https://github.com/tedinburgh/ads2023

# Clustering: k-means

**Tom Edinburgh**
**te269**

# Example classes

- There are 2 more small group classes, next week and the following week
- Groups/times will be confirmed very soon
- Next week's class: decision trees
  - Chapter 8 of Introduction to Statistical Learning with Python
  - Questions 4, 5, 9, 10, 12 (ignore BART in Q12)
  - Pdf of the book available at https://www.statlearning.com/
  - .csv files are at https://www.statlearning.com/resources-python

# Recap recap: do you need to mean-centre for PCA?

- It depends how you define things
- $Q = X^T X/(n-1)$ is the sample covariance, but only because we'd mean-centred first
- Initially, we defined $Q$ as the sample covariance for non-centred data
- Sometimes, implementations use $X^T X$ without subtracting the mean from $X$ first
- This will lead to the issue with the first component

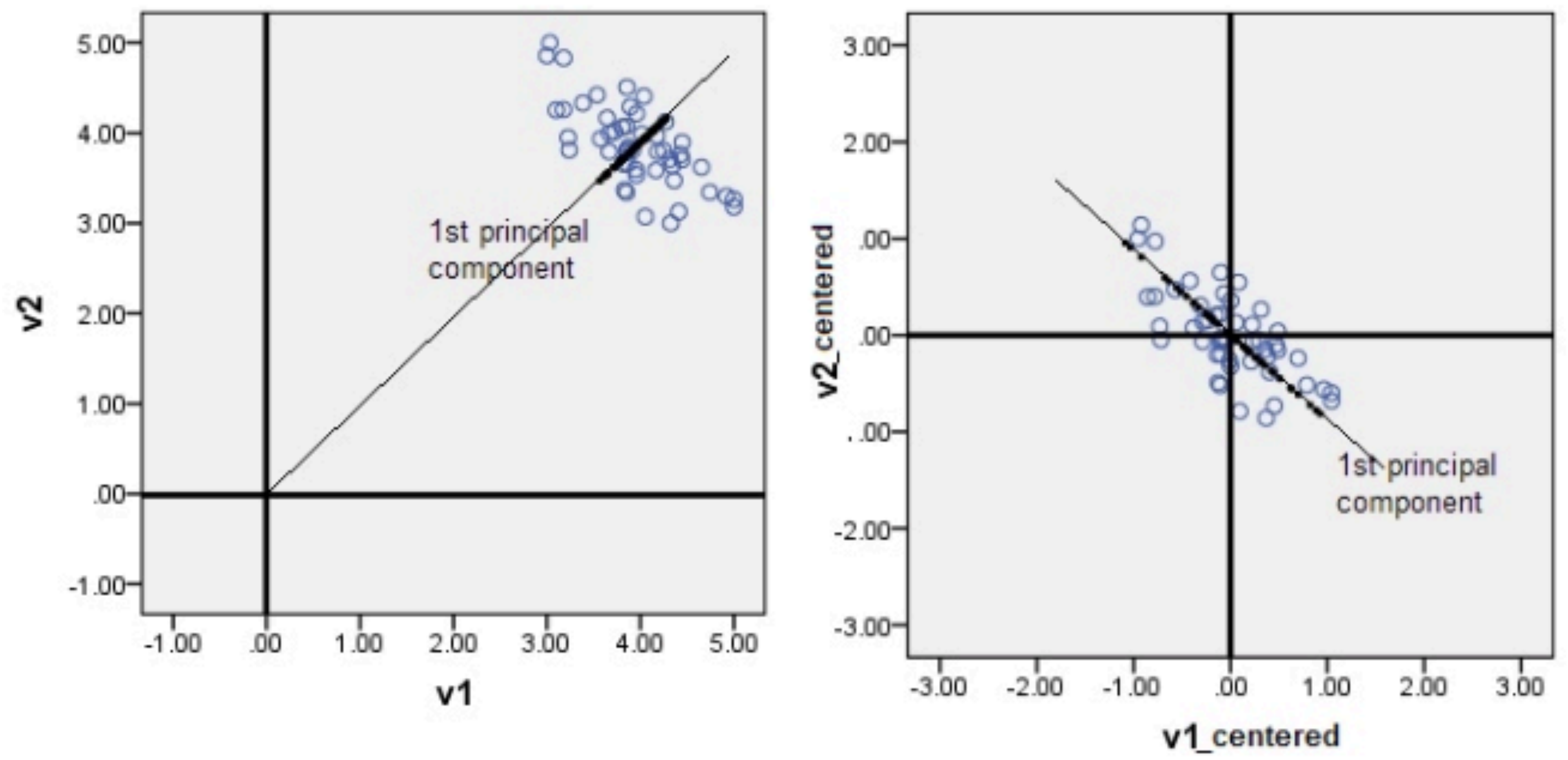
## Maximal variance

- We want to choose a vector $w$ so that it's as 'informative' as possible i.e. it maximises the variance of the projections of the data onto $w$.
- Suppose $\alpha_i = w \cdot x_i = w^T x_i$ is the projection for observation $x_i$, where $w$ has unit length. We want to maximise the variance of $\alpha = (\alpha_1, \ldots, \alpha_n)$.

