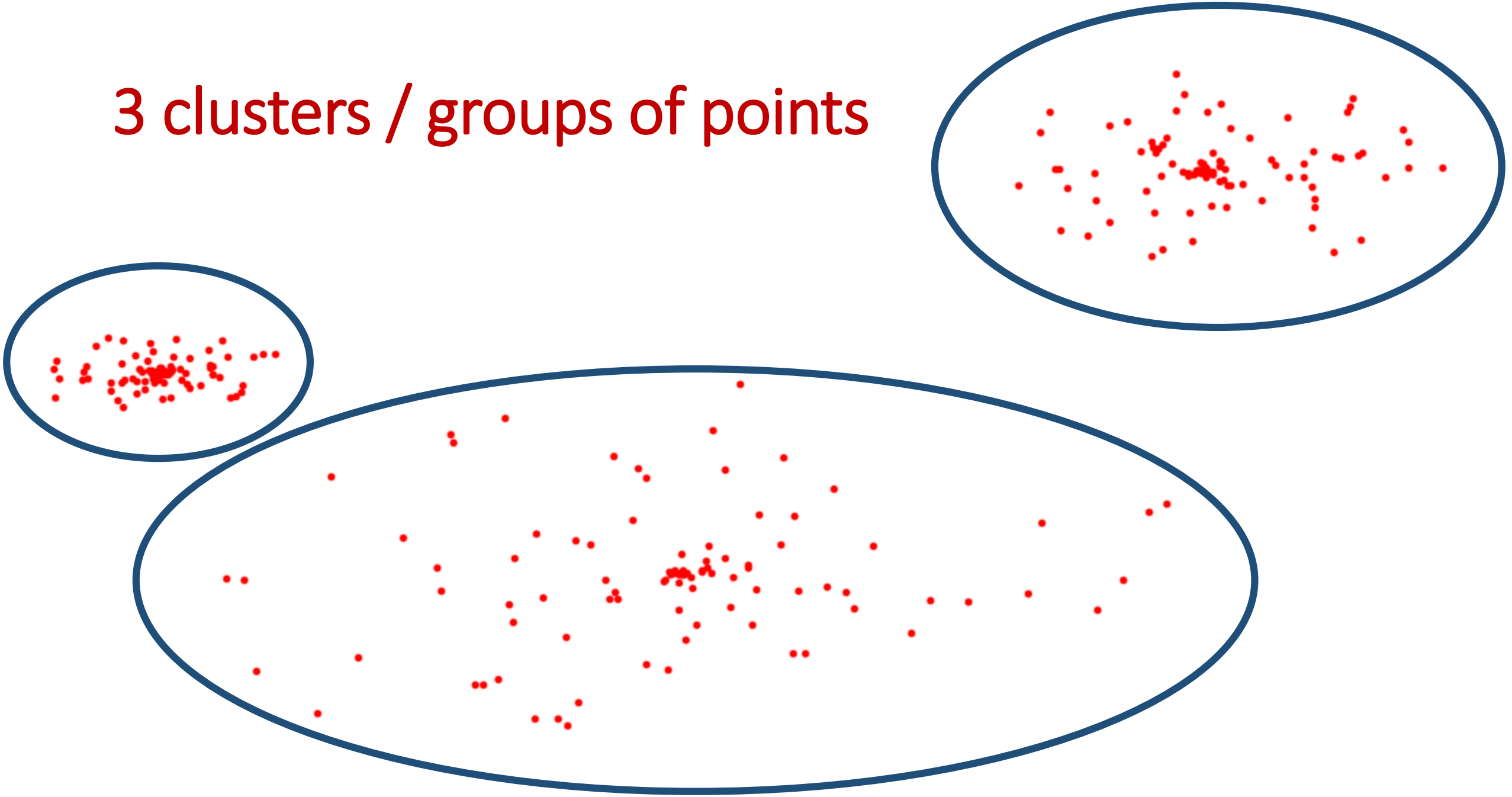
Clustering

- k-means
- scipy.cluster.vq.kmeans

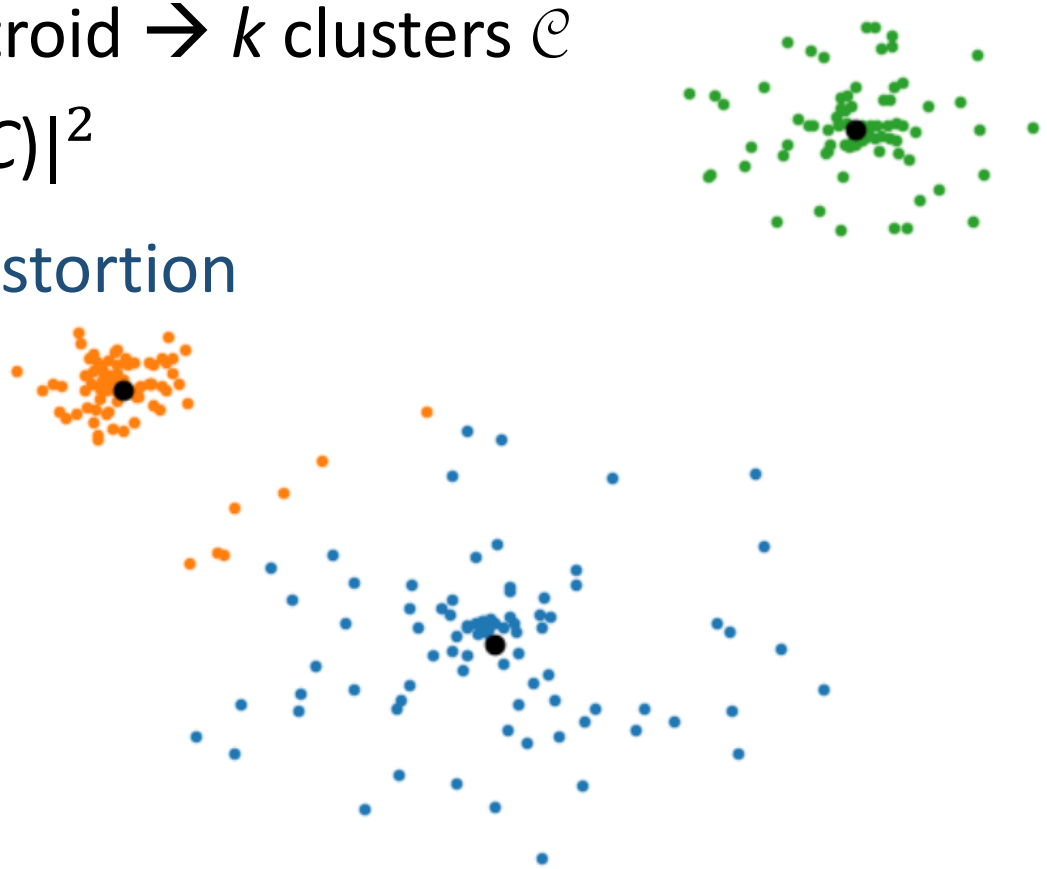
3 clusters / groups of points



Clustering = Optimization problem

Example: k-means

- Find *k* points *centroids*
- Assign each input point to nearest centroid → *k* clusters $\mathcal{C}$
- distortion = $\sum_{C \in \mathcal{C}} \sum_{p \in C} |p - \text{centroid}(C)|^2$
- Goal : Find *k* centroids that minimize distortion



