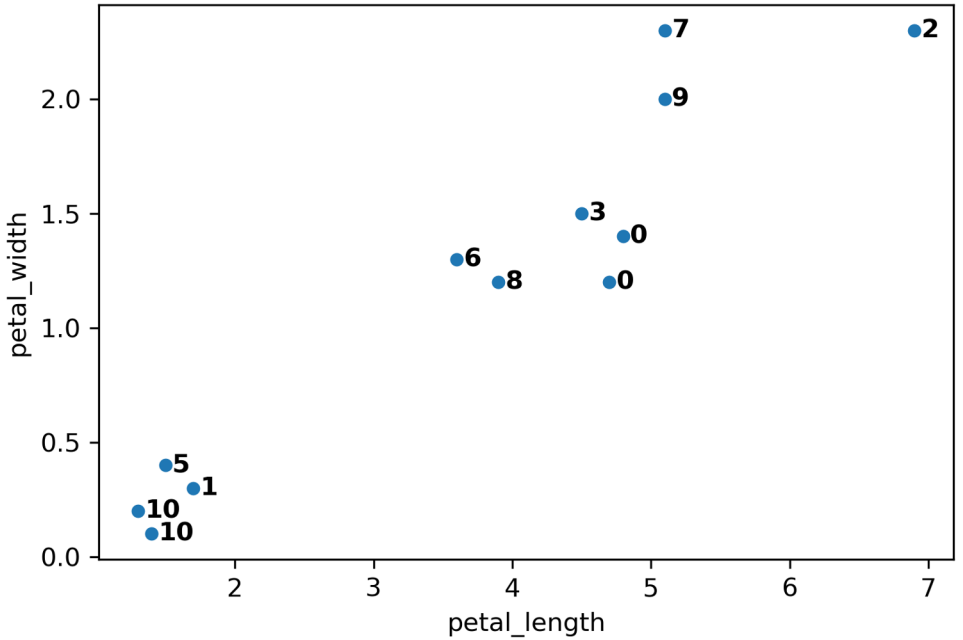
Next two closest are 0 and 4, so merge them



Agglomerative Clustering Example

At this point we have 10 clusters:

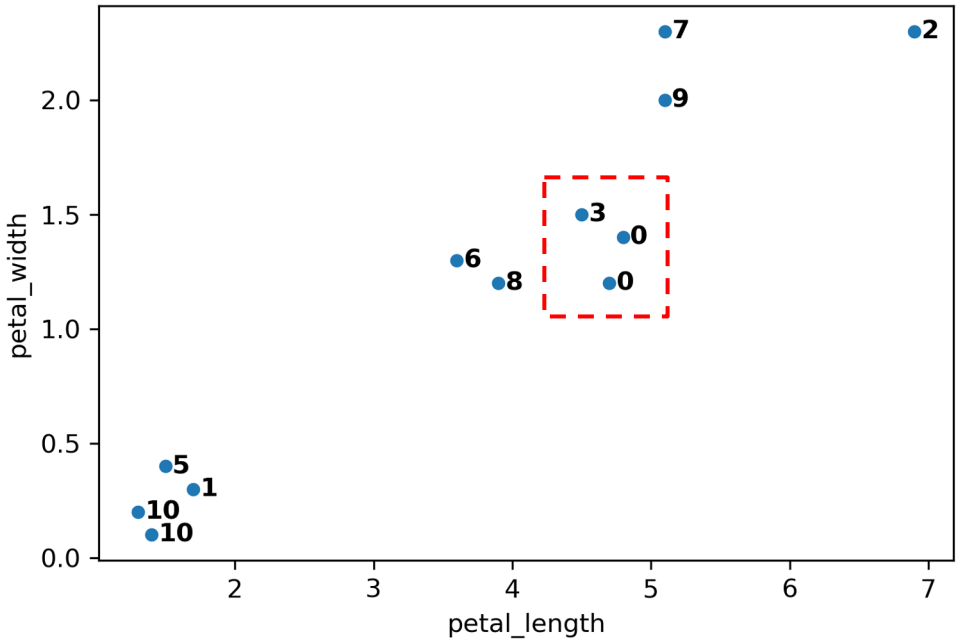
- 8 with a single point {1, 2, 3, 5, 6, 7, 8, 9}
- 2 with two points {0/0, 10/10}



Agglomerative Clustering Example

Tricky question:

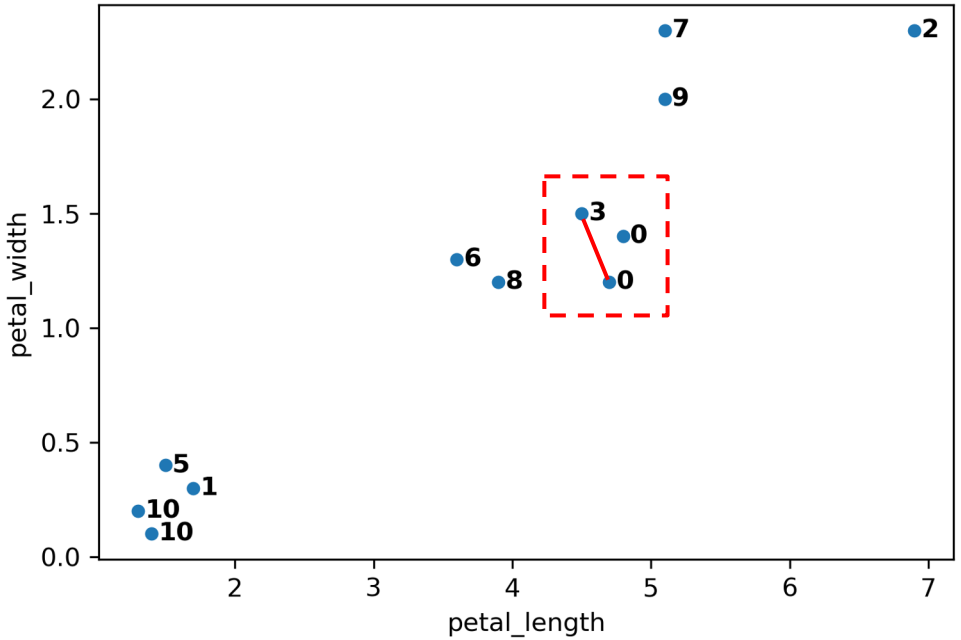
- What is the distance between clusters 0 and 3?



Agglomerative Clustering Example

Tricky question:

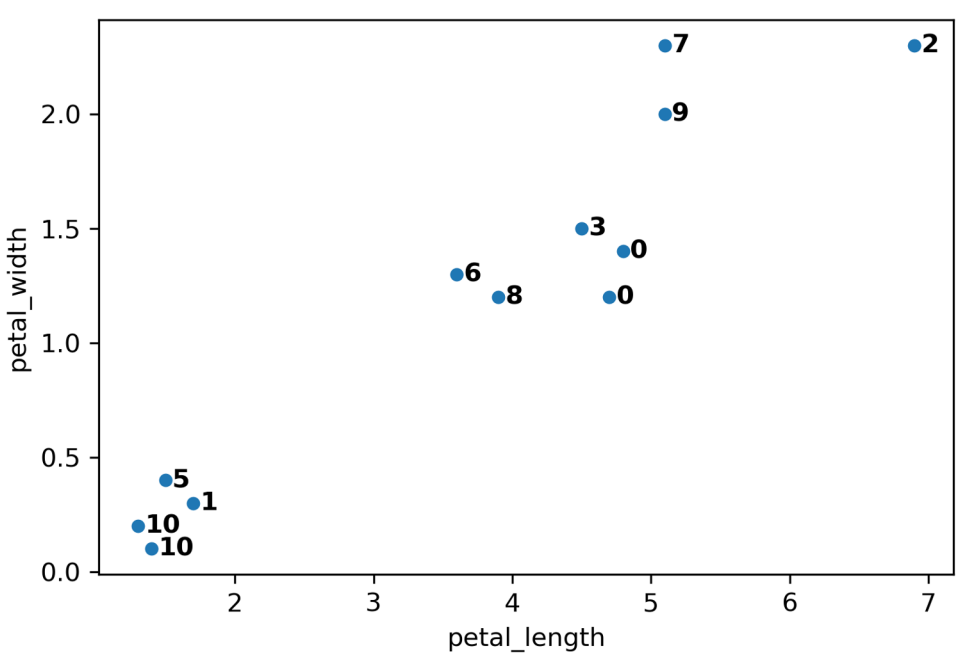
- What is the distance between clusters 0 and 3?
- There is no right answer. Common choice, use the **max**