$$\bar{\alpha} = \frac{1}{n}\sum_i \alpha_i = \frac{1}{n}\sum_i w^T x_i = w^T\left(\frac{1}{n}\sum_i x_i\right) = w^T \bar{x}$$

$$\mathrm{var}(\alpha) = \frac{1}{n-1}\sum_i (\alpha_i - \bar{\alpha})^2 = \frac{1}{n-1}\sum_i (w^T x_i - w^T \bar{x})^2 = \frac{1}{n-1}\sum_i w^T (x_i - \bar{x})(x_i - \bar{x})^T w = w^T Q w$$

$$\frac{1}{n-1}\sum_i w^T (x_i - \bar{x})(x_i - \bar{x})^T w = w^T Q w$$



# Recap: MDS vs Sammon mapping

- Metric MDS stress $S^2(y) = \dfrac{\sum_{i,j}(d_{ij} - o_{ij})^2}{\sum_{i,j} o_{ij}^2}$ with $d_{ij} = d(y_i, y_j),\ o_{ij} = d(x_i, x_j)$

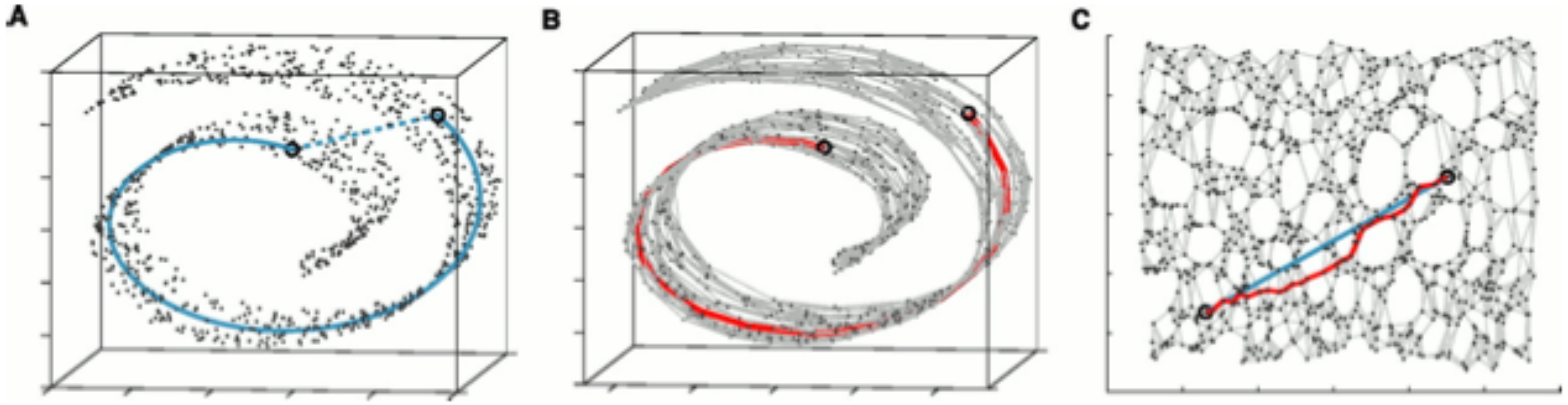
- Sammon's stress $S^2(y) = \dfrac{1}{\sum_{i=1}^{n}\sum_{j=i+1}^{n} o_{ij}} \sum_{i=1}^{n}\sum_{j=i+1}^{n} \dfrac{1}{o_{ij}}(d_{ij} - o_{ij})^2$

- Sammon mapping gives more weight to small distances, so preserves local structure more than other MDS methods

- I.e. if $o_{ij}$ is small, then it up-weights the contribution from $(d_{ij} - o_{ij})^2 / o_{ij}$

# Recap: Isomap

- We need a connected neighbourhood graph, so that it’s possible to go from any point to any other point along edges
- Each point connected to k nearest neighbours, k is a hyperparameter to specify
- If the neighbourhood graph is disconnected, increase k (scikit-learn default: 5)
- Approximate geodesic distance (blue) by sum of Euclidean distances between neighbours using e.g. Dijsktra’s algorithm (it may have to double back on itself!)