k-means for $k = 1$

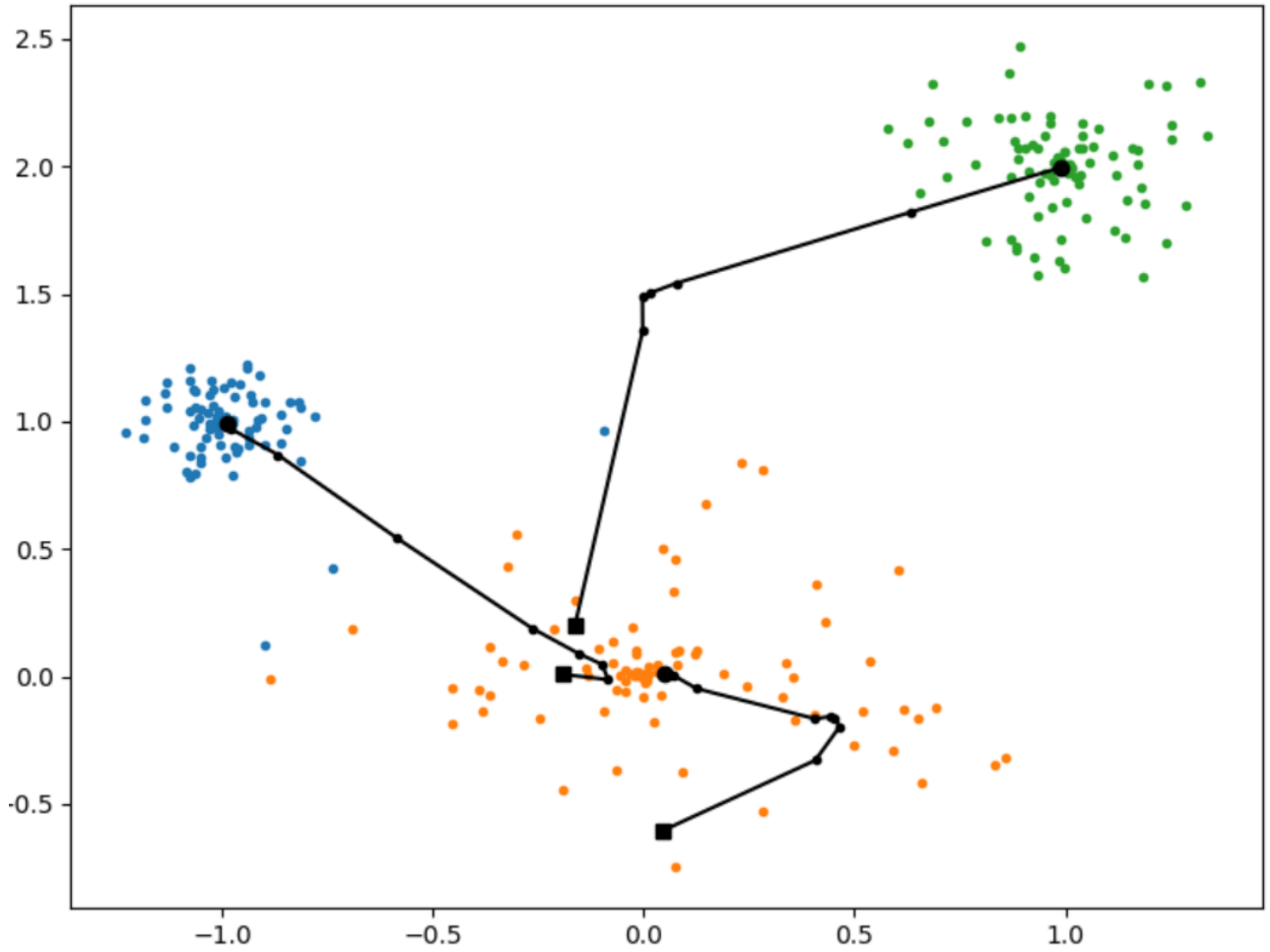
- Let the centroid point c for a point set C be the point minimizing the