Would be perfectly reasonable to do something else, e.g. average or minimum

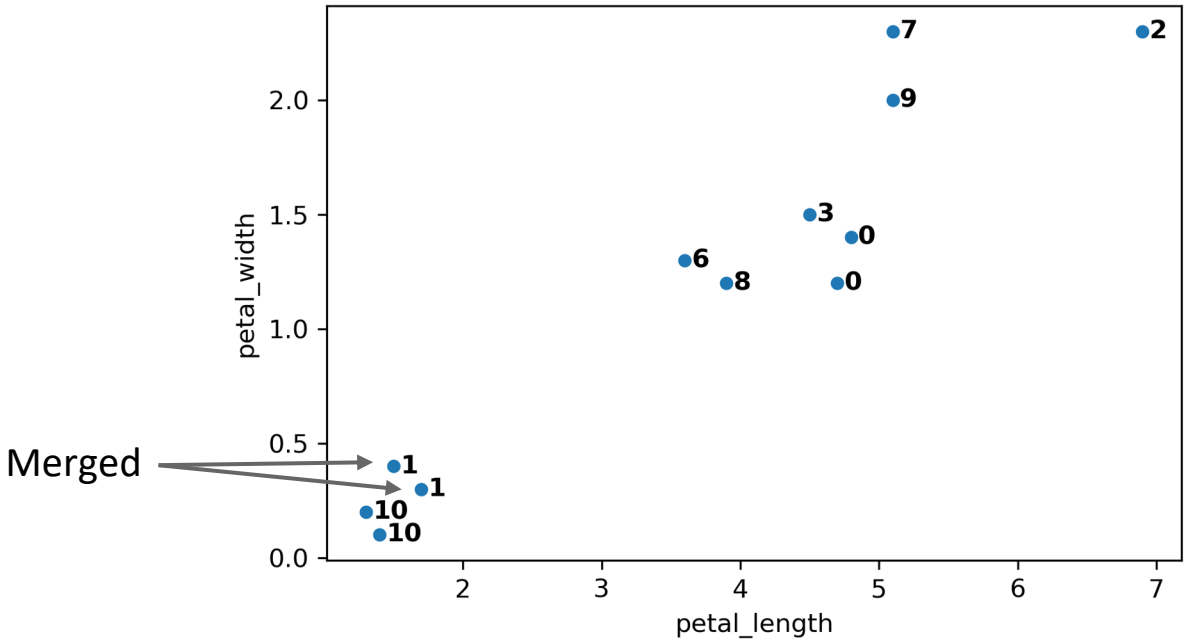
Agglomerative Clustering Example

Next two closest clusters are 1 and 5



Agglomerative Clustering Example

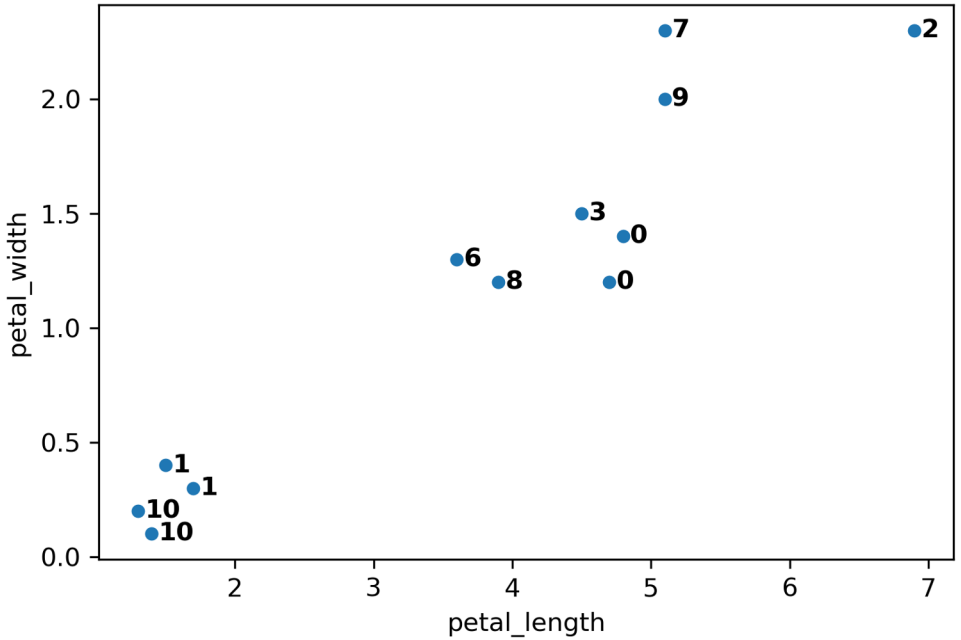
Next two closest clusters are 1 and 5



Agglomerative Clustering Example

Next two closest clusters are 7 and 9

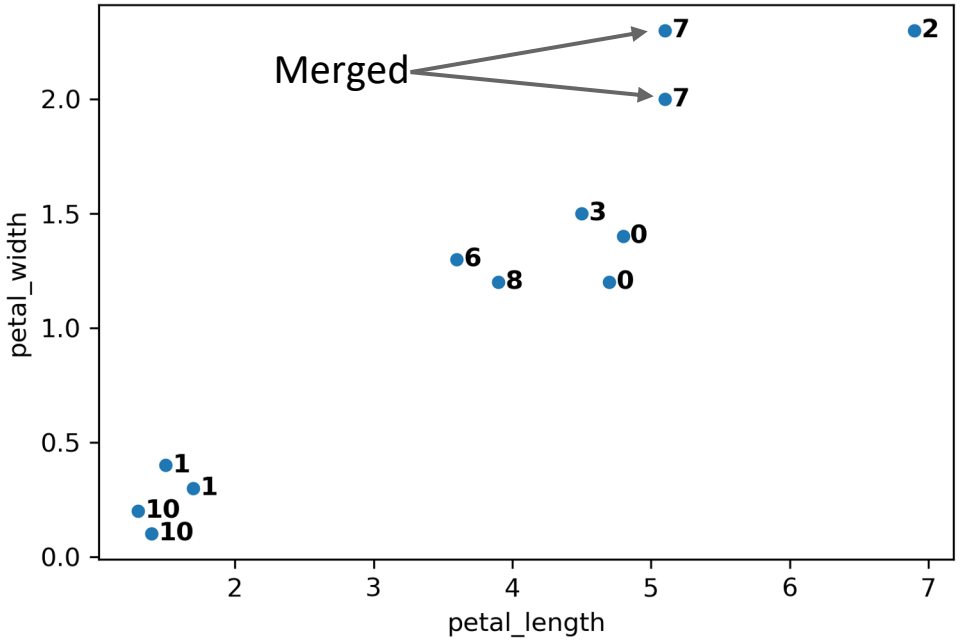
- Note: Might not look that way, but axes are not on the same scale! Y-axis goes only up to 2.5, and x axis goes up to more than 7



Agglomerative Clustering Example

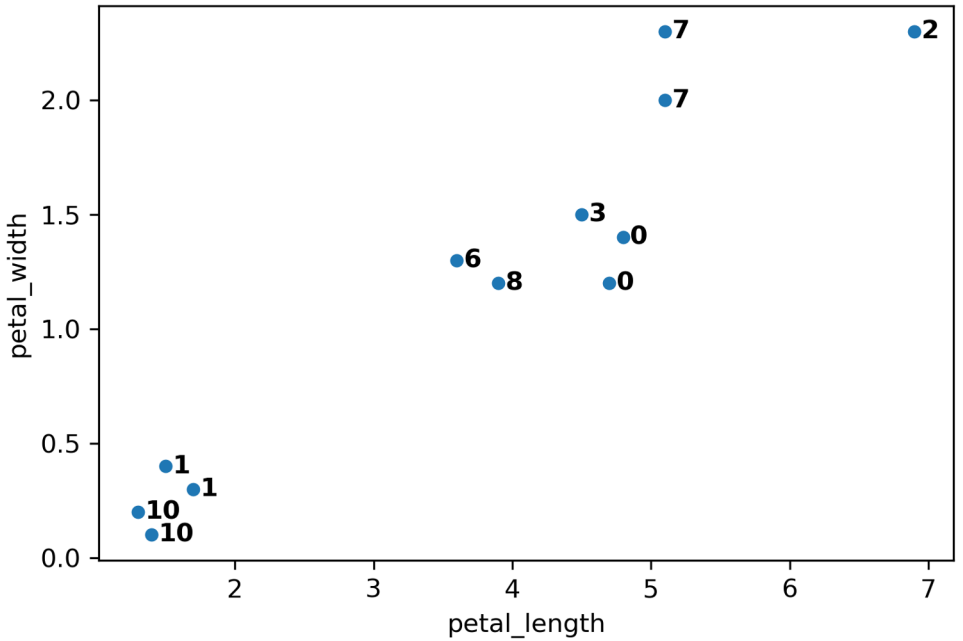
Next two closest clusters are 7 and 9

- Note: Might not look that way, but axes are not on the same scale! Y-axis goes only up to 2.5, and x axis goes up to more than 7



