FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG

SCHOOL OF ENGINEERING

Lecture Pattern Analysis

Part 05: Introduction: Simplifications of the Feature Space

Christian Riess

IT Security Infrastructures Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg

May 11, 2022

Introduction

- So far, we looked at different options to represent a set of samples:
  - Local operators from fixed neighborhood relationships:
    Non-parametric Density Estimation via Kernels and k-NN
  - Local operators from learning-based sample space partitioning:
    Trees and Random Forests
- Compared to our kernels, tree-based methods
  - do not need to store all samples to respond to a query
  - subdivide the sample space dynamically with an objective function
- However, beyond the representation itself, these methods do not provide much information about the distributions of samples
- In the upcoming second part of the lecture, we will look at representational simplifications to make the distributions interpretable:
  - **Clustering** segments the data into few meaningful groups
  - **Manifold Learning** reduces the sample space dimensionality while preserving the structure of the data

Clustering

- The goal is to assign identical labels to similar samples
- Difference to classification/regression: Clustering is unsupervised
- Hence, clustering applications oftentimes explore data, e.g.:
  - Which gene expressions cause which type of cancer[1]?
  - Which other products attract customers who buy coffee when it is discounted?
- We will investigate these specific algorithms:
  - Gaussian Mixture Models
  - k-means
  - Mean Shift
- We will also address the model selection problem, i.e., the selection of the hyperparameters

[1] See Hastie/Tibshirani/Friedman Sec. 14.3.8 for a k-means example

Manifold Learning

- The goal is to represent the data manifold in a lower dimensional space, i.e., to perform a structure-preserving mapping to a lower dimension
- Oftentimes, manifold learning is directly integrated into a PR pipeline, e.g.,
  - as pre-processing step to reduce the dimensionality of the input, e.g., the 100s of spectral bands in remote sensing are highly correlated
  - within the feature extraction step to make the classifier input “denser”
- For deep neural networks, manifold learning is oftentimes used to visualize that good features have been learned
- We will investigate these specific algorithms:
  - PCA (known, I guess?)
  - Multi-dimensional Scaling
  - ISOMAP
  - Laplacian Eigenmaps
- If time permits, we can also touch applications of spectral graph processing

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG

SCHOOL OF ENGINEERING

Lecture Pattern Analysis

Part 06: Gaussian Mixture Models

Christian Riess
IT Security Infrastructures Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg
May 13, 2022

Introduction

- Gaussian Mixture Models (GMMs) are covered in Pattern Recognition
- Nevertheless, let's do a quick recap in this lecture[1]
- A GMM models a PDF as sum of K normal distributions $\mathcal{N}$ with weights π_k:

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k) \tag{1}$$

Note hereby that $0 \leq \pi_k \leq 1$ and $\sum_{k=1}^{K} \pi_k = 1$

- GMMs can be seen as density representation or as clustering method
- GMMs are fitted to data with an **Expectation-Maximization** (EM) algorithm

[1] We follow Bishop Sec. 9.2 (including 9.2.1 and 9.2.2).
Hastie/Tibshirani/Friedman Sec. 8.5 and 8.5.1 starts with an instructive 2-component mixture model, but then hastes within only 2 pages through content that is covered in two full sections of Bishop (Sec. 9.2 and 9.3.), so this is probably a little bit too fast.

Preparations for the Probabilistic Model: Hidden Variable z

- We need a K-dim. hidden variable $\mathbf{z}$ to derive the EM algorithm
- Properties of $\mathbf{z}$:
  - $\mathbf{z}$ is a binary indicator vector ("one-hot vector"), i.e.,

$$z_k = \{0, 1\} \tag{2}$$

and

$$\sum_{k=1}^{K} z_k = 1 \ , \tag{3}$$

  - The marginal probability of z_k is π_k, i.e., $p(z_k = 1) = \pi_k$, such that

$$p(\mathbf{z}) = \prod_{k=1}^{K} \pi_k^{z_k} \tag{4}$$

is the probability mass function over the one-hot vector $\mathbf{z}$

Joint Distribution over x and z

- The joint distribution

$$p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z}) \cdot p(\mathbf{z}) \tag{5}$$

  models the distribution for each data point to belong to each component

- Here, we set

$$p(\mathbf{x}|\mathbf{z}) = \prod_{k=1}^{K} \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k)^{z_k} \quad , \tag{6}$$

  which results in a single Gaussian component at $z_k = 1$,

$$p(\mathbf{x}|z_k = 1) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k) \tag{7}$$

- Insert for the prior $p(\mathbf{z}) = \prod_{k=1}^{K} \pi_k^{z_k}$ as in Eqn. 4 (i.e. $p(z_k = 1) = \pi_k$)

Assembling Everything in the EM Algorithm

- The GMM formulation turns out to be the marginalization over $\mathbf{z}$,

$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{x}, \mathbf{z}) = \sum_{\mathbf{z}} p(\mathbf{z}) p(\mathbf{x}|\mathbf{z}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k) \tag{8}$$

- EM iteratively fits a GMM to data via Maximum Likelihood:
  1. Initialize $\boldsymbol{\theta} = (\pi_k, \boldsymbol{\mu}_k, \Sigma_k)$
  2. Expectation: determine membership of sample to GMM component, i.e., the posterior distribution of the latent variables $p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})$
  3. Maximization: from those memberships, improve $\boldsymbol{\theta}$ via maximizing the expectation of the complete-data log likelihood $\sum_{\mathbf{z}} p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}) \ln p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta}^{\text{new}})$
  4. Set $\boldsymbol{\theta} = \boldsymbol{\theta}^{\text{new}}$ and goto 2) until convergence
- This is essentially the **soft clustering** variant of k-means

GMM Fitting: Expectation Step

- Introduce **responsibilities** $\gamma(z_k)$ that indicate the degree of membership of a sample to a component
- More formally, the responsibility is $p(z_k = 1|\mathbf{x})$ that a sample $\mathbf{x}$ belongs to component k:

$$\gamma(z_k) \equiv p(z_k = 1|\mathbf{x}) \overset{Bayes}{=} \frac{p(z_k = 1)p(\mathbf{x}|z_k = 1)}{\sum_{j=1}^{K} p(z_j = 1)p(\mathbf{x}|z_j = 1)} \tag{9}$$

$$= \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \Sigma_j)} \tag{10}$$

- So, to clarify, π_k is the prior probability of $z_k = 1$, and $\gamma(z_k)$ is the posterior of $z_k = 1$ after having observed the data $\mathbf{x}$

GMM Fitting: Maximization Step (1/2)

- The parameter updates for $\boldsymbol{\mu}_k$, Σ_k, π_k are calculated from maximizing the expectation of the log likelihood for all samples $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$,

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma) = \sum_{i=1}^{N} \ln \left(\sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_k, \Sigma_k) \right) \tag{11}$$

- Finding the maximum: set derivatives w.r.t. $\boldsymbol{\mu}_k$, Σ_k, π_k to 0.
- For $\boldsymbol{\mu}_k$:

$$\frac{\partial \ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma)}{\partial \boldsymbol{\mu}_k} = \sum_{i=1}^{N} \frac{\pi_k \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_j, \Sigma_j)} \cdot \Sigma_k^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k) \tag{12}$$

GMM Fitting: Maximization Step (2/2)

- Setting the derivative to 0 gives

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k} \cdot \sum_{i=1}^{N} \gamma(z_{ik}) \cdot \mathbf{x}_i \tag{13}$$

  where N_k is the total responsibility of component k, $N_k = \sum_{i=1}^{N} \gamma(z_{nk})$

- The new maxima for Σ_k and $\boldsymbol{\mu}_k$ are found analogously:

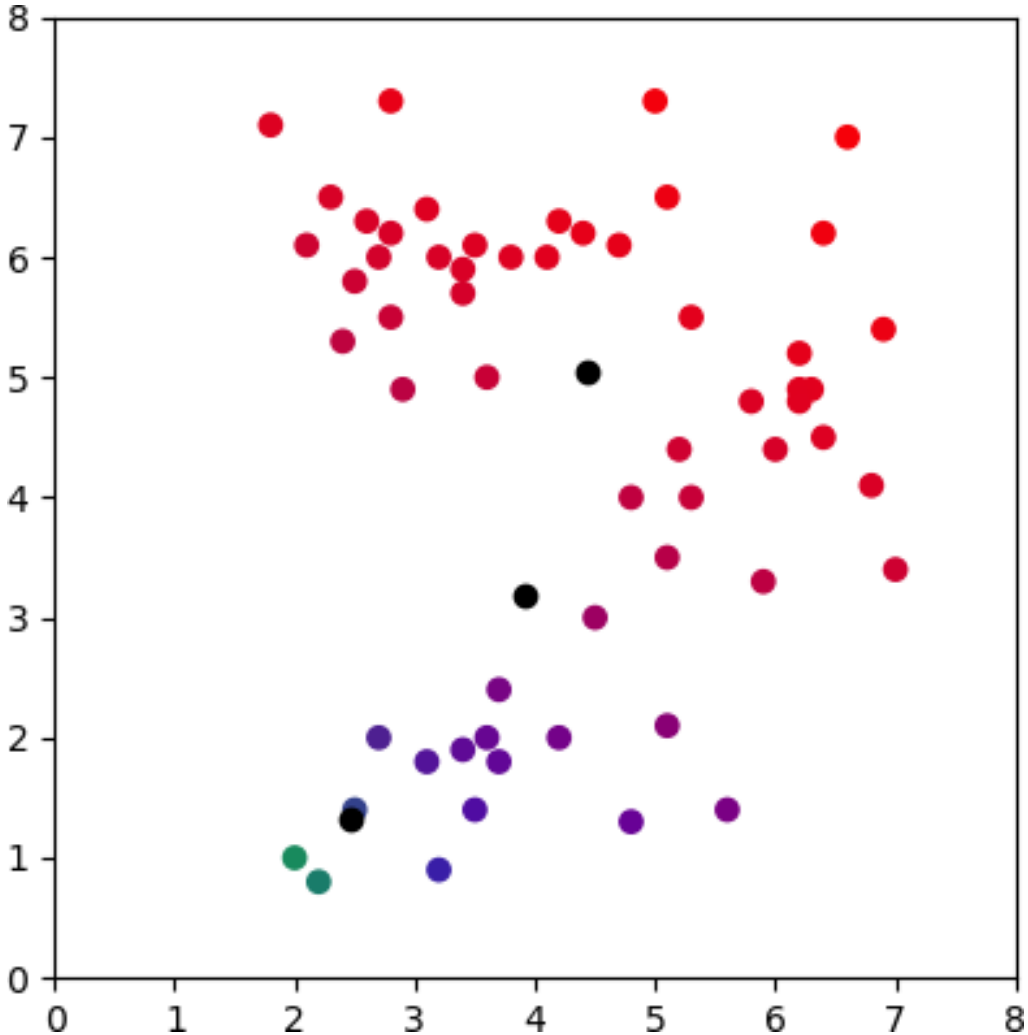
$$\Sigma_k^{\text{new}} = \frac{1}{N_k} \cdot \sum_{i=1}^{N} \gamma(z_{ik}) \cdot (\mathbf{x}_i - \boldsymbol{\mu}_k^{\text{new}}) \cdot (\mathbf{x}_i - \boldsymbol{\mu}_k^{\text{new}})^{\mathrm{T}} \tag{14}$$

$$\pi_k^{\text{new}} = \frac{N_k}{\sum_{k=1}^{K} N_k} \tag{15}$$

- GMM fitting is only locally optimal unless operating on Gaussian distributions

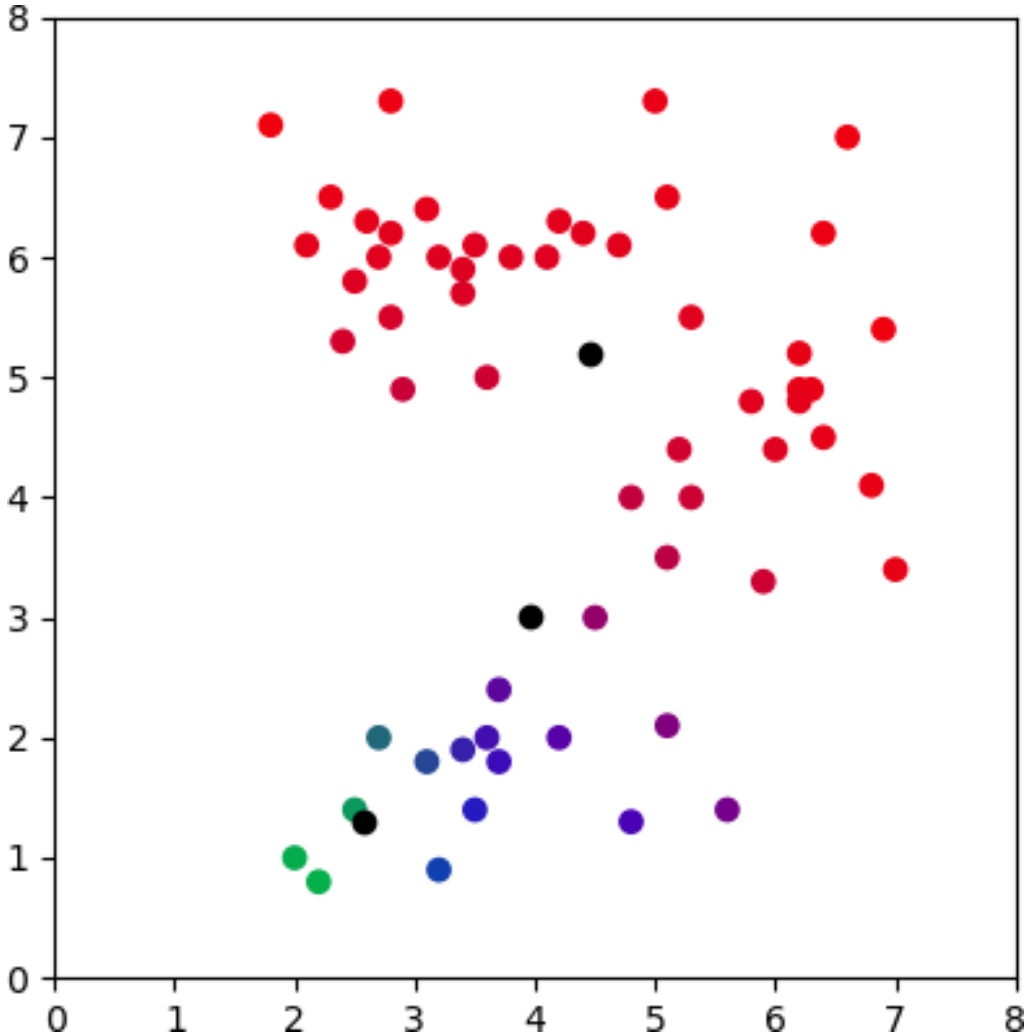
Example Run for $K = 3$

- Black: $\boldsymbol{\mu}_k$ (same starting positions as for k-means)
- Sample chromaticities: Color-coded responsibilities (base colors: red, green, blue)



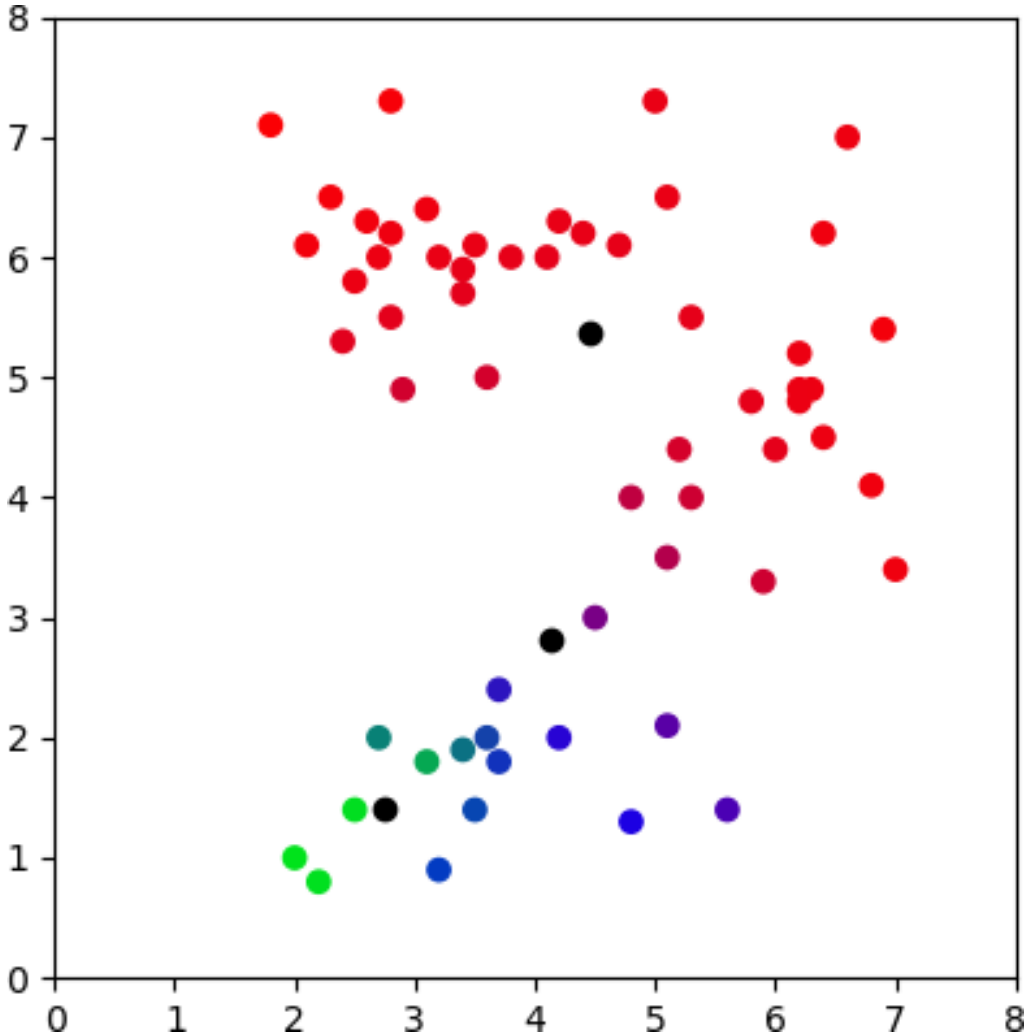
Example Run for $K = 3$

- Black: $\boldsymbol{\mu}_k$ (same starting positions as for k-means)
- Sample chromaticities: Color-coded responsibilities (base colors: red, green, blue)



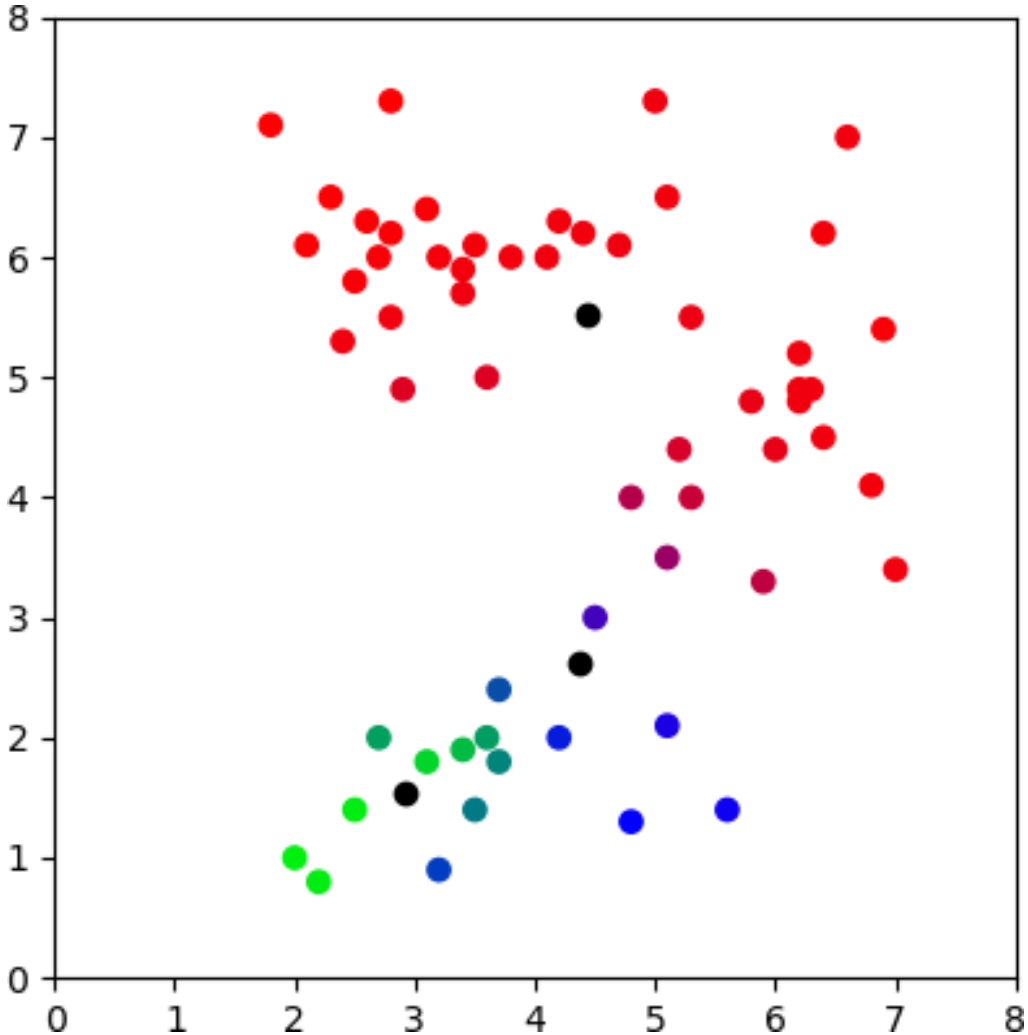
Example Run for $K = 3$

- Black: $\boldsymbol{\mu}_k$ (same starting positions as for k-means)
- Sample chromaticities: Color-coded responsibilities (base colors: red, green, blue)



