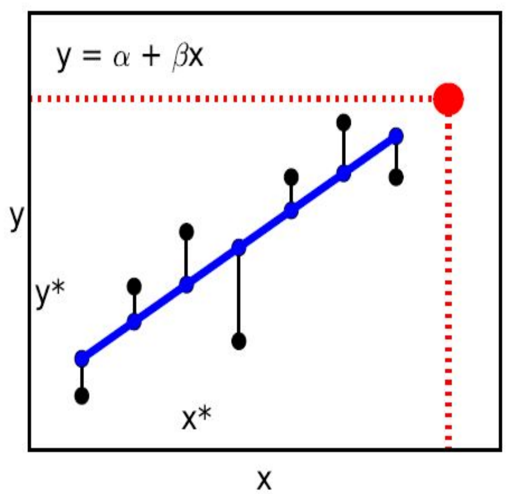
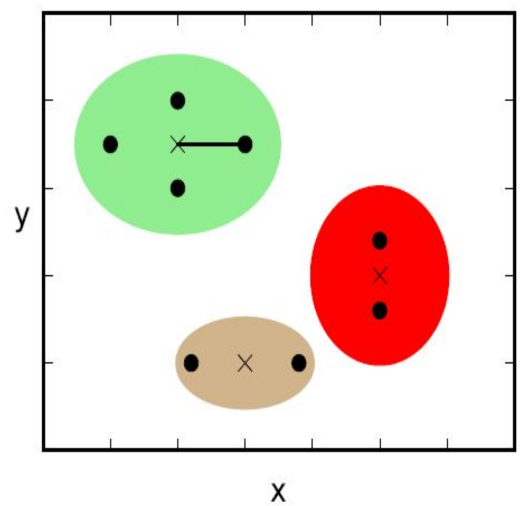
k-means

Clustering

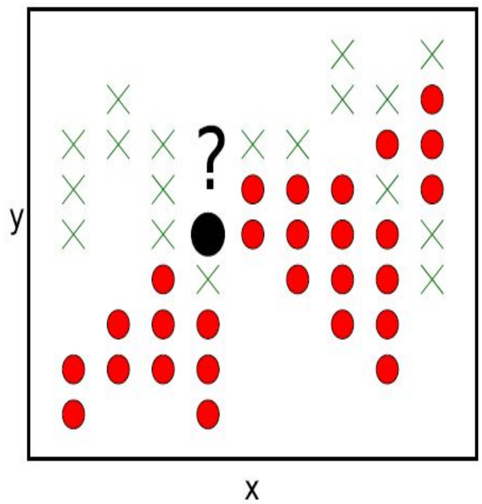
Prediction vs. Classification

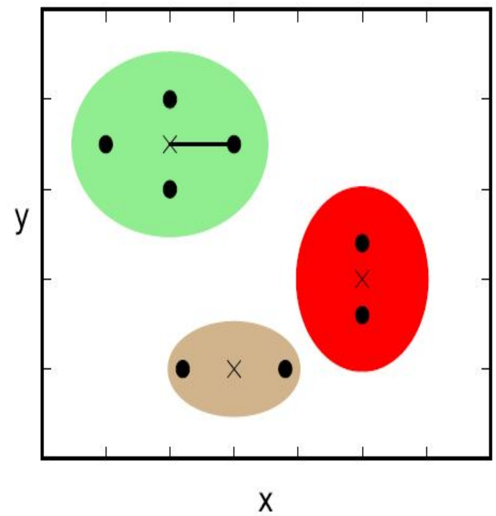




- two main goals in machine learning

Classification with Clustering



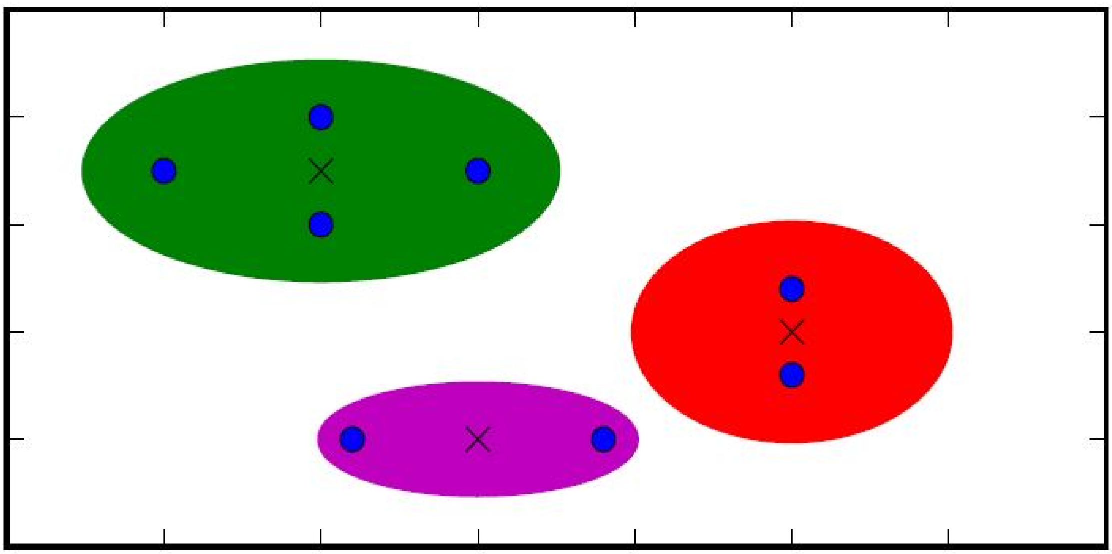


- split objects into clusters
- describe clusters by centroids
- need similarity (distance)
- most widely used: k-means

k-means Clustering

- unlabeled data
- assume distances between points
- goal: split data into k disjoint groups
- points in each cluster are closer to its center than to other centers
- computes allocation by iteration

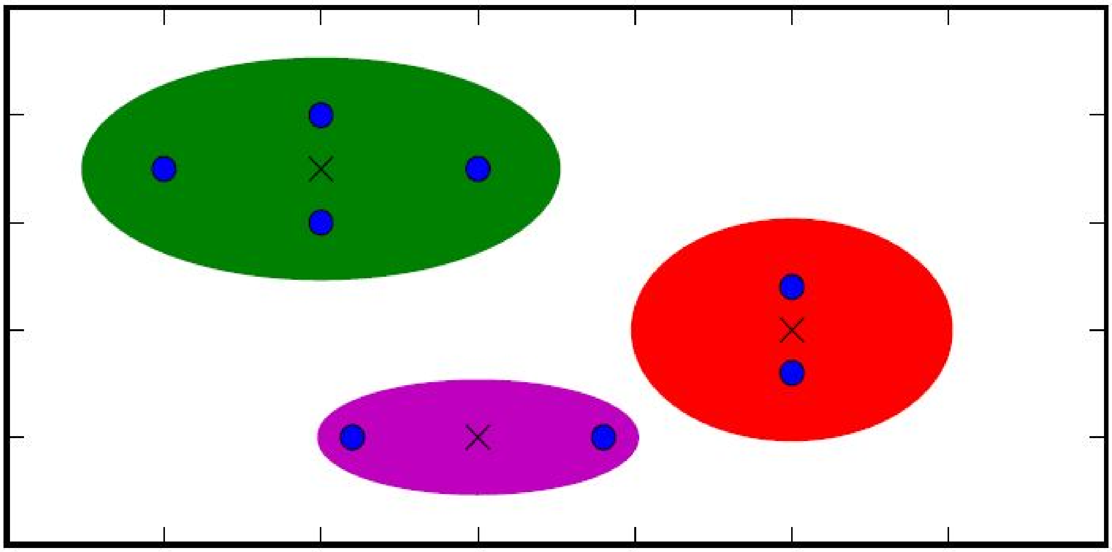
Formal Description



- N data points $x_1, \ldots, x_N$
- k cluster centers $C_1, \ldots, C_k$
- want to minimize "inertia"

$$E = \sum_{j=1}^{k} \left(\sum_{x \in C_i} ||x - \mu(C_i)||^2 \right)$$

Formal Description



- point x is assigned to a nearest cluster C (by distance to its cluster center $||x - C||$)
- default distance is (L_2) Euclidean