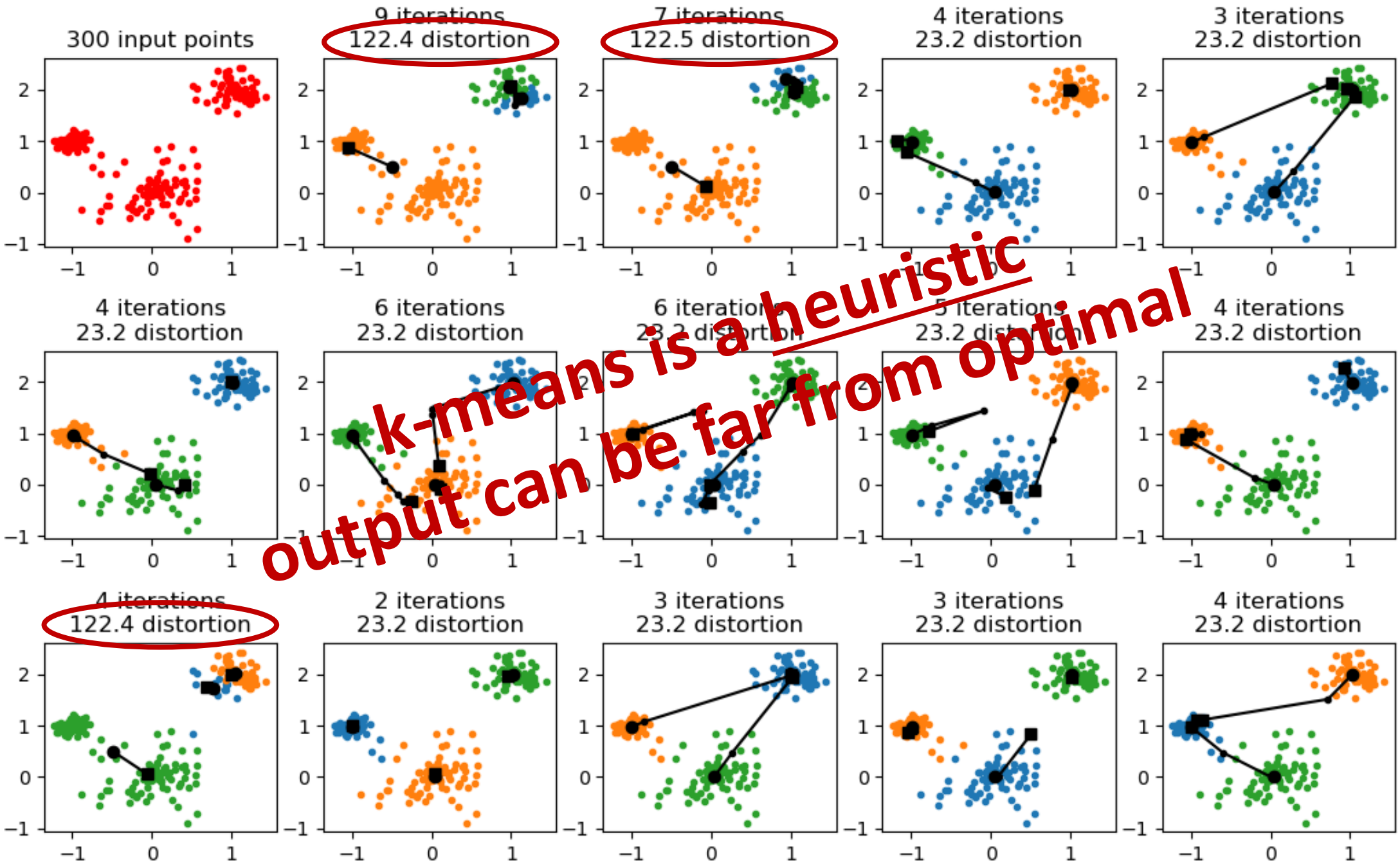
$$\text{distortion} = \sum_{p \in C} |p - c|^2$$

- Theorem $c = \text{average}(C)$

k-means - Lloyd's method (pseudo code)

```
centroids = k distinct random input points
while centroids change:
    create clusters C by assigning points to the nearest centroid
    centroids = average of each cluster
```




300 input points
9 iterations
122.4 distortion
7 iterations
122.5 distortion
4 iterations
23.2 distortion
3 iterations
23.2 distortion
4 iterations
23.2 distortion
6 iterations
23.2 distortion
6 iterations
23.2 distortion
5 iterations
23.2 distortion
4 iterations
23.2 distortion
4 iterations
122.4 distortion
2 iterations
23.2 distortion
3 iterations
23.2 distortion
3 iterations
23.2 distortion
4 iterations
23.2 distortion
k-means is a heuristic
output can be far from optimal