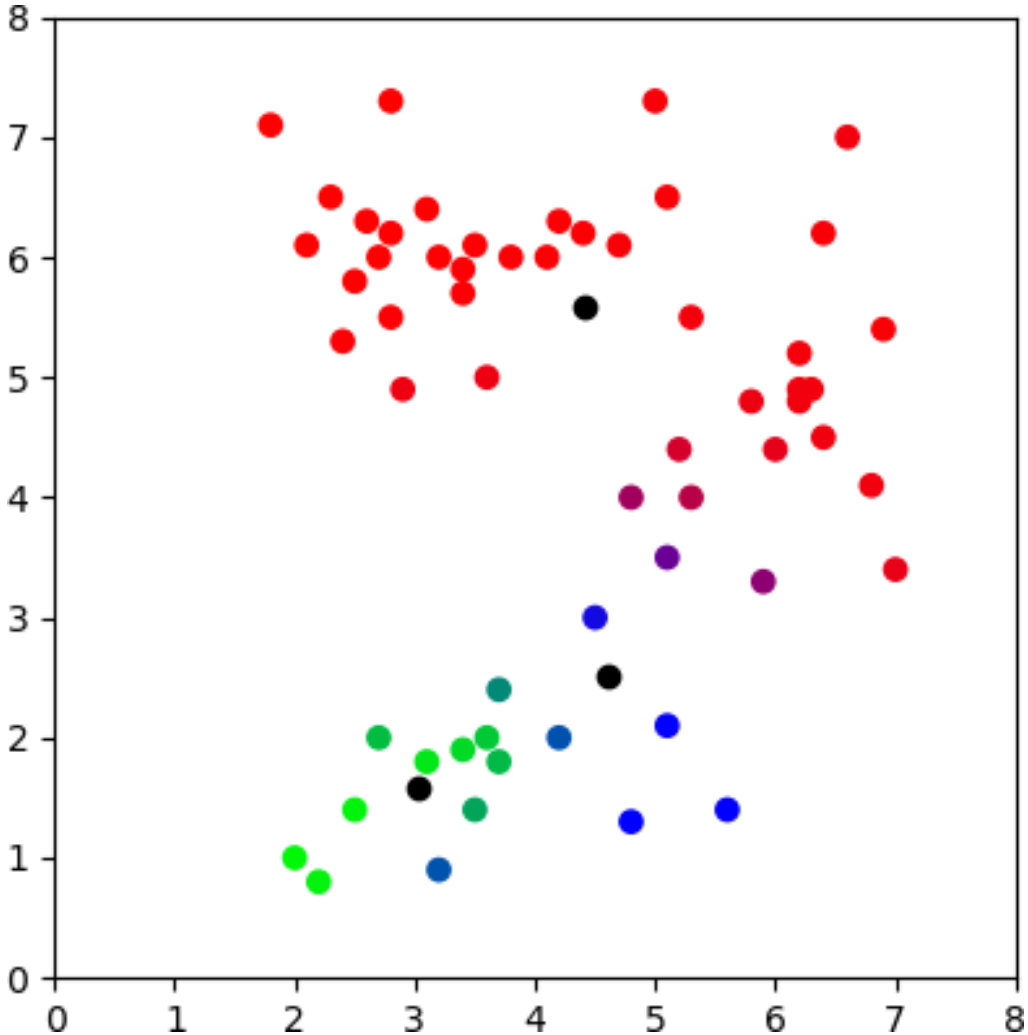
Example Run for $K = 3$

- Black: $\boldsymbol{\mu}_k$ (same starting positions as for k-means)
- Sample chromaticities: Color-coded responsibilities (base colors: red, green, blue)



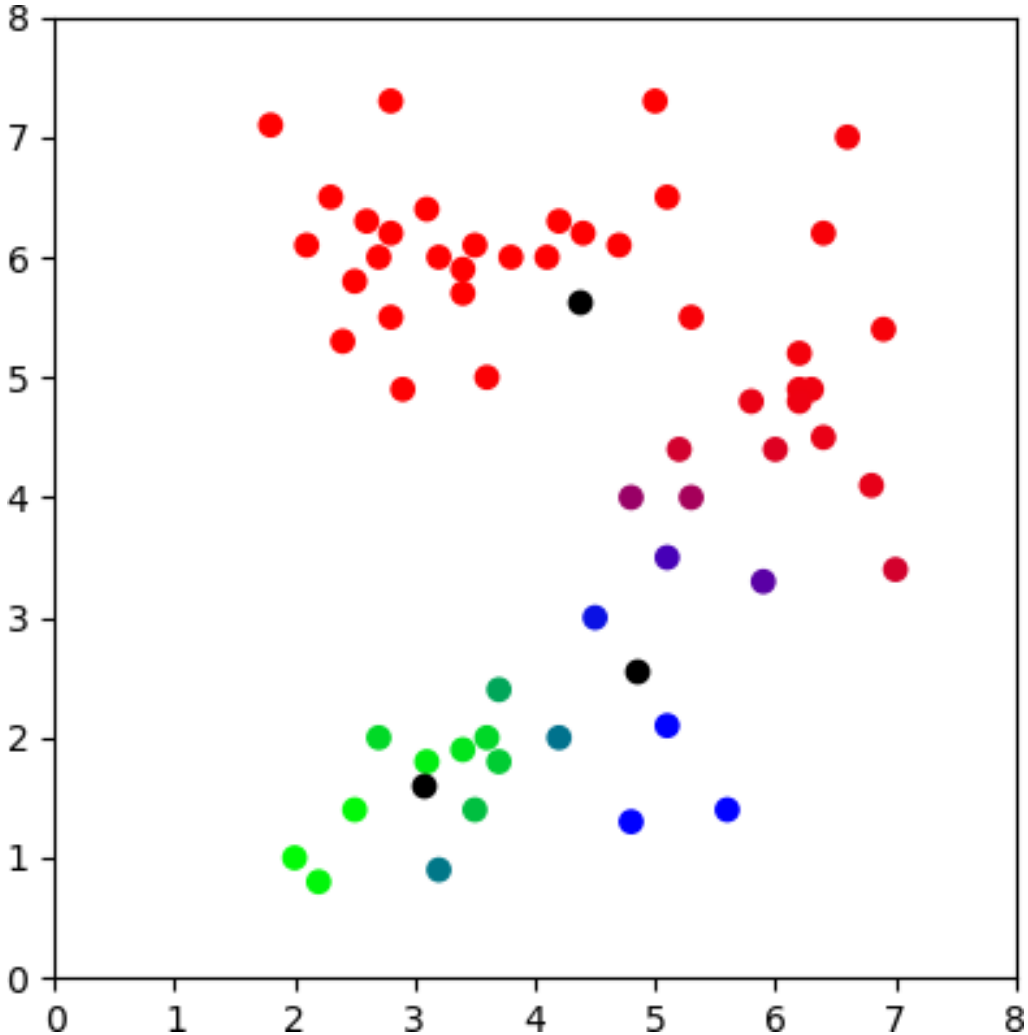
Example Run for $K = 3$

- Black: $\boldsymbol{\mu}_k$ (same starting positions as for k-means)
- Sample chromaticities: Color-coded responsibilities (base colors: red, green, blue)



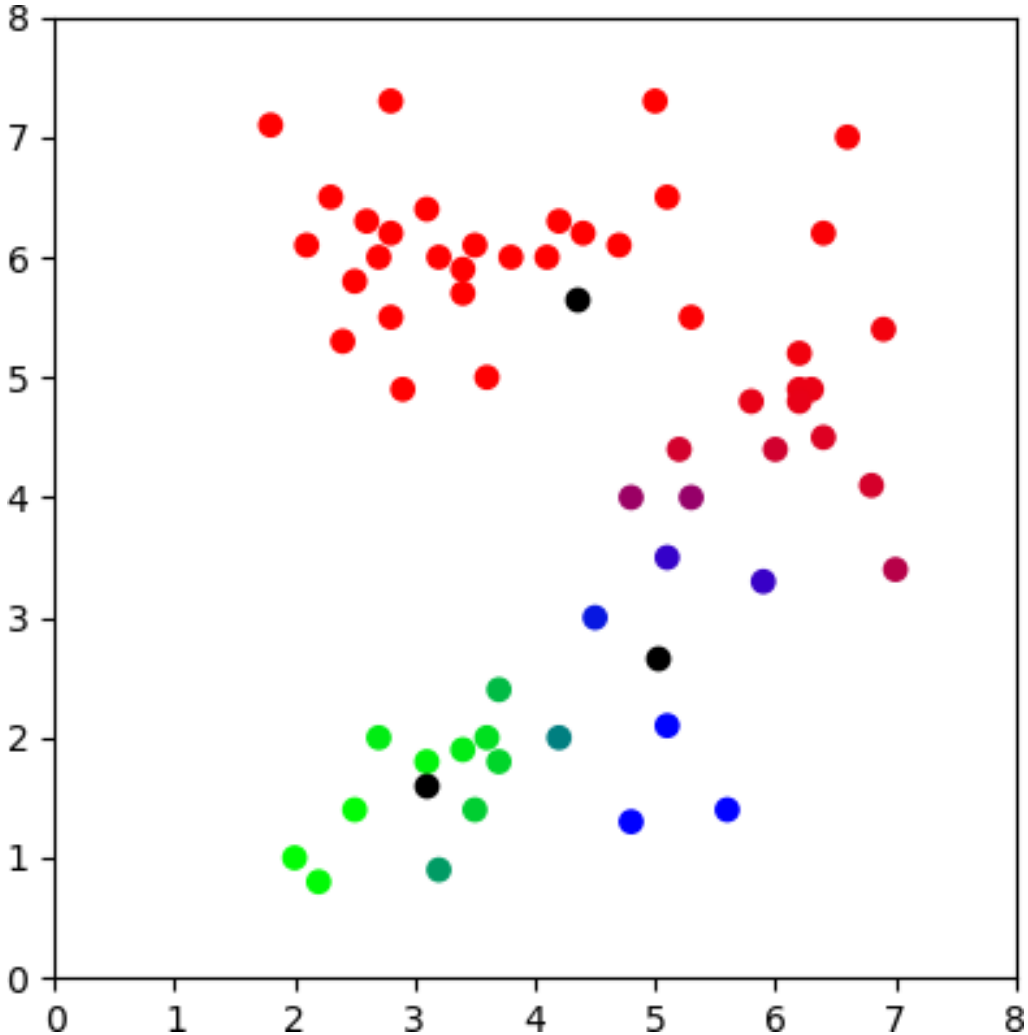
Example Run for $K = 3$

- Black: $\boldsymbol{\mu}_k$ (same starting positions as for k-means)
- Sample chromaticities: Color-coded responsibilities (base colors: red, green, blue)



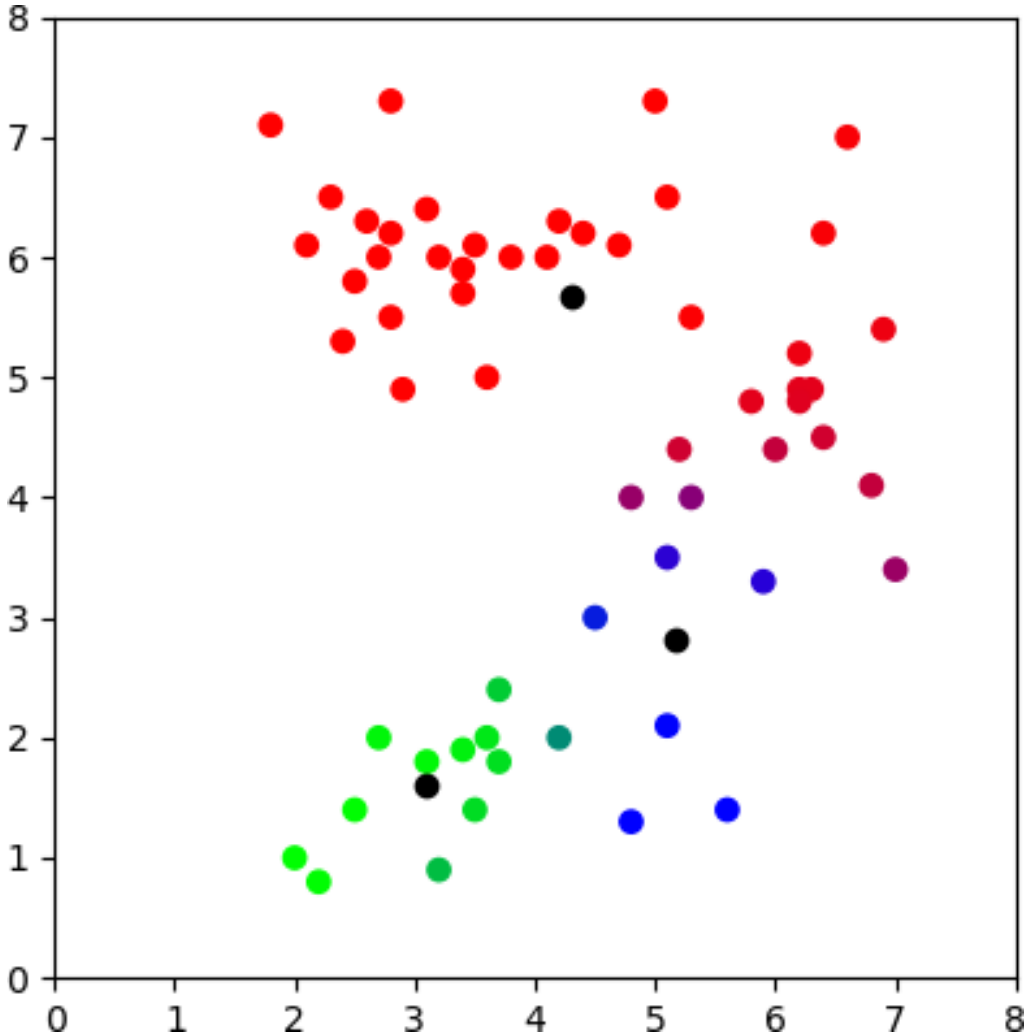
Example Run for $K = 3$

- Black: $\boldsymbol{\mu}_k$ (same starting positions as for k-means)
- Sample chromaticities: Color-coded responsibilities (base colors: red, green, blue)



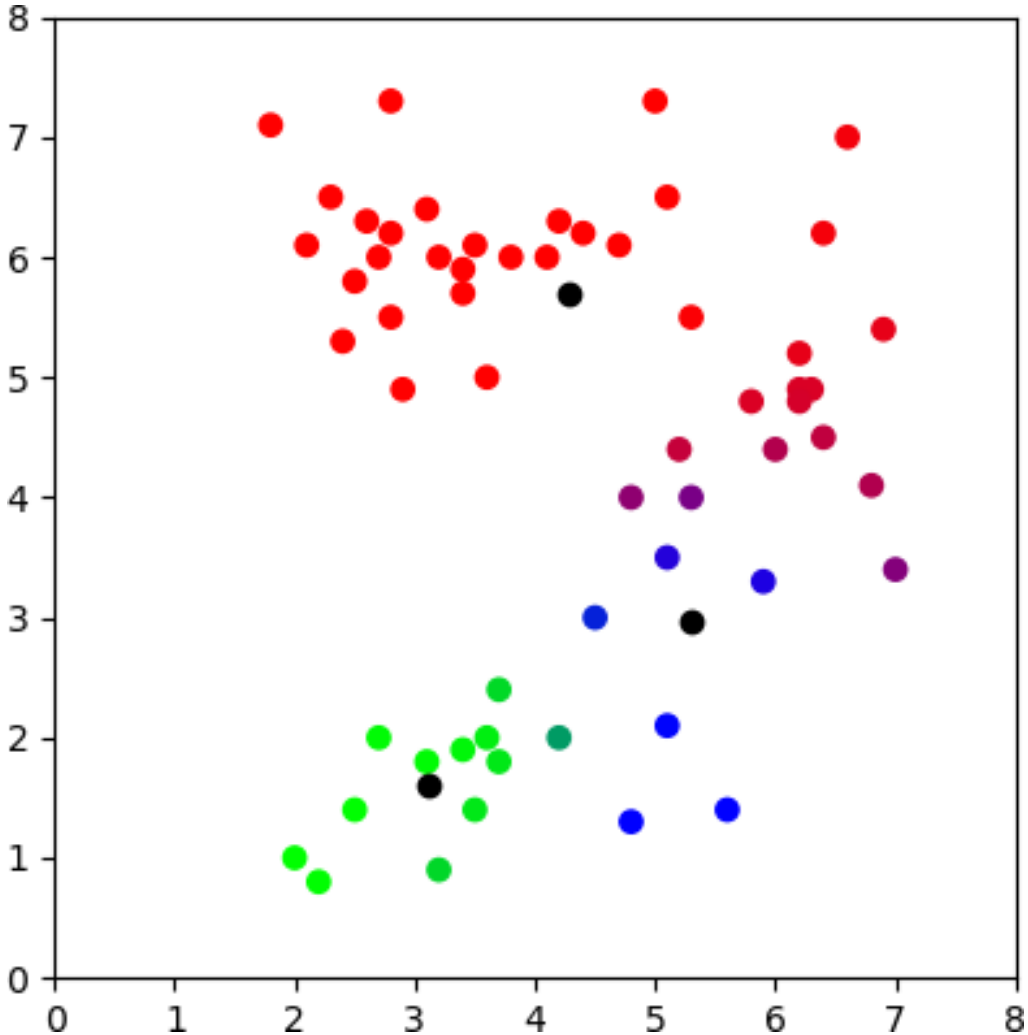
Example Run for $K = 3$

- Black: $\boldsymbol{\mu}_k$ (same starting positions as for k-means)
- Sample chromaticities: Color-coded responsibilities (base colors: red, green, blue)



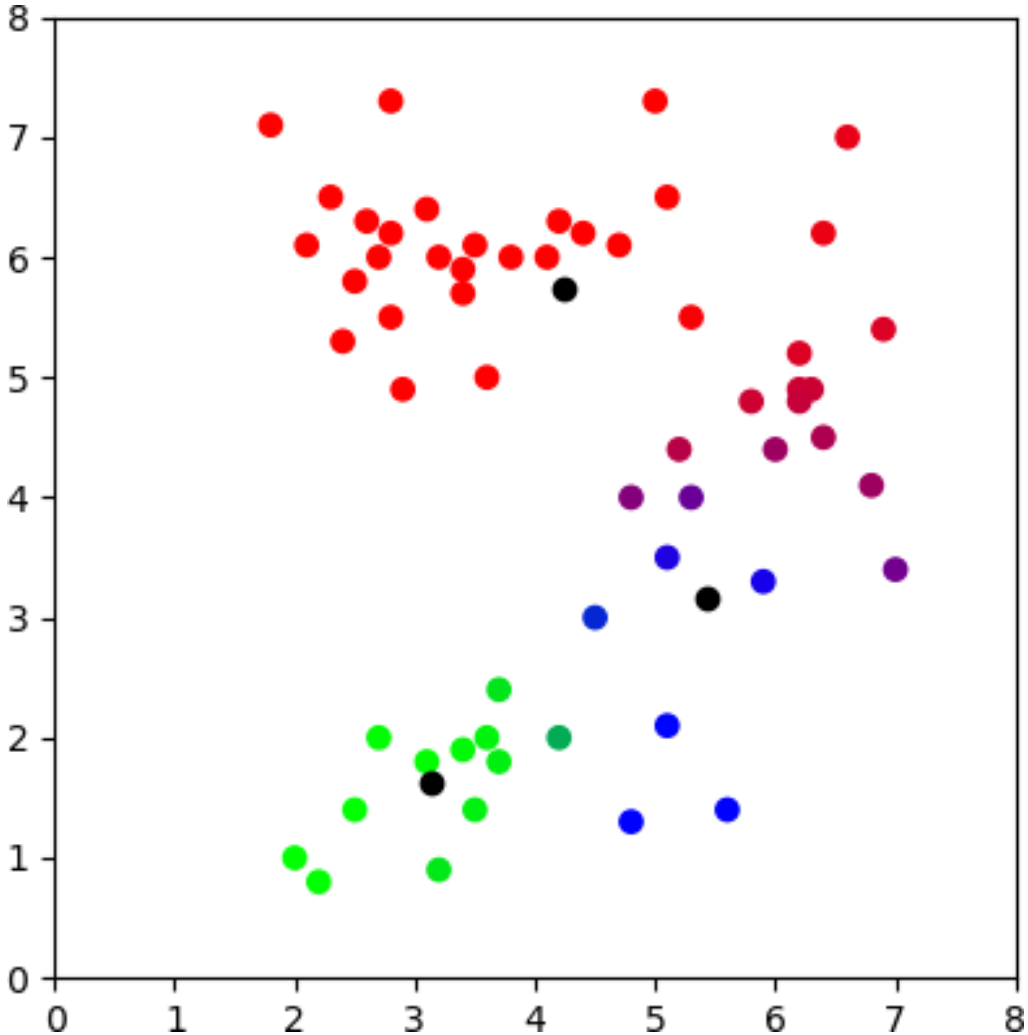
Example Run for $K = 3$

- Black: $\boldsymbol{\mu}_k$ (same starting positions as for k-means)
- Sample chromaticities: Color-coded responsibilities (base colors: red, green, blue)



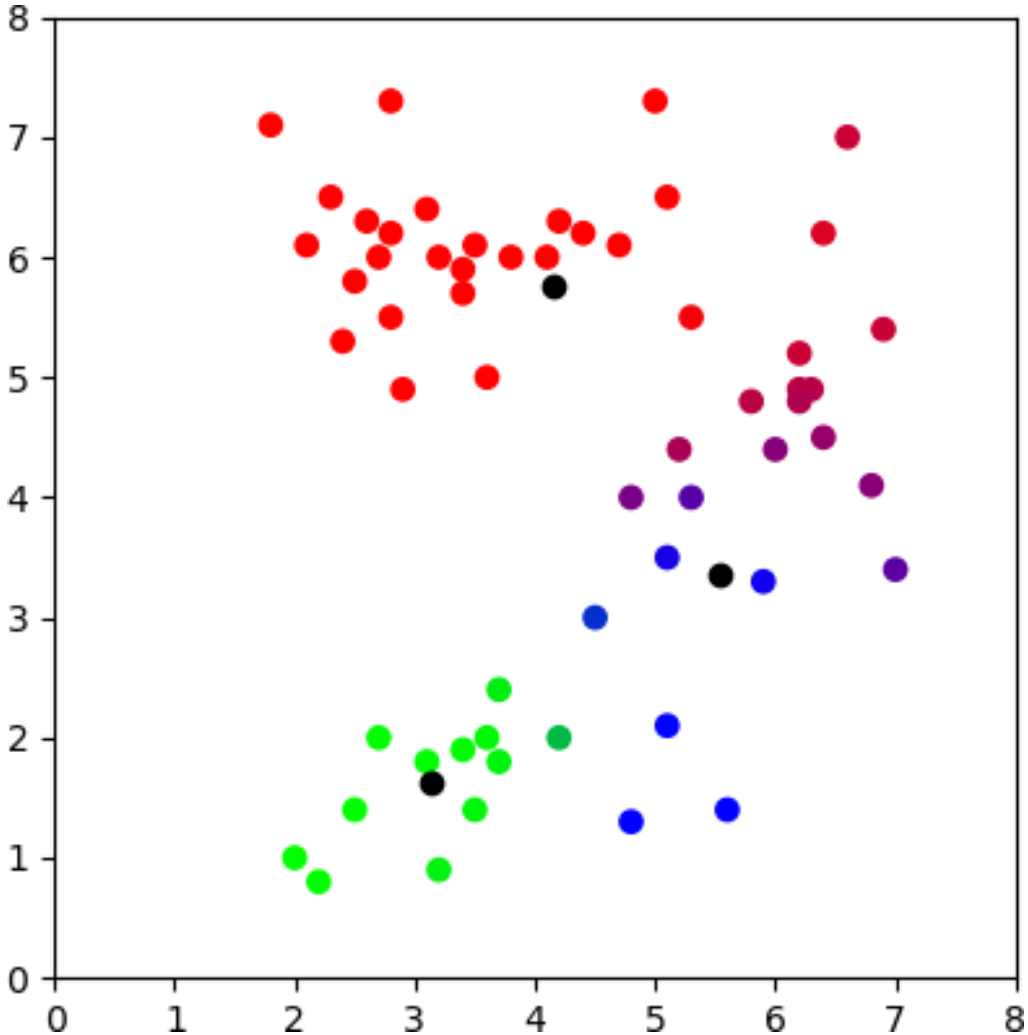
Example Run for $K = 3$

- Black: $\boldsymbol{\mu}_k$ (same starting positions as for k-means)
- Sample chromaticities: Color-coded responsibilities (base colors: red, green, blue)