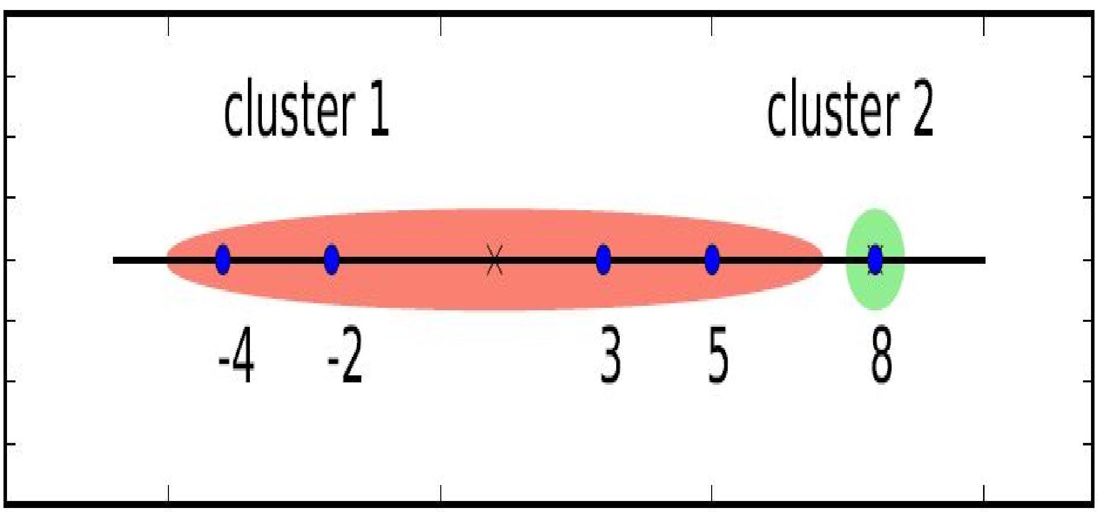
k-means Algorithm

1. choose k initial centroids
2. assign points to nearest centroids
3. recalculate centroids
4. repeat steps 2 and 3 until no more updates to clusters are possible

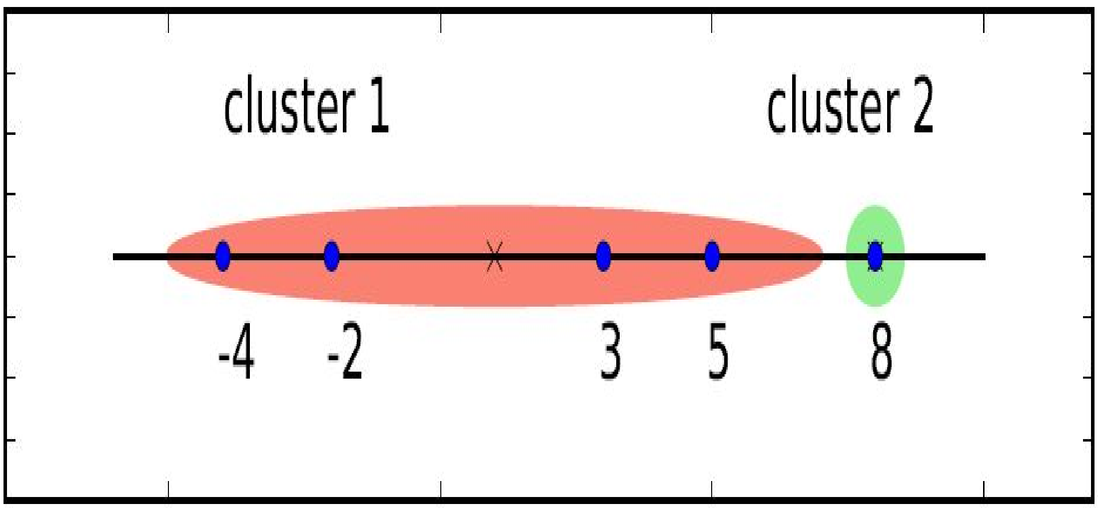
- k-means algorithm is guaranteed to converge

A Simple Example

- 5 points: -4,-2,3,5,8
- choose initial clusters:
  $C_1 = \{-4, -2, 3, 5\}, ;\ C_2 = \{8\}$



Initial Centers & Inertia

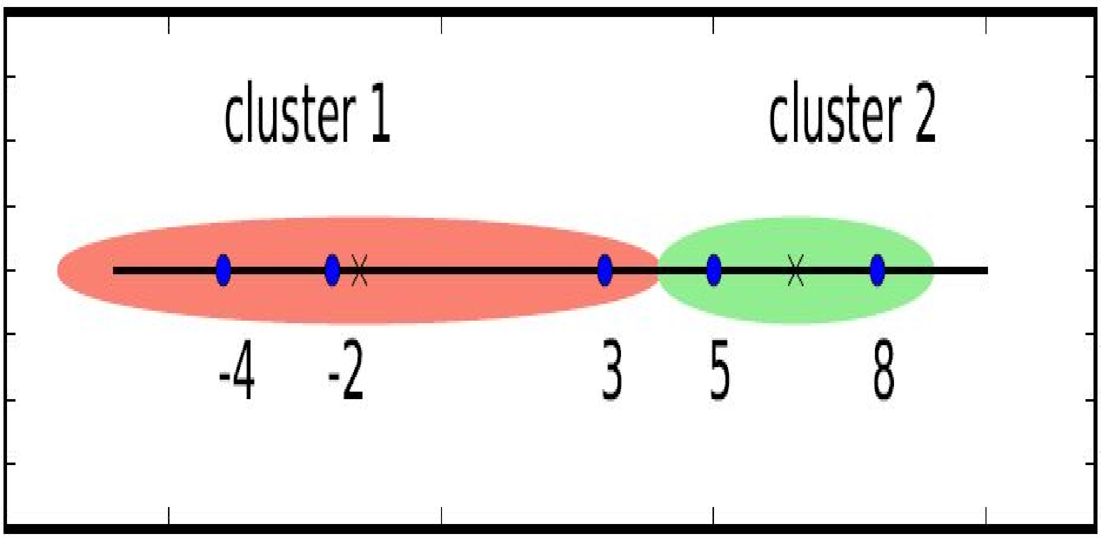


$$\mu_1 = \frac{(-4 - 2 + 3 + 5)}{4} = 0.5$$

$$\mu_2 = 8$$

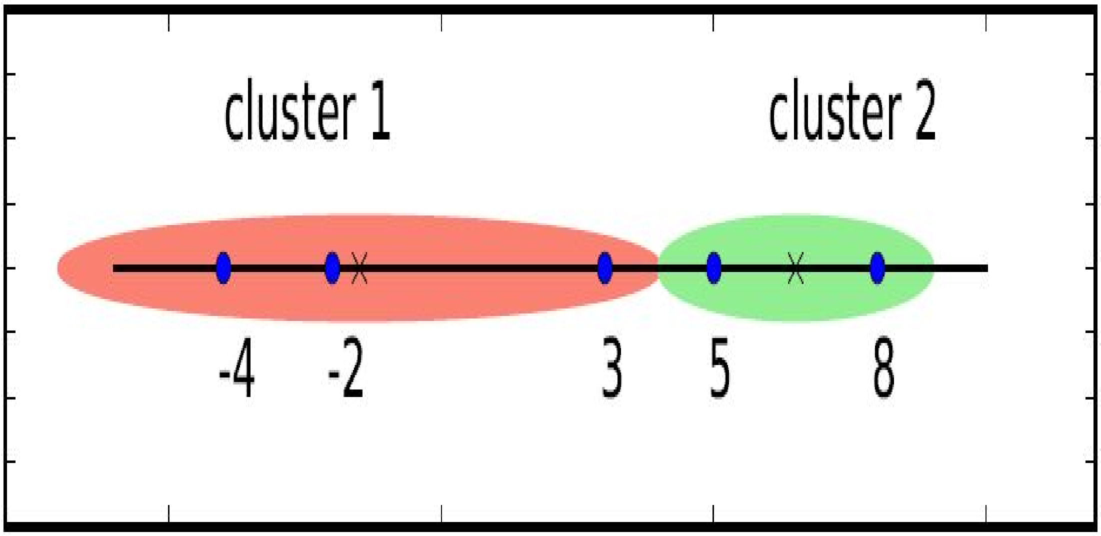
$$\begin{aligned} E = & \left[(-4 - 0.5)^2 + (-2 - 0.5)^2 \right. \\ & \left. + (3 - 0.5)^2 + (5 - 0.5)^2\right] \\ & + \left[(8 - 8)^2\right] = 53 \end{aligned}$$