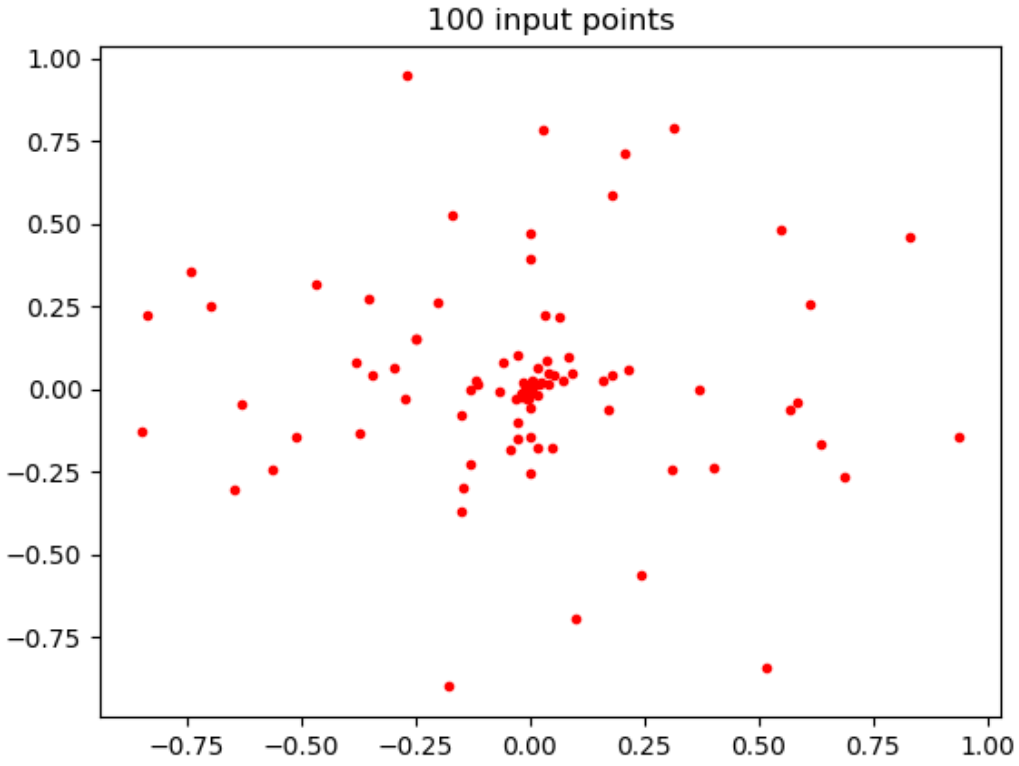
Generating random points (just one random approach)

k_means.py

```
from random import random
from math import pi, cos, sin

def random_point(x, y, radius):
    angle = 2 * pi * random()
    r = radius * random() ** 2
    return x + r * cos(angle), y + r * sin(angle)

def random_points(n, x, y, radius):
    for _ in range(n):
        yield random_point(x, y, radius)
```



k-means

k_means.py

```
from random import sample
from numpy import argmin, mean

def k_means(points, k):
    centroid = sample(points, k)
    centroids = [ centroid ]

    while True:
        clusters = [[] for _ in centroid]
        for p in points:
            i = argmin([dist(p, c) for c in centroid])
            clusters[i].append(p)

        centroid = [tuple(map(mean, zip(*c))) for c in clusters]

        if centroid == centroids[-1]:
            break

        centroids.append(centroid)
        if min(len(c) for c in clusters) == 0:
            print("Not good - empty cluster")
            break

    return clusters
```

k-mean limitations

- Can easily converge to a solution far from a global minimum
  - Solution – try several times and take the best
    (possibly since we can measure the quality (= distortion) of a solution)

- Clusters can become empty
  - Solution – discard and restart / take a random point out as a new centroid / take point furthest away from existing centroids / ....