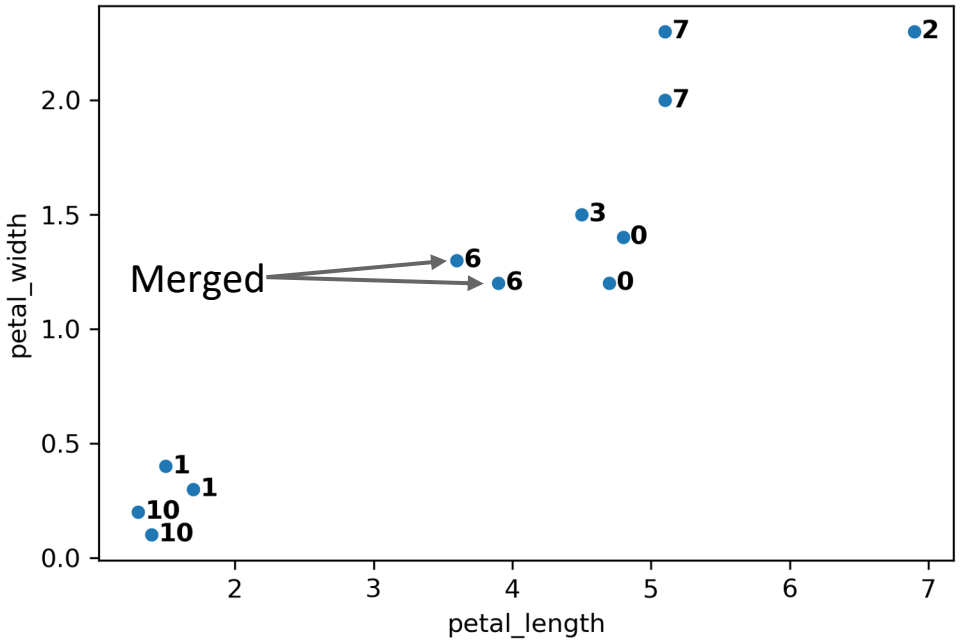
Agglomerative Clustering Example

Next closest are 6 and 8



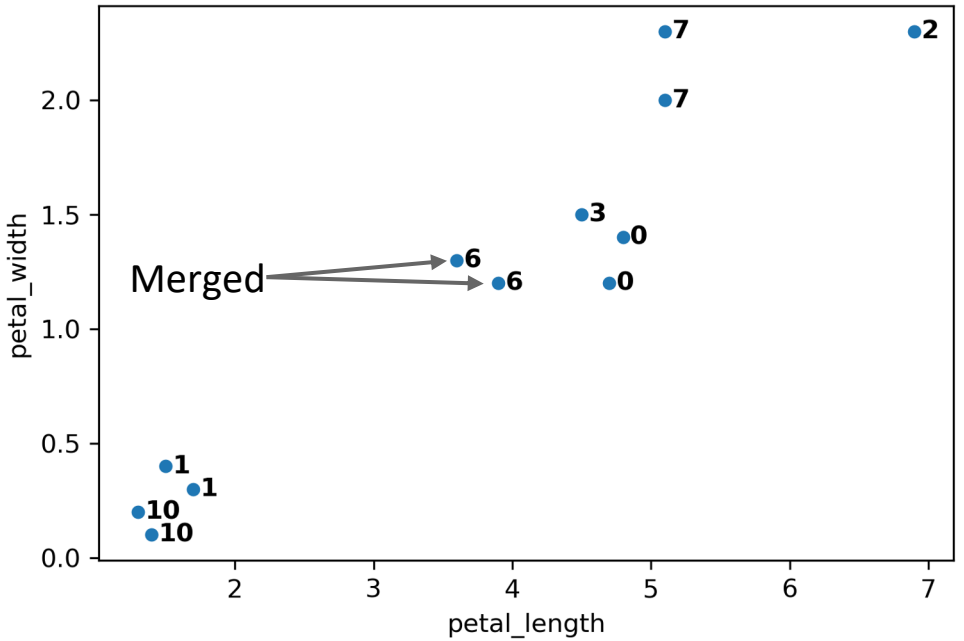
Agglomerative Clustering Example

Next closest are 6 and 8



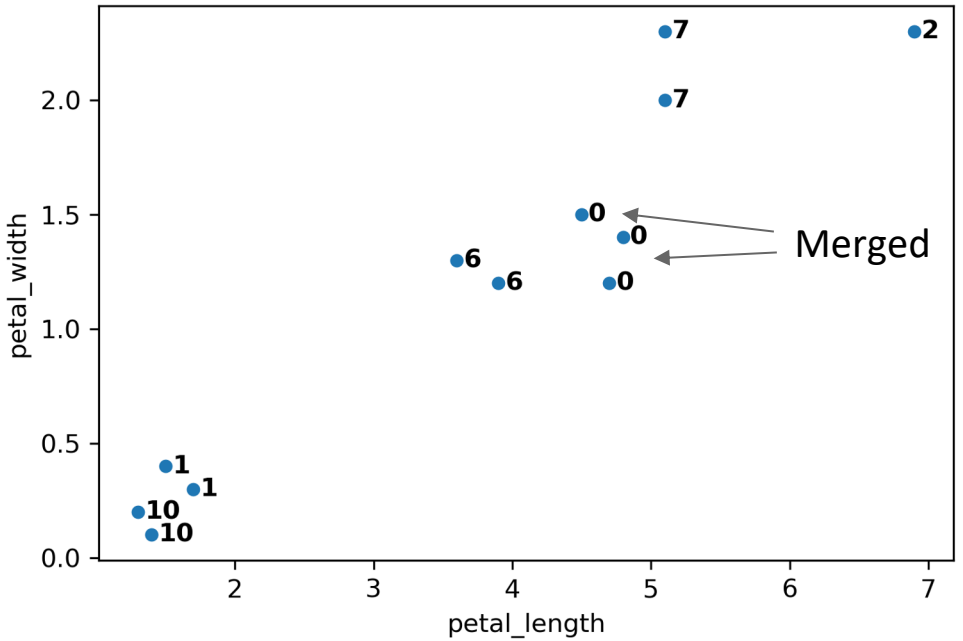
Agglomerative Clustering Example

Now 0 and 3 are closest. Merge them next



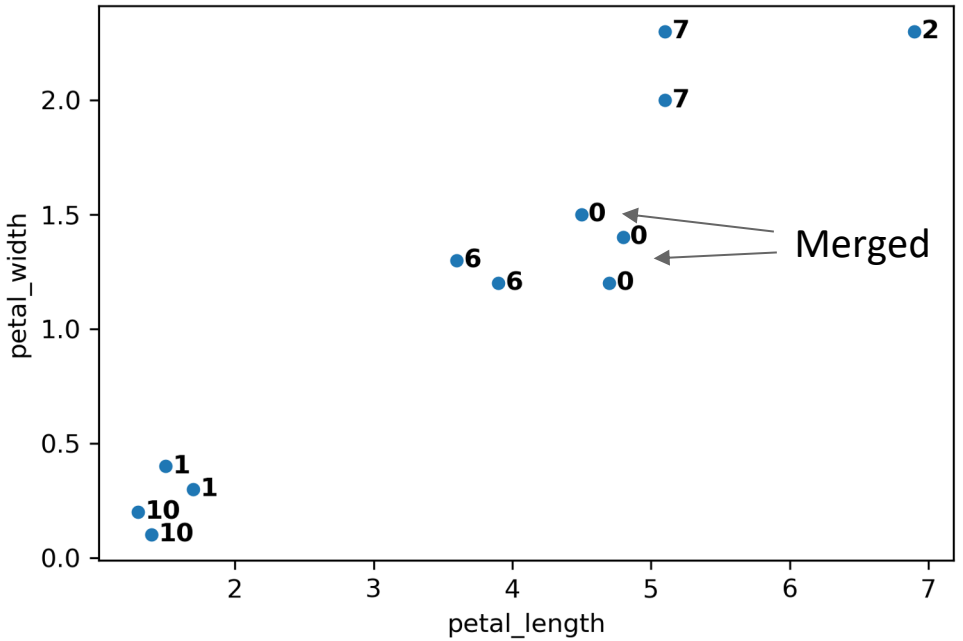
Agglomerative Clustering Example

Now 0 and 3 are closest. Merge them next



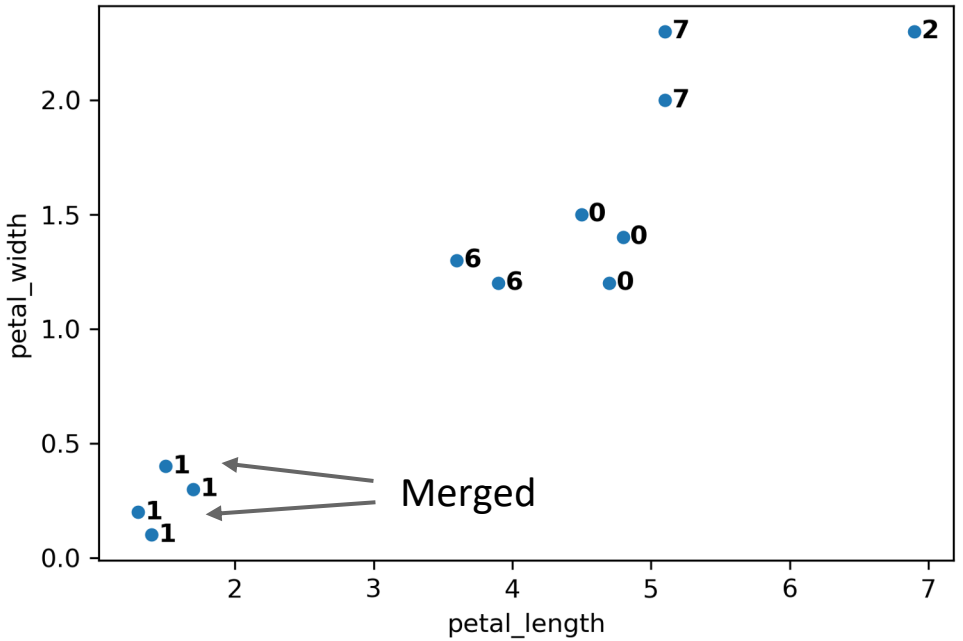
Agglomerative Clustering Example

Next up are 1 and 10



Agglomerative Clustering Example