Updating Clusters



- 5 is closer to 8 than to 0.5
- re-assign 5 to cluster 2

New Centers & Inertia

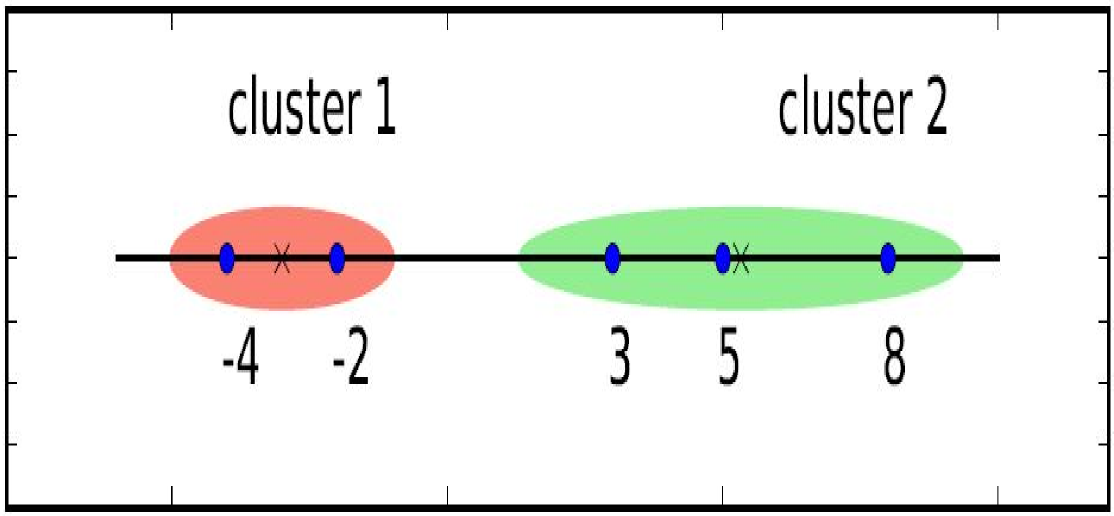


$$\mu_1 = \frac{(-4-2+3)}{3} = -1$$

$$\mu_2 = \frac{(5+8)}{2} = 6.5$$

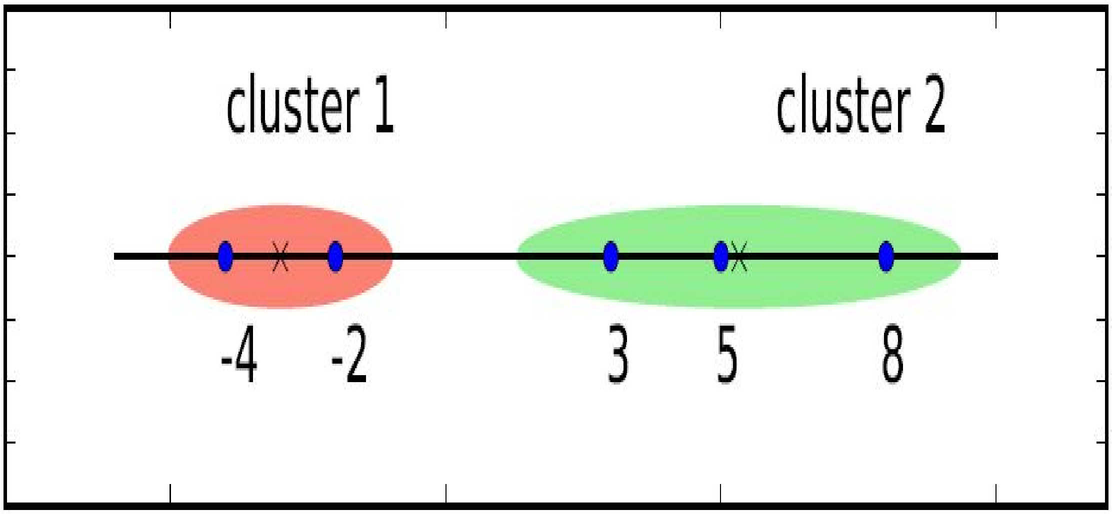
$$\begin{aligned} E = & \left[(-4-(-1))^2 + (-2-(-1))^2 \right. \\ & \left. + (3-(-1))^2\right] + \left[(5-6.5)^2\right] \\ & + \left[(8-6.5)^2\right] = 30.5 \end{aligned}$$

Updating Clusters



- 3 is closer to 6.5 than to -1
- re-assign 3 to cluster 2

New Centers & Inertia

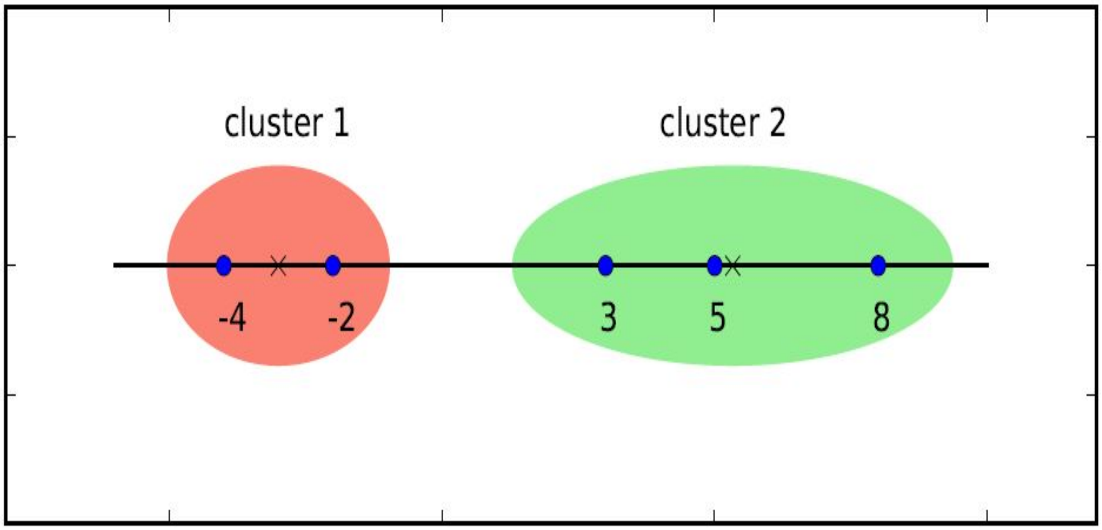


$$\mu_1 = \frac{(-4-2)}{2} = -3$$

$$\mu_2 = \frac{(3+5+8)}{3} = 5.3$$

$$\begin{aligned} E = &\left[(-4-(-3))^2 + (-2-(-3))^2\right] \\ &+ \left[(3-5.3)^2\right] + (5-5.3)^2] \\ &+ \left[(8-5.3)^2\right] = 14.7 \end{aligned}$$

Final Allocation



- no points need to be re-allocated
- this is final allocation

$$C_1 = \{-4, -2\} \quad C_2 = \{3, 5, 8\}$$

Notes on the Algorithm

- optimal solution is intractable
- allocation depends on the choice of initial clusters
- want moderate values of k
- k is typically chosen "visually" by the "knee" method

A Numerical Dataset

object x_i	Height (H)	Weight (W)	Foot (F)	Label (L)
x_1	5.00	100	6	green
x_2	5.50	150	8	green
x_3	5.33	130	7	green
x_4	5.75	150	9	green
x_5	6.00	180	13	red
x_6	5.92	190	11	red
x_7	5.58	170	12	red
x_8	5.92	165	10	red

