Lecture: K-Means Clustering

Rodrigo Canaan
rodrigocanaan.com
rocanaan@gmail.com
@rocanaan

Cal Poly State University, May 26th

Slides and Code

Types of Machine Learning

- **Supervised Learning**
  Learning to predict values or classify objects based on labeled data

- **Unsupervised Learning**
  Learning patterns based on unlabeled data

- **Reinforcement Learning**
  Learning to act intelligently based on interaction with an environment

Types of Machine Learning

- **Supervised Learning**

  Learning to predict values or classify objects based on labeled data

- **Unsupervised Learning**

  Learning patterns based on unlabeled data

- **Reinforcement Learning**

  Learning to act intelligently based on interaction with an environment

Why do Unsupervised Learning?

- To gain insight about the data
- To compress and/or visualize the data
- To generate examples that are like the data
- To label data for downstream tasks

Examples in various domains

- **Recommender Systems**

Group similar products and/or users together

- **Visual Processing**

Group similar images even without labels

- **Games**

Build models of different player styles

- **Science**

Create a taxonomy of phenomena (e.g. stars) based on their observed properties (magnitude, spectrum, distance…)

Unsupervised Learning Tasks

- **Clustering**
  Partitioning the data into “clusters” based on a measure of similarity
- **Dimensionality reduction**
  Transforming the data from a high dimension to a low dimension
- **Generative models**
  Learning a “model” that can be used to produce new examples that are similar to the data

Unsupervised Learning Tasks

- **Clustering**

Partitioning the data into “clusters” based on a measure of similarity

- **Dimensionality reduction**

Transforming the data from a high dimension to a low dimension

- **Generative models**

Learning a “model” that can be used to produce new examples that are similar to the data

Visual Example

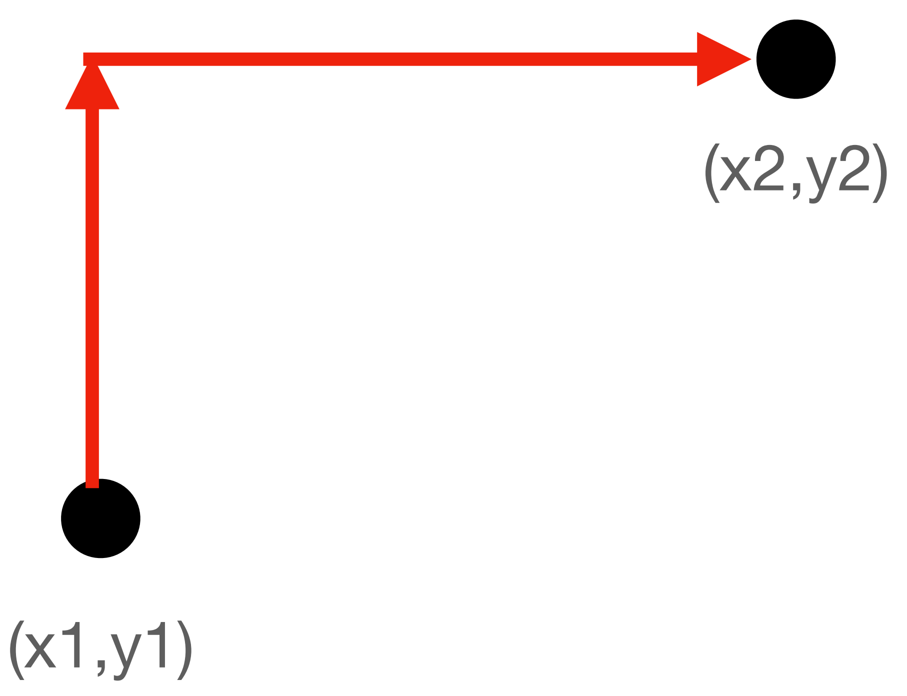
Clustering - Distances

- **What does it mean for two data points to be "similar"?**

Usually, it means to have a small "distance" according to some metric

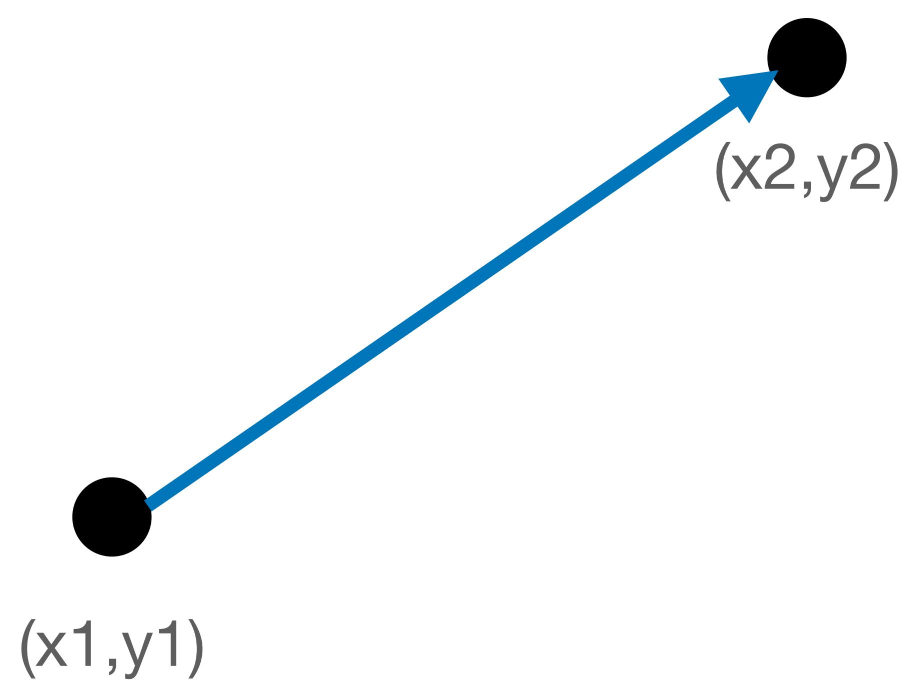
Examples (in 2-dimensions)