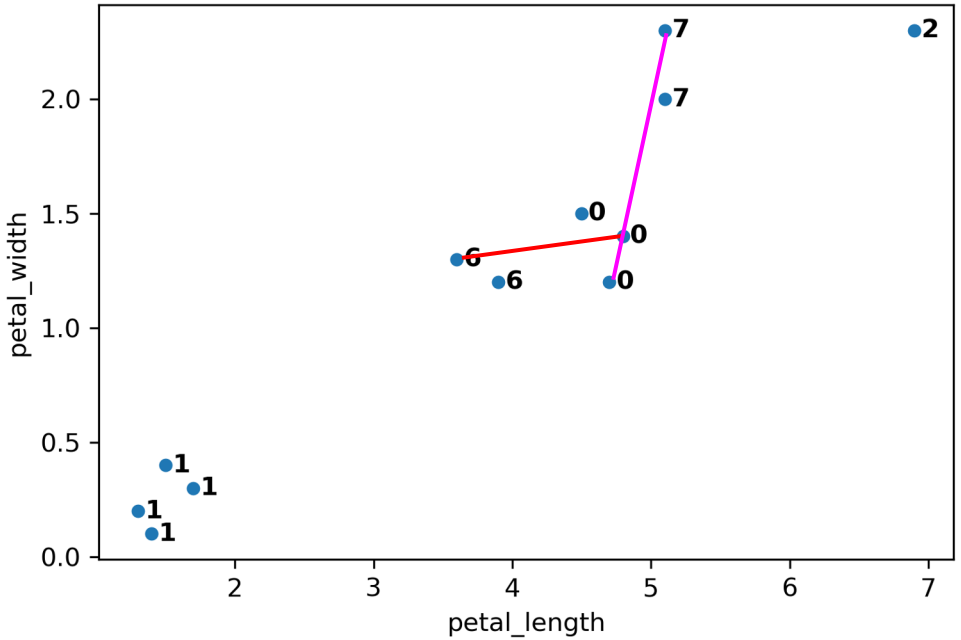
Next up are 1 and 10



Agglomerative Clustering Example

Next up are 0 and 7. Why?

- Max line between any member of 0 and 6 is longer than max line between any member of 0 and 7



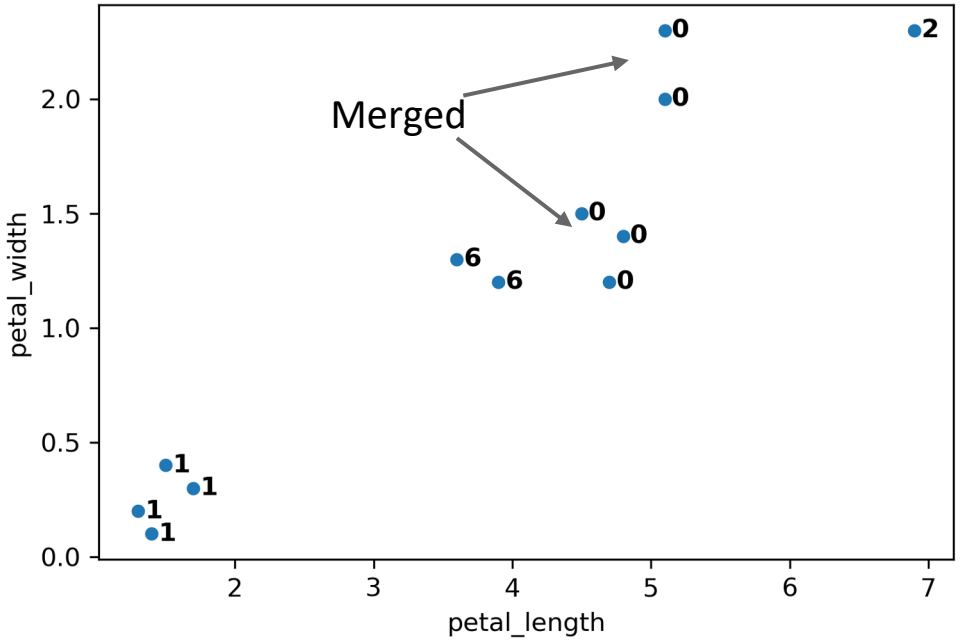
Purple line is shorter than red line

Note: Doesn't look visually shorter due to different scales for x and y axes

Agglomerative Clustering Example

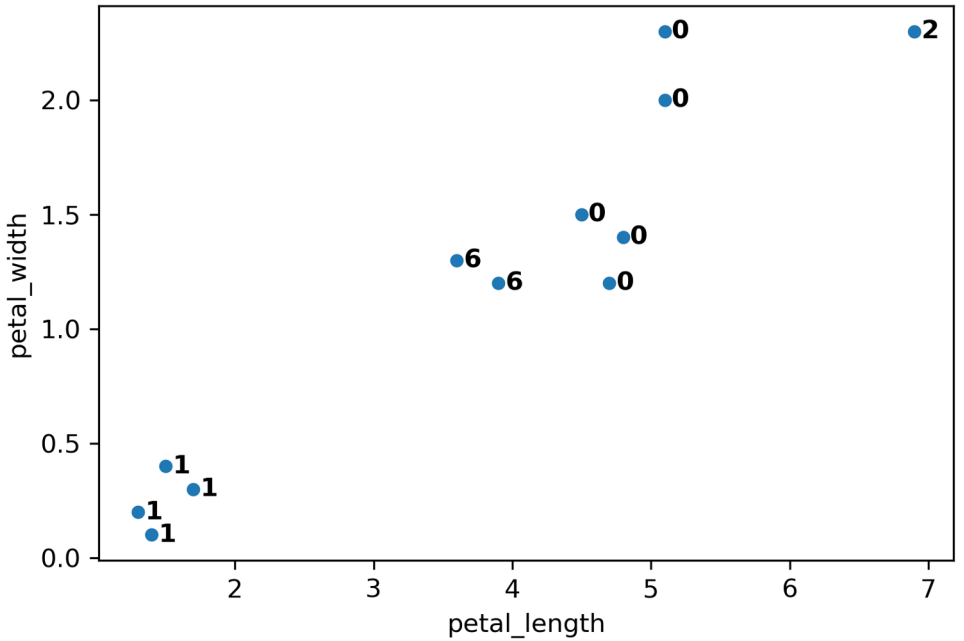
Next up are 0 and 7. Why?

- Max line between any member of 0 and 6 is longer than max line between any member of 0 and 7