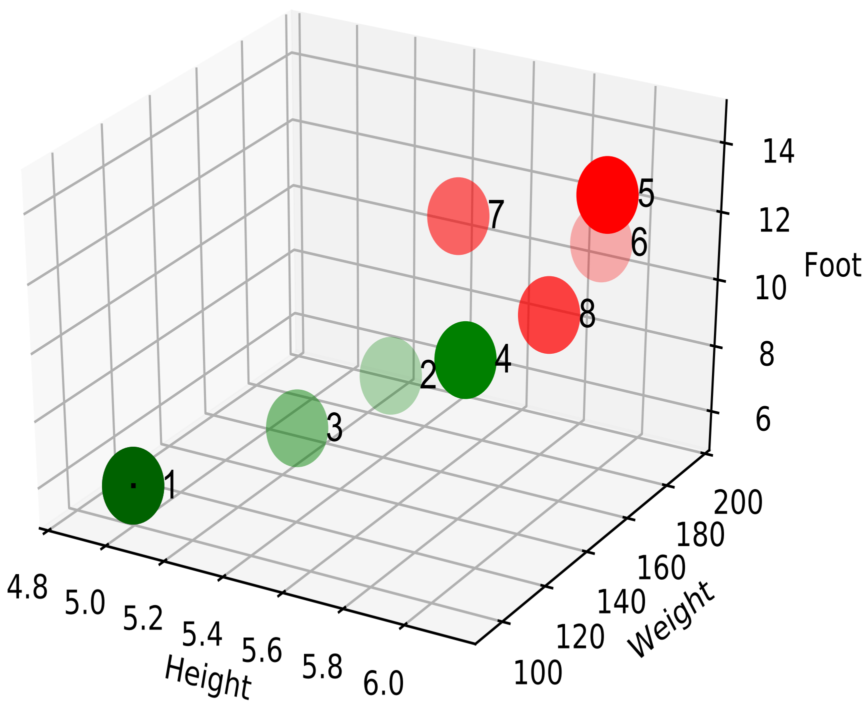
- $N = 8$ items
- $M = 3$ (unscaled) attributes

Code for the Dataset

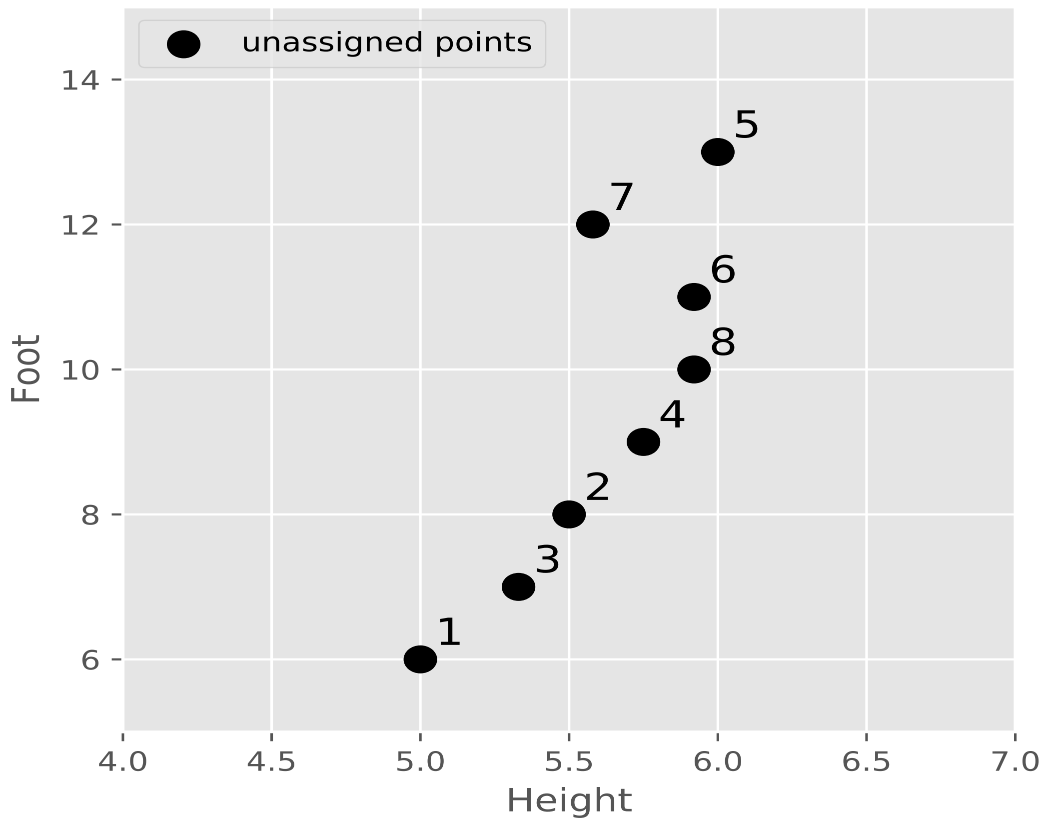
```
import pandas as pd
data = pd.DataFrame(
        {'id': [ 1,2,3,4,5,6,7,8],
        'Label': ['green','green','green','green',
                   'red','red','red','red'],
        'Height': [5, 5.5, 5.33, 5.75,
                   6.00, 5.92,  5.58, 5.92],
        'Weight': [100, 150, 130, 150,
                   180, 190, 170, 165],
        'Foot': [6, 8, 7, 9, 13, 11, 12, 10]},
         columns = ['id', 'Height', 'Weight',
                    'Foot', 'Label'] )
```

```
ipdb> data
  id Height Weight Foot Label
0  1   5.00    100    6 green
1  2   5.50    150    8 green
2  3   5.33    130    7 green
3  4   5.75    150    9 green
4  5   6.00    180   13   red
5  6   5.92    190   11   red
6  7   5.58    170   12   red
7  8   5.92    165   10   red
```