Manhattan Distance



$$D_M = |x2 - x1| + |y2 - y1|$$

Euclidean Distance



$$D_E = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}$$

Clustering - Partitions

What does it mean for a partition to be “good”?

Intuitively:

- Elements in the same cluster are similar.
- Elements in different clusters are dissimilar.

What is the simplest-possible clustering algorithm?

Clustering by exhaustive search

1. Generate all ___ possible partitions
2. Calculate clustering metric for each partition
3. Return the partition with best clustering metric

Clustering by exhaustive search

1. Generate all k^n possible partitions

2. Calculate WCSS for each partition

3. Return the partition with minimum WCSS

Clustering by exhaustive search

1. Generate all k^n possible partitions

2. Calculate WCSS for each partition

3. Return the partition with minimum WCSS

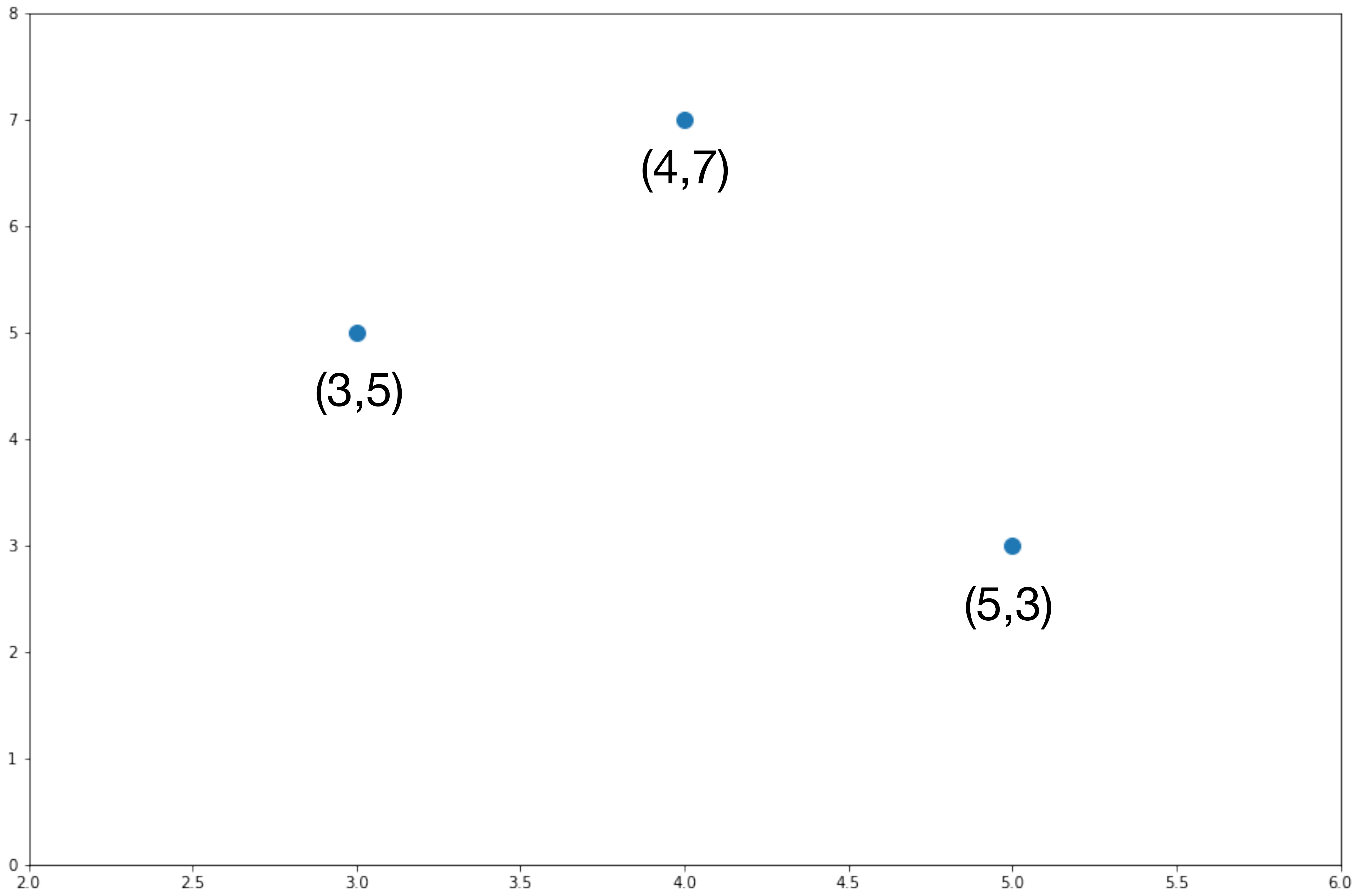
Very expensive!
Need **heuristic** for approximate (but faster) solution!