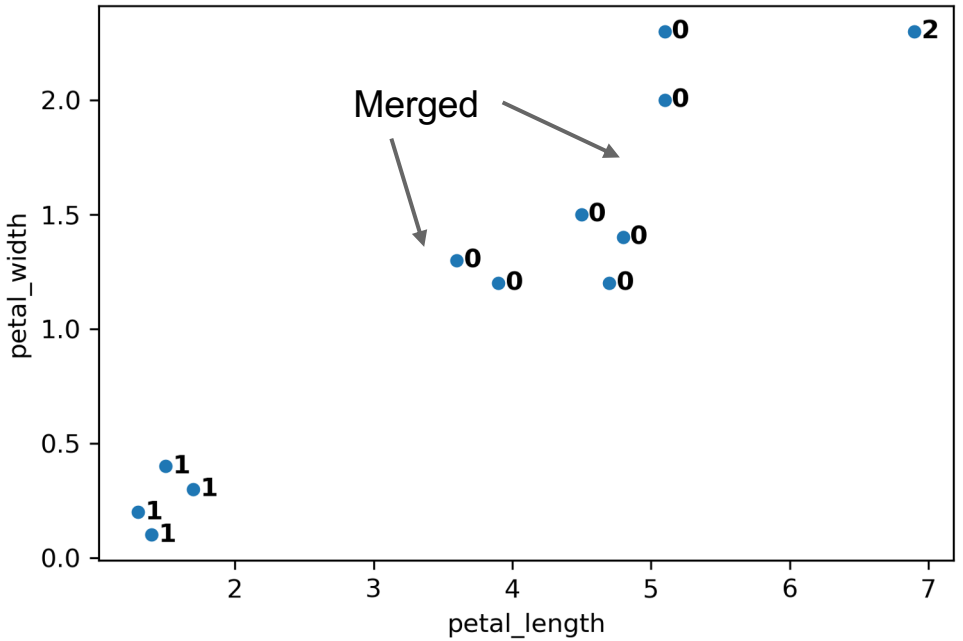
Agglomerative Clustering Example

Next up are 0 and 6



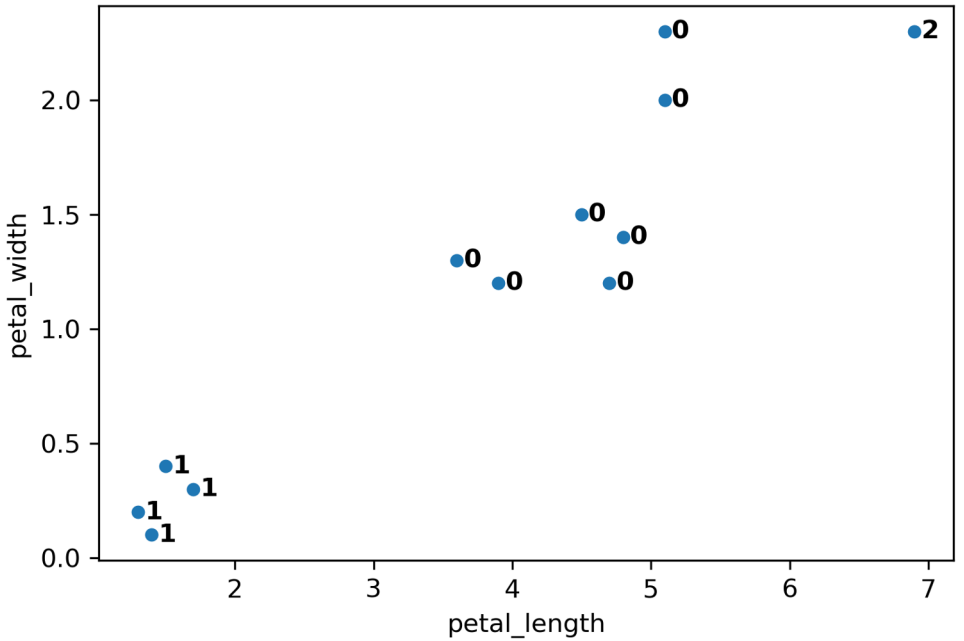
Agglomerative Clustering Example

Next up are 0 and 6



Agglomerative Clustering Example

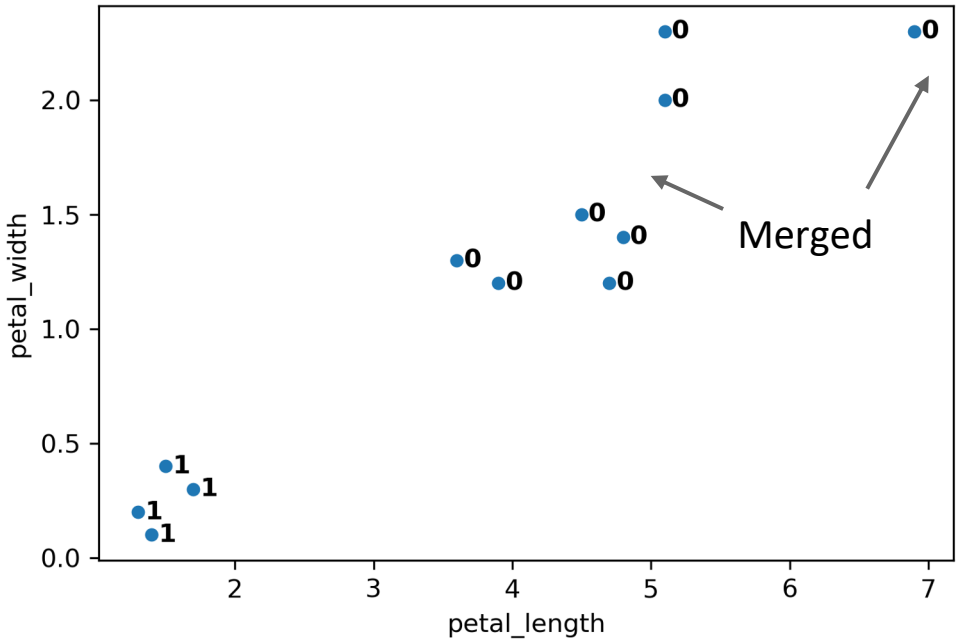
Next up are 0 and 2



Agglomerative Clustering Example

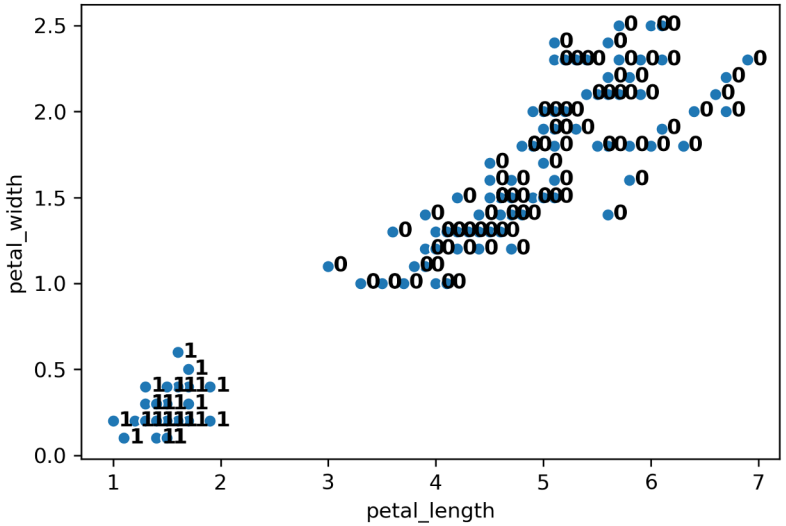
Next up are 0 and 2

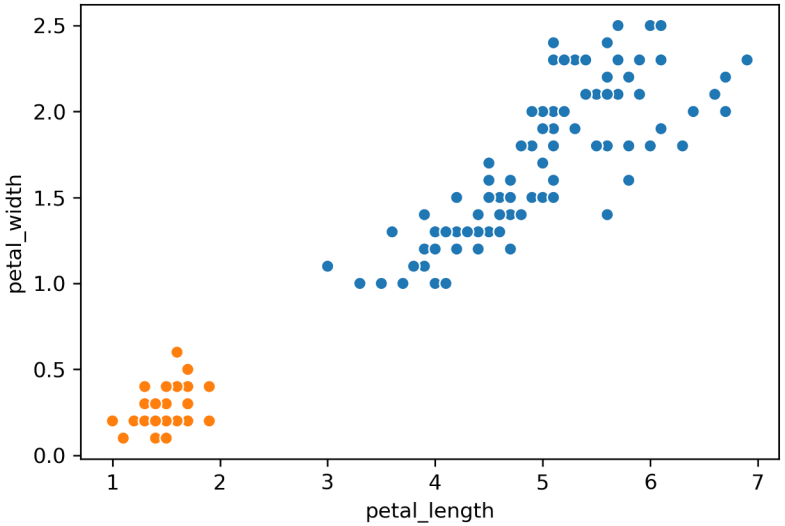
- At this point, we are done, because we only have two clusters left



Agglomerative Clustering Example

On the full dataset, our agglomerative clustering algorithm gets the “right” output