A Dataset Illustration

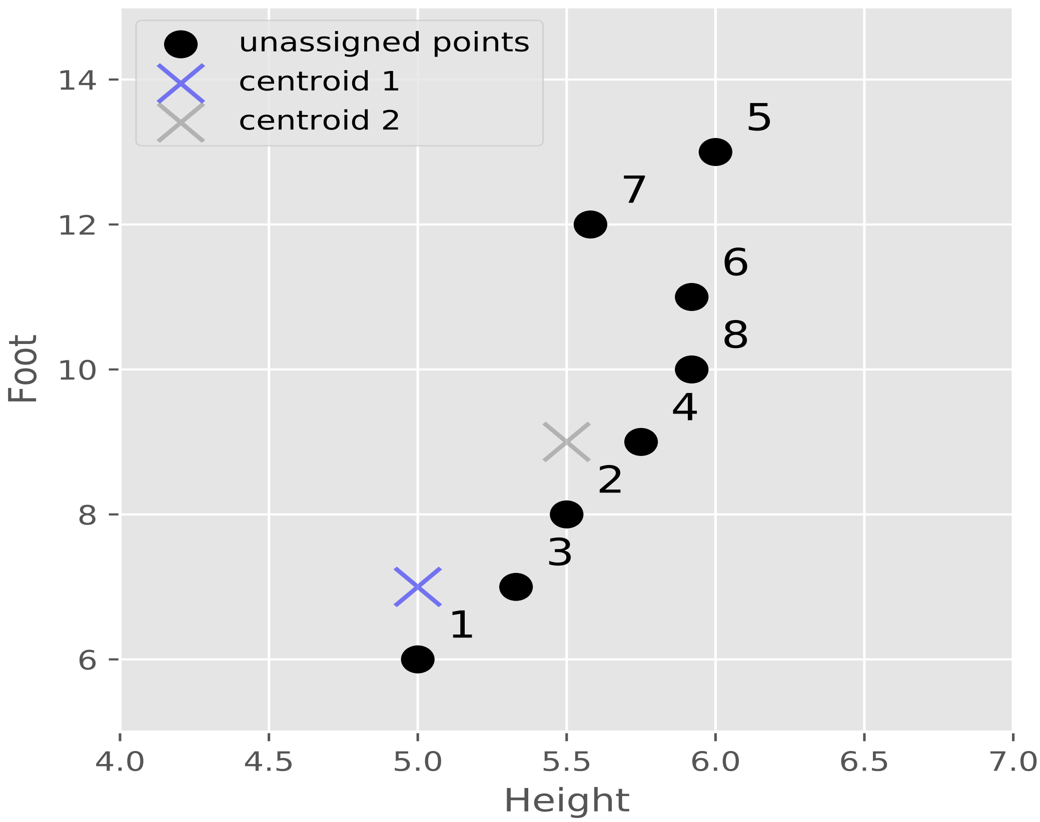


Initial Dataset



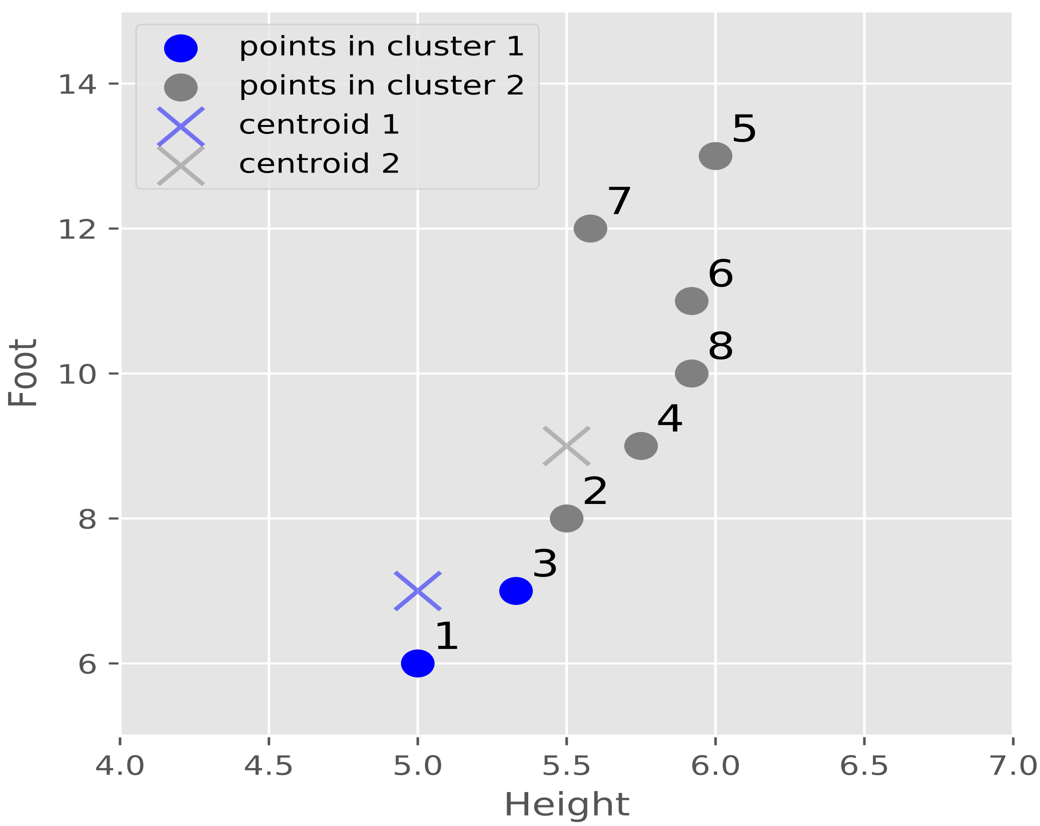
- points are not labelled
- want to split into $k = 2$ clusters

Initial Centroids



- difficult problem to choose
- solution may depend on this choice

Initial Assignment



- can assign at random

$C_1 = (5.0, 7.0)$ and $C_2 = (5.5, 9.0)$

A Initial Assignment

$$C_1 = (5.0, 7.0) \text{ and } C_2 = (5.5, 9.0)$$

x_i	Height (H)	Foot (F)	Cluster C_k
x_1	5.00	6	1
x_2	5.50	8	2
x_3	5.33	7	1
x_4	5.75	9	2
x_5	6.00	13	2
x_6	5.92	11	2
x_7	5.58	12	2
x_8	5.92	10	2

A Initial Assignment

x_i	H	F	C	$d(x_i, C_1)$	$d(x_i, C_2)$	C'
x_1	5.00	6	1	1.00	3.04	1
x_2	5.50	8	2	1.12	1.00	2
x_3	5.33	7	1	0.33	2.01	1
x_4	5.75	9	2	2.13	0.25	2
x_5	6.00	13	2	6.08	4.03	2
x_6	5.92	11	2	4.10	2.04	2
x_7	5.58	12	2	5.03	3.00	2
x_8	5.92	10	2	3.14	1.08	2

Updating Centroids for Initial Assignment

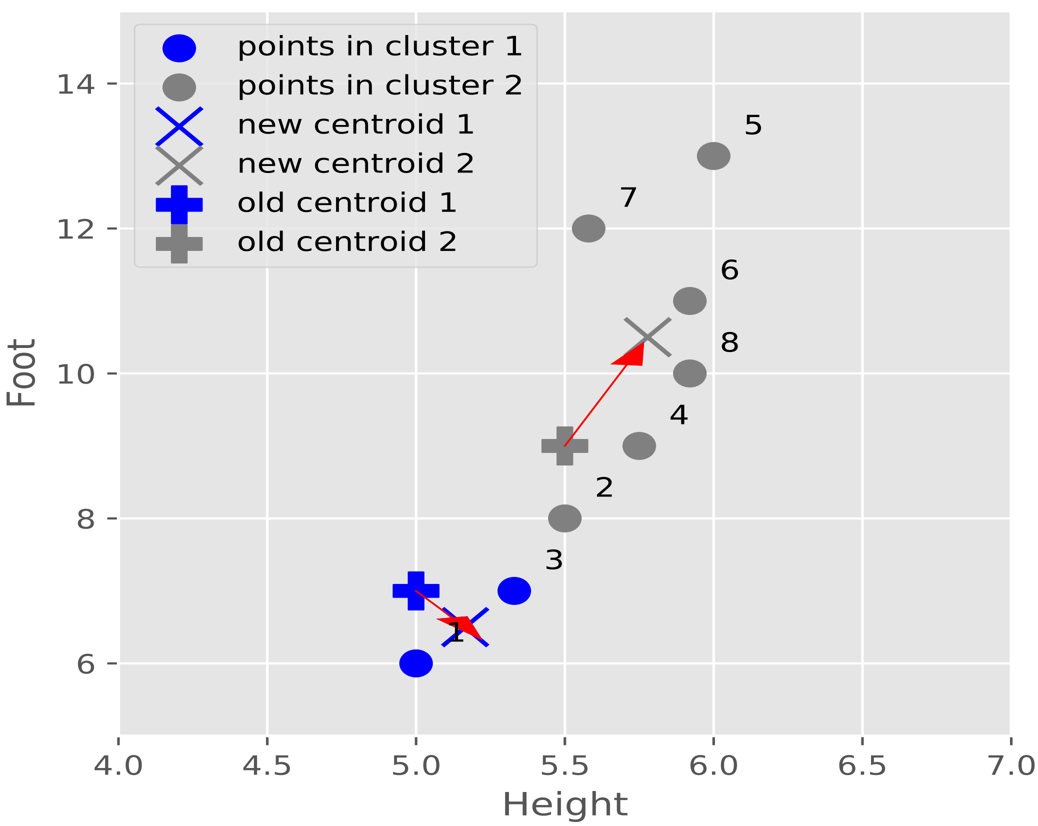
- cluster 1 has two points: x_1, x_3

$$C_1 = \frac{(x_1 + x_3)}{2} = \frac{(5,6) + (5.33,7)}{2}$$
$$= (5.17, 6.5)$$

- cluster 2 has 6 points: $x_2, x_4, x_5, x_6, x_7, x_8$

$$C_2 = \frac{(x_2 + x_4 + x_5 + x_6 + x_7 + x_8)}{6}$$
$$= (5.78, 10.5)$$

Updating Centroids



$$C_1: \ (5.0, 7) \mapsto (5.17, 6.5)$$
$$C_2: \ (5.5, 9) \mapsto (5.78, 10.5)$$

How to re-assign Points to Clusters

- $C_1 = (5.17, 6.5)$ and $C_2 = (5.78, 10.5)$
- for each x_i, assign it to cluster C_k with minimum $d(x_i, C_k)$
- for point $x_1 = (5, 6)$:

$$d(x_1, C_1) = \sqrt{(5 - 5.17)^2 + (6 - 6.5)^2} = 0.53$$
$$d(x_1, C_2) = \sqrt{(5 - 5.78)^2 + (6 - 10.5)^2} = 4.57$$

$\Rightarrow x_1$ remains in cluster 1

- for point $x_2 = (5.5, 8)$:

$$d(x_2, C_1) = \sqrt{(5.5 - 5.17)^2 + (8 - 6.5)^2} = 1.54$$
$$d(x_2, C_2) = \sqrt{(5.5 - 5.78)^2 + (8 - 10.5)^2} = 2.52$$