Clustering by exhaustive search

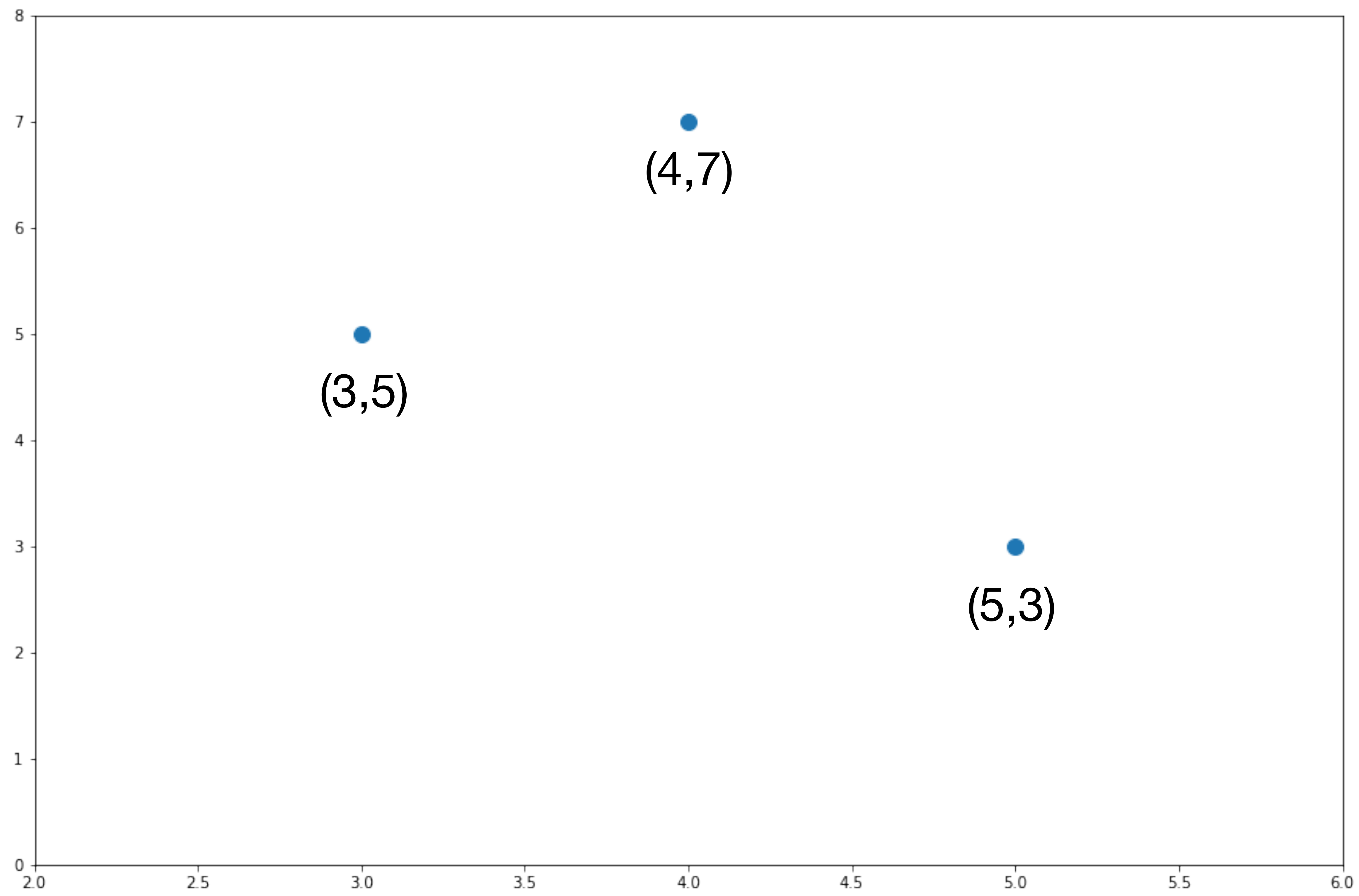
1. **Generate all k^n possible partitions**
2. **Calculate WCSS for each partition**
3. **Return the partition with minimum WCSS**

Very expensive!
Need **heuristic** for approximate (but faster) solution!
Example: assign each point to cluster with nearest centroid