# Today: k-means clustering

- k-means
- Voronoi
- Initialisation
- Extensions (e.g. weighted kernel k-means)
- Fuzzy c-means

Questions: halfway through, at the end, or by email (te269)

# Resources

- Introduction to Statistical Learning with Python, Chapter 12

- Slides adapted from:
  - Prof Stephen Eglen, Cambridge
  - Ethan Fetaya/James Lucas/Emad Andrews, Toronto
  - Ronan Cummins, Cambridge

# Overview: clustering

- Grouping $n$ observations into $K$ clusters is one of the common/major problems in unsupervised learning
- We assume the data was generated from a number of different classes, we want to cluster observations (objects) from the same class together (without necessarily describing the classes)
- Objects that are close to each other in high-dimensional space should be in the same cluster (objects in the same cluster should be similar and objects in different clusters should be dissimilar), we want to find a 'natural' grouping
- How is this different from classification? What is $K$?

# Types of clustering

- Hard vs soft
  - Do objects belong to only one cluster or can they belong to more than one?
- Hierarchical vs non-hierarchical
  - How are clusters related to each other (i.e. are there parent clusters)?
- Agglomerative vs partitioning/divisive:
  - Do you lump together or split up?
- Centroid vs distribution-based vs density vs graph-based vs spectral

# Motivation and examples

- Pattern recognition
- Computer vision, e.g. detect moving objects in videos (self-driving cars)
- Personalised medicine/phenotypes (groups of patients who are similar)
- Bioinformatics, e.g. gene expression
- Cosmology

- **Warning**: clustering methods can find structure where there isn’t actually any. We should be wary of making strong conclusions about the output of clustering methods!

# Overview: k-means

- Partitioning method: assign each point to one of $K$ non-nested clusters
- k-means (unsupervised) is similar to k-nearest neighbour classifier (supervised)
- We represent each cluster by a single representative point (the cluster centroid)
- This is an NP-hard problem (finding a global minimum is computationally difficult)
- $K$ is the only parameter, how should we choose $K$?

# Overview: k-means

- Partitioning method: assign each point to one of $K$ non-nested clusters
- k-means (unsupervised) is similar to k-nearest neighbour classifier (supervised)
- We represent each cluster by a single representative point (the cluster centroid)
- This is an NP-hard problem (there are a lot of possible clusterings and finding a global minimum is computationally difficult)
- $K$ is the only parameter, how should we choose $K$?

# Lloyd's algorithm (naive k-means)

- Observations $x_i = (x_1, \ldots, x_p)$ for $i = 1,\ldots,n$
- $K$ clusters, with centroids $c_k$ for $k = 1,\ldots,K$
- Cluster membership matrix $M$ (size $n \times K$) or cluster sets $C_1, \ldots, C_k$, with $x_i \in C_k$ and $m_{ik} = 1$ if $x_i$ belongs to cluster $k$ and $m_{ik} = 0$ otherwise
- Two steps:
  1. Assignment
  2. Centroid update

# Lloyd’s algorithm (naïve k-means)

- Initialise clusters or centroids (various approaches, e.g. uniform random)
- Two steps:
  1. Assignment:
     - Assign each observation to the cluster with the nearest* centroid
  2. Centroid update
     - Recalculate the centroids as the mean of all observations in that cluster

- *we’re using the squared Euclidean distance $d(x_i, x_j) = ||x_i - x_j||_2^2$

# Lloyd’s algorithm (naïve k-means)

- Initialise clusters or centroids (various approaches, e.g. uniform random)
- Two steps (repeat until $M$ stops changing):
  1. Assignment:
     - Assign $x_i$ to cluster $j$ if $j = \arg\min_k d(x_i, c_k)$, i.e. $m_{ij} = 1, m_{il} = 0\ \forall l \neq k$
  2. Centroid update
     - New cluster centroids are $c_k = \dfrac{\sum_{i=1}^{n} m_{ik} x_i}{\sum_{i=1}^{n} m_{ik}}$