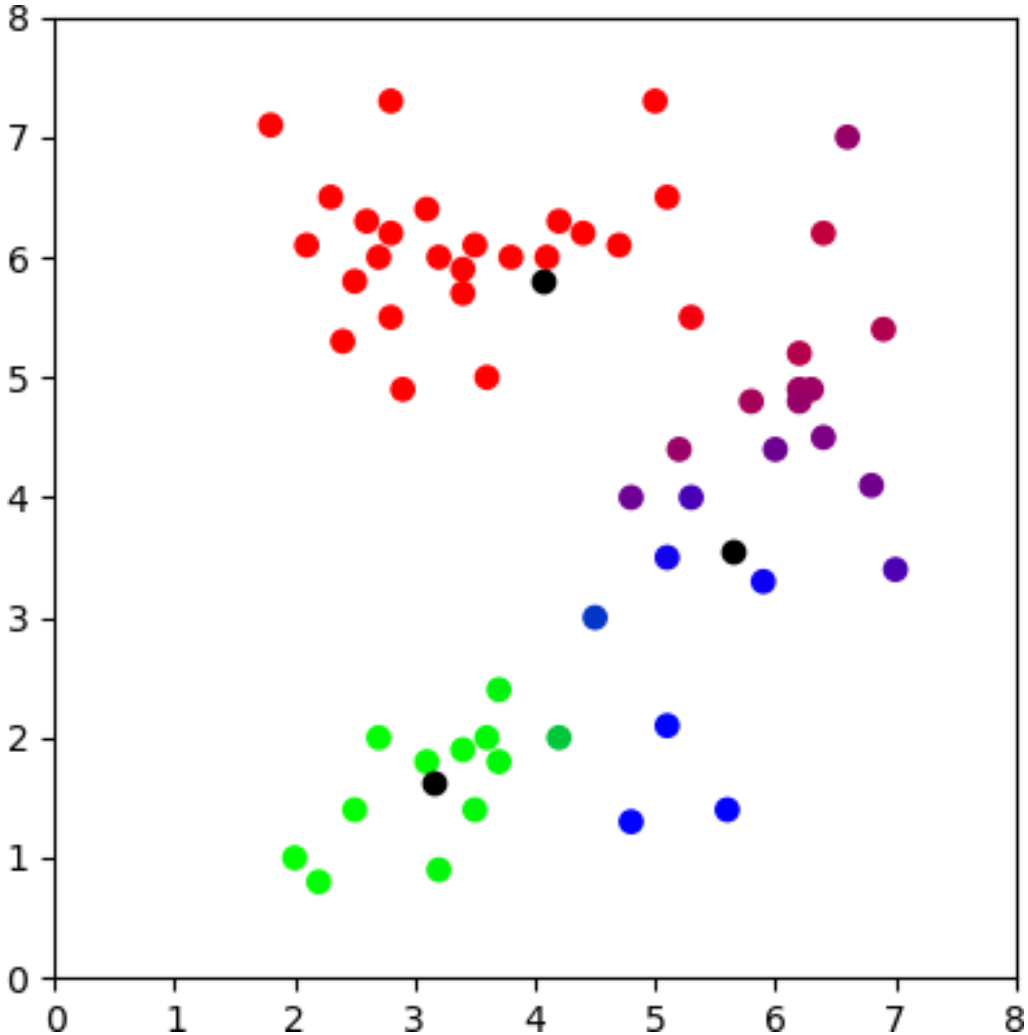
Example Run for $K = 3$

- Black: $\boldsymbol{\mu}_k$ (same starting positions as for k-means)
- Sample chromaticities: Color-coded responsibilities (base colors: red, green, blue)



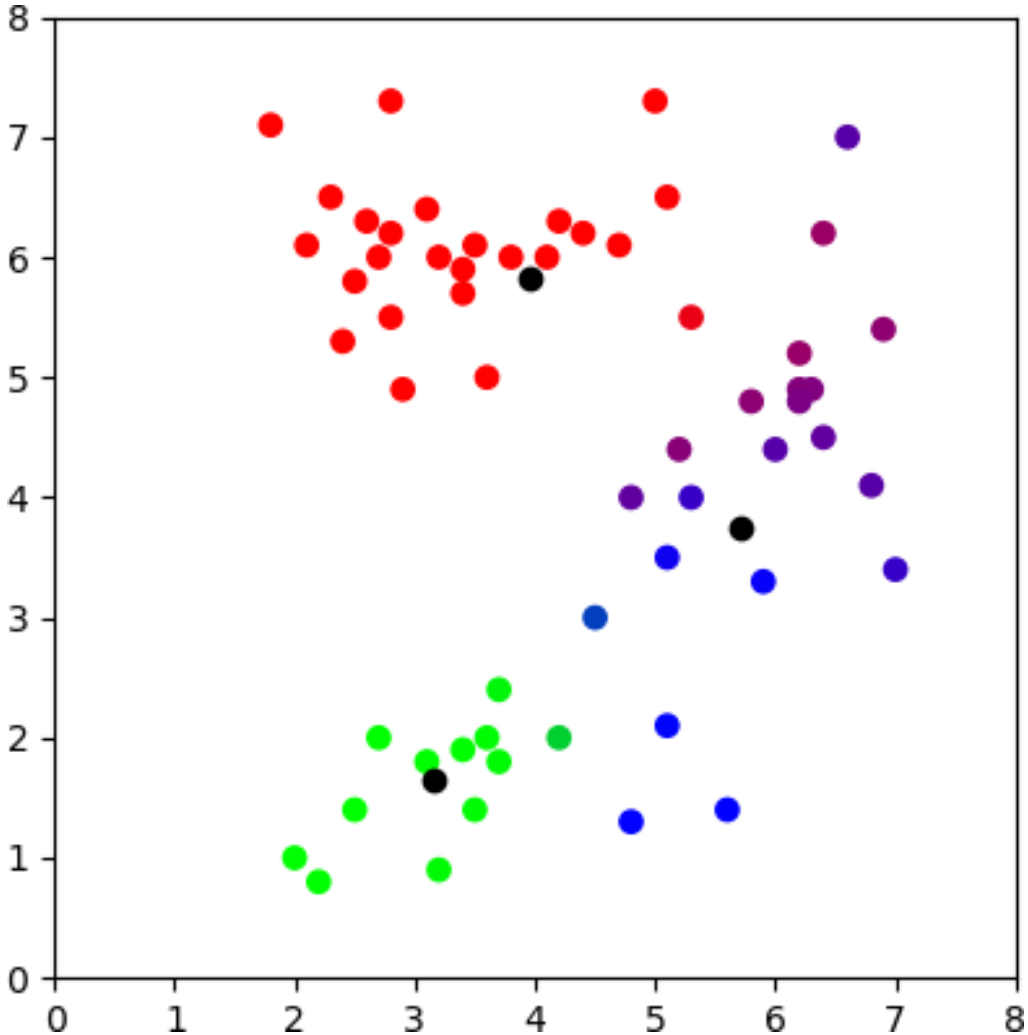
Example Run for $K = 3$

- Black: $\boldsymbol{\mu}_k$ (same starting positions as for k-means)
- Sample chromaticities: Color-coded responsibilities (base colors: red, green, blue)



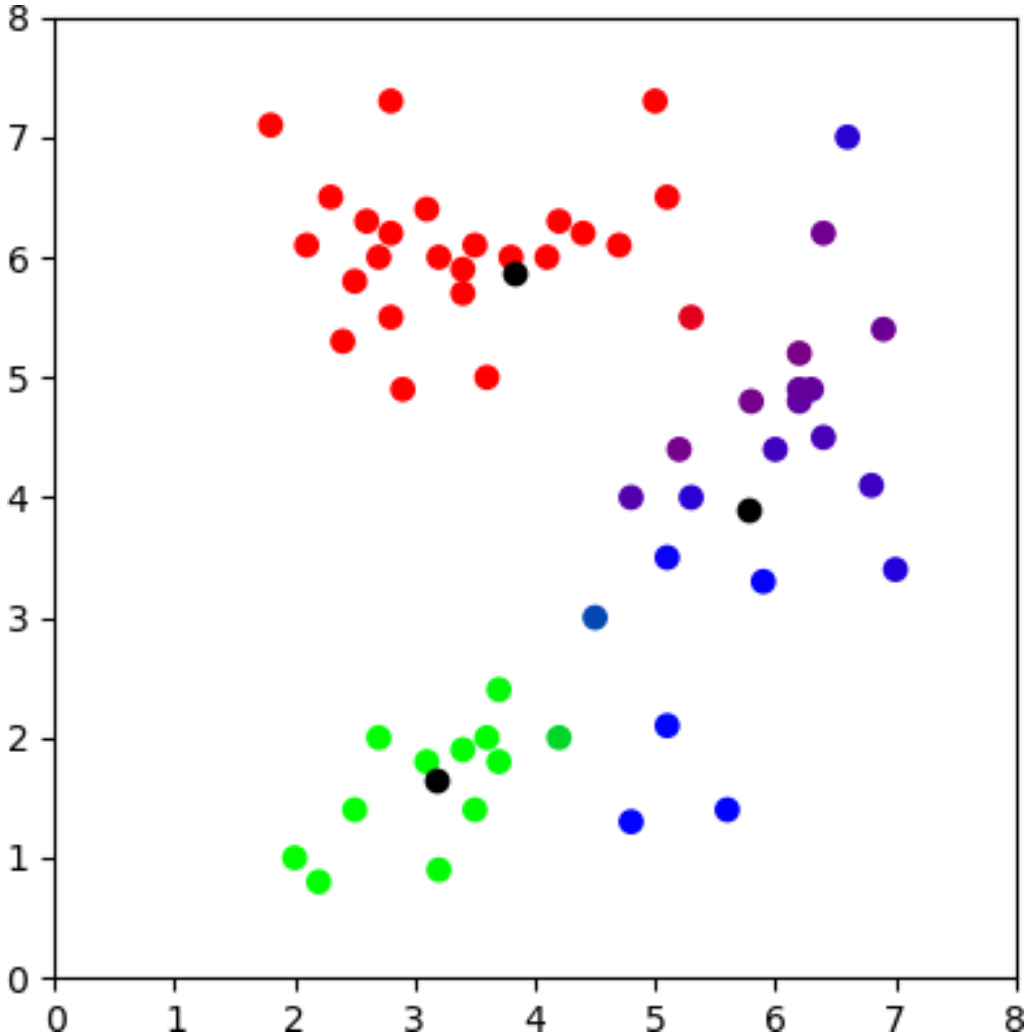
Example Run for $K = 3$

- Black: $\boldsymbol{\mu}_k$ (same starting positions as for k-means)
- Sample chromaticities: Color-coded responsibilities (base colors: red, green, blue)



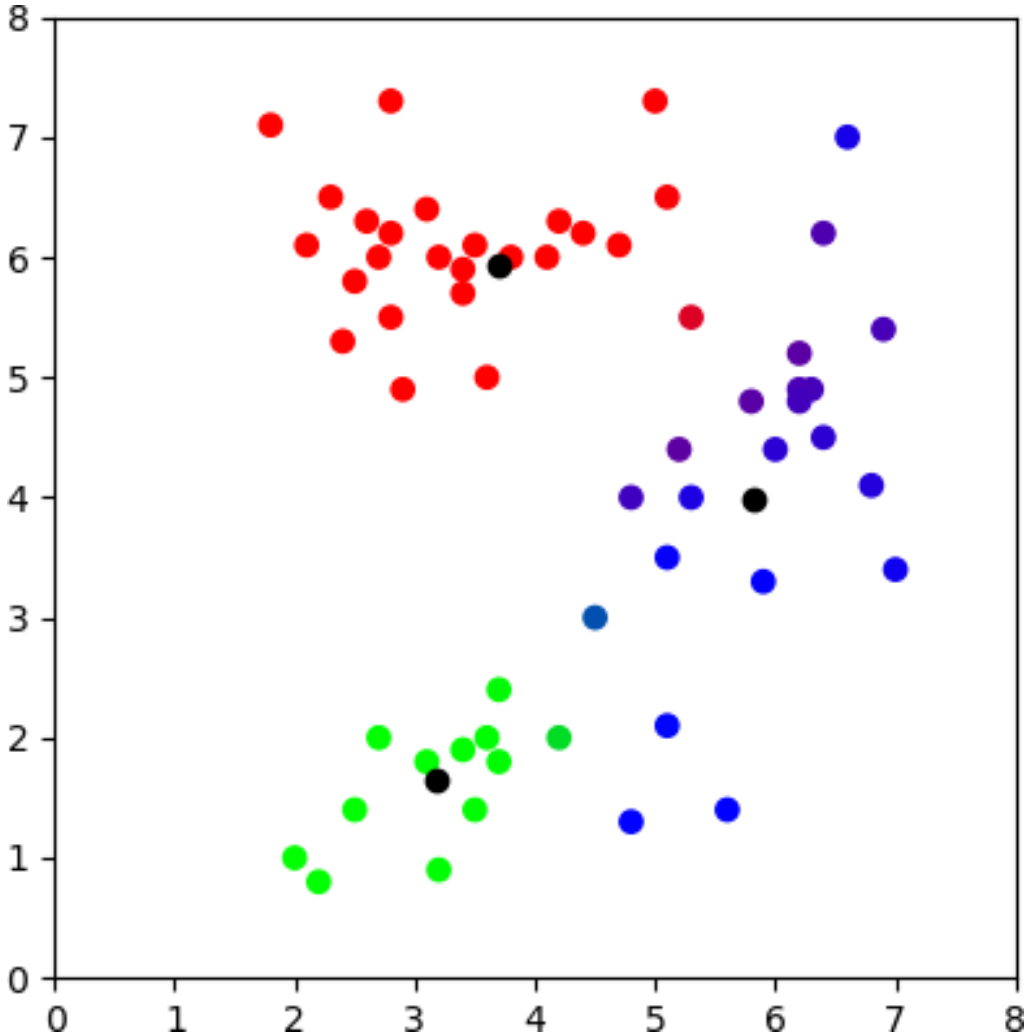
Example Run for $K = 3$

- Black: $\boldsymbol{\mu}_k$ (same starting positions as for k-means)
- Sample chromaticities: Color-coded responsibilities (base colors: red, green, blue)



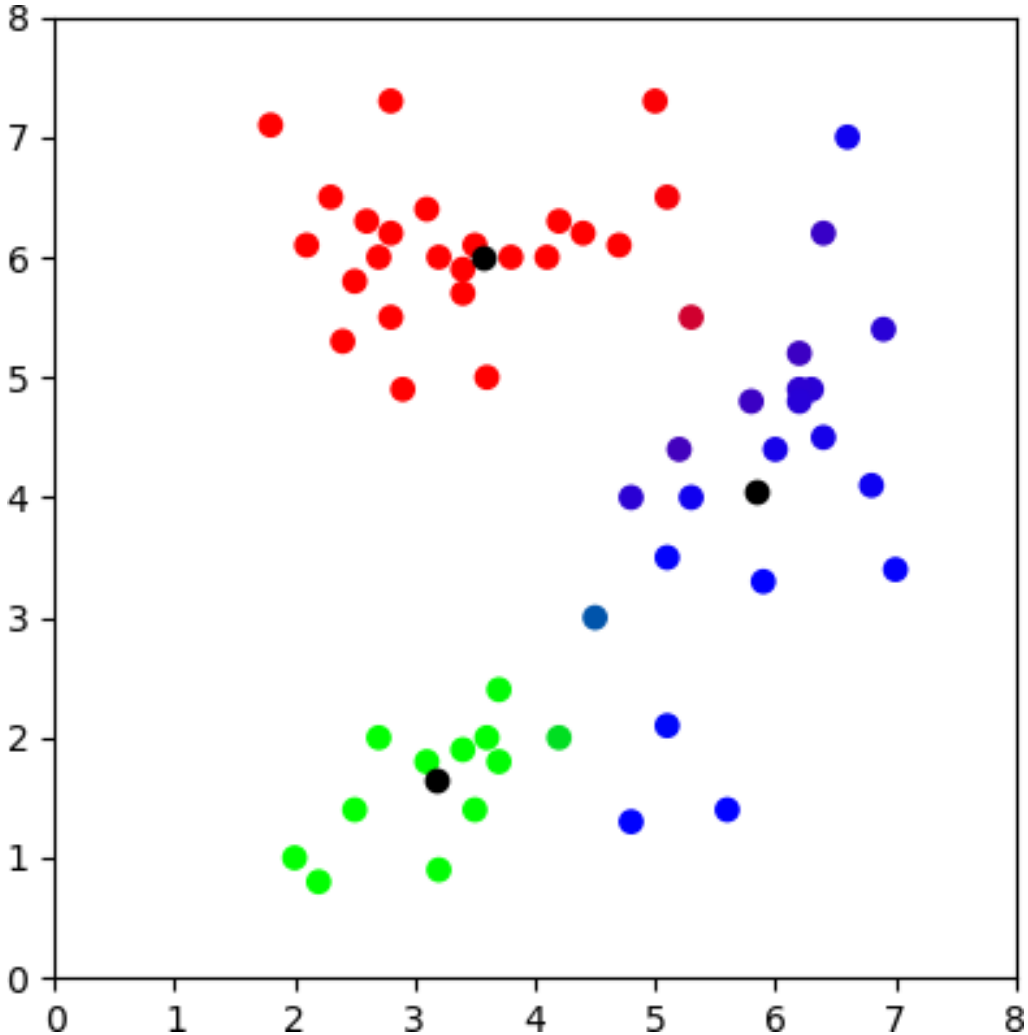
Example Run for $K = 3$

- Black: $\boldsymbol{\mu}_k$ (same starting positions as for k-means)
- Sample chromaticities: Color-coded responsibilities (base colors: red, green, blue)



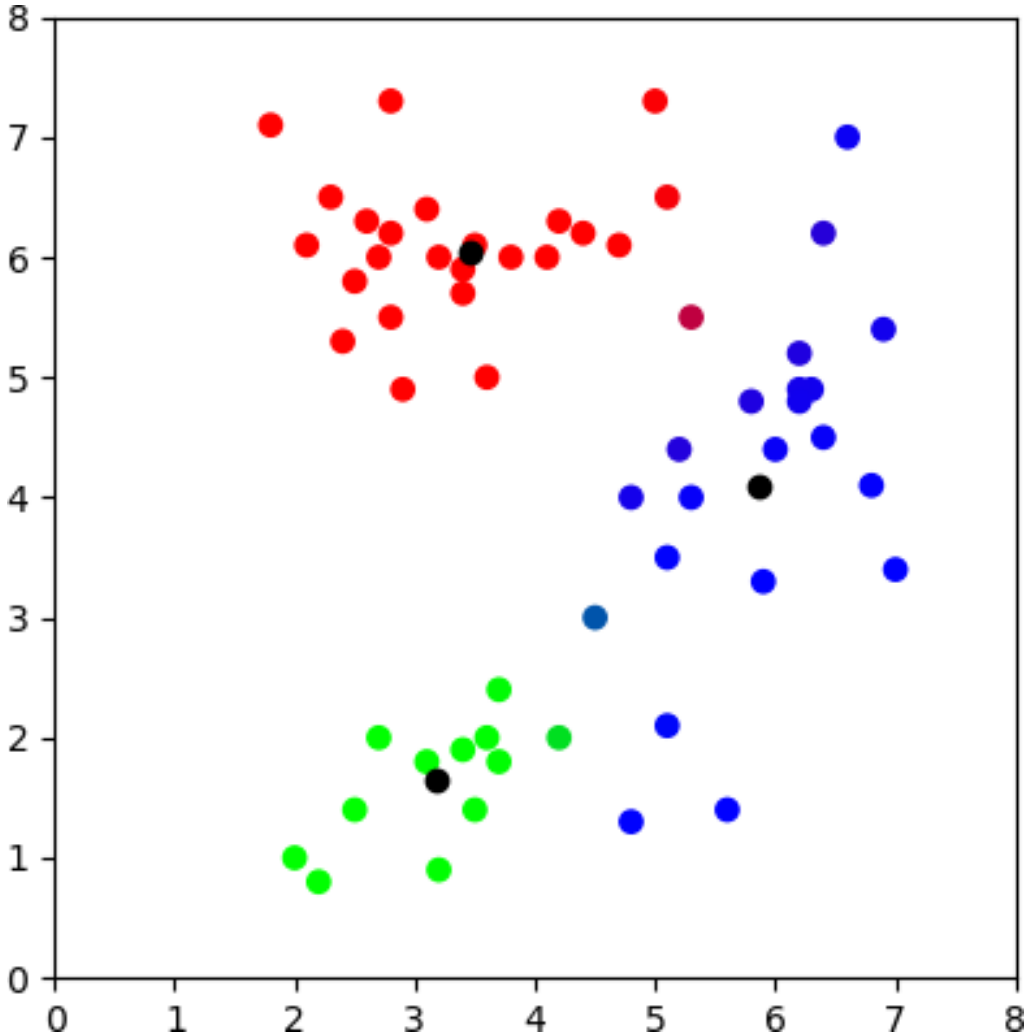
Example Run for $K = 3$

- Black: $\boldsymbol{\mu}_k$ (same starting positions as for k-means)
- Sample chromaticities: Color-coded responsibilities (base colors: red, green, blue)