- Sensitive to the scales of the different dimensions
  - Solution – apply some kind of initial normalization of coordinates

k-means - better bounds

- The k-means++ algorithm achieves an expected guarantee to be at most a factor 8(ln *k* +2) from the optimal [Vassilvitskii & Arthur]

- There exist polynomial time approximation schemes that find a solution that is guaranteed 1 + ε of the optiomal (but running time exponential in *k* and dimension of points) [Har Peled et al.]

- **In practice: A heuristic is most often the algorithm of choice**

scipy.cluster.vq.kmeans

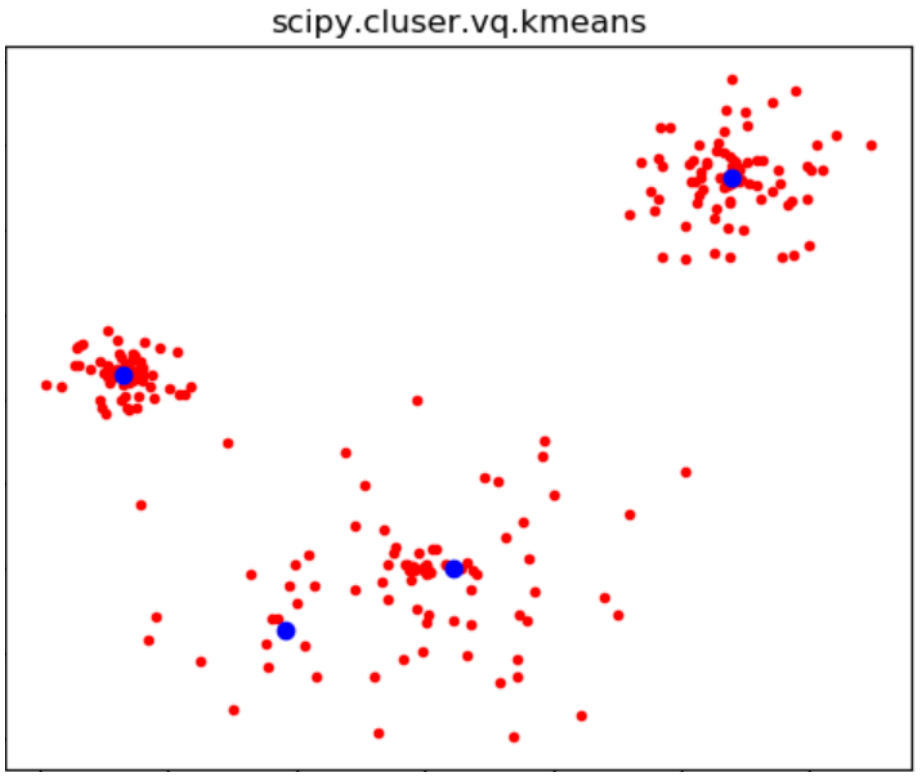
k_means.py

```
from scipy.cluster.vq import kmeans, whiten
import matplotlib.pyplot as plt

points = whiten(points)  # normalize variance of points

plt.plot(*zip(*points), 'r.')
plt.plot(*zip(*kmeans(points, K)[0]), "bo")
plt.title("scipy.cluster.vq.kmeans")
```

Note: According to the documentation "`whiten` must be called prior to passing an observation matrix to `kmeans`"



docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.vq.kmeans.html

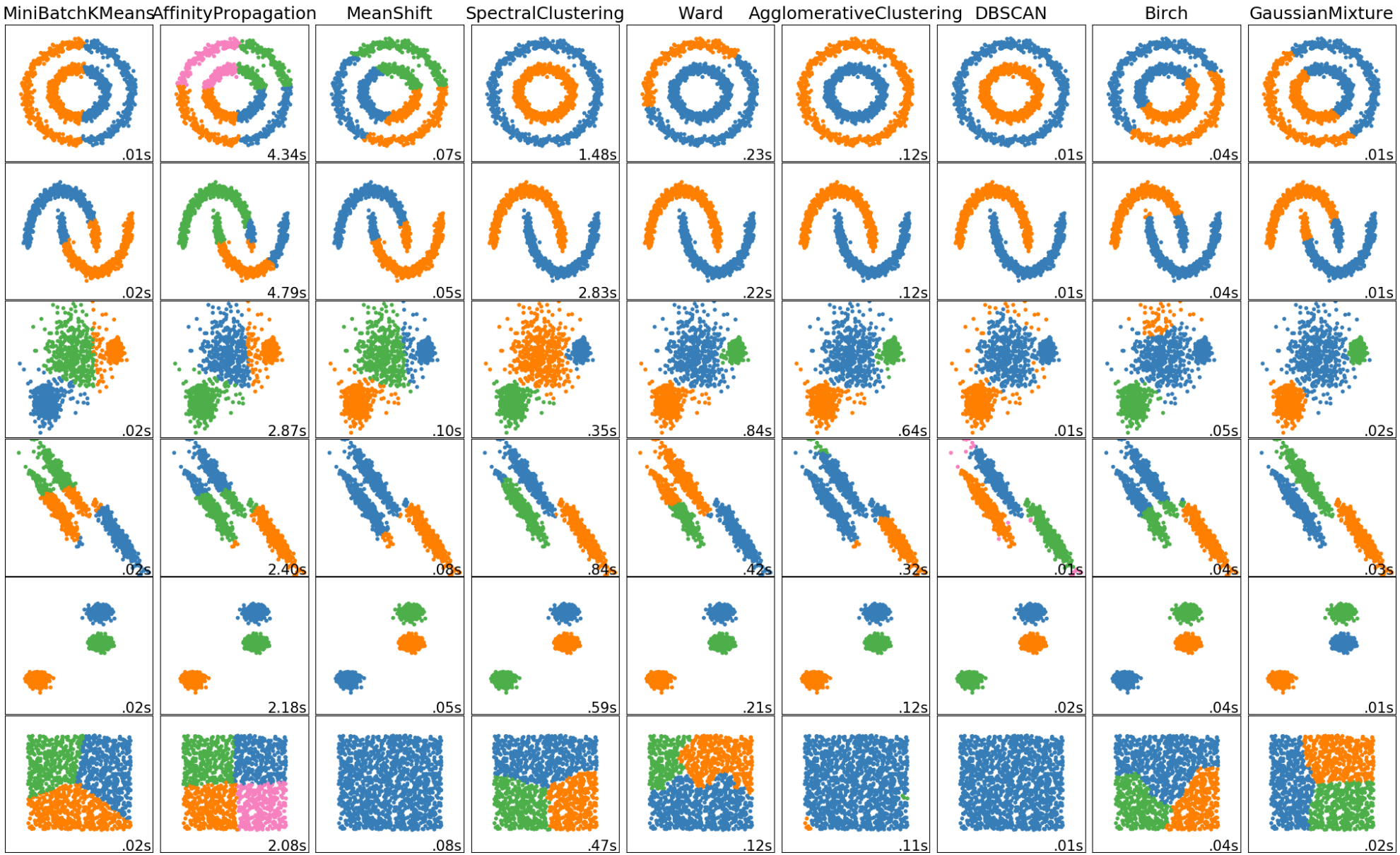
scipy.cluster.vq.whiten

- Normalizes / scales each dimension to have unit variance 1.0

$$\mathrm{Var}(X) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2 \qquad \mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.vq.kmeans.html

Other Python clustering methods - `sklearn.cluster`



http://scikit-learn.org/stable/modules/clustering.html

Data Mining Algorithms

- k-means, and more generally clustering, is just one field in the area of *Data Mining*

- For more information see the webpage
  Top 10 Data Mining Algorithms, Explained
  a follow up to the below paper

- X. Wu et al., *Top 10 algorithms in data mining*,
  Knowledge and Information Systems, 14(1):1–37, 2008.
  DOI 10.1007/s10115-007-0114-2

www.kdnuggets.com/2015/05/top-10-data-mining-algorithms-explained.html