Centroid Calculation

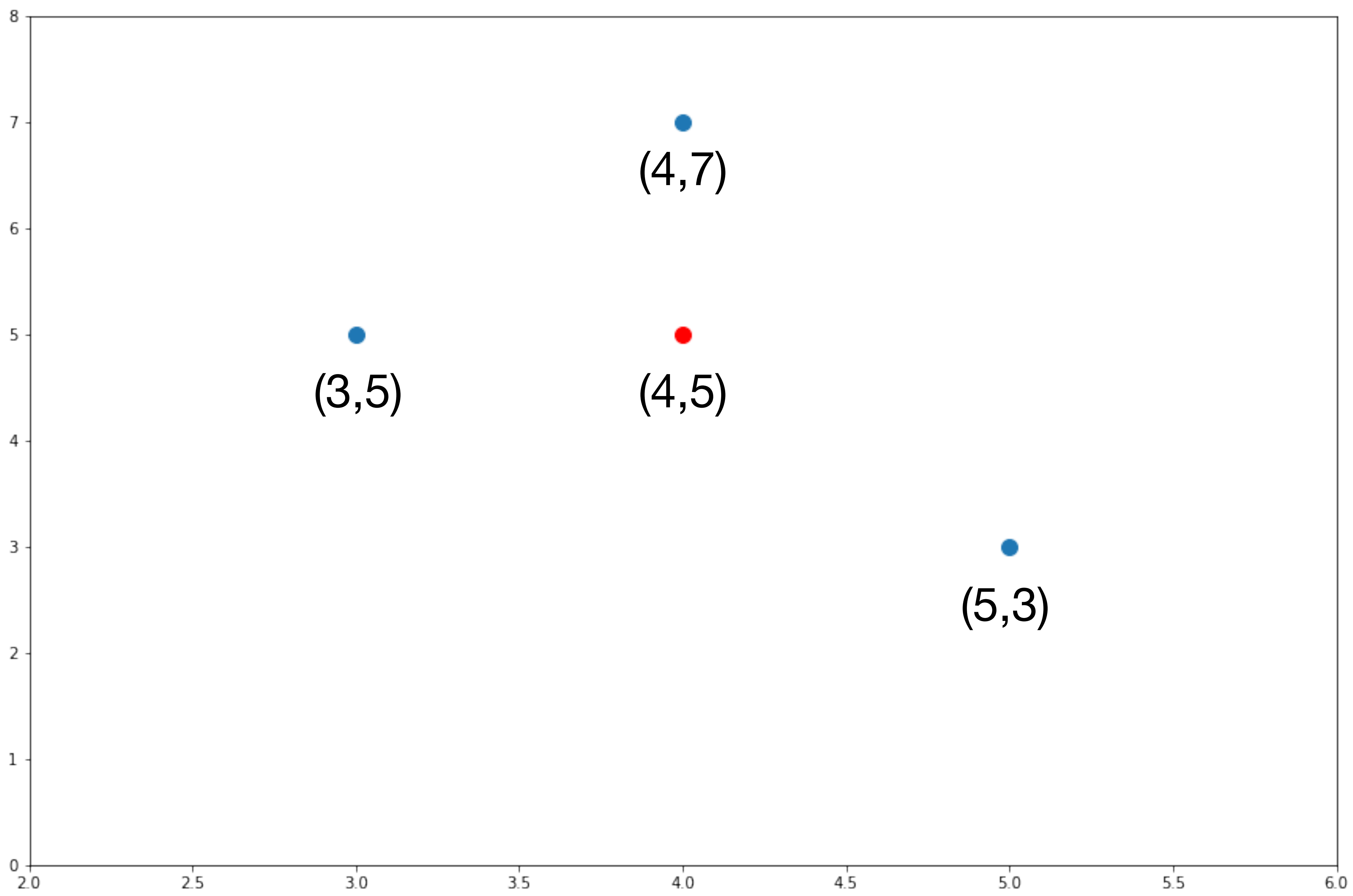


Centroid Calculation



x_mean = (3+4+5)/3 = 4
y_mean = (5+7+3)/3 = 5

Centroid Calculation



x_mean = (3+4+5)/3 = 4
y_mean = (5+7+3)/3 = 5

Centroid = (4,5)

Outline of K-Means Clustering Algorithm

1. **Generate *k* initial centroids**
2. **Re-assign points based on heuristic**
   Each point is assigned to cluster with nearest centroid
3. **Recalculate centroids**
4. **Repeat (2, 3) until no re-assignments**
   Alternatively, up to some maximum number of repetitions

(Naive) K-Means Pseudo-Code

```
# Pseudo-code of K-Means algorithm

Function k-means (data,k)
    P <- initialize_partition(data,k)
    stop <- False

    while not stop # runs until no new assignments
                   # optionally, until some max_iterations is reached
        stop = True
        P_new <- empty_partition(k)
        for d in data
            new_label <- P.find_min_sq_distance_cluster(d)
            P_new.add_element(d,new_label)
            if new_cluster <> P.get_label(d)    # new assignment, don't stop!
                stop = False
            Endif
        Endfor
        P <- P_new
        P.compute_centroids()
    Endwhile

    return P
Endfunction
```

K-Means Initialization

How to initialize clusters?

- **Forgy method:** Choose *k* initial centroids randomly (from the data), assign other points according to distance to centroids

- **Random Partition:** Assign each datapoint to a random cluster label, then compute centroids

- Other options: see comparative study by (Celebi et al., 2013)