# Lloyd's algorithm (naïve k-means)

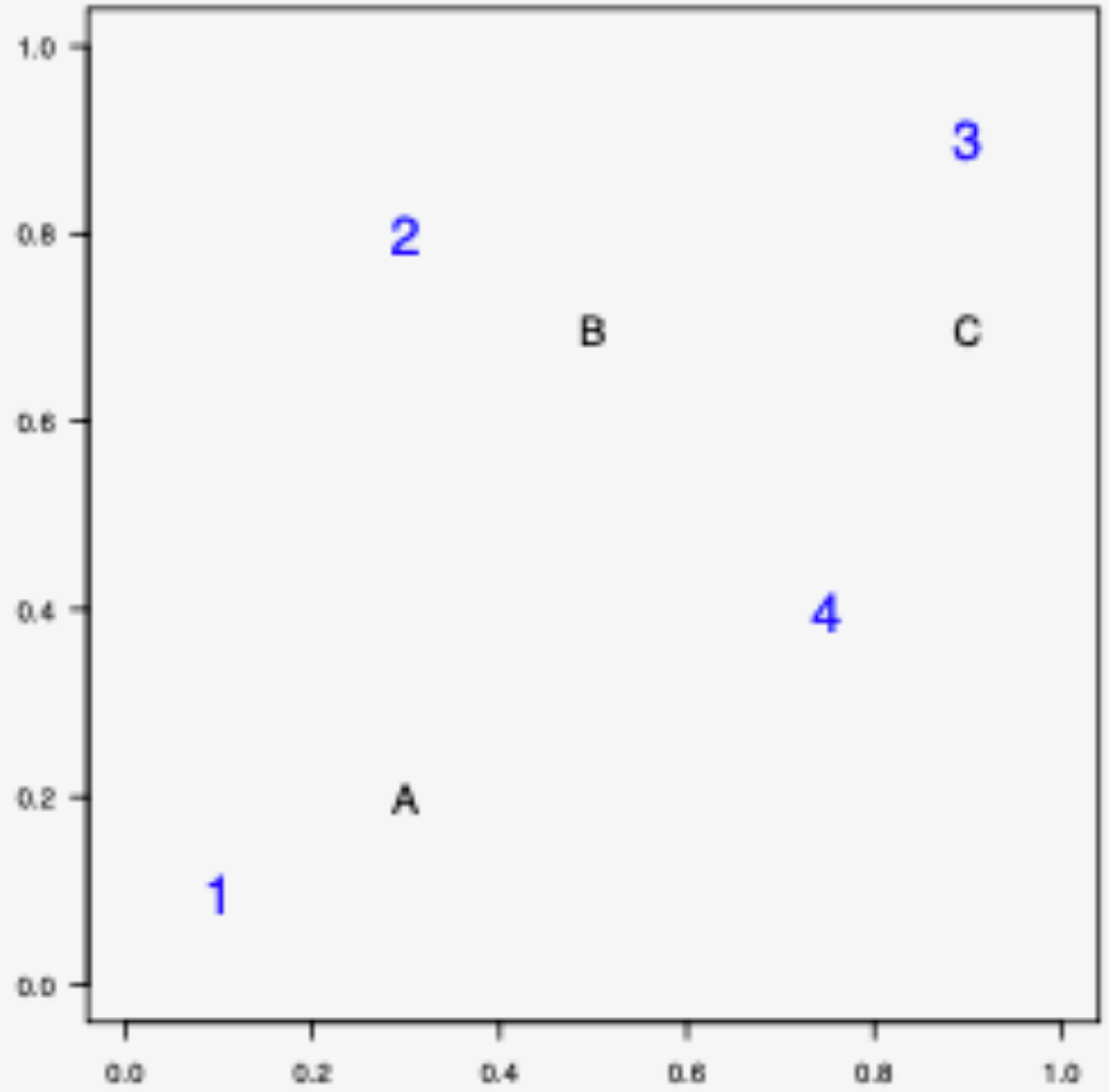
- Initialise clusters or centroids (various approaches, e.g. uniform random)
- Two steps (repeat until $C_k$'s stop changing):
  1. Assignment:
     - Assign points to cluster set $C_k = \{x_i : d(x_i, c_k) \leq d(x_i, c_j)\ \forall j = 1,\ldots,K\}$
  2. Centroid update
     - New cluster centroids are $c_k = 1/n_k \sum_{x_i \in C_k} x_i$, where $n_k$ is the size of set $C_k$

# Aside: distance metrics

- In principle, we can use any measure for the distance between $x_i$ and $x_j$, as long as it's a **metric**
- Metrics have a few properties
  - $d(x_i, x_j) = 0 \iff x_i = x_j$ (identity)
  - $d(x_i, x_j) = d(x_j, x_i)$ (symmetry)
  - $d(x_i, x_j) \leq d(x_i, x_k) + d(x_k, x_j)$ (triangle inequality)