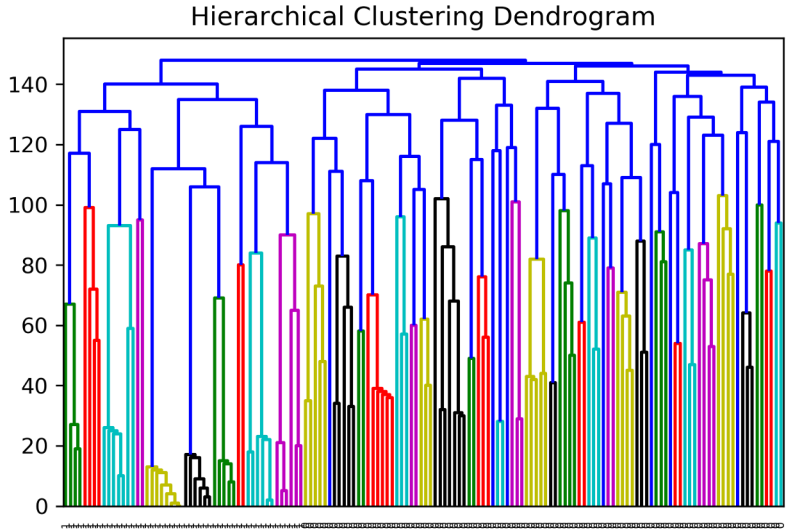


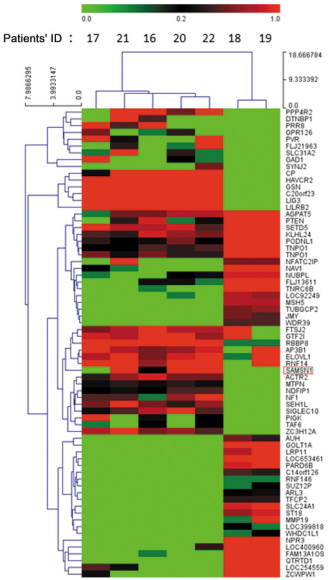


Clustering and Dendrograms

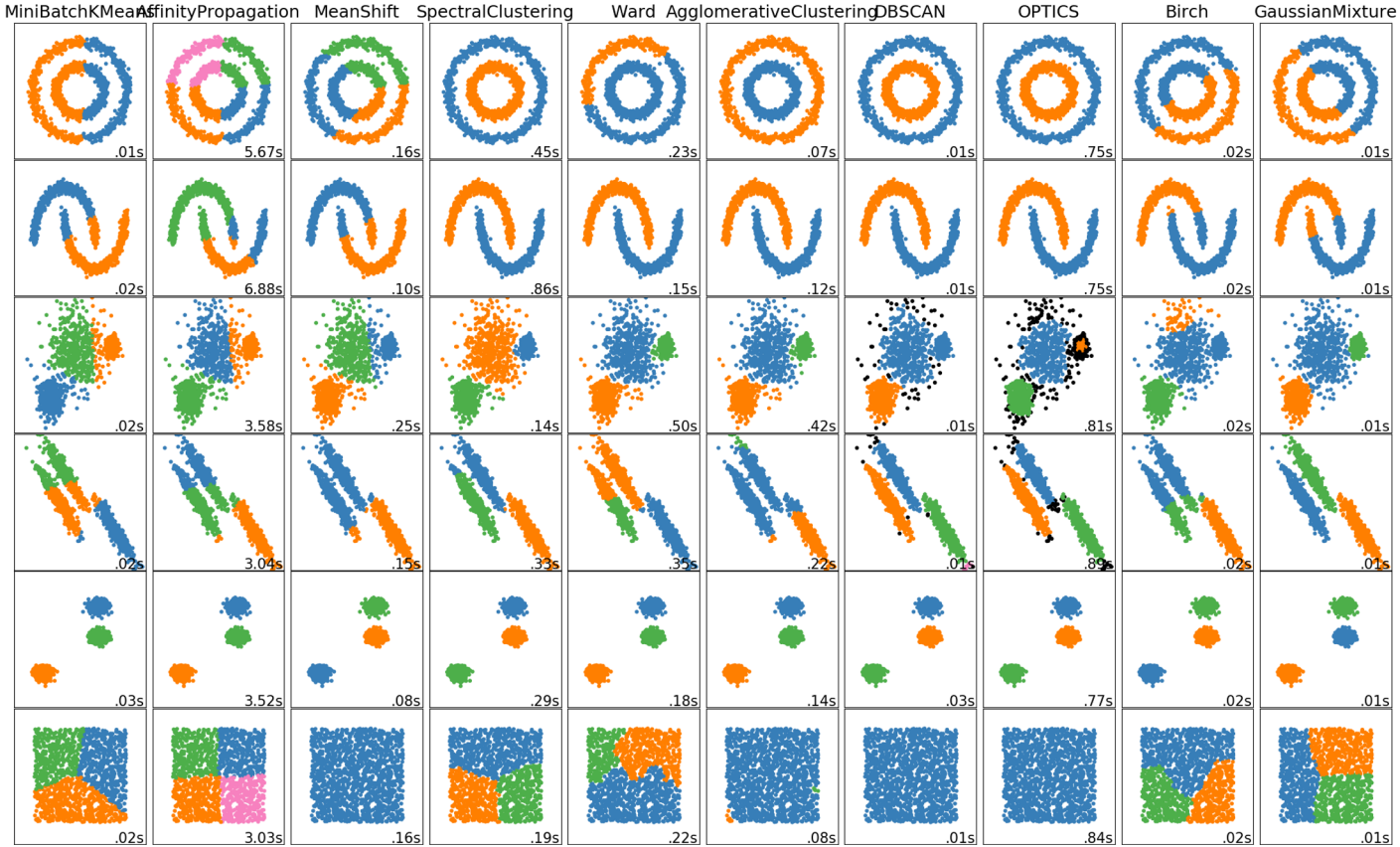
Agglomerative clustering is one form of “hierarchical clustering”

- Can keep track of when two clusters got merged
  - Each cluster is a tree
- Can visualize merging hierarchy, resulting in a “dendrogram”
  - Won’t discuss any further, but you might see these in the wild





Clustering Algorithms: Many More We Haven't Seen



K Means

Agglomerative

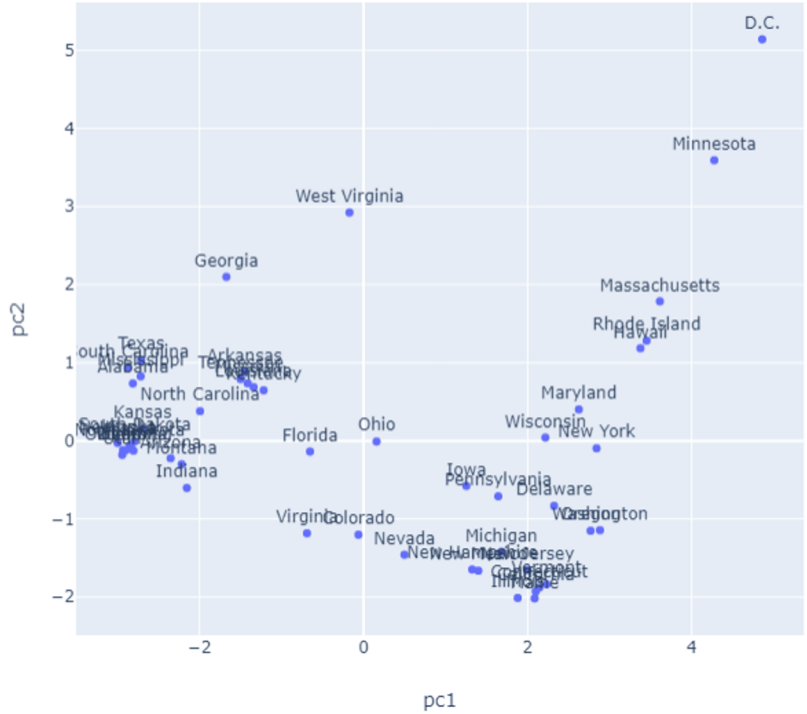
https://scikit-learn.org/stable/_images/sphx_glr_plot_cluster_comparison_001.png

Picking K

Picking K

The algorithms we've discussed today require us to pick a K before we start

- But how do we pick K?



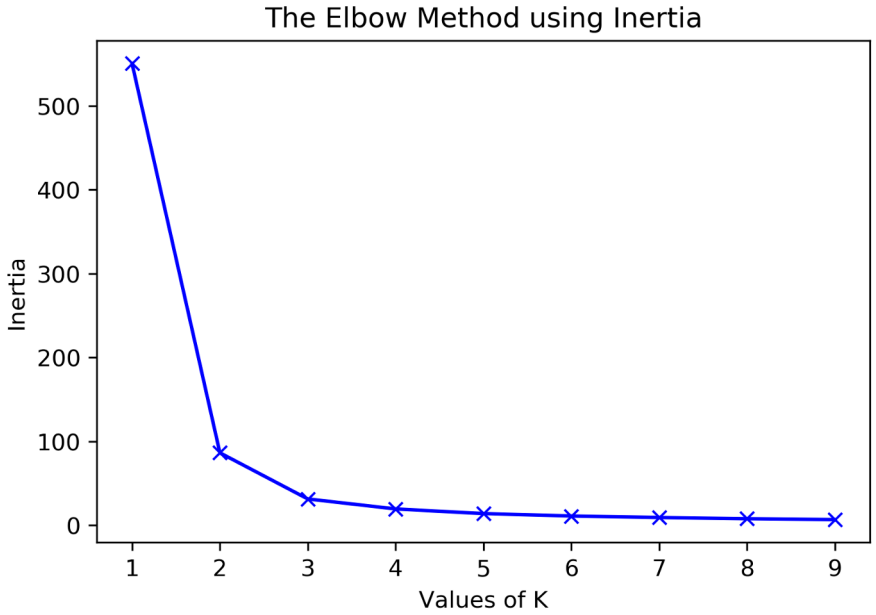
Often, best K is subjective

- How many clusters are there here?