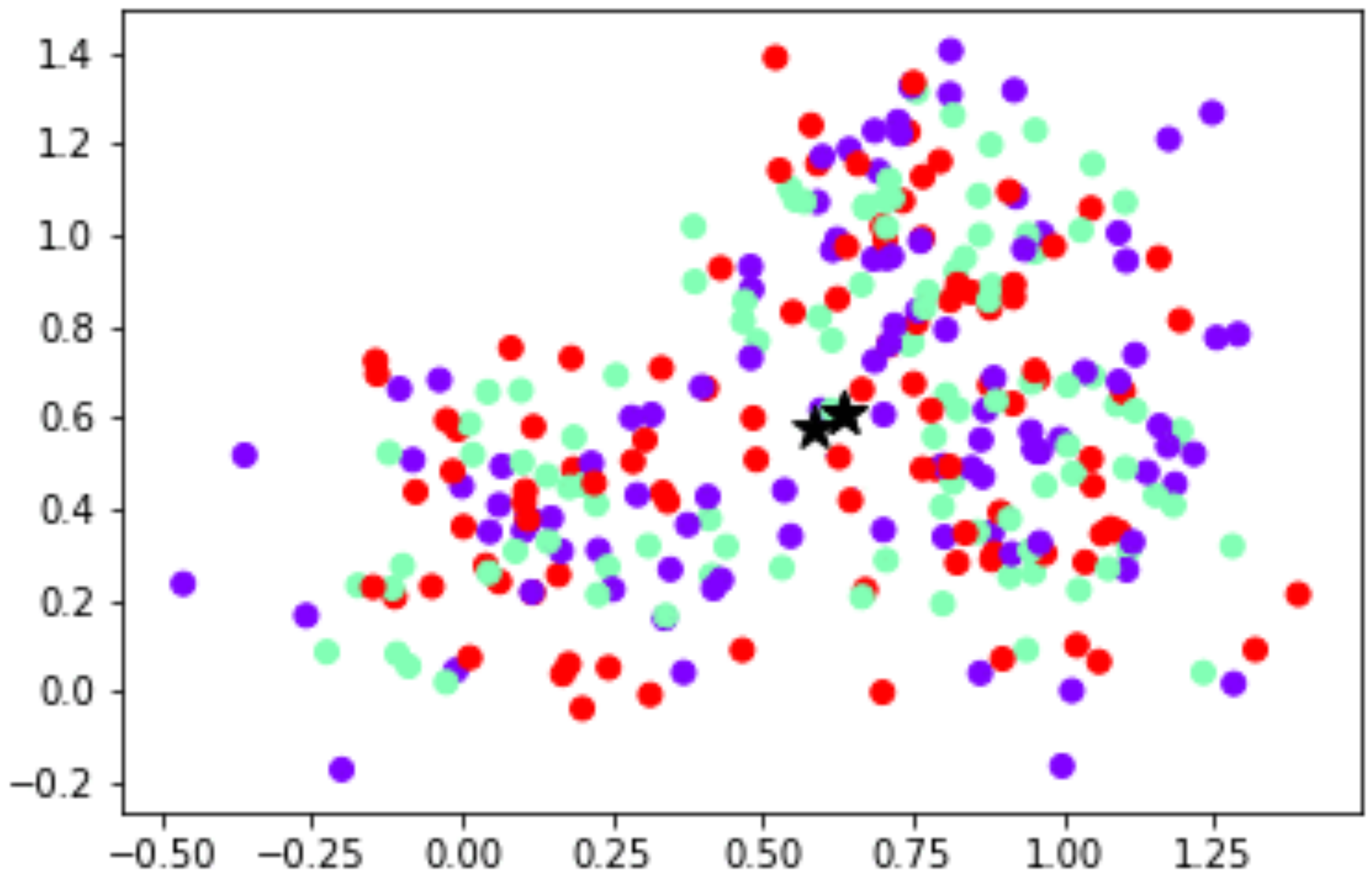
Forgy Method

```
# Forgy initialization
# Select k random points as initial centroids,
# then assign remaining elements to closest centroid

Function initialize_partitions(data,k)
    P <- empty_partition(k)
    medoids <- sample_without_replacement(data,k) # get k points from data
    for i in 0..k-1
        P.add_element(medoids[i],i)
    Endfor
    P.compute_centroids()

    for d in data
        label <- P.find_min_sq_distance_cluster(d)
        P.add_element(d,label)
    Endfor
    P.compute_centroids
    return P

Endfunction
```