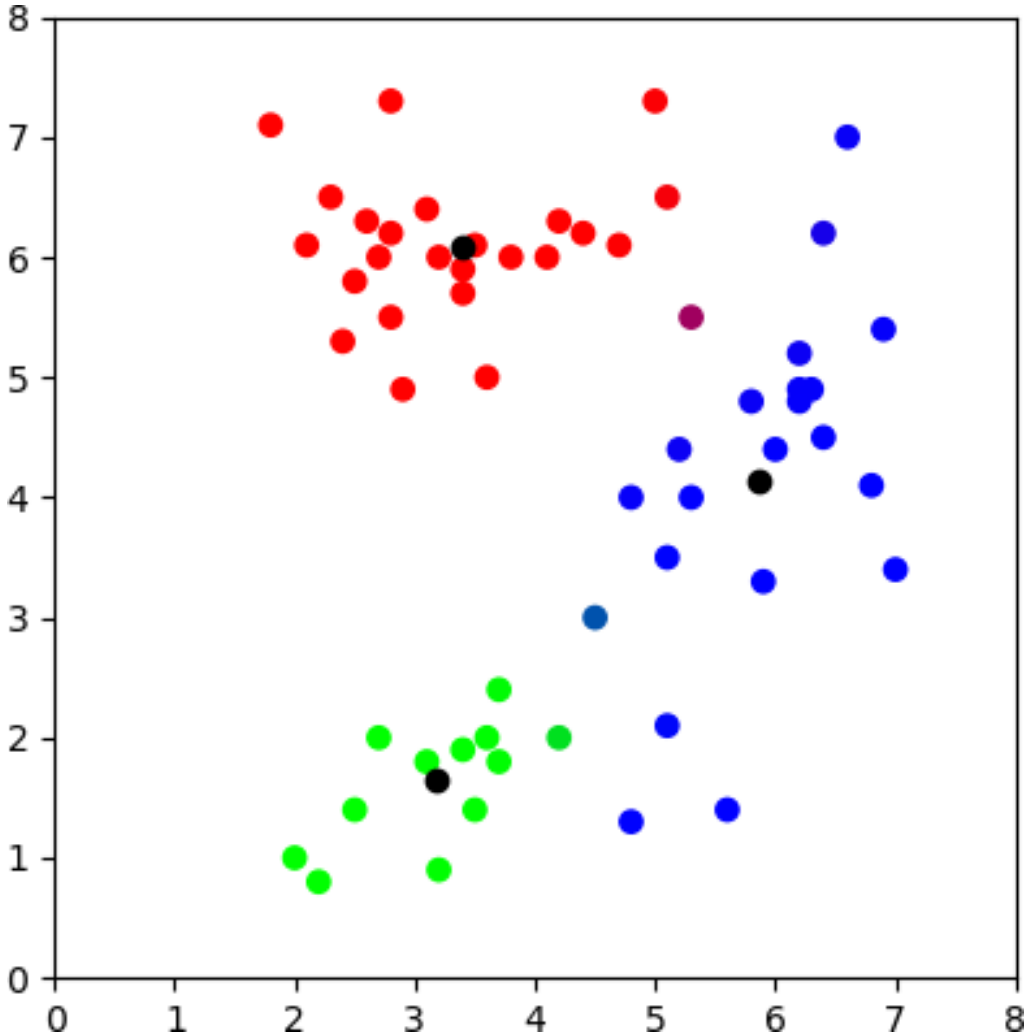
Example Run for $K = 3$

- Black: $\boldsymbol{\mu}_k$ (same starting positions as for k-means)
- Sample chromaticities: Color-coded responsibilities (base colors: red, green, blue)



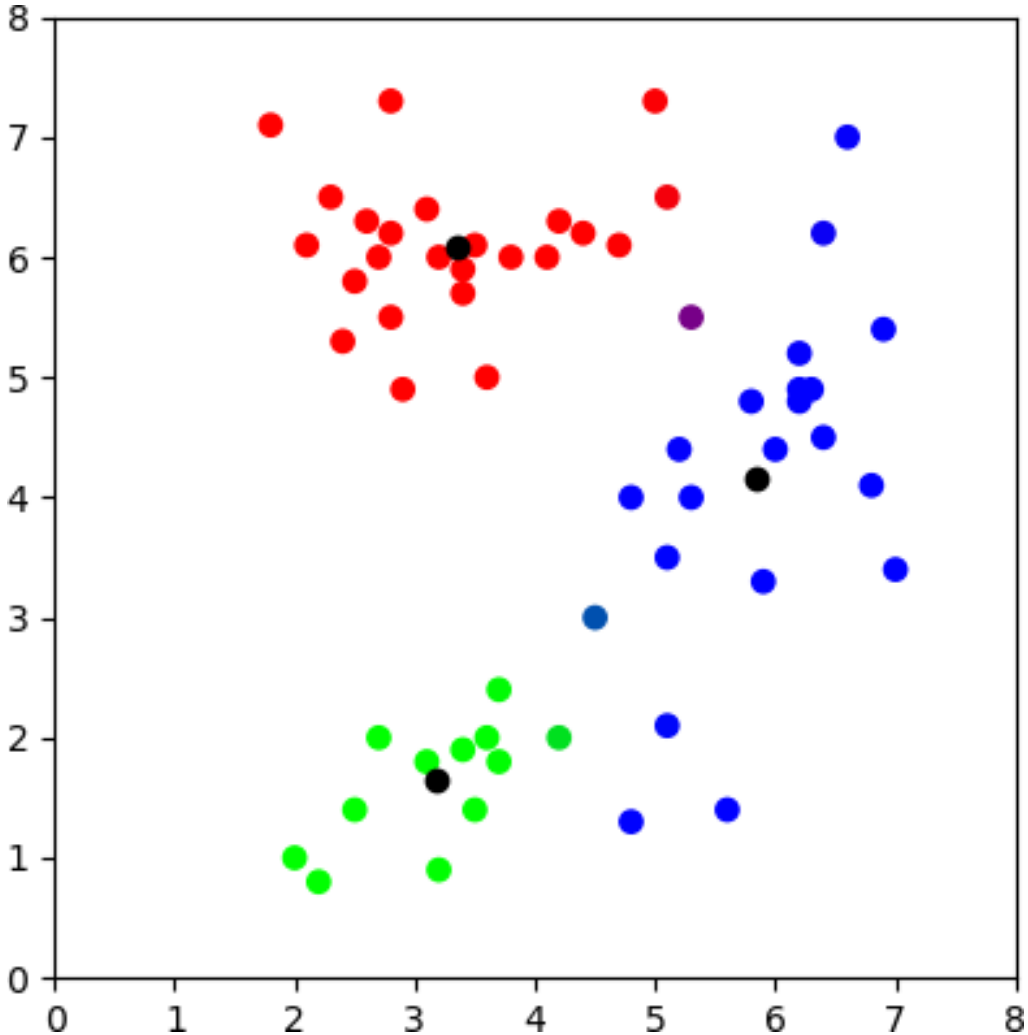
Example Run for $K = 3$

- Black: $\boldsymbol{\mu}_k$ (same starting positions as for k-means)
- Sample chromaticities: Color-coded responsibilities (base colors: red, green, blue)



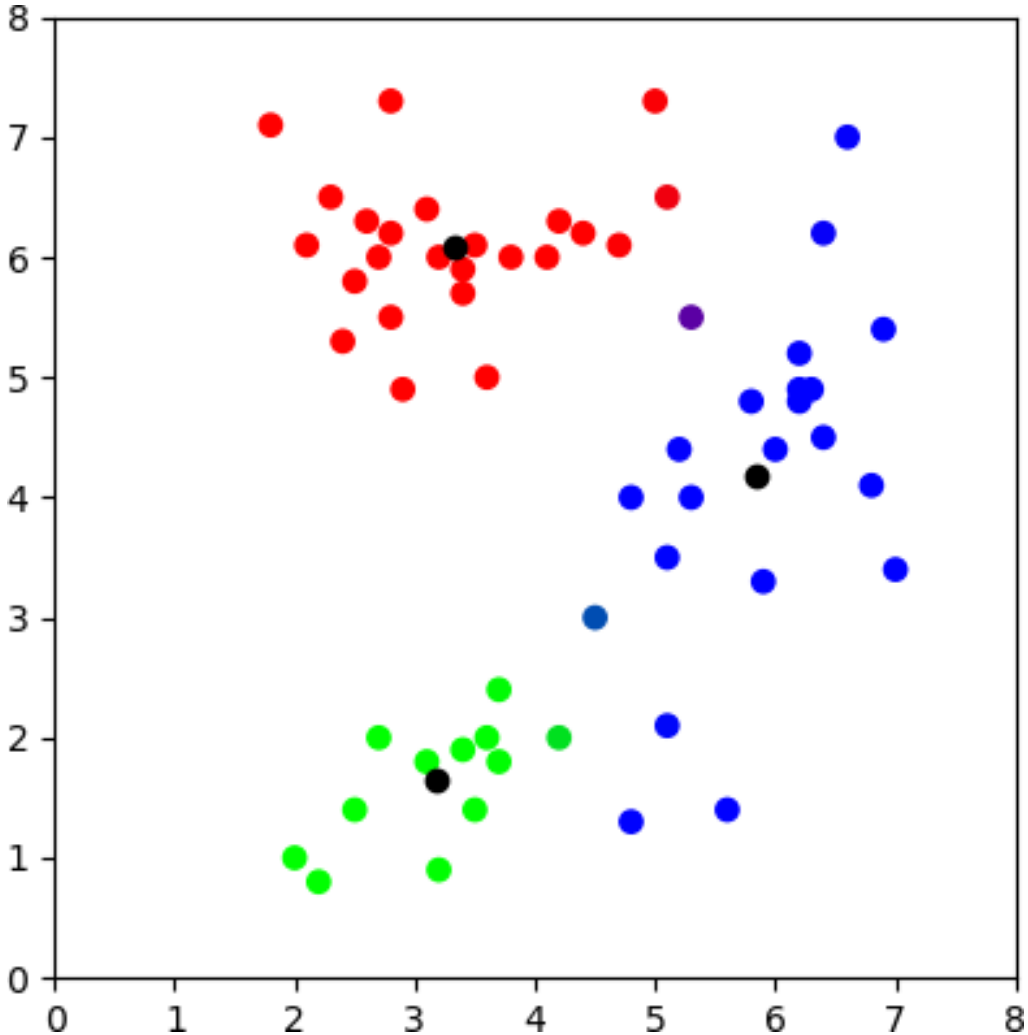
Example Run for $K = 3$

- Black: $\boldsymbol{\mu}_k$ (same starting positions as for k-means)
- Sample chromaticities: Color-coded responsibilities (base colors: red, green, blue)



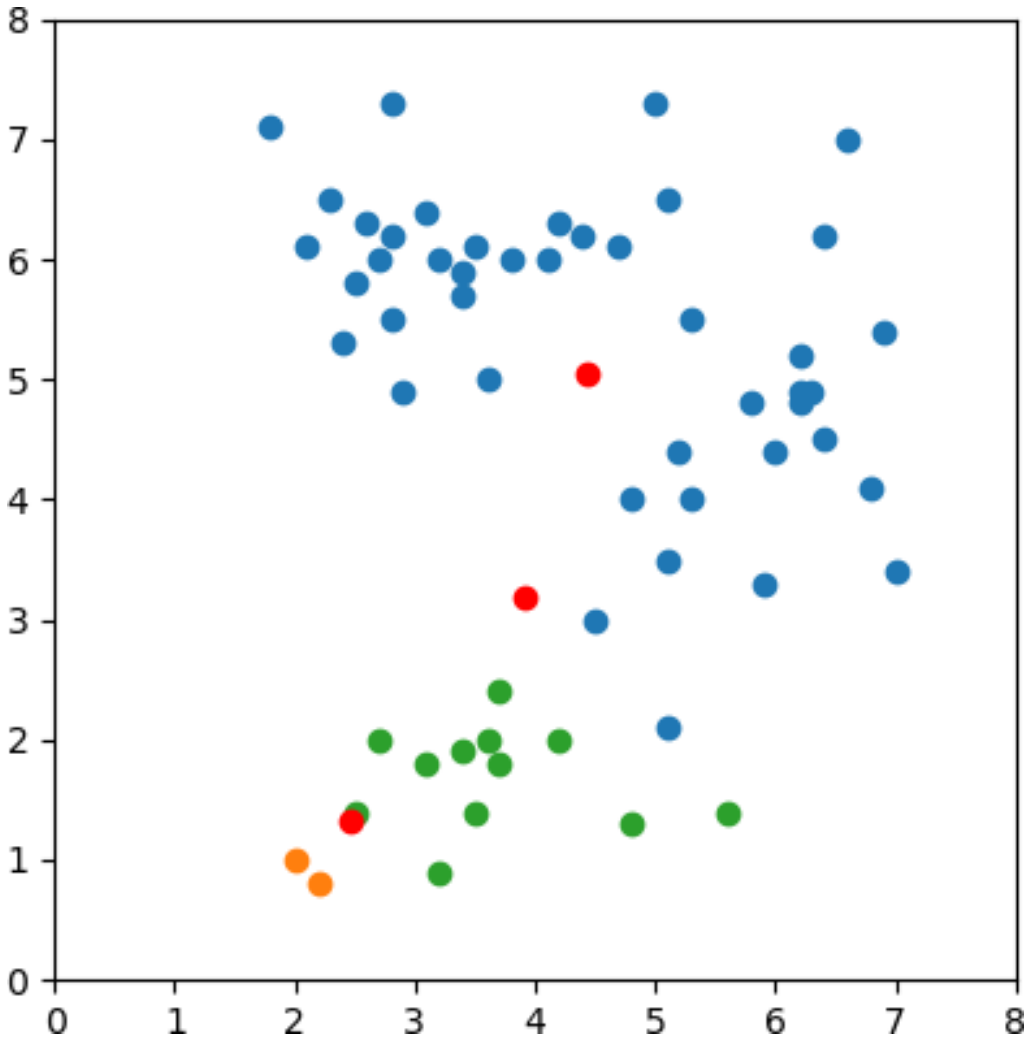
Example Run for $K = 3$

- Black: $\boldsymbol{\mu}_k$ (same starting positions as for k-means)
- Sample chromaticities: Color-coded responsibilities (base colors: red, green, blue)



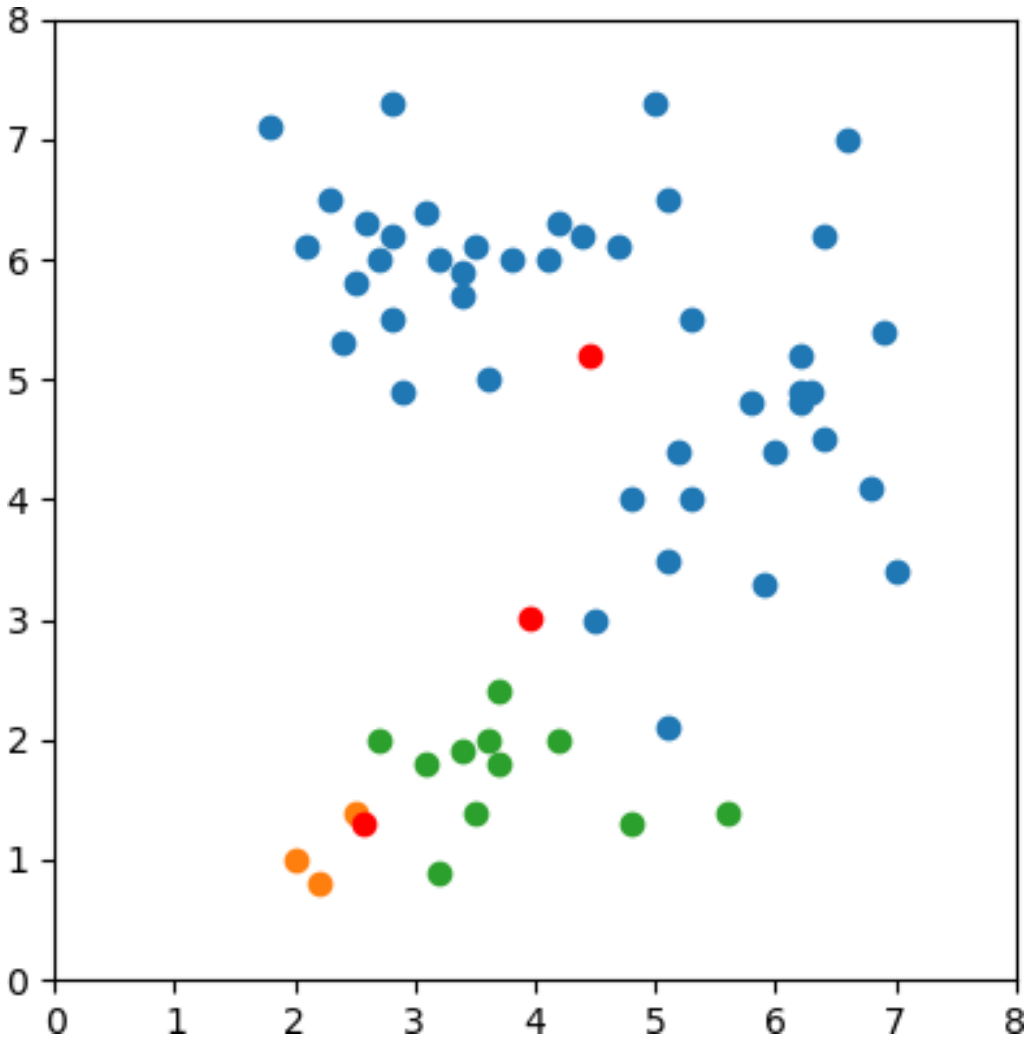
Example Run for $K = 3$, ML label assignment

- Identical run, but visualization shows component color of maximum responsibility
- Safety notice: only take the maximum on the output. The iteration itself has to use continuous responsibilities



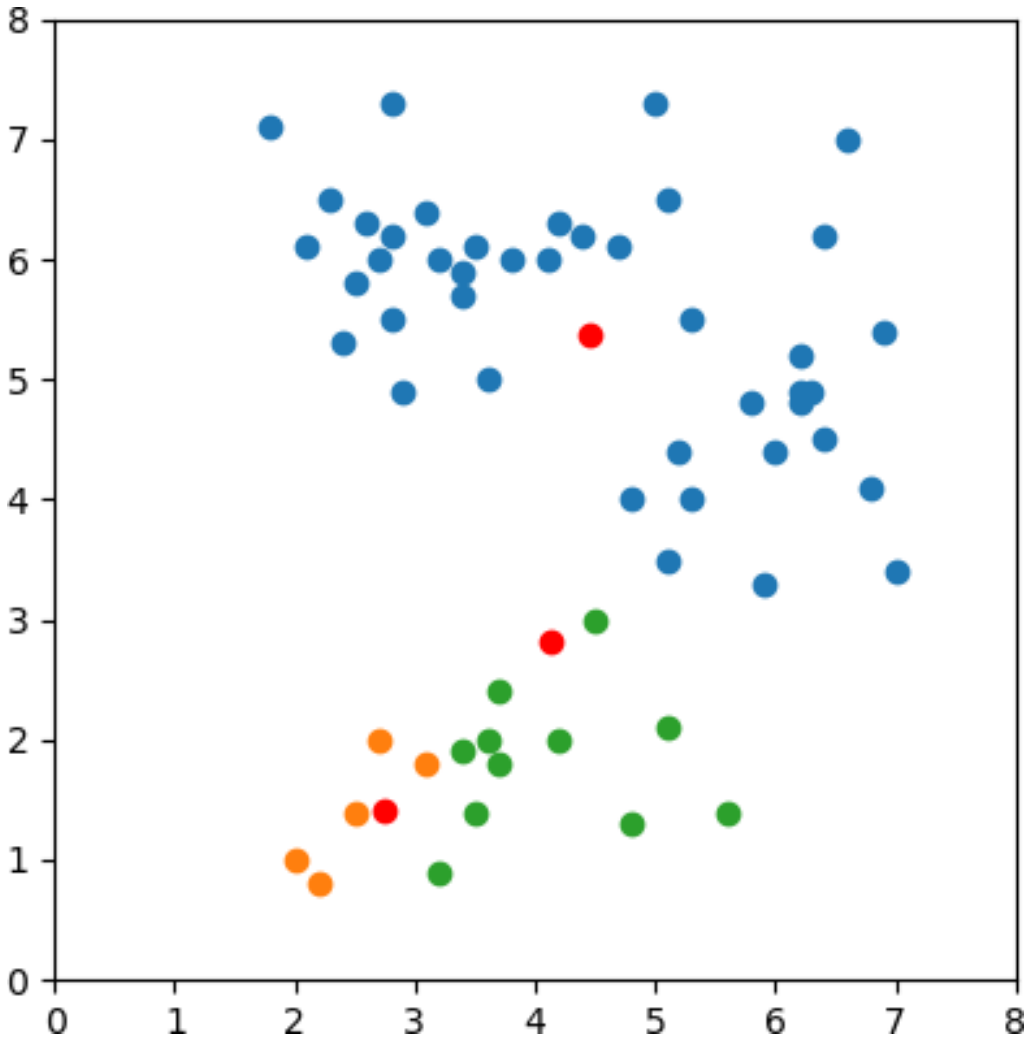
Example Run for $K = 3$, ML label assignment

- Identical run, but visualization shows component color of maximum responsibility
- Safety notice: only take the maximum on the output. The iteration itself has to use continuous responsibilities



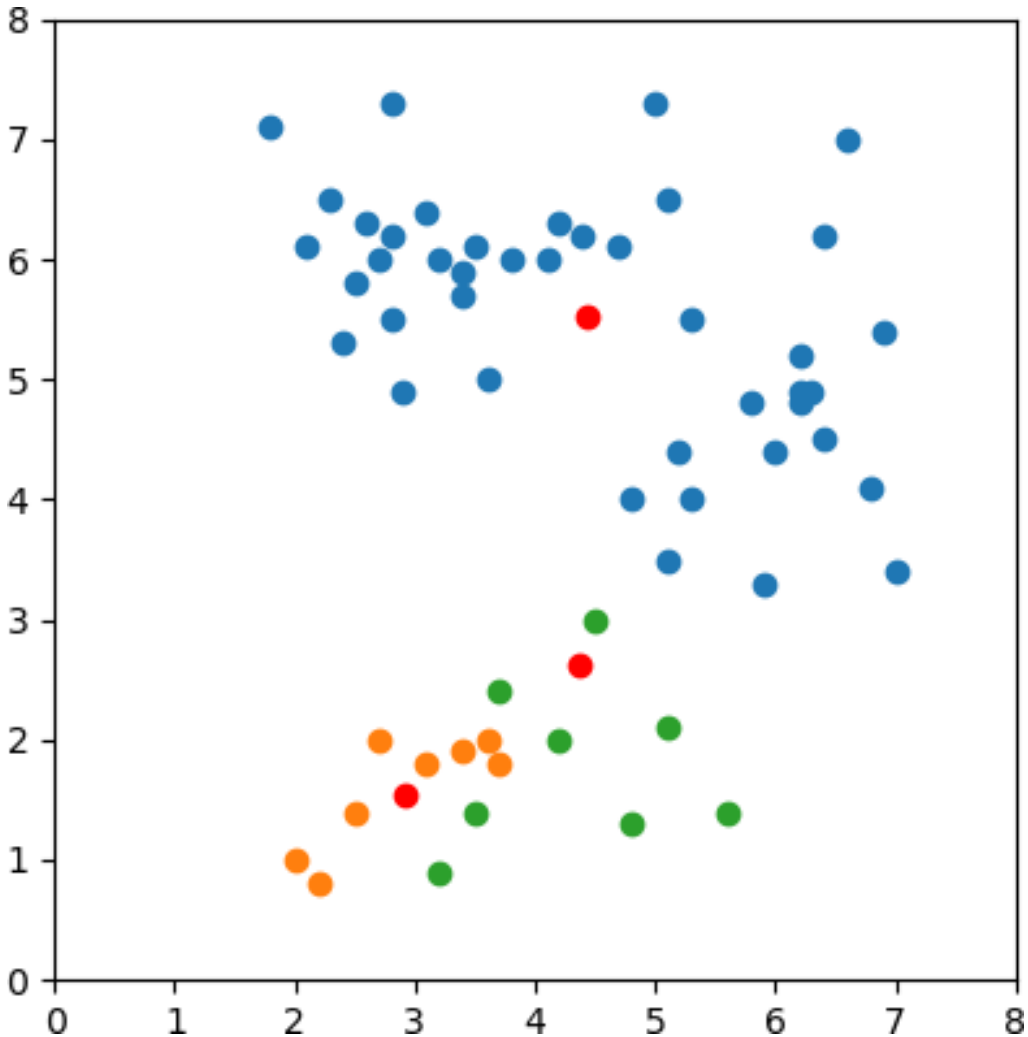
Example Run for $K = 3$, ML label assignment