Picking K: Elbow Method

For K-Means, one approach is to plot inertia versus many different K values

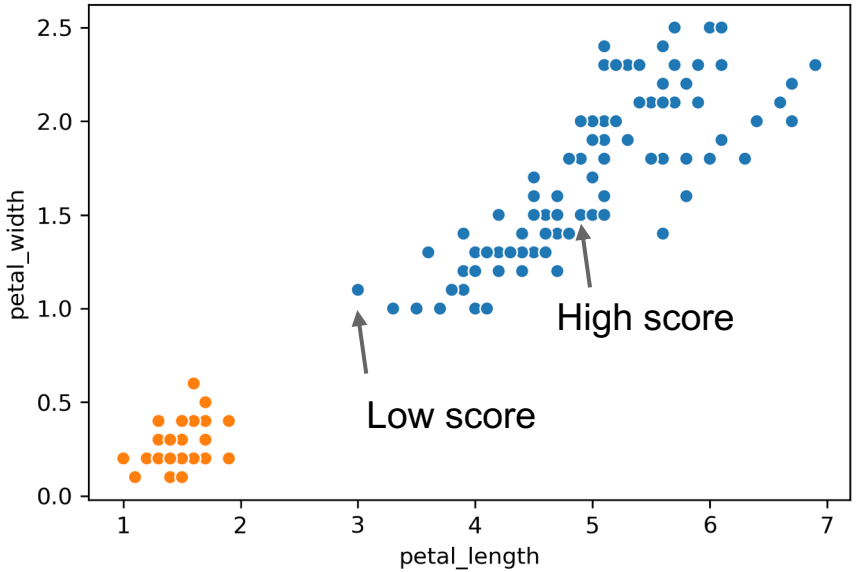
- Pick the K in the “elbow”, where we get diminishing returns afterwards
- Note: Big complicated data often lacks an elbow



Silhouette Scores

To evaluate how “well clustered” a specific data point is, we can use the “silhouette score”, a.k.a. the “silhouette width”

- High score: Near the other points in its X’s cluster
- Low score: Far from the other points in its cluster



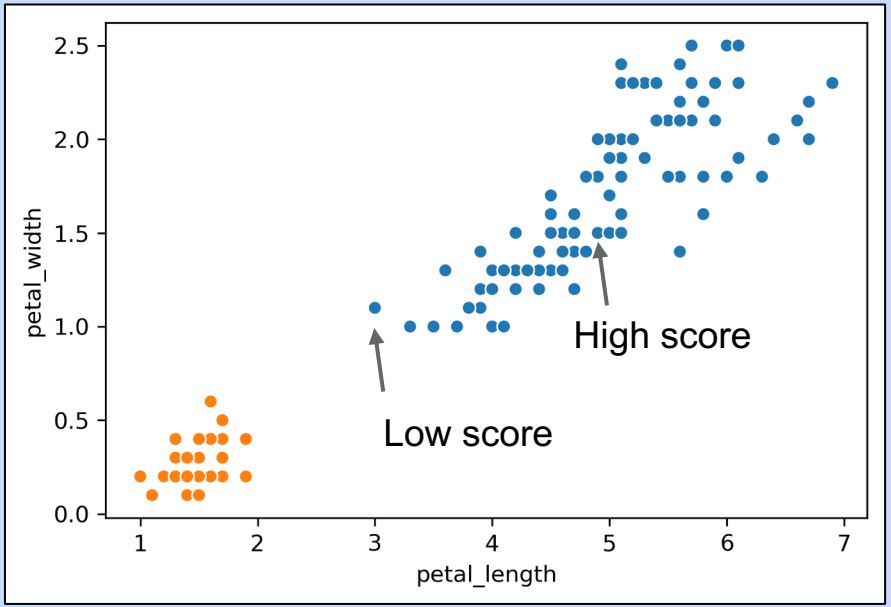
For a data point X, score S is:

- A = avg distance to other points in cluster
- B = avg distance to points in closest cluster
- S = (B - A) / max(A, B)

Silhouette Scores

For a data point X:

- A = average distance to other points in X's cluster
- B = average distance to points in closest cluster
- S = (B - A) / max(A, B)



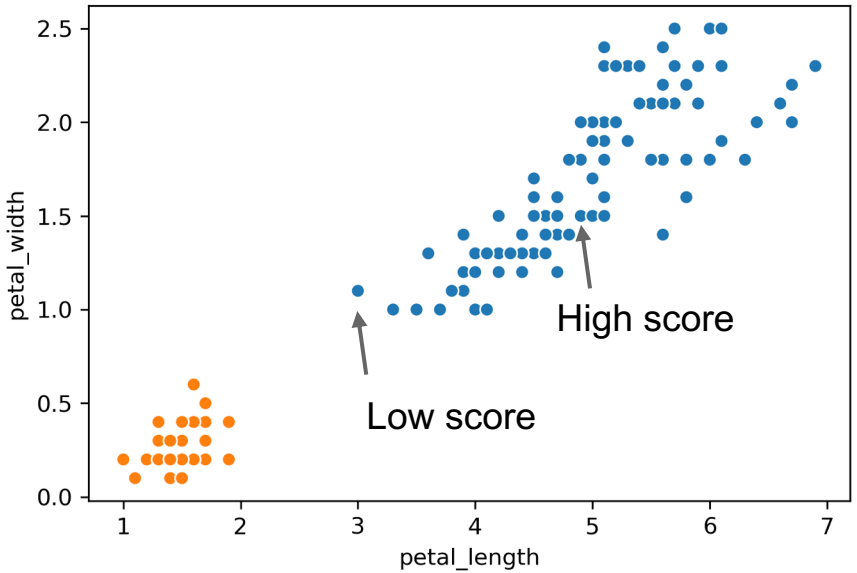
What is the highest possible S?

- How can this happen?

Silhouette Scores

For a data point X:

- A = average distance to other points in X's cluster
- B = average distance to points in closest cluster
- S = (B - A) / max(A, B)



What is the highest possible S?

- How can this happen?

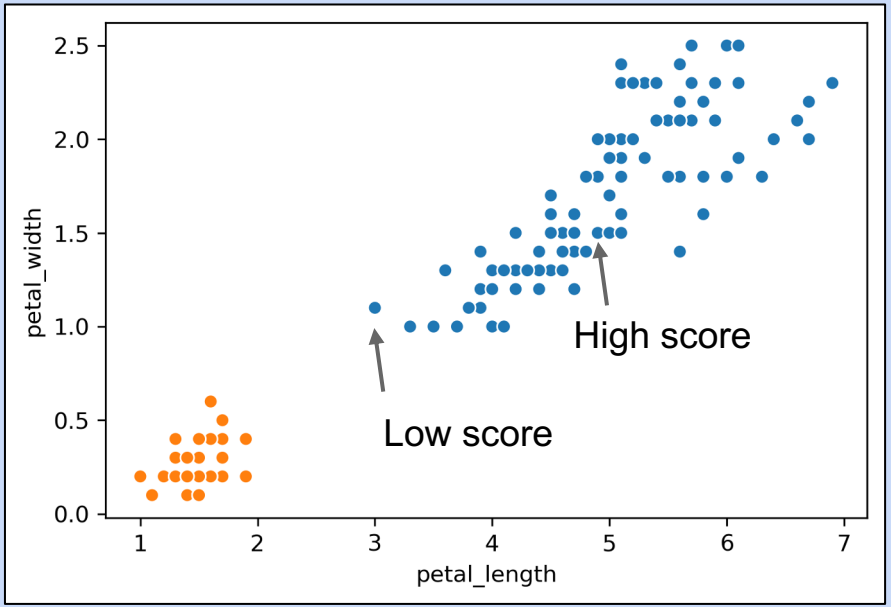
Highest possible S is 1

- This happens if every point in X's cluster is right on top of X

Silhouette Scores

For a data point X:

- A = average distance to other points in X's cluster
- B = average distance to points in closest cluster
- S = (B - A) / max(A, B)

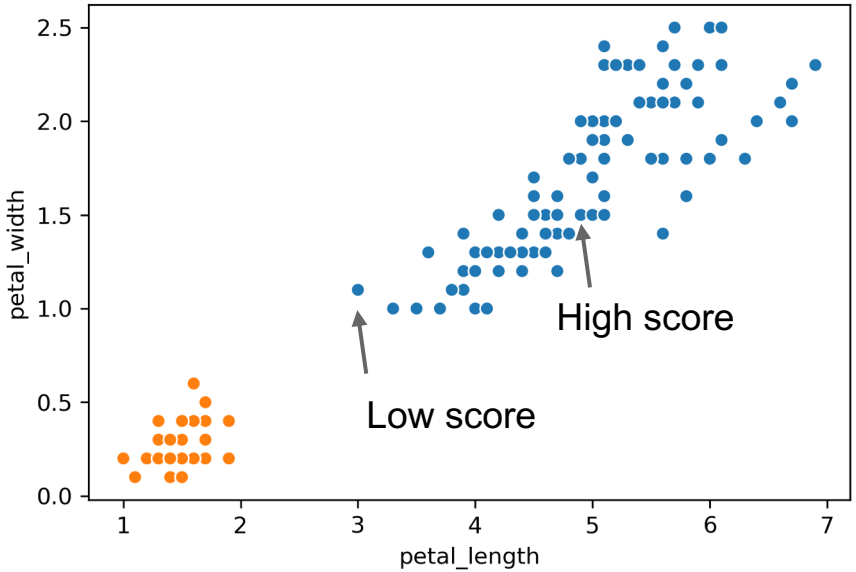


Can S be negative?