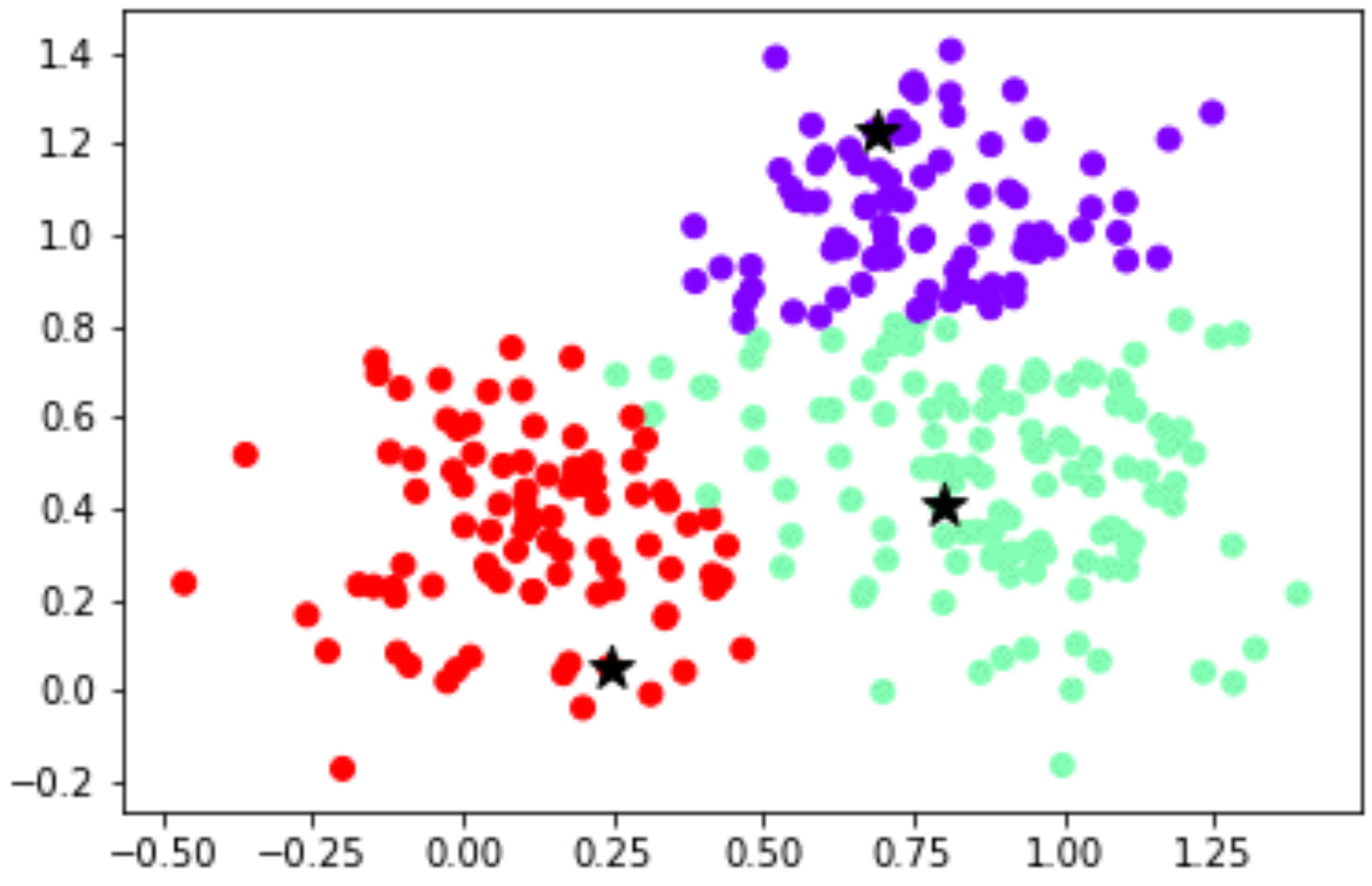
Random Partition

```
# Random Initialization
# Assign each point to a random label, then compute centroids

Function initialize_partitions(data,k)
    P <- empty_partition(k)

    for d in data
        label <- random_uniform(k)  # select a random int uniformly in 0..k-1
        P.add_element(d,label)
    Endfor

    P.compute_centroids
    return P
Endfunction
```

Demo



1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.50 −0.25 0.00 0.25 0.50 0.75 1.00 1.25

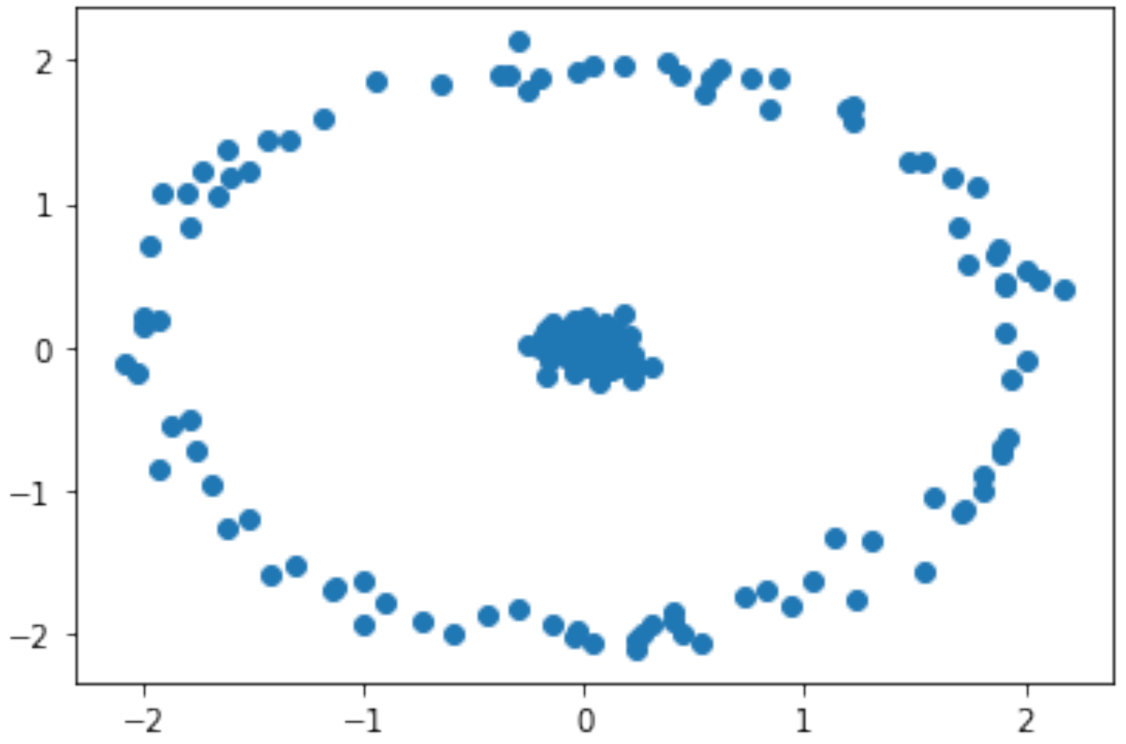


1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.50 −0.25 0.00 0.25 0.50 0.75 1.00 1.25