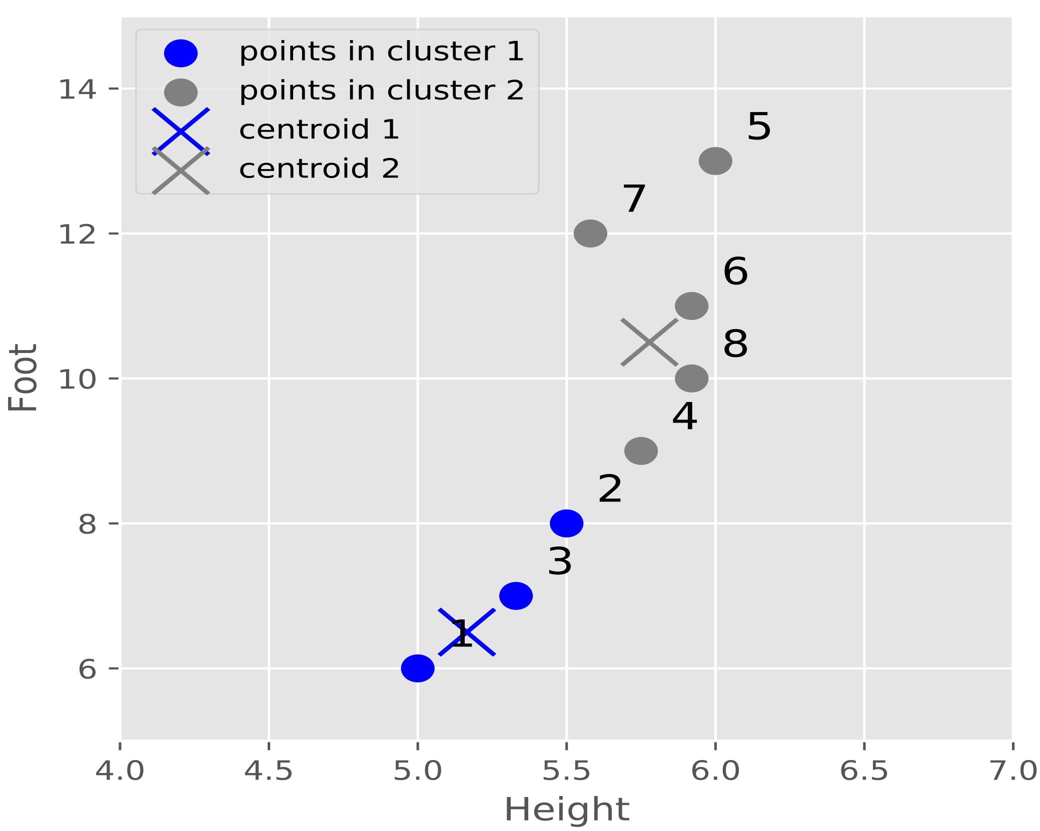
$\Rightarrow x_2$ is re-assigned to cluster 1

Re-assigning Points to Clusters

x_i	H	F	C	$d(x_i, C_1)$	$d(x_i, C_2)$	C'	updated
x_1	5.00	6	1	0.53	4.57	1	no
x_2	5.50	8	2	1.54	2.52	1	yes
x_3	5.33	7	1	0.53	3.53	1	no
x_4	5.75	9	2	2.57	1.50	2	no
x_5	6.00	13	2	6.55	2.51	2	no
x_6	5.92	11	2	4.56	0.52	2	no
x_7	5.58	12	2	5.52	1.51	2	no
x_8	5.92	10	2	3.58	0.52	2	no

- if some points are re-assigned to new cluster(s) then we need to compute new centroids

New Assignment



- compute distances to each center
- re-assign points to clusters

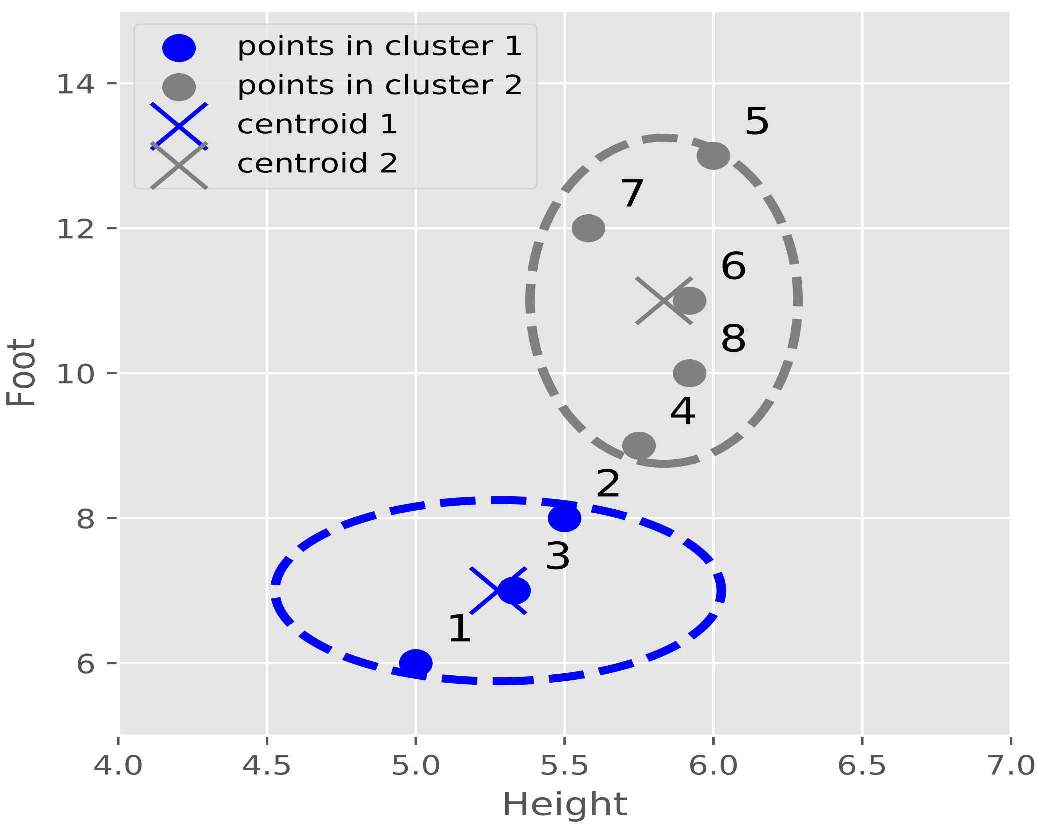
Final Assignment

- $C_1 = (5.28, 7), C_2 = (5.83, 11)$

x_i	H	F	C	$d(x_i, C_1)$	$d(x_i, C_2)$	C'	updated
x_1	5.00	6	1	1.04	5.07	1	no
x_2	5.50	8	1	1.02	3.02	1	no
x_3	5.33	7	1	0.05	4.03	1	no
x_4	5.75	9	2	2.06	2.00	2	no
x_5	6.00	13	2	6.04	2.01	2	no
x_6	5.92	11	2	4.05	0.09	2	no
x_7	5.58	12	2	5.01	1.03	2	no
x_8	5.92	10	2	3.07	1.00	2	no

- points cannot be updated
- this is the final clustering

Final Assignment



- points allocated to right cluster
- no more updates are possible

k-means in Python

- many parameters
- can specify number of clusters and initial centroids
- computes the cluster labels, centroids and inertia
- can choose best algorithm out of n_init different initalizations