- Identical run, but visualization shows component color of maximum responsibility
- Safety notice: only take the maximum on the output. The iteration itself has to use continuous responsibilities



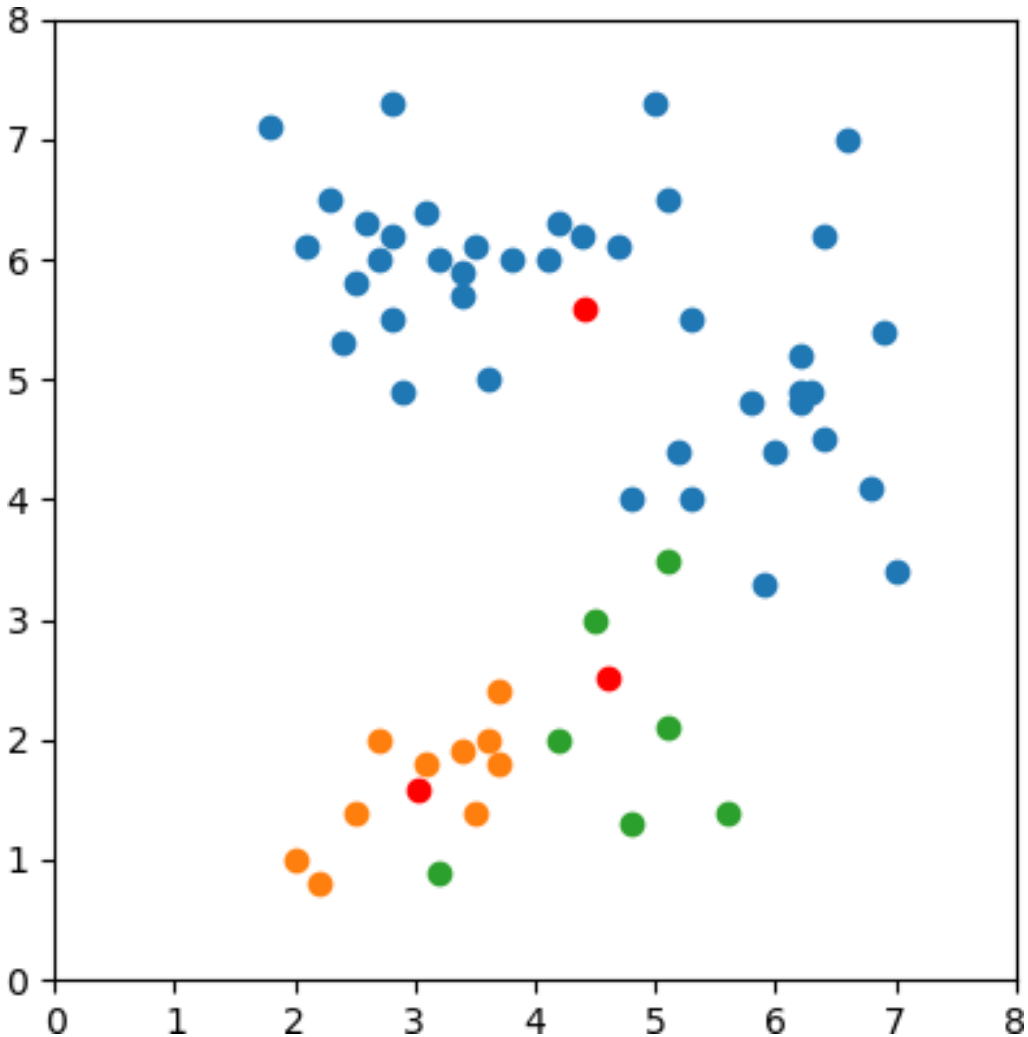
Example Run for $K = 3$, ML label assignment

- Identical run, but visualization shows component color of maximum responsibility
- Safety notice: only take the maximum on the output. The iteration itself has to use continuous responsibilities



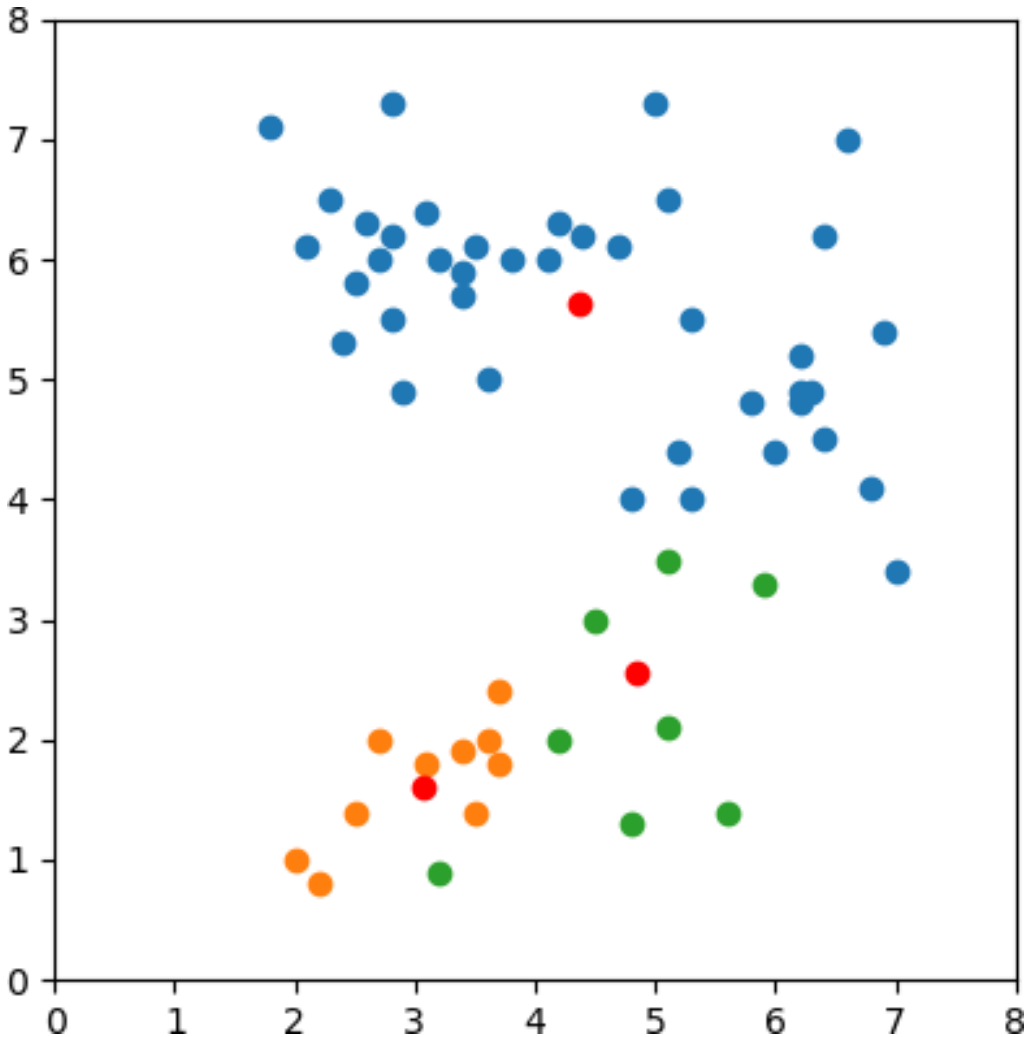
Example Run for $K = 3$, ML label assignment

- Identical run, but visualization shows component color of maximum responsibility
- Safety notice: only take the maximum on the output. The iteration itself has to use continuous responsibilities



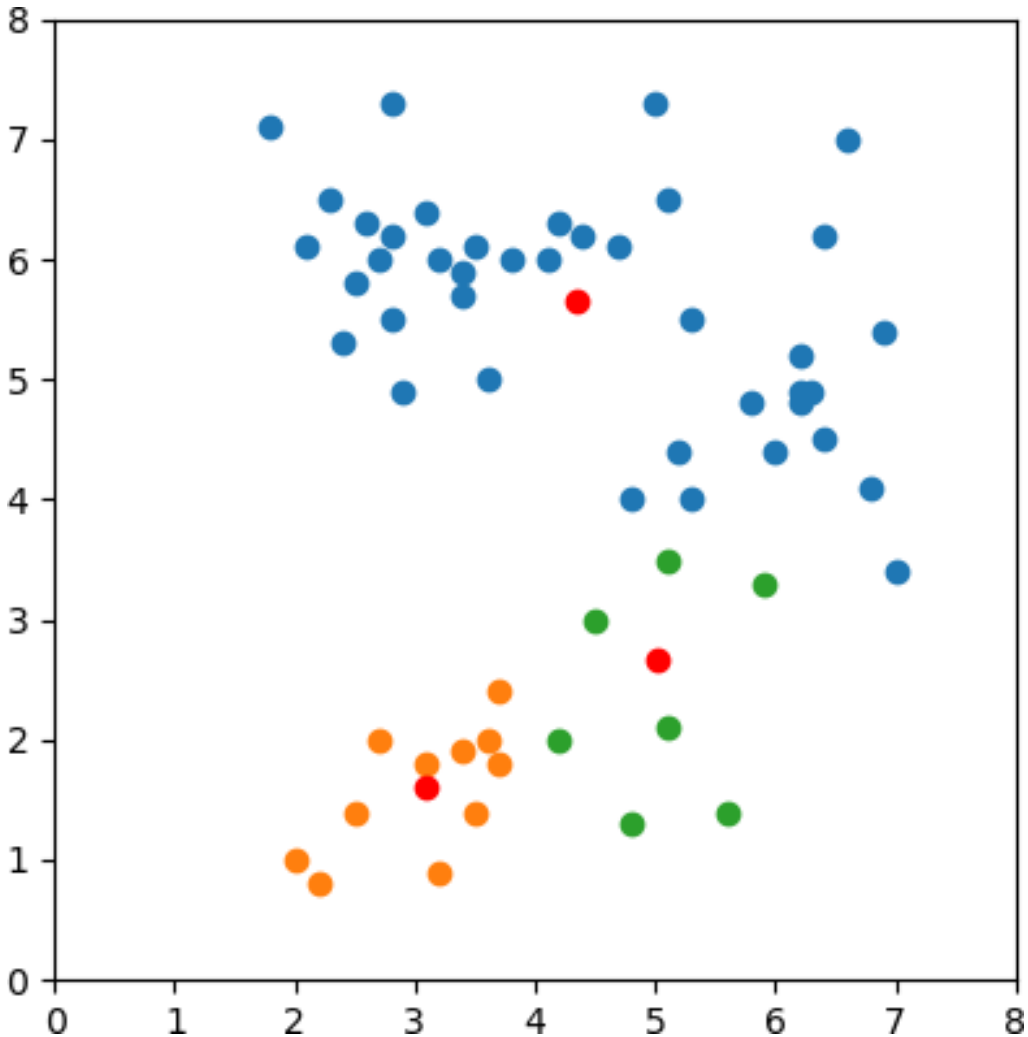
Example Run for $K = 3$, ML label assignment

- Identical run, but visualization shows component color of maximum responsibility
- Safety notice: only take the maximum on the output. The iteration itself has to use continuous responsibilities



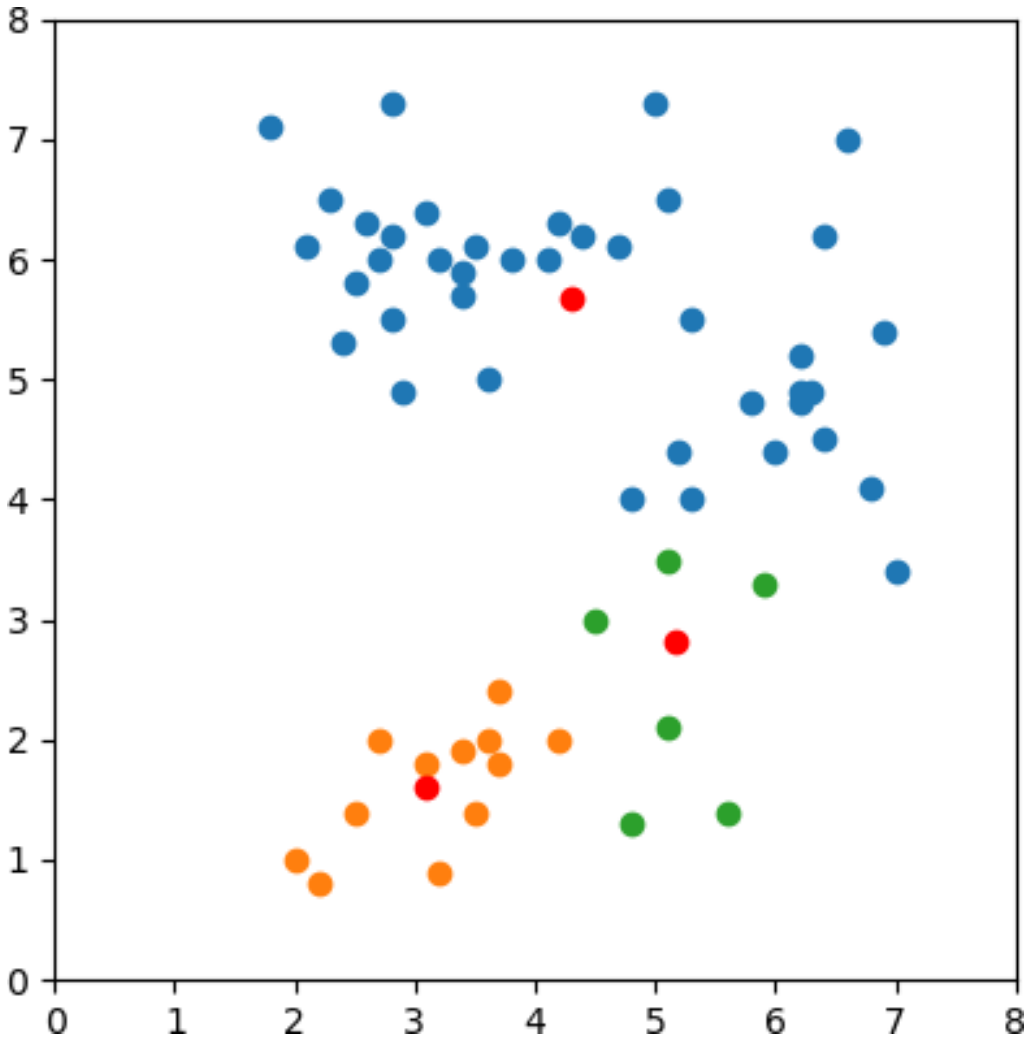
Example Run for $K = 3$, ML label assignment

- Identical run, but visualization shows component color of maximum responsibility
- Safety notice: only take the maximum on the output. The iteration itself has to use continuous responsibilities



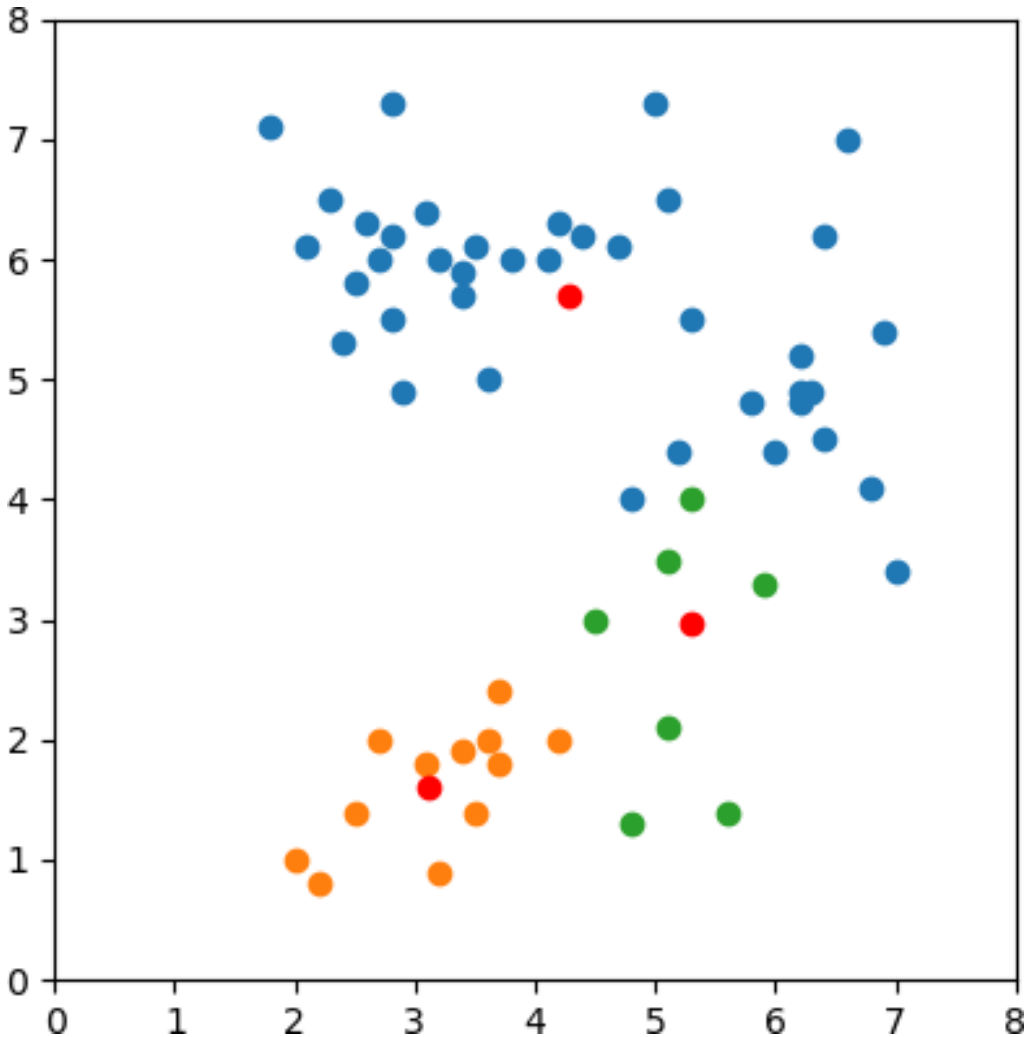
Example Run for $K = 3$, ML label assignment

- Identical run, but visualization shows component color of maximum responsibility
- Safety notice: only take the maximum on the output. The iteration itself has to use continuous responsibilities



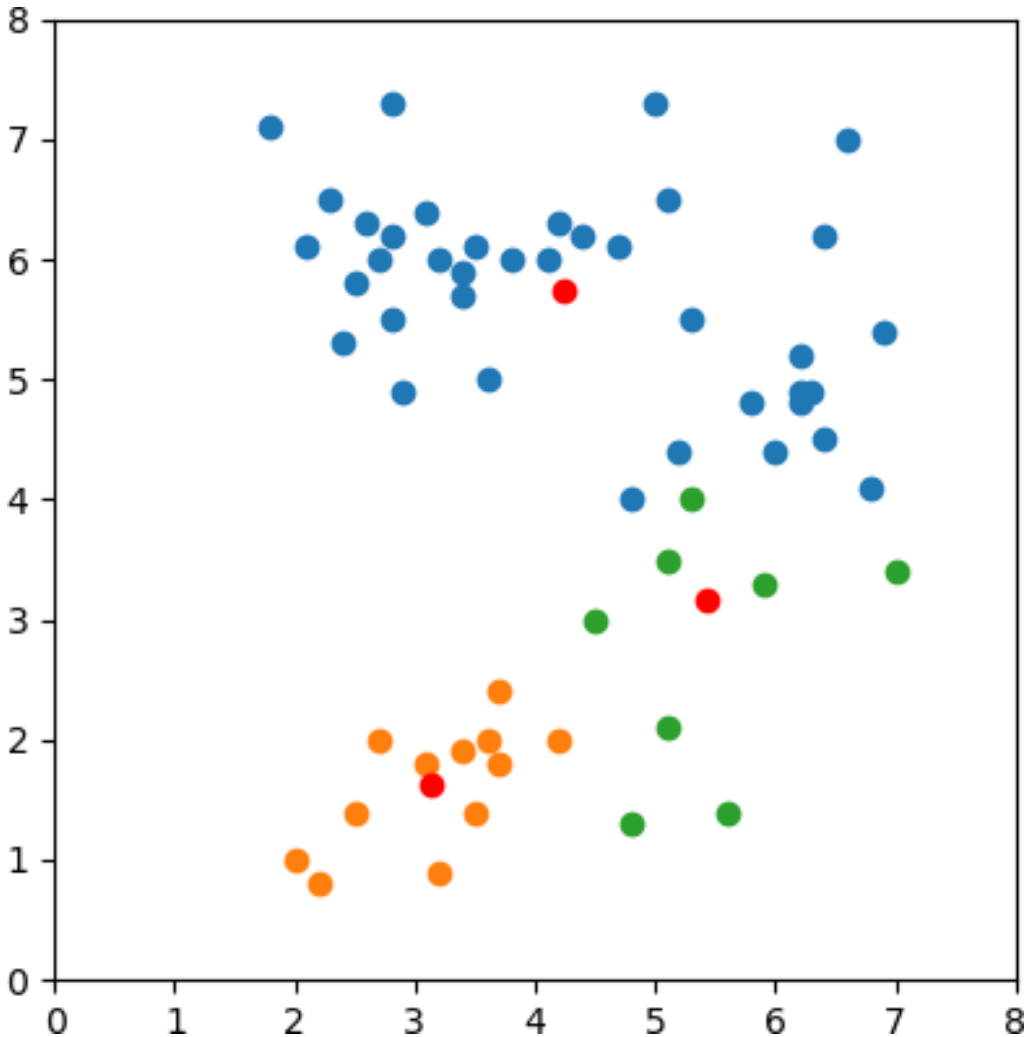
Example Run for $K = 3$, ML label assignment

- Identical run, but visualization shows component color of maximum responsibility
- Safety notice: only take the maximum on the output. The iteration itself has to use continuous responsibilities