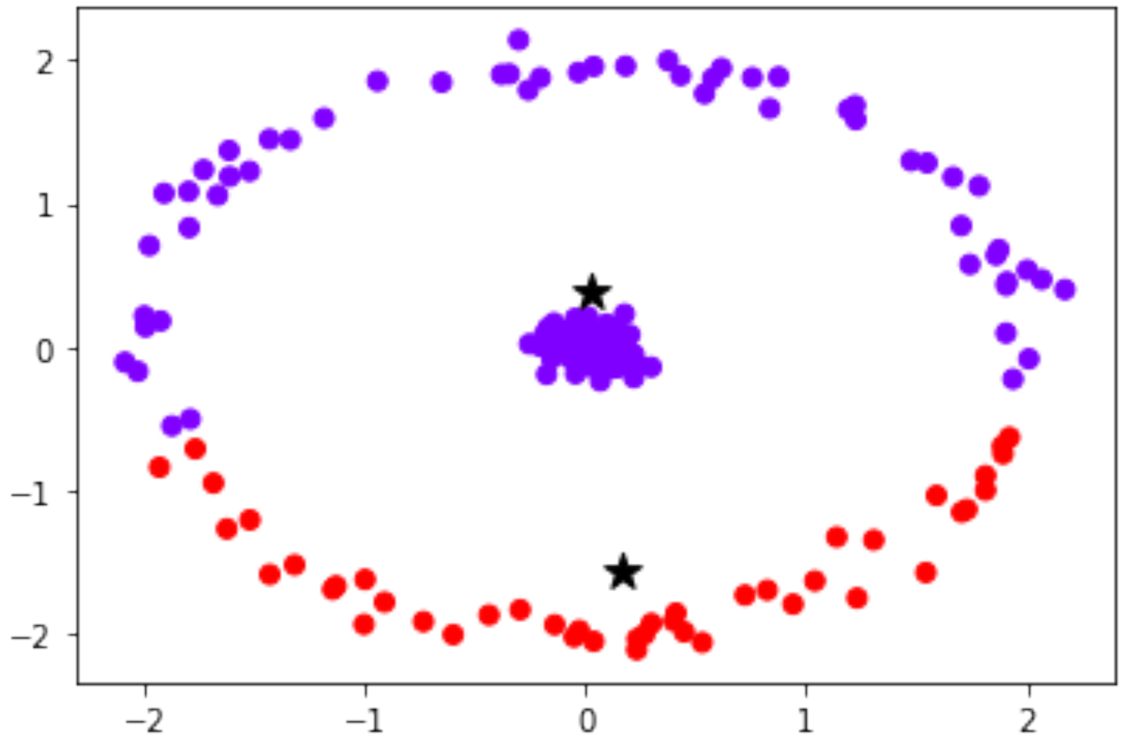
(Naive) K-Means Disadvantages

- **Struggles with some cluster shapes, sizes and outliers**

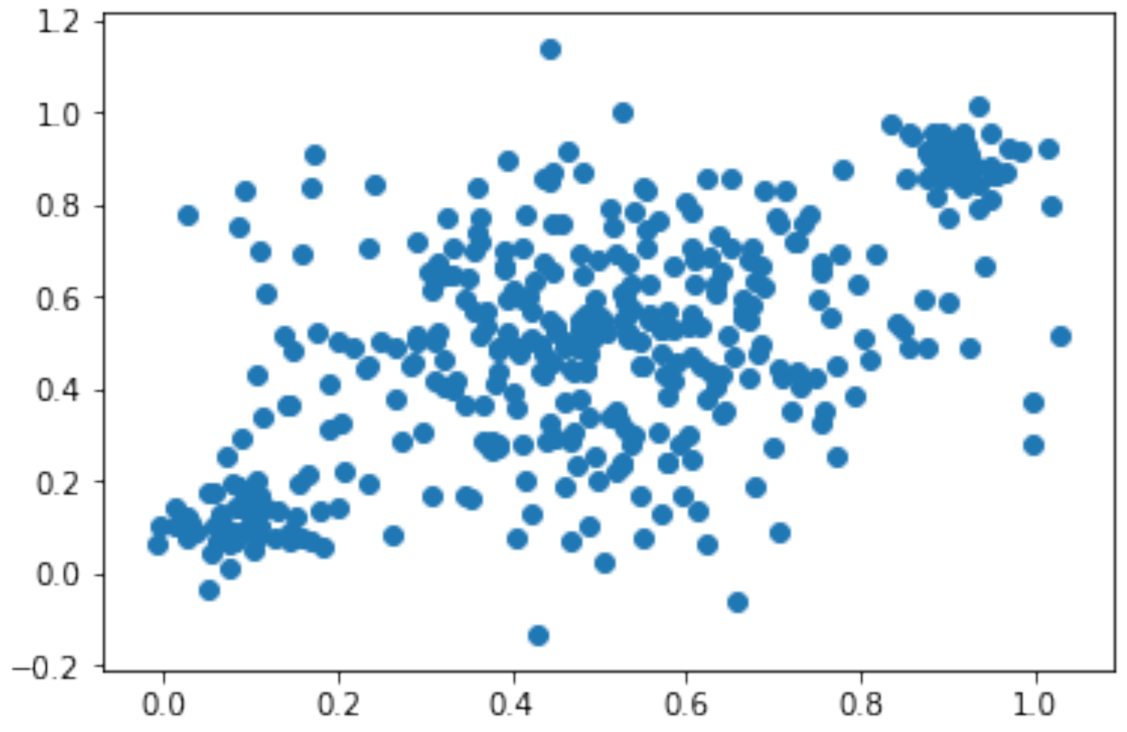
Non-Convex Clusters



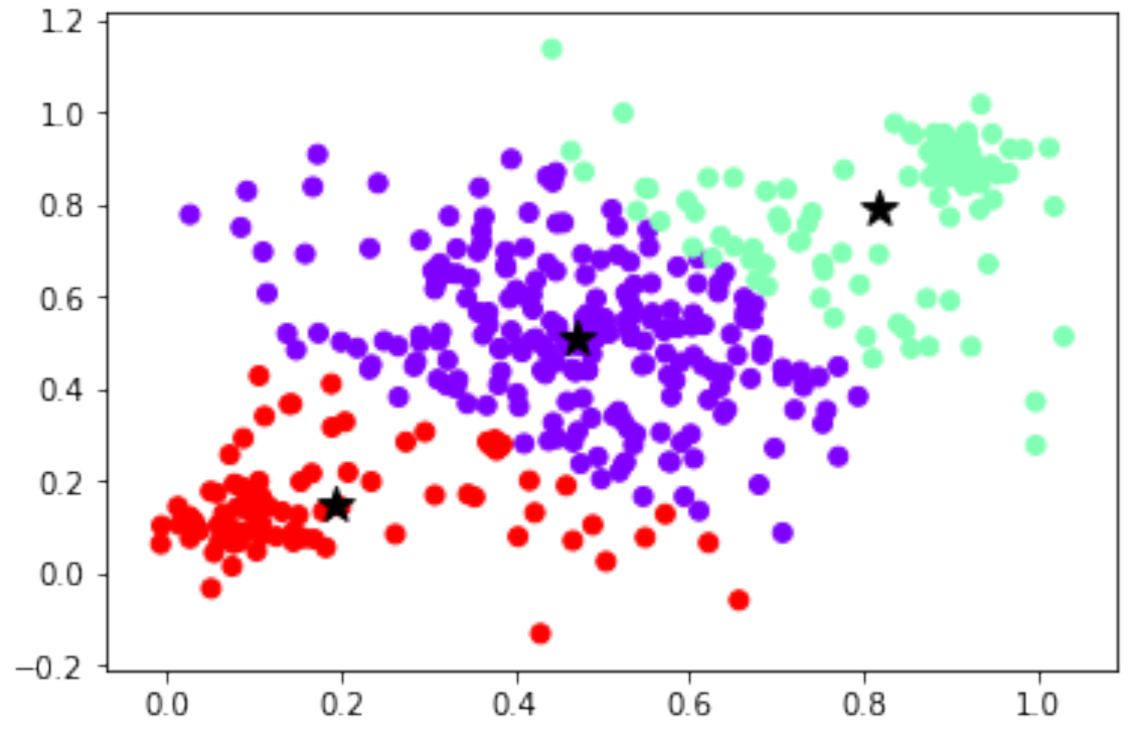
Partition with k=2



Clusters of different sizes



Partition with k=3



(Naive) K-Means Disadvantages

- Struggles with some cluster shapes, sizes and outliers
- Struggles with categorical data
- Guaranteed convergence only to local (not global optimum)
- Need to specify *k* in advance
- “Curse of dimensionality”
- Worst-case performance?

Time Complexity

O(i) Outer-loop iterations

O(n) Inner-loop iterations

```
# Pseudo-code of K-Means algorithm

Function k-means (data,k)
    P <- initialize_partition(data,k)
    stop <- False

    while not stop # runs until no new assignments
                   # optionally, until some max_iterations is reached
        stop = True
        P_new <- empty_partition(k)
        for d in data
            new_label <- P.find_min_sq_distance_cluster(d)
            P_new.add_element(d,new_label)
            if new_cluster <> P.get_label(d)   # new assignment, don't stop!
                stop = False
            Endif
        Endfor
        P <- P_new
        P.compute_centroids()
    Endwhile

    return P
Endfunction
```