Python Code

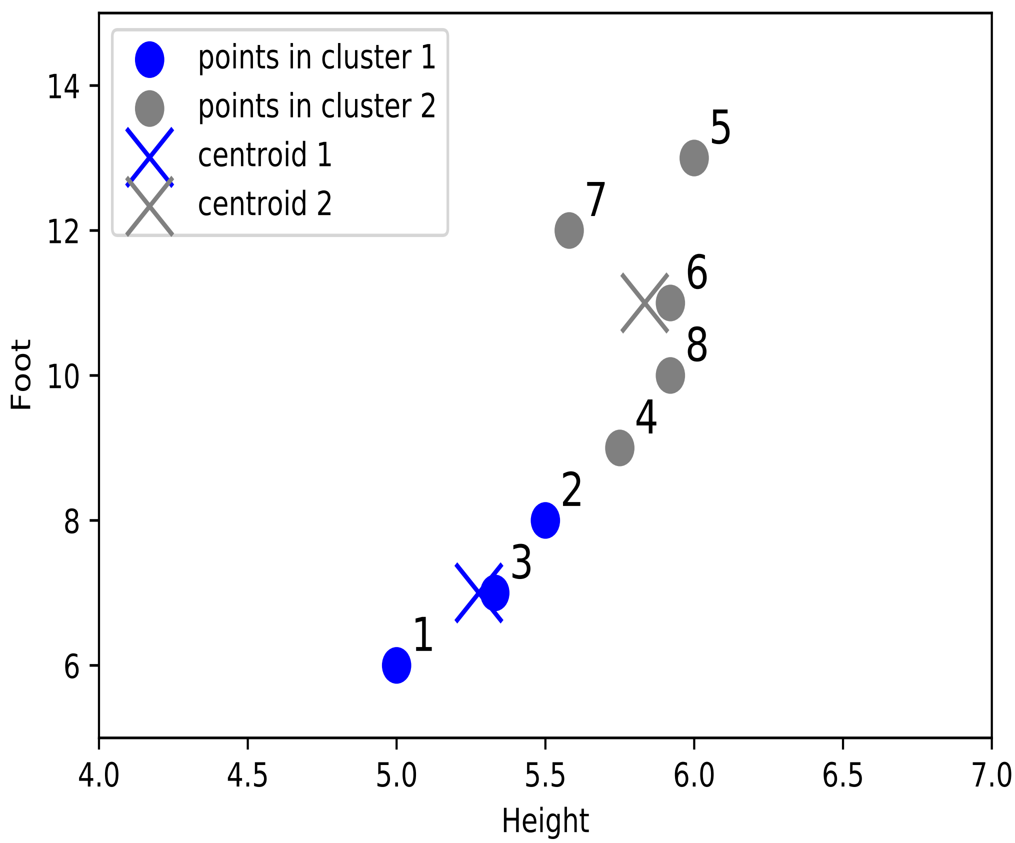
```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans

data = pd.DataFrame(
    {'id': [ 1,2,3,4,5,6,7,8],
     'Label': ['green', 'green', 'green', 'green', 'red',
                                           'red', 'red', 'red'],
     'Height': [5, 5.5, 5.33, 5.75, 6.00, 5.92,  5.58, 5.92],
     'Weight': [100, 150, 130, 150, 180, 190, 170, 165],
     'Foot': [6, 8, 7, 9, 13, 11, 12, 10]},
      columns = ['id', 'Height', 'Weight', 'Foot', 'Label']
        )
init_centers = np.array([[5.0,7.0],[5.5, 9.0]])
colmap = {0: 'blue', 1: 'grey'}
x = data[['Height', 'Foot']].values
kmeans_classifier = KMeans(n_clusters=n_clusters,init=init_centers)
y_means=kmeans_classifier.fit_predict(x)
```

```
>>> centroids
array([[  5.28, 7.0],
       [  5.83, 11.0]])
>>> y_kmeans
array([0, 0, 0, 1, 1, 1, 1, 1])
```

Python Code for Plotting

```
import matplotlib.pyplot as plt

fig = plt.figure()

for i in range(n_clusters):
    new_df = data[data['cluster']==i]
    plt.scatter(new_df['Height'],new_df['Foot'], color=colmap[i],
                s=100, label='points in cluster ' + str(i+1))

for i in range(n_clusters):
    plt.scatter(centroids[i][0], centroids[i][1], color=colmap[i],
                marker='x', s=300,label='centroid ' + str(i+1))

for i in range(len(data)):
    x_text = data['Height'].iloc[i] + 0.05
    y_text = data['Foot'].iloc[i] + 0.2
    id_text = data['id'].iloc[i]
    plt.text(x_text, y_text, str(id_text), fontsize=14)

plt.legend(loc='upper left')
plt.xlim(4, 7)
plt.ylim(5, 15)
plt.xlabel('Height')
plt.ylabel('Foot')
plt.show()
```

Python Code for Plotting



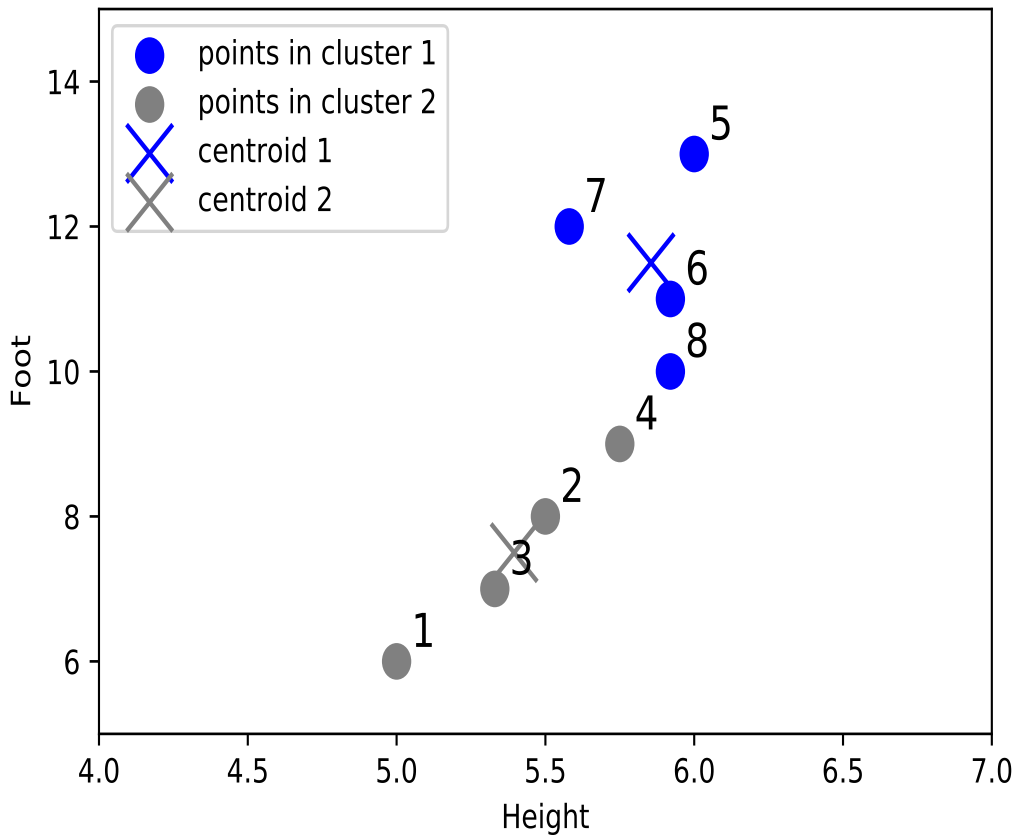
Python Code

- random initial centroids

```
import pandas as pd
from sklearn.cluster import KMeans

data = pd.DataFrame(
    {'id': [ 1,2,3,4,5,6,7,8],
     'Label': ['green', 'green', 'green', 'green', 'red',
                                                 'red', 'red', 'red'],
     'Height': [5, 5.5, 5.33, 5.75, 6.00, 5.92,  5.58, 5.92],
     'Weight': [100, 150, 130, 150, 180, 190, 170, 165],
     'Foot': [6, 8, 7, 9, 13, 11, 12, 10]},
     columns = ['id', 'Height', 'Weight', 'Foot', 'Label']
        )
colmap = {0: 'blue', 1: 'grey'}
x = data[['Height', 'Foot']].values
kmeans_classifier = KMeans(n_clusters=n_clusters)
y_means=kmeans_classifier.fit_predict(x)
centroids = kmeans_classifier.cluster_centers_
```

```
>>> centroids
array([[  5.395,   7.5  ],
       [  5.855,  11.5  ]])
>>> y_kmeans
array([0, 0, 0, 0, 1, 1, 1, 1])
```