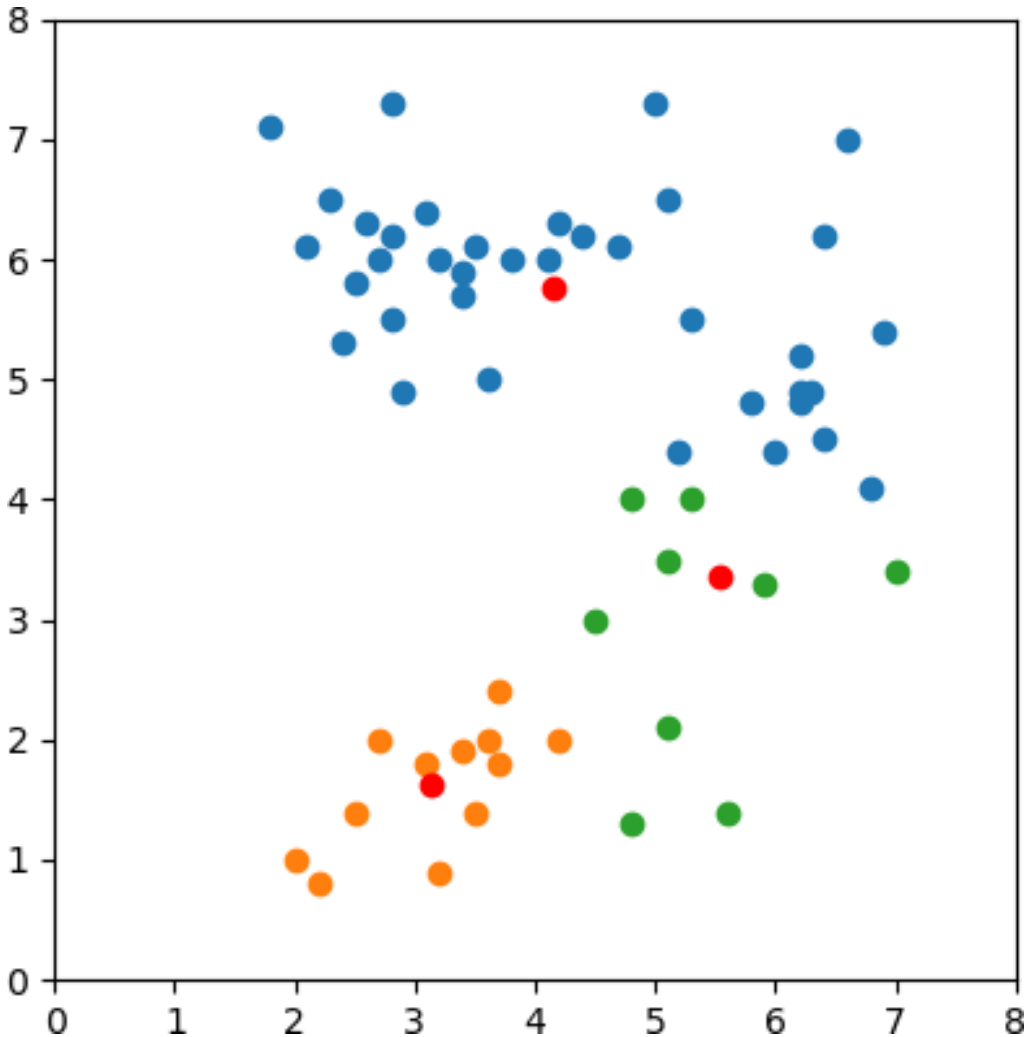
Example Run for $K = 3$, ML label assignment

- Identical run, but visualization shows component color of maximum responsibility
- Safety notice: only take the maximum on the output. The iteration itself has to use continuous responsibilities



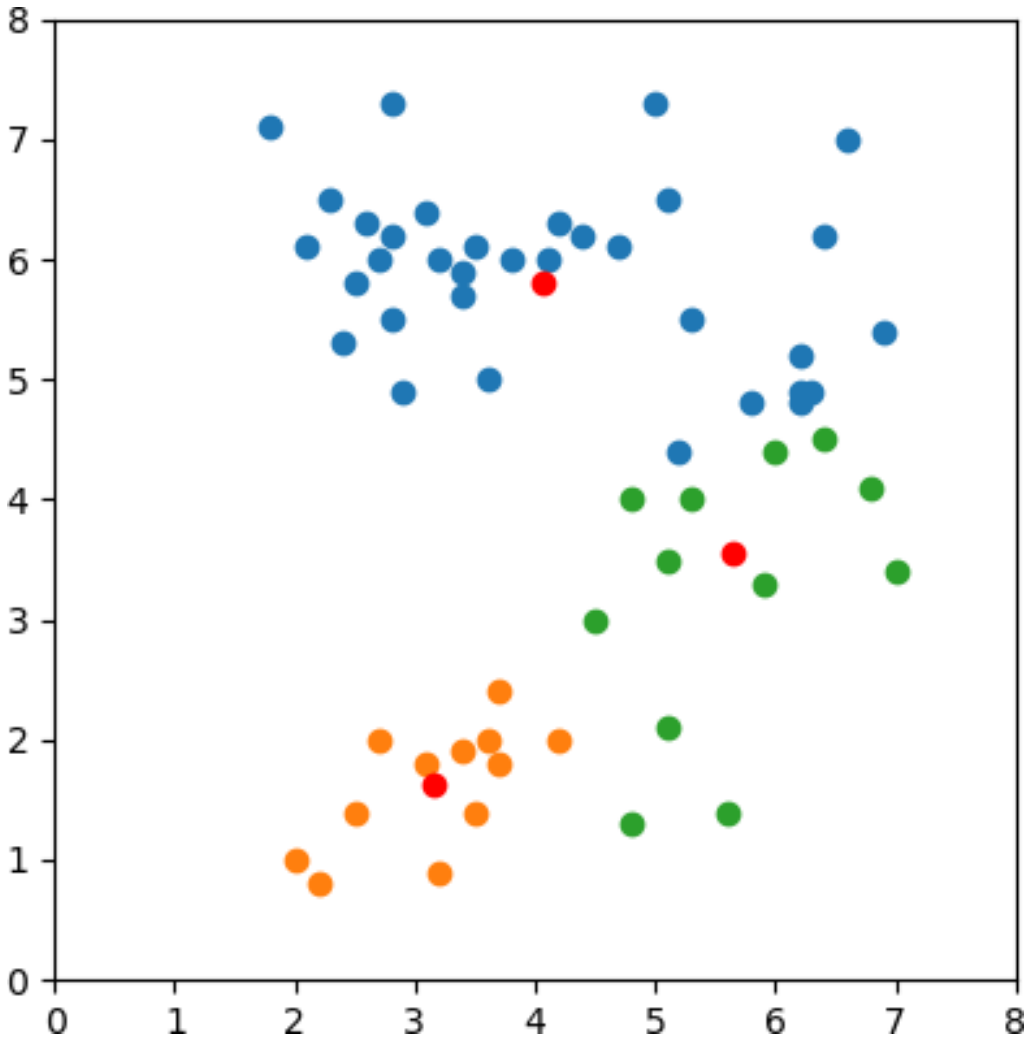
Example Run for $K = 3$, ML label assignment

- Identical run, but visualization shows component color of maximum responsibility
- Safety notice: only take the maximum on the output. The iteration itself has to use continuous responsibilities



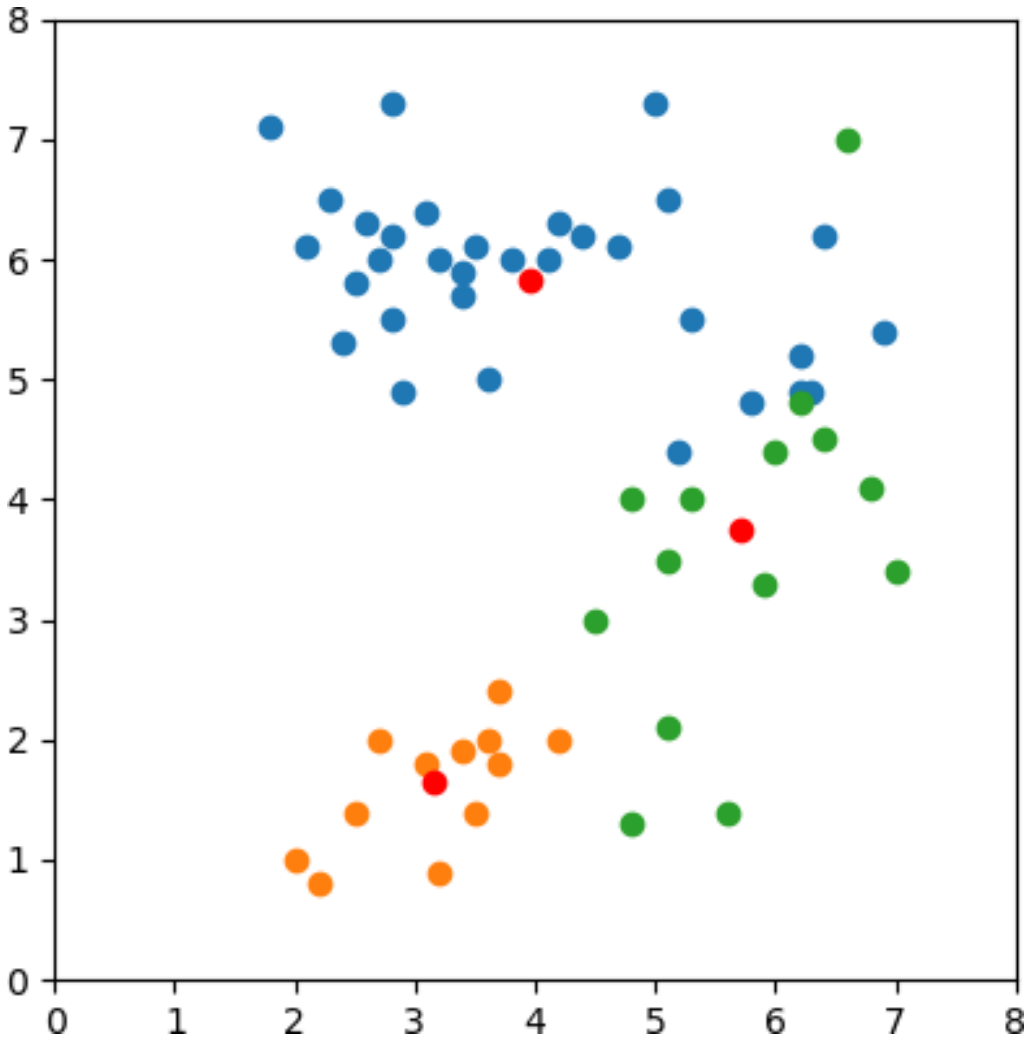
Example Run for $K = 3$, ML label assignment

- Identical run, but visualization shows component color of maximum responsibility
- Safety notice: only take the maximum on the output. The iteration itself has to use continuous responsibilities



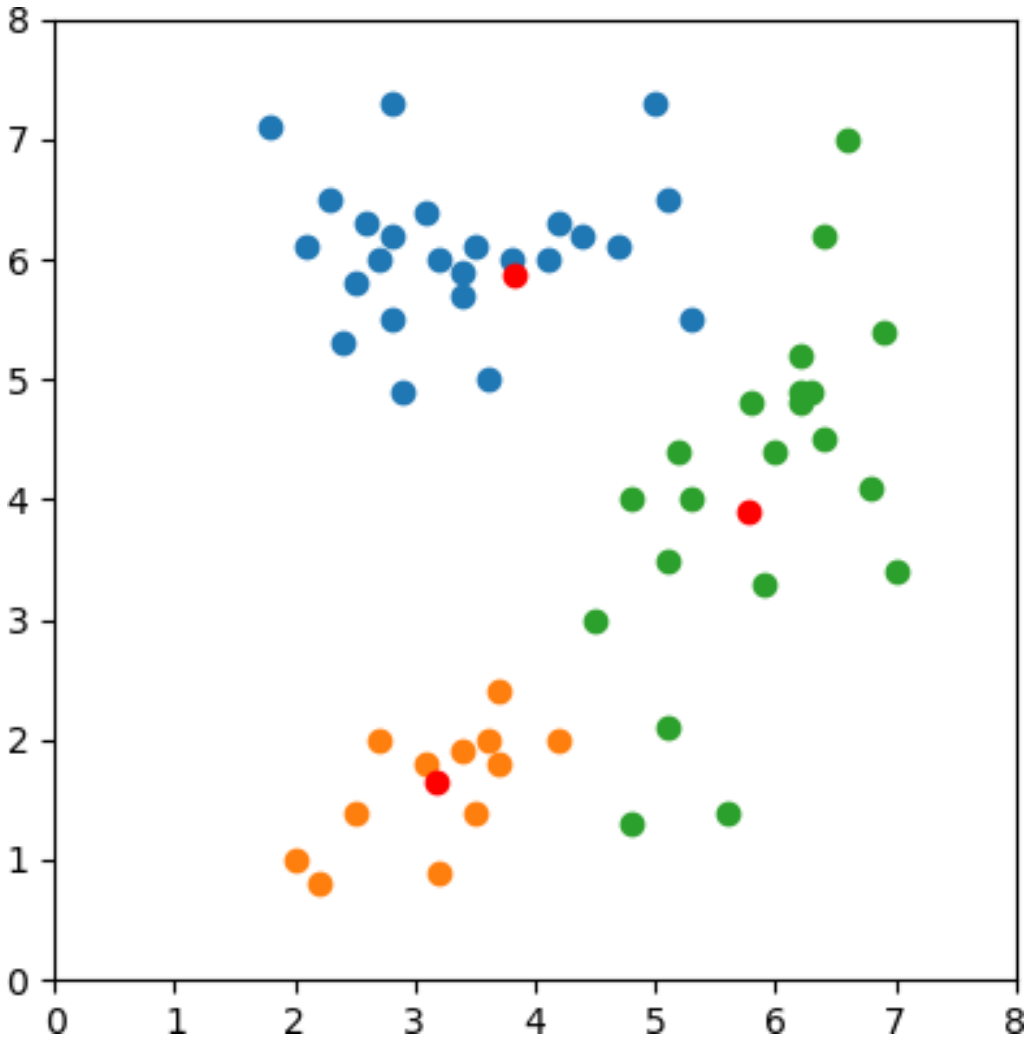
Example Run for $K = 3$, ML label assignment

- Identical run, but visualization shows component color of maximum responsibility
- Safety notice: only take the maximum on the output. The iteration itself has to use continuous responsibilities



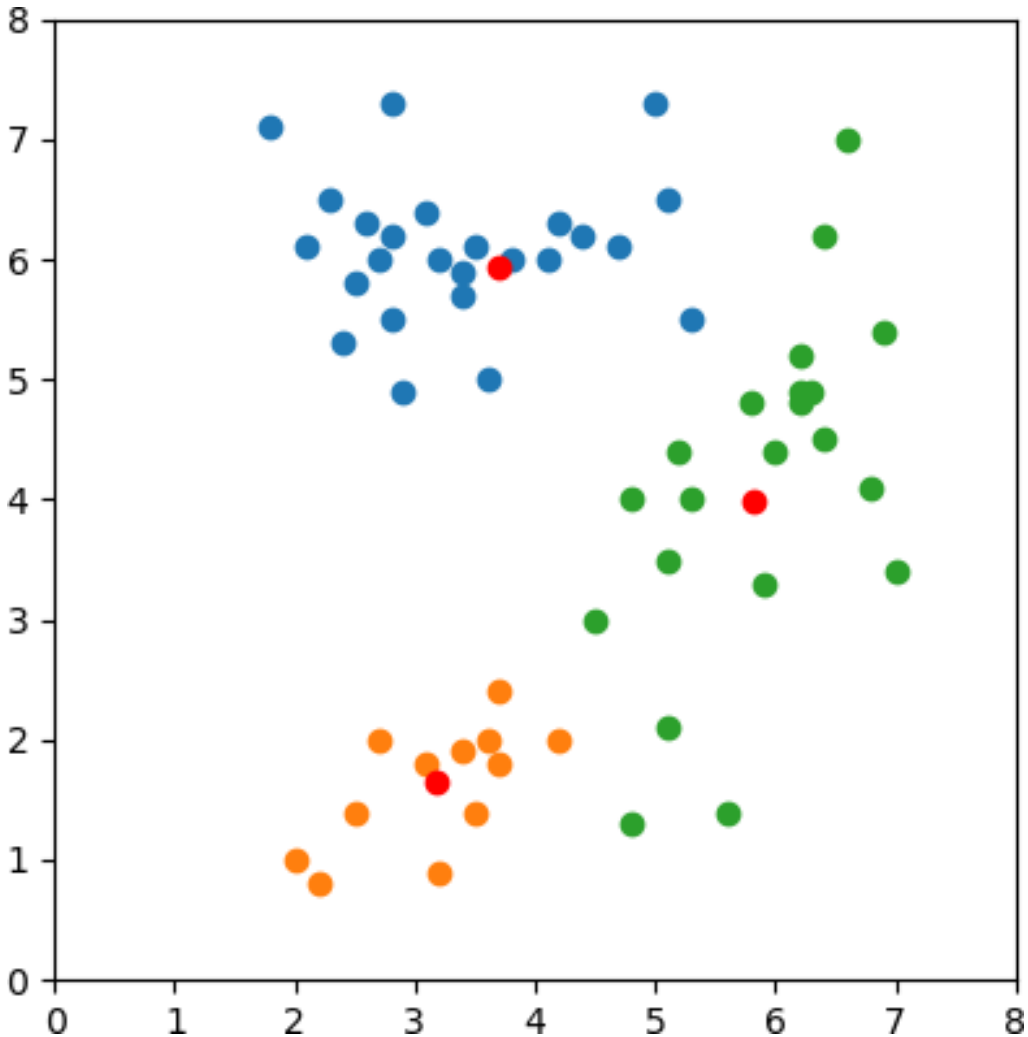
Example Run for $K = 3$, ML label assignment

- Identical run, but visualization shows component color of maximum responsibility
- Safety notice: only take the maximum on the output. The iteration itself has to use continuous responsibilities



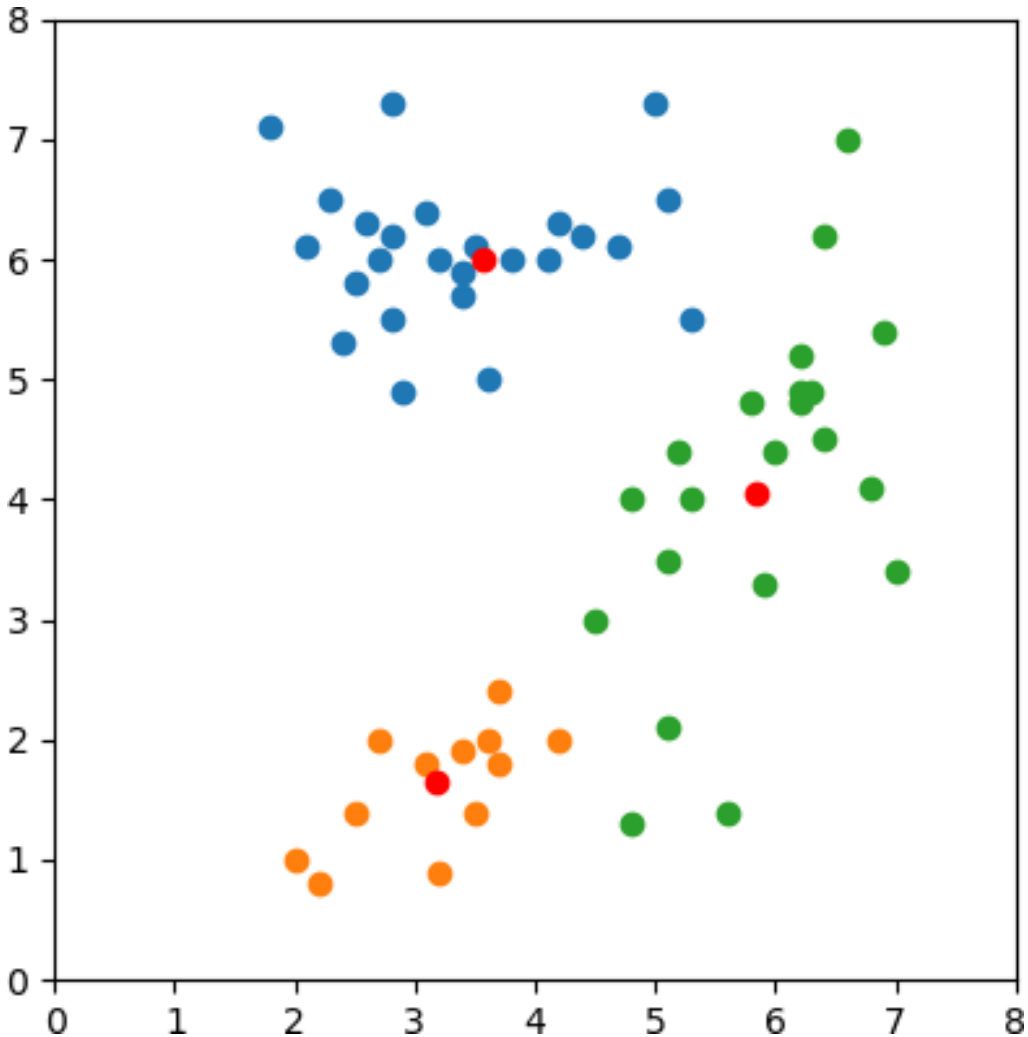
Example Run for $K = 3$, ML label assignment

- Identical run, but visualization shows component color of maximum responsibility
- Safety notice: only take the maximum on the output. The iteration itself has to use continuous responsibilities



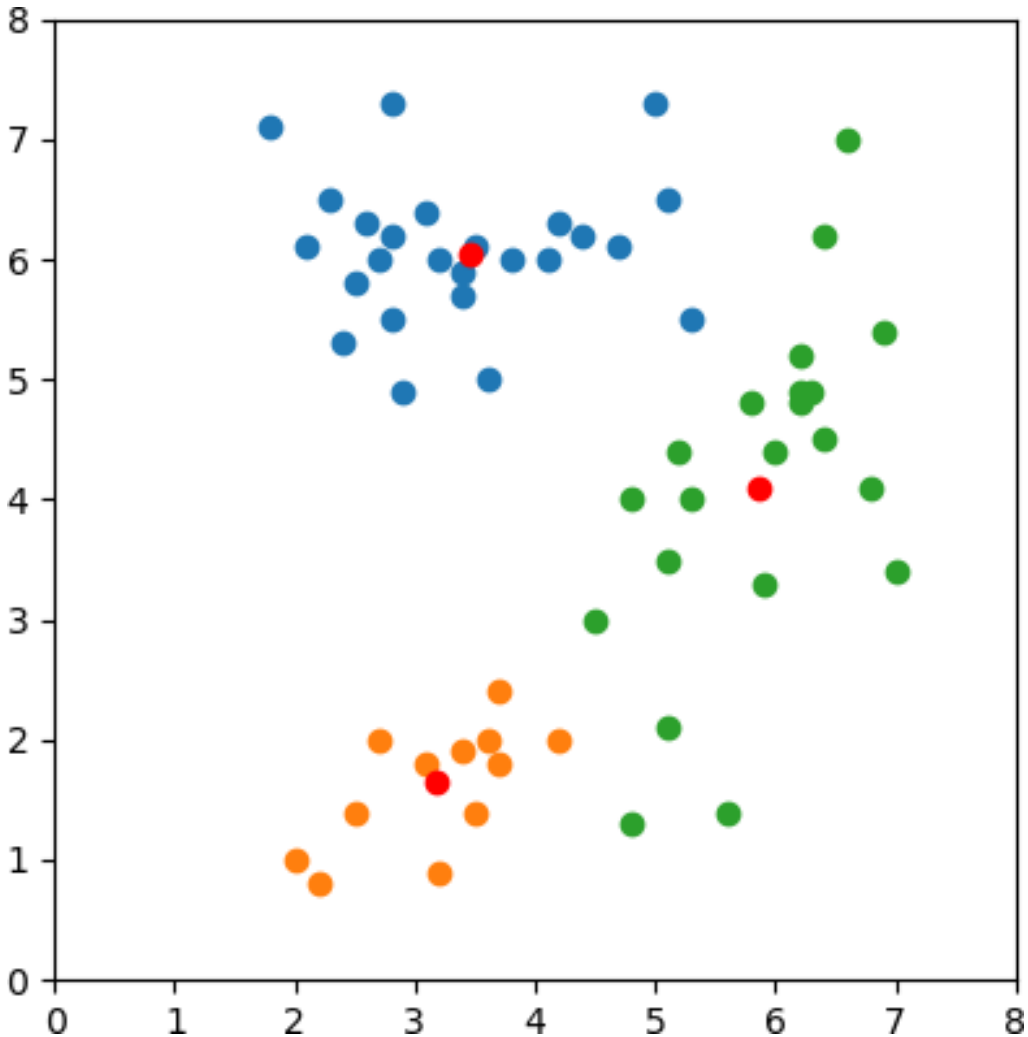
Example Run for $K = 3$, ML label assignment