O(k)

Time Complexity = O(nki)

O(i)
Outer-loop iterations

O(n)
Inner-loop iterations

```
# Pseudo-code of K-Means algorithm

Function k-means (data,k)
    P <- initialize_partition(data,k)
    stop <- False

    while not stop # runs until no new assignments
                   # optionally, until some max_iterations is reached
        stop = True
        P_new <- empty_partition(k)
        for d in data
            new_label <- P.find_min_sq_distance_cluster(d)
            P_new.add_element(d,new_label)
            if new_cluster <> P.get_label(d)   # new assignment, don't stop!
                stop = False
            Endif
        Endfor
        P <- P_new
        P.compute_centroids()
    Endwhile

    return P
Endfunction
```

O(k)

(Naive) K-Means Disadvantages

- Struggles with some cluster shapes, sizes and outliers
- Guaranteed convergence only to local (not global optimum)
- Need to specify *k* in advance
- “Curse of dimensionality”
- High worst-case performance

K-Means Advantages

- Simple to implement
- Good performance in many practical scenarios
- Adaptations can handle outliers, different shapes and sizes, higher dimensions, categorical data…
- Performance can be improved by non-naive implementations

Thank you!

Scholar Page

www.rodrigocanaan.com

rodrigo.canaan@nyu.edu