# Membership matrix

Example with 4 inputs (1:4 in blue) and 3 centroids (A, B, C):




Distance matrix

| **D** input i | centroid k A | B | C |
|---|---|---|---|
| 1 | 0.2 | 0.7 | 1.0 |
| 2 | 0.6 | 0.2 | 0.6 |
| 3 | 0.9 | 0.5 | 0.2 |
| 4 | 0.5 | 0.4 | 0.3 |

Membership matrix

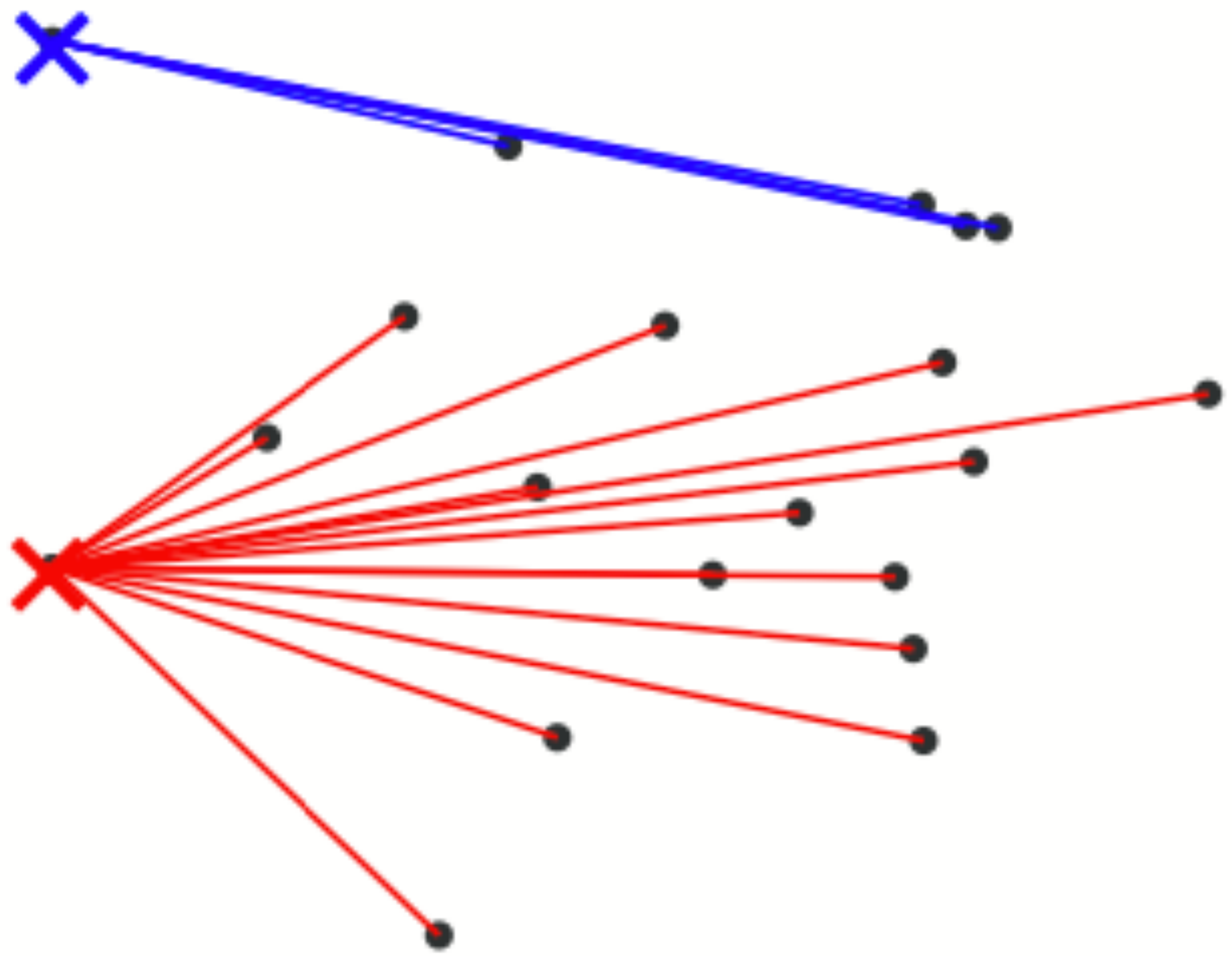
| **M** input i | centroid k A | B | C |
|---|---|---|---|
| 1 | 1 | 0 | 0 |
| 2 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 |
| 4 | 0 | 0 | 1 |

# Within-cluster sum of squares