- Identical run, but visualization shows component color of maximum responsibility
- Safety notice: only take the maximum on the output. The iteration itself has to use continuous responsibilities



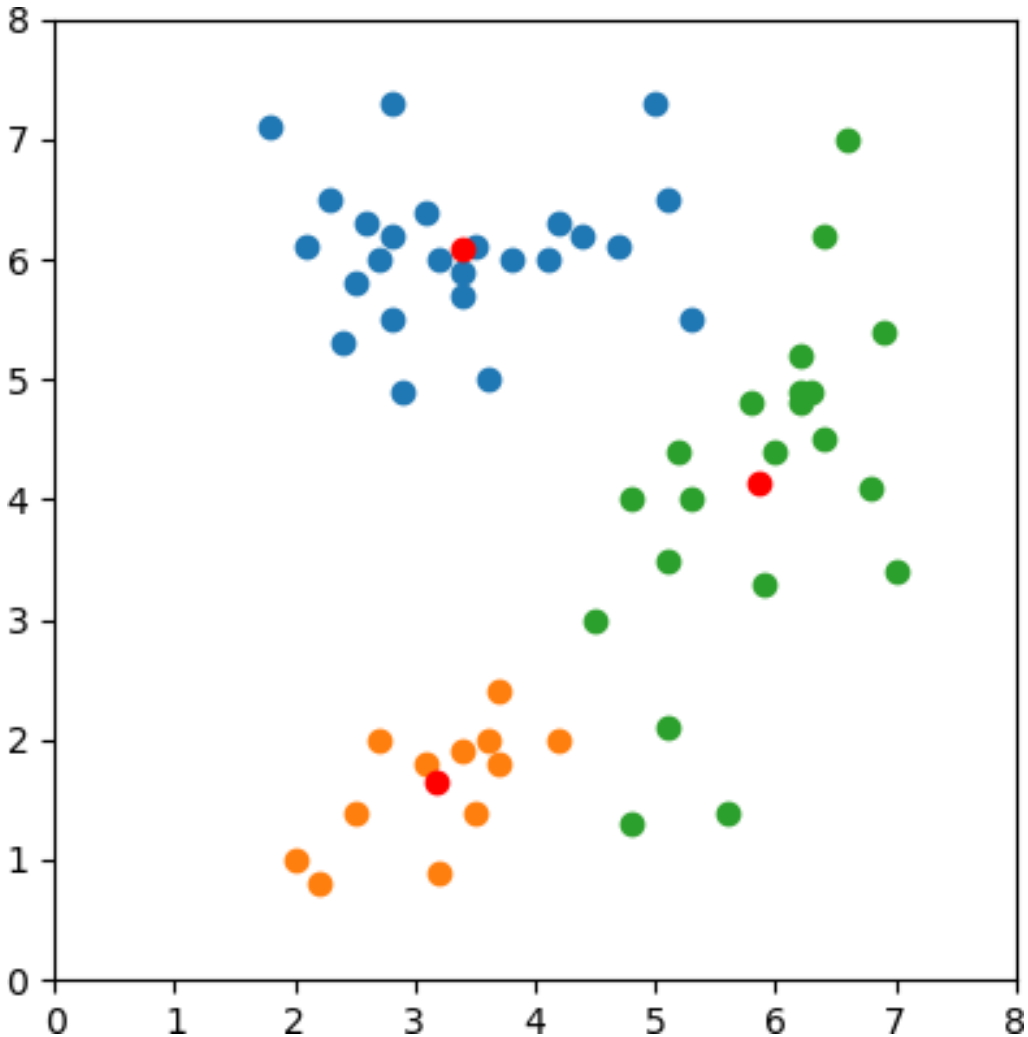
Example Run for $K = 3$, ML label assignment

- Identical run, but visualization shows component color of maximum responsibility
- Safety notice: only take the maximum on the output. The iteration itself has to use continuous responsibilities



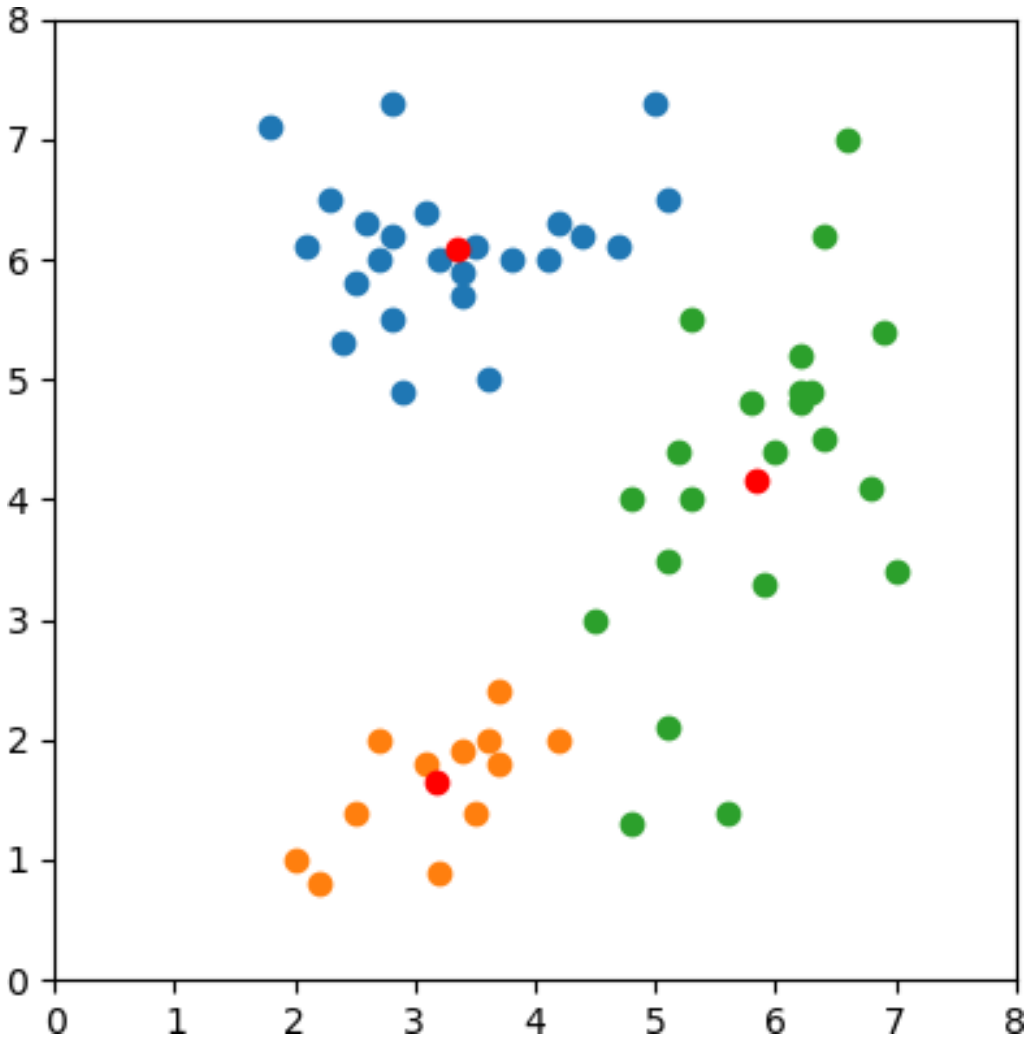
Example Run for $K = 3$, ML label assignment

- Identical run, but visualization shows component color of maximum responsibility
- Safety notice: only take the maximum on the output. The iteration itself has to use continuous responsibilities



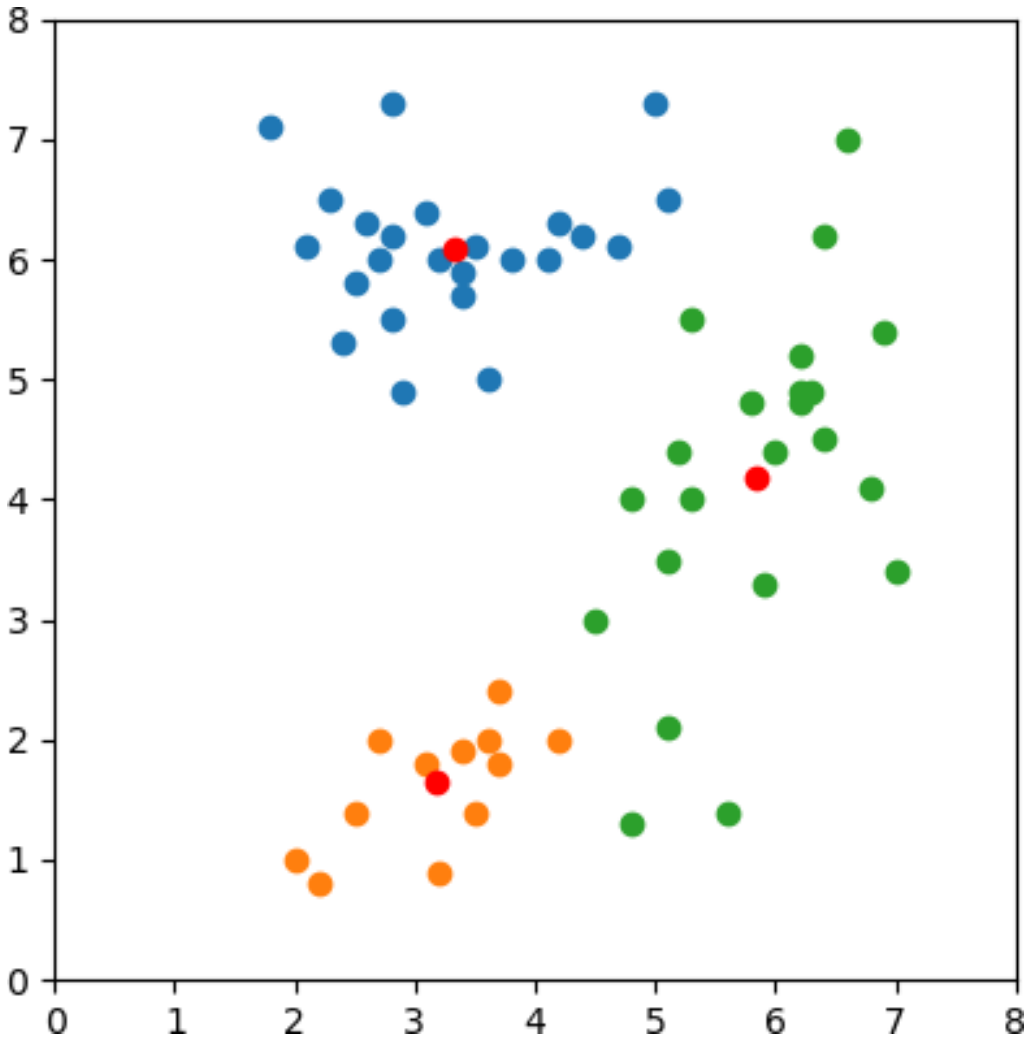
Example Run for $K = 3$, ML label assignment

- Identical run, but visualization shows component color of maximum responsibility
- Safety notice: only take the maximum on the output. The iteration itself has to use continuous responsibilities



Example Run for $K = 3$, ML label assignment

- Identical run, but visualization shows component color of maximum responsibility
- Safety notice: only take the maximum on the output. The iteration itself has to use continuous responsibilities



FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG

SCHOOL OF ENGINEERING

Lecture Pattern Analysis

Part 07: K-Means

Christian Riess
IT Security Infrastructures Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg
May 13, 2022

K-Means at a Glance

- K-means is arguably the most well-known clustering algorithm[1]
- **Hard-clustering** method, i.e., each sample gets a discrete cluster label assigned
- Idea: minimize Euclidean **Within-Cluster Distance** $W(C)$:

$$W(C) = \frac{1}{2} \cdot \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(j)=k} \|\mathbf{x}_i - \mathbf{x}_j\|^2 \tag{1}$$

$$= \sum_{k=1}^{K} N_k \cdot \sum_{C(i)=k} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|^2 \ , \tag{2}$$

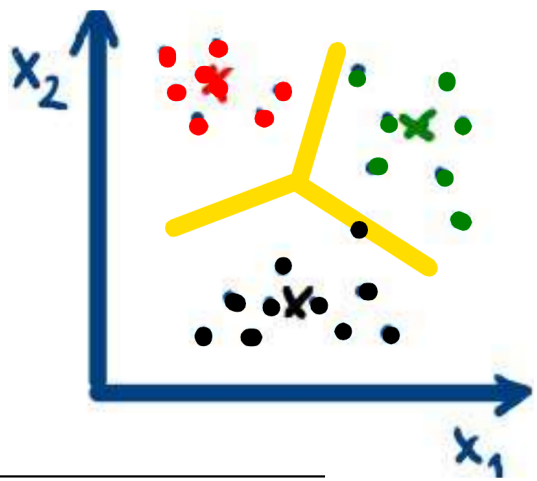
where K is the total number of clusters, $C(i)$ the cluster ID for sample $\mathbf{x}_i$, N_k the number of points in cluster k, and $\boldsymbol{\mu}_k$ the mean of all points in cluster k.

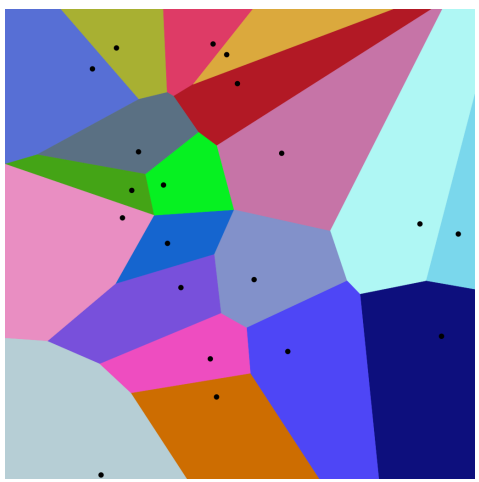
[1] Literature references are, e.g., Hastie/Tibshirani/Friedman Sec. 14.3.6 or Bishop Sec. 9.1

K-Means Algorithm

1. Initialization: set K cluster centers in sample space (e.g., randomly selected)
2. Assign each sample to the nearest cluster center w.r.t. Euclidean distance
3. Calculate the mean of each cluster from its assigned samples
4. goto 2) until convergence

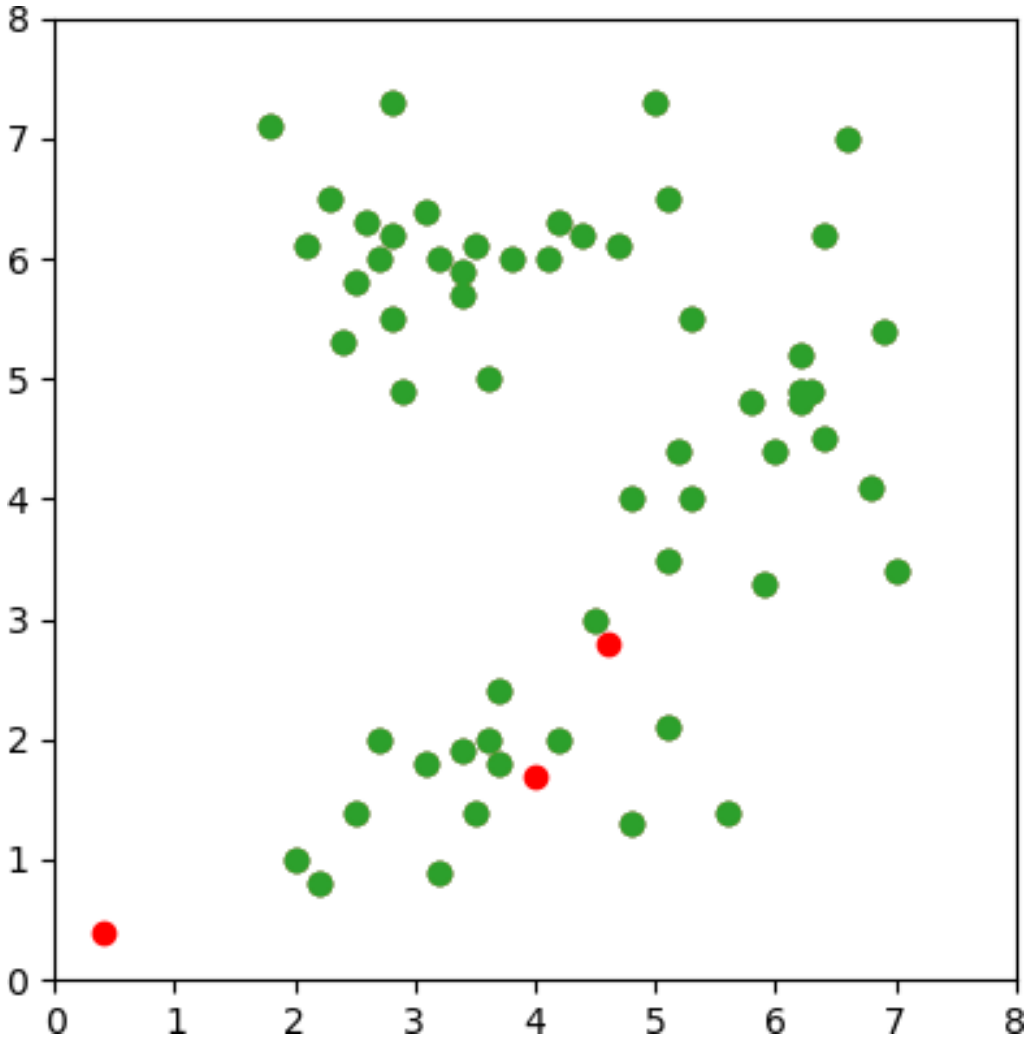
- Remarks:
  - K-means is locally optimal $\rightarrow$ different initializations $\overset{?}{=}$ different results
  - The clusters partition the space, the partitioning is called Voronoi tesselation[2]



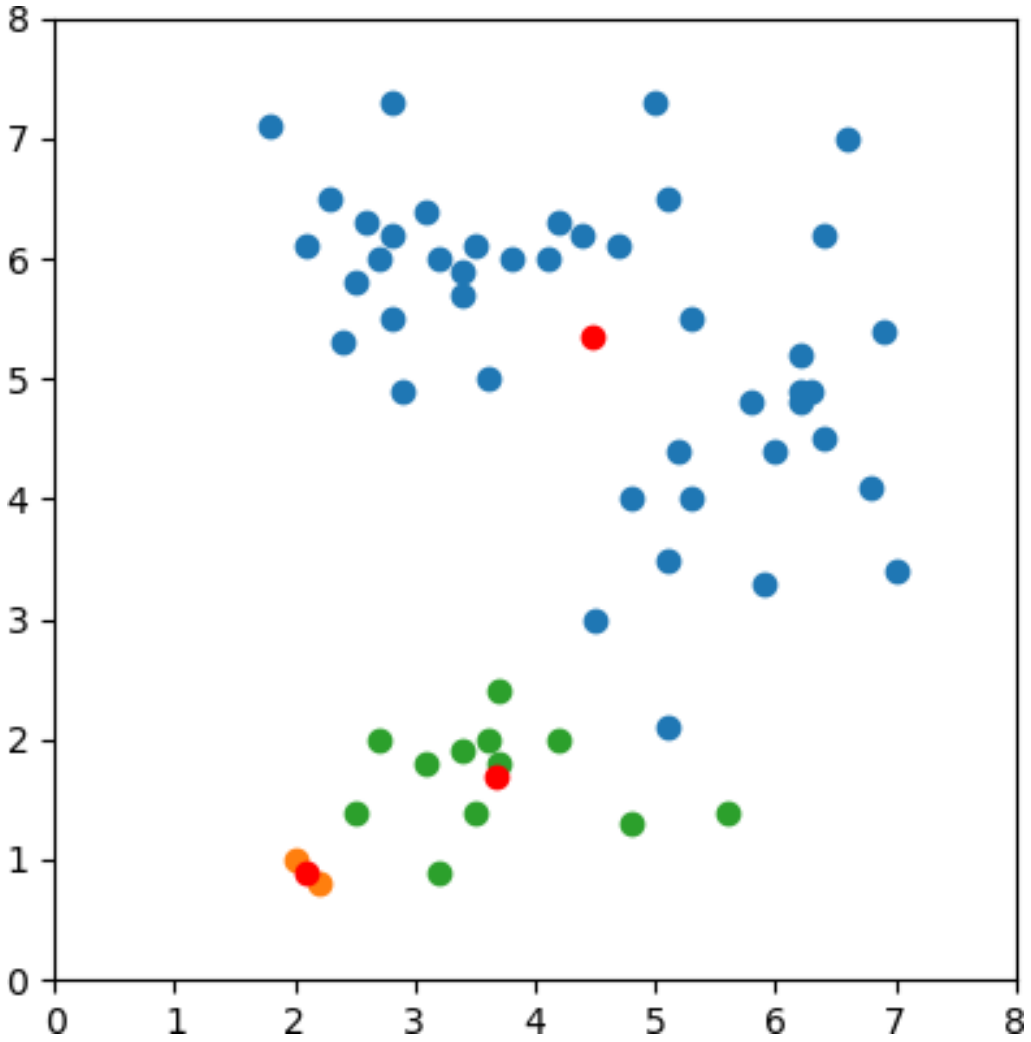


[2] Picture on the right is from wikipedia (CC BY-SA 4.0): https://upload.wikimedia.org/wikipedia/commons/5/54/Euclidean_Voronoi_diagram.svg

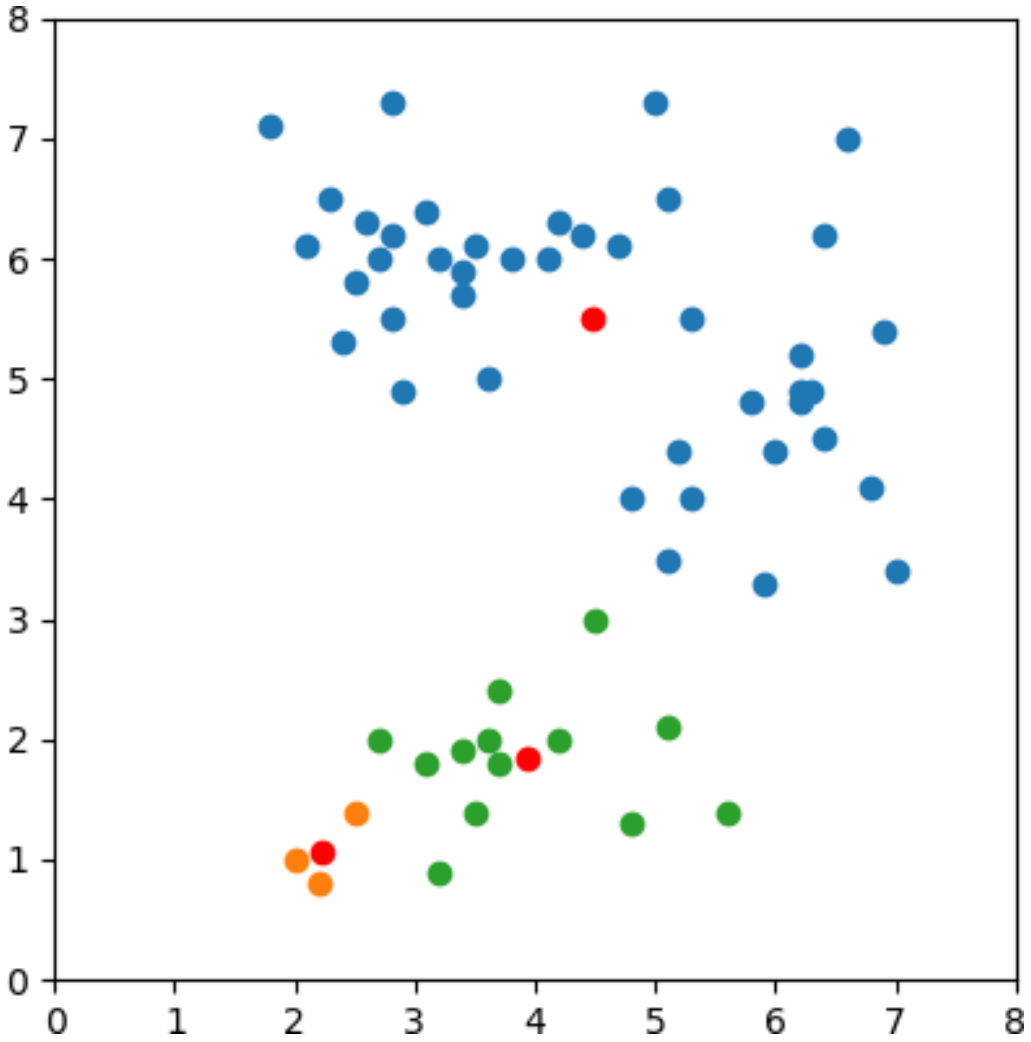
Example Run for $k = 3$, Random Starting Positions



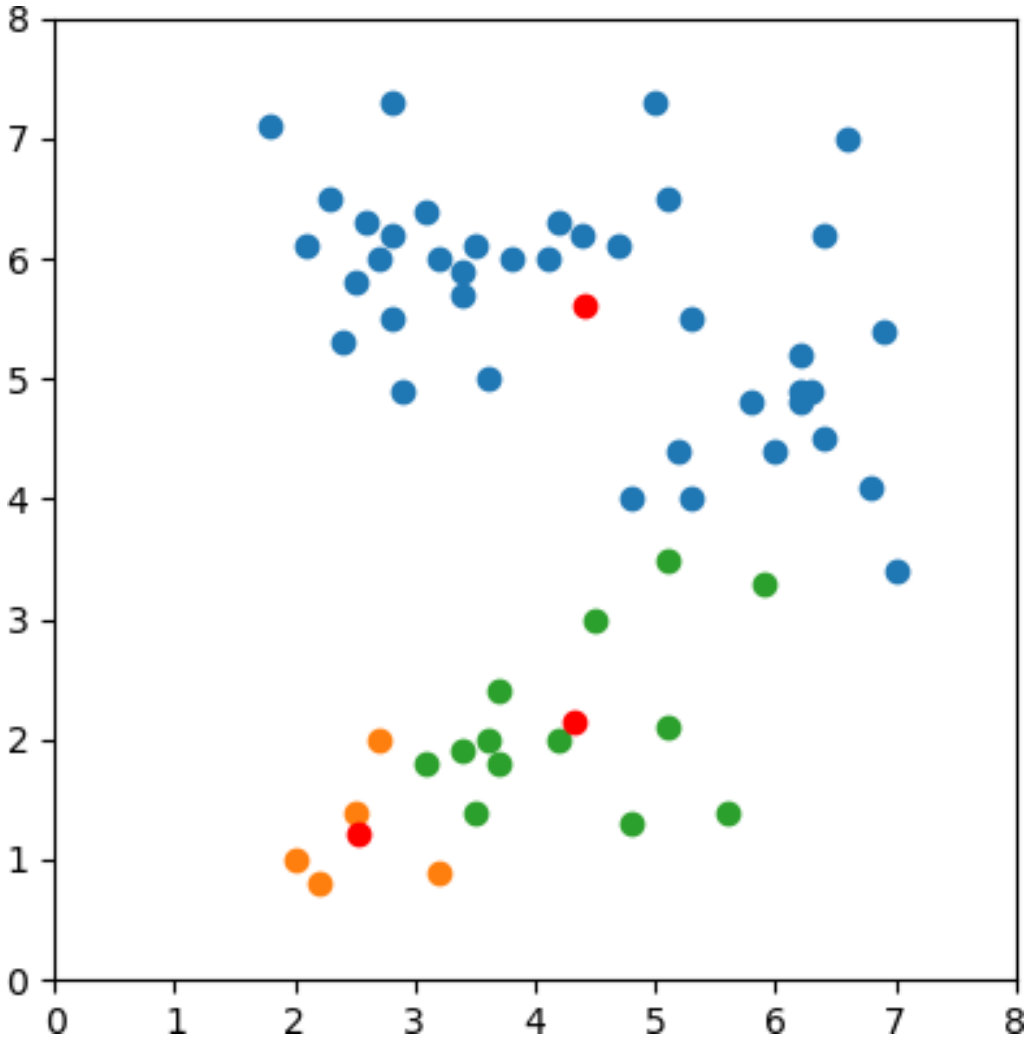
Example Run for $k = 3$, Random Starting Positions



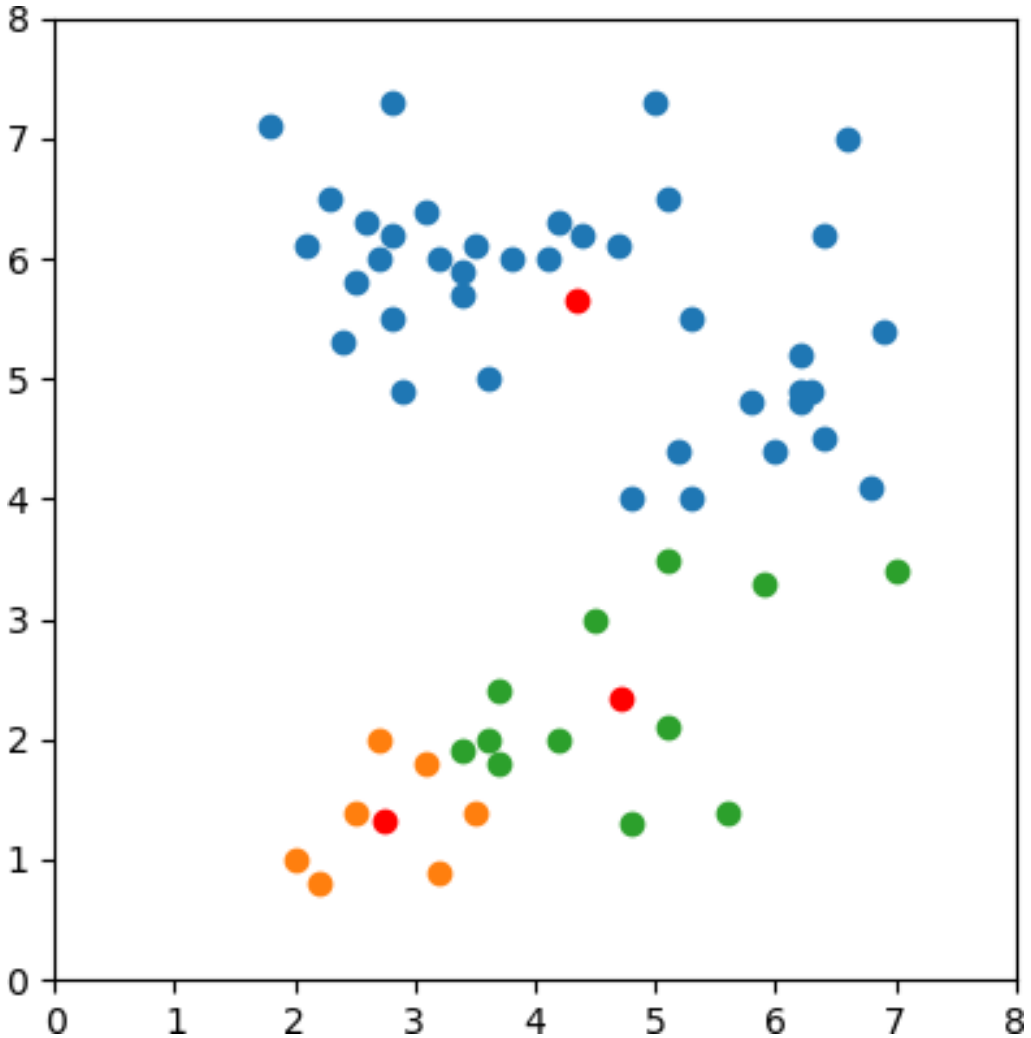
Example Run for $k = 3$, Random Starting Positions



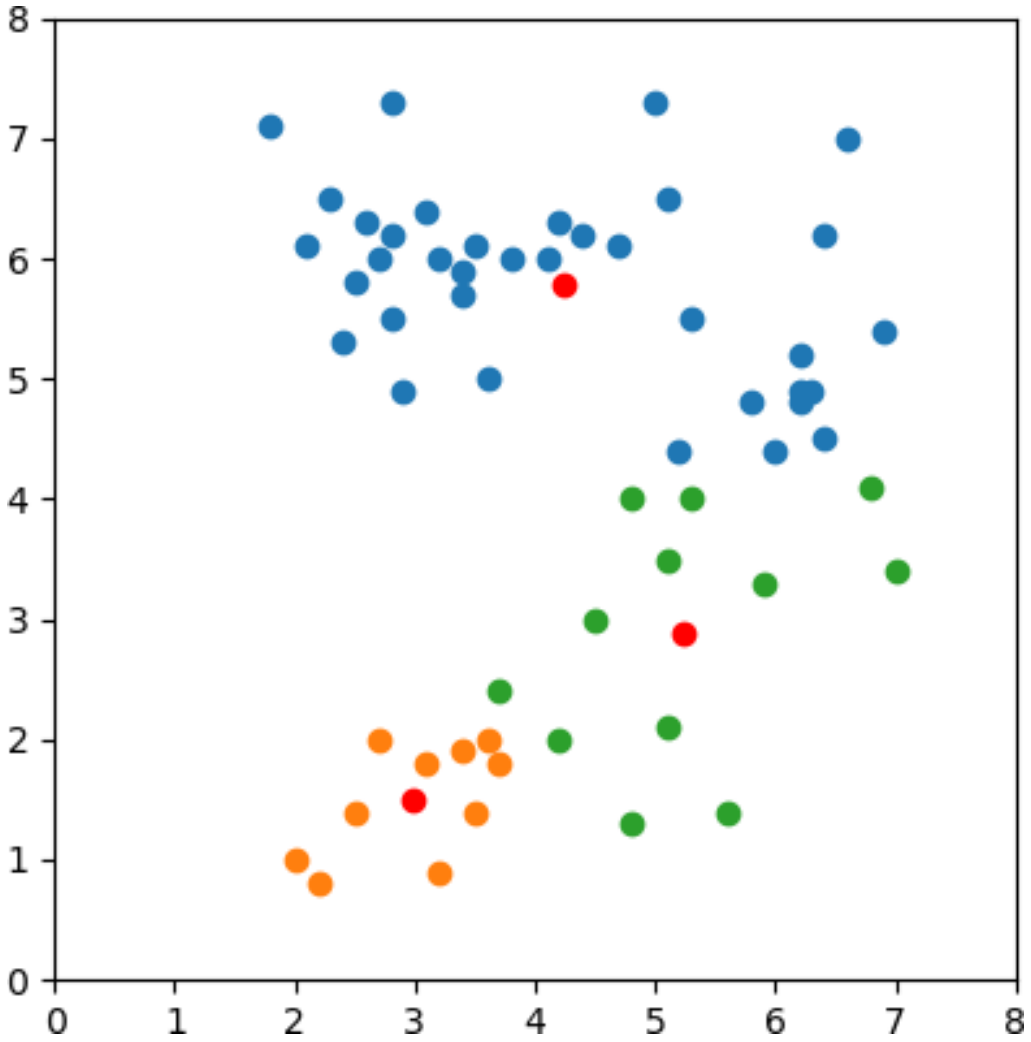
Example Run for $k = 3$, Random Starting Positions



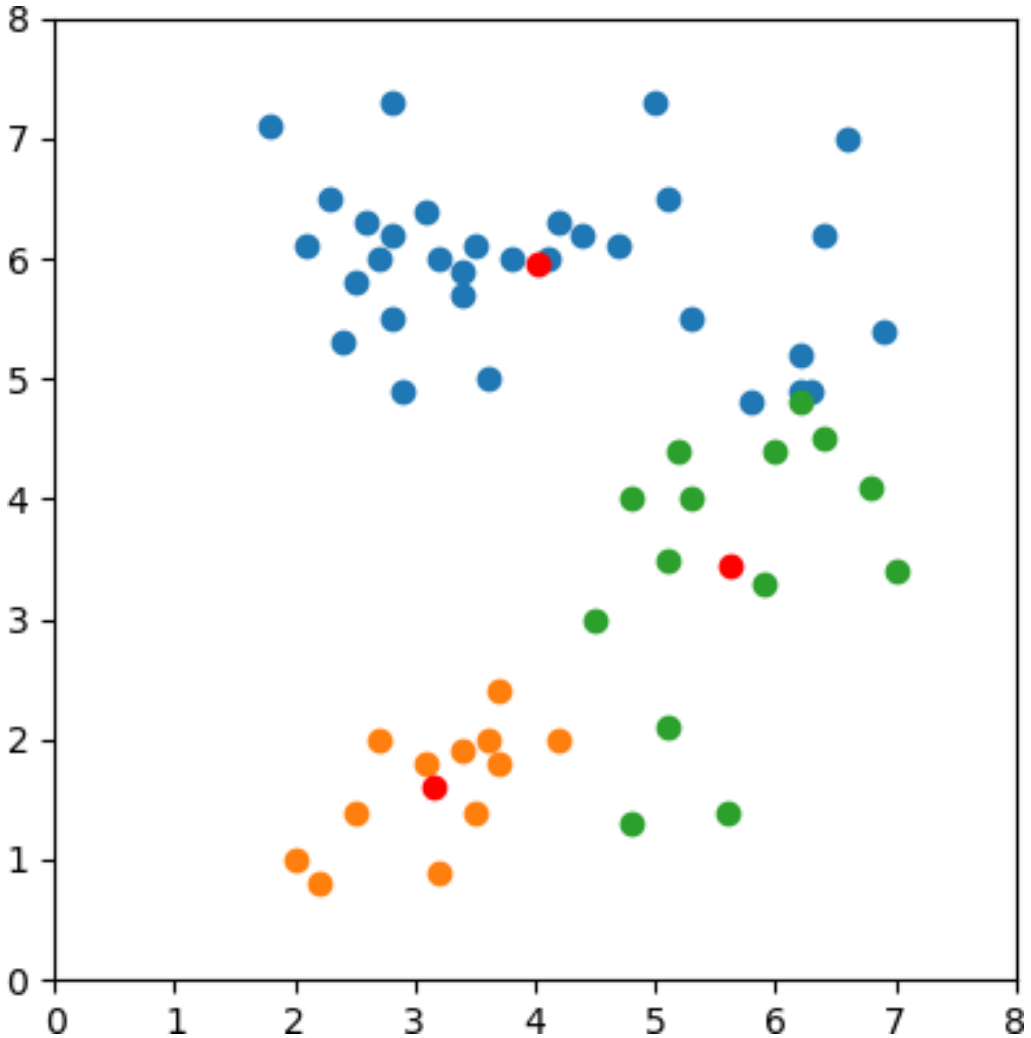
Example Run for $k = 3$, Random Starting Positions



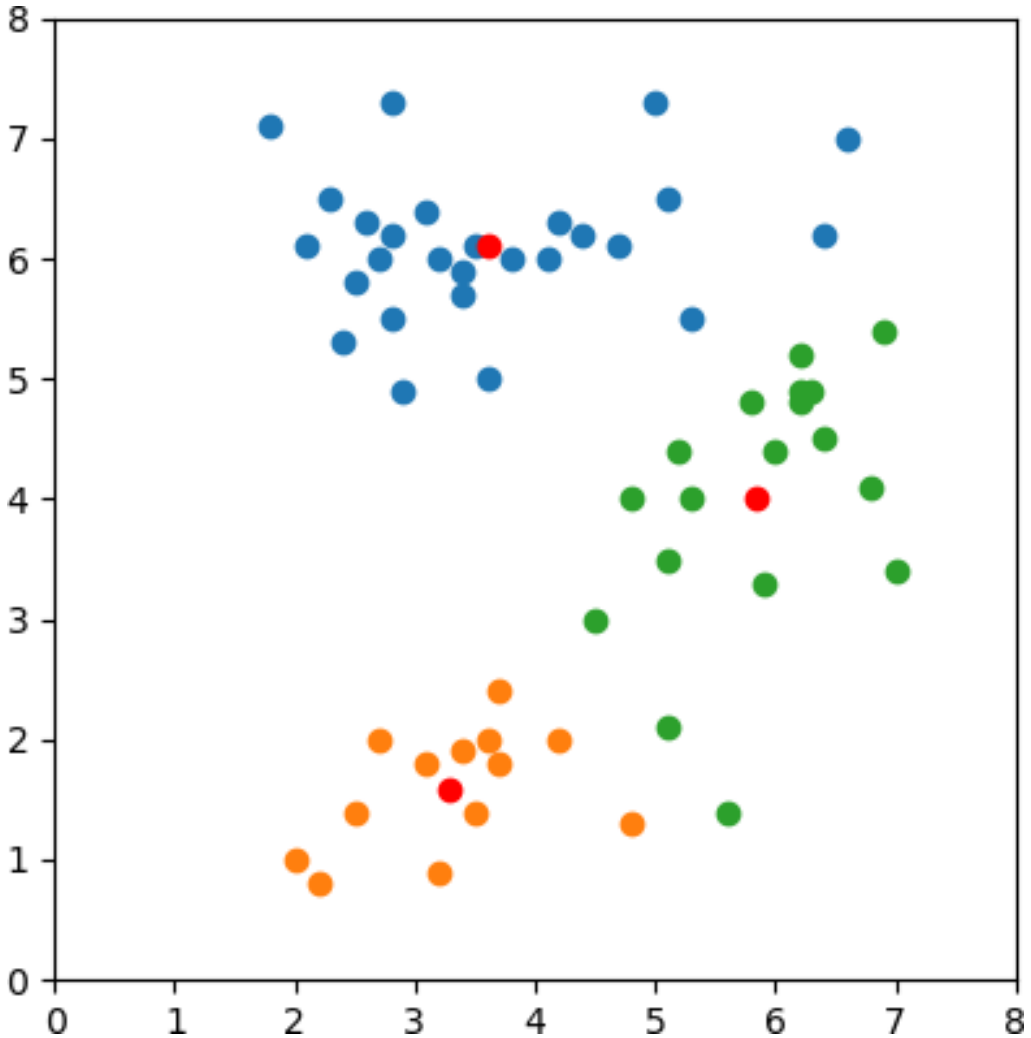
Example Run for $k = 3$, Random Starting Positions



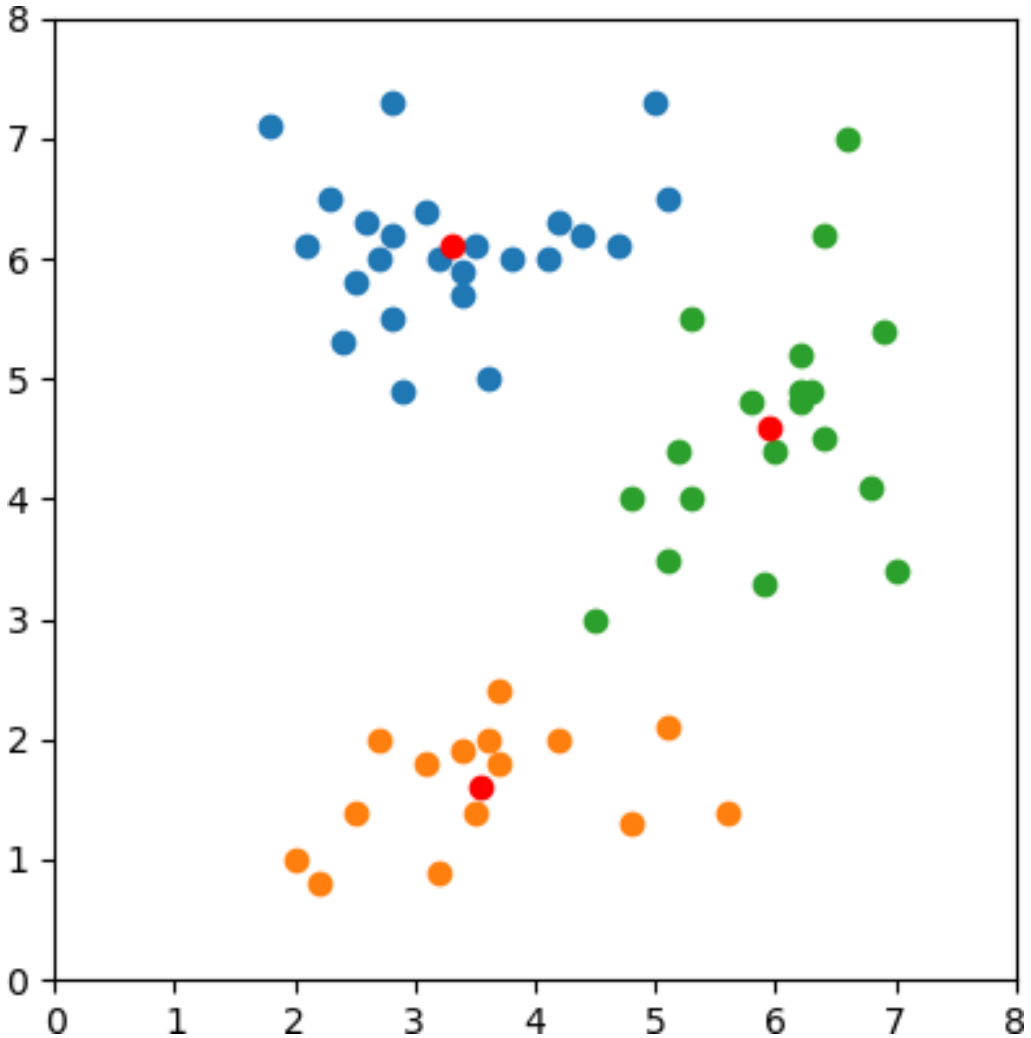
Example Run for $k = 3$, Random Starting Positions



Example Run for $k = 3$, Random Starting Positions



Example Run for $k = 3$, Random Starting Positions



Example Run for $k = 3$, Random Starting Positions