Results for Random Initialization



Python Code for IRIS

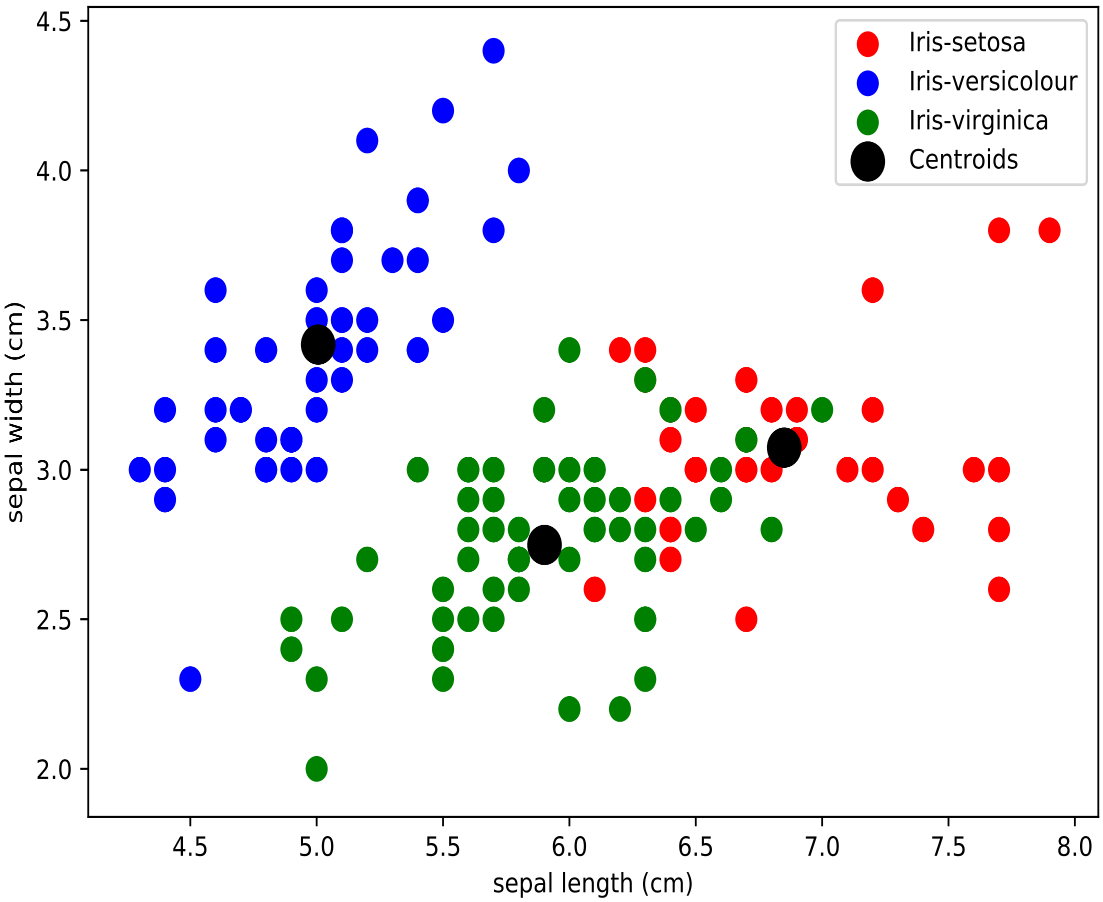
```python
from sklearn import datasets
from sklearn.cluster import KMeans

iris = datasets.load_iris()
x = iris.data

kmeans_classifier = KMeans(n_clusters=3)
y_kmeans = kmeans_classifier.fit_predict(x)
centroids = kmeans_classifier.cluster_centers_
```

```
>>> centroids
array([[ 5.90, 2.75, 4.39, 1.43],
       [ 5.01, 3.42, 1.46, 0.24],
       [ 6.85, 3.07, 5.74, 2.07]])
```

Python Code for Plotting

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots(1,figsize=(7,5))
plt.scatter(x[y_kmeans == 0, 0], x[y_kmeans == 0, 1],
                s = 75, c = 'red', label = 'Iris-setosa')
plt.scatter(x[y_kmeans == 1, 0], x[y_kmeans == 1, 1],
                s = 75, c = 'blue', label = 'Iris-versicolour')
plt.scatter(x[y_kmeans == 2, 0], x[y_kmeans == 2, 1],
                s = 75, c = 'green', label = 'Iris-virginica')
plt.scatter(centroids[:, 0], centroids[:,1],
                s = 200, c = 'black', label = 'Centroids')

x_label = iris.feature_names[0]
y_label = iris.feature_names[1]

plt.legend()
plt.xlabel(x_label)
plt.ylabel(y_label)
plt.tight_layout()
plt.show()
```

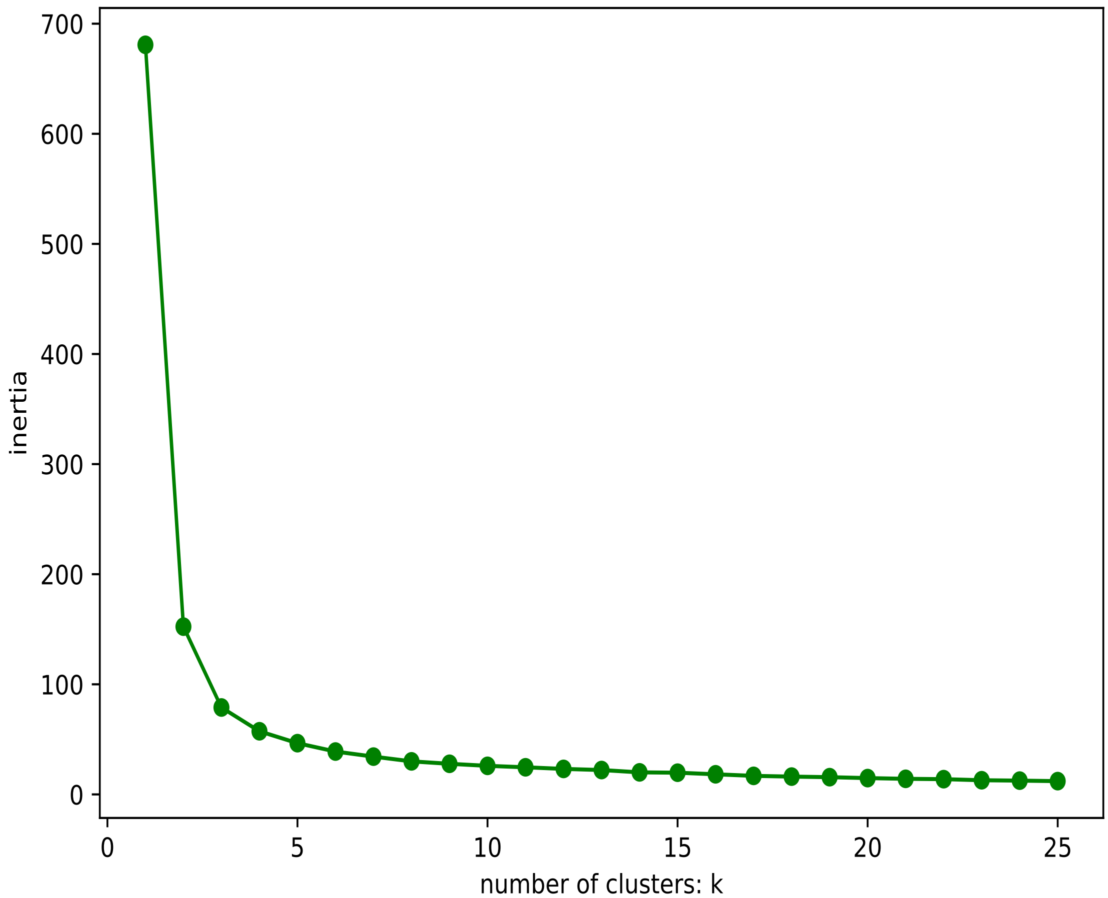
k-means for IRIS



Code to Compute k

```
from sklearn import datasets
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

iris = datasets.load_iris()

x = iris.data
inertia_list = []
for k in range(1,26):
    kmeans_classifier = KMeans(n_clusters=k)
    y_kmeans = kmeans_classifier.fit_predict(x)
    inertia = kmeans_classifier.inertia_
    inertia_list.append(inertia)

fig, ax = plt.subplots(1,figsize=(7,5))
plt.plot(range(1, 26), inertia_list, marker='o',
         color='green')

plt.legend()
plt.xlabel('number of clusters: k')
plt.ylabel('inertia')
plt.tight_layout()

plt.show()
```

Computing k for IRIS



- choose k visually - no more significant decline in inertia