Silhouette Scores

For a data point X:

- A = average distance to other points in X's cluster
- B = average distance to points in closest cluster
- S = (B - A) / max(A, B)



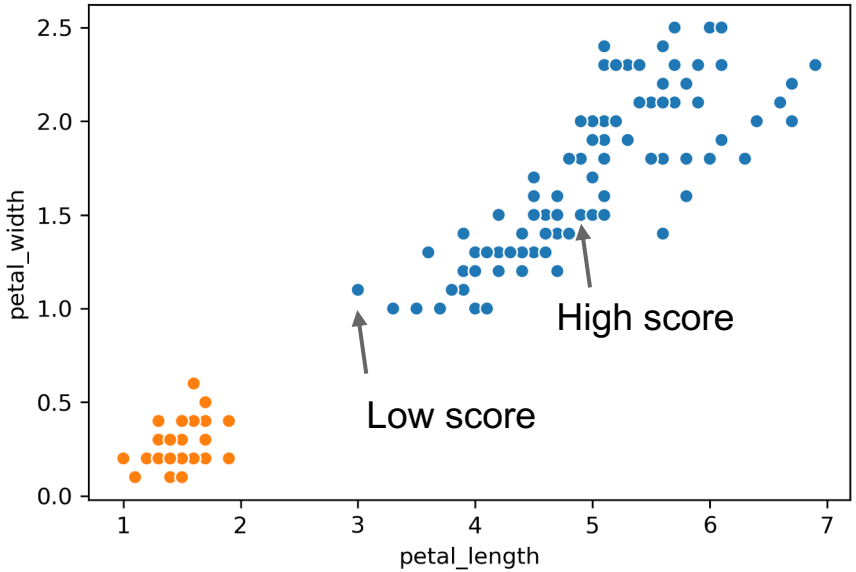
Can S be negative?

- Yes. Average distance to X's clustermates is larger than distance to the closest cluster

Silhouette Scores

For a data point X:

- A = average distance to other points in X's cluster
- B = average distance to points in closest cluster
- S = (B - A) / max(A, B)



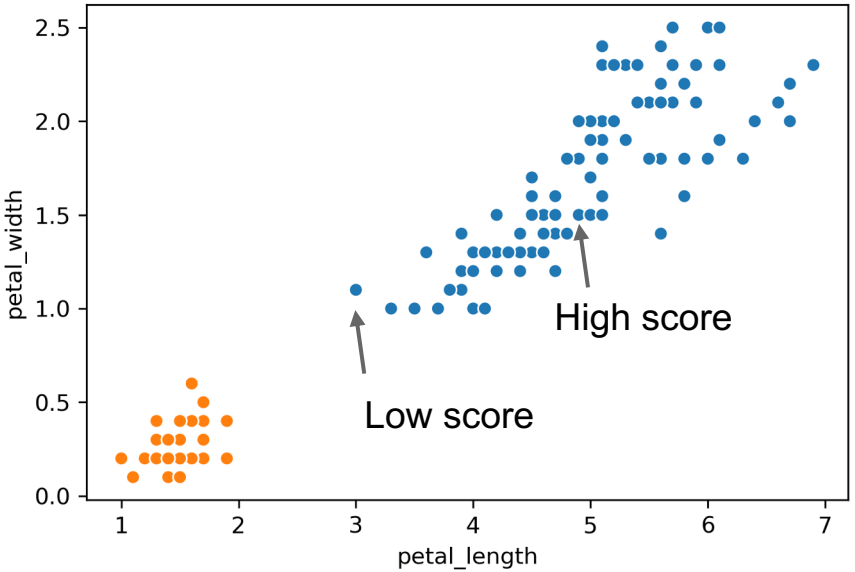
Can S be negative?

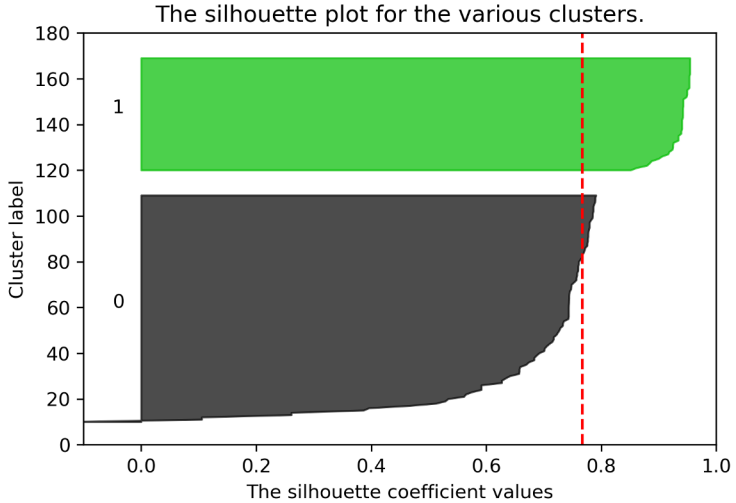
- Yes. Average distance to X's clustermates is larger than distance to the closest cluster
- Example: The "low score" point on the right has S = -0.13

Silhouette Plot

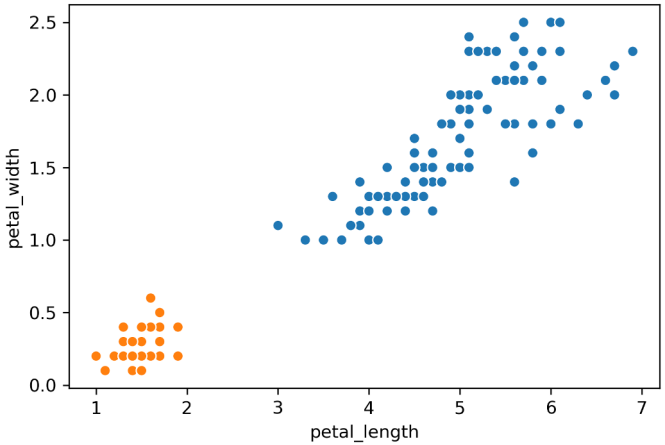
We can plot the Silhouette Scores for all of our data points

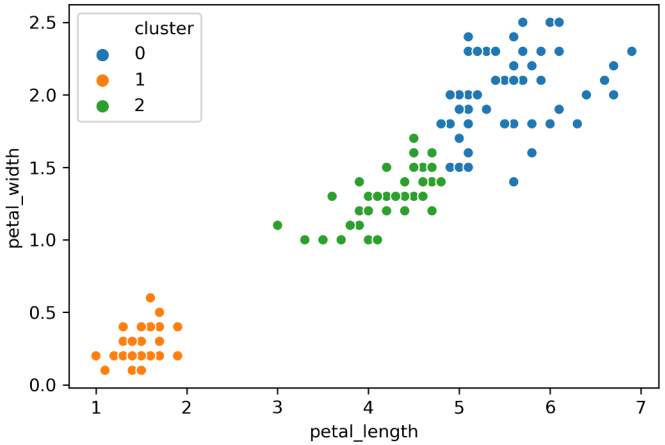
- Points with large silhouette widths are deeply embedded in their cluster
- Red dotted line shows the average





Silhouette Scores, K = 2 vs. K = 3





Average silhouette score is lower

Picking K: Real World Metrics

Sometimes you can rely on real world metrics to guide your choice of K

Perform 2 clusterings:

- Cluster heights and weights of customers with K = 3 to design Small, Medium, and Large shirts
- Cluster heights and weights of customers with K = 5 to design XS, S, M, L, and XL shirts

To pick K:

- Consider projected costs and sales for the 2 different Ks
- Pick the one that maximizes profit

Summary

Today we discussed a new machine learning goal: Clustering

Saw two solutions:

- K-Means
  - Tries to optimize a loss function called inertia -- no known algorithm to do this optimally
- Agglomerative

Our version of these algorithms required a hyperparameter k

- 4 ways to pick K: Intuitively, Elbow method, Silhouette scores, Harnessing real world metrics

Bigger Picture Summary

There are many Machine Learning problems

- Each can be addressed by many different solution techniques
- Each has many metrics for evaluating success / loss

Many solution technique can be used for multiple problem types

- Example: Linear models can be used for regression and classification

We've only scratched the surface. Haven't discussed many important ideas

- One hugely important solution technique: Neural Networks / Deep Learning
  - See other courses in CS and Data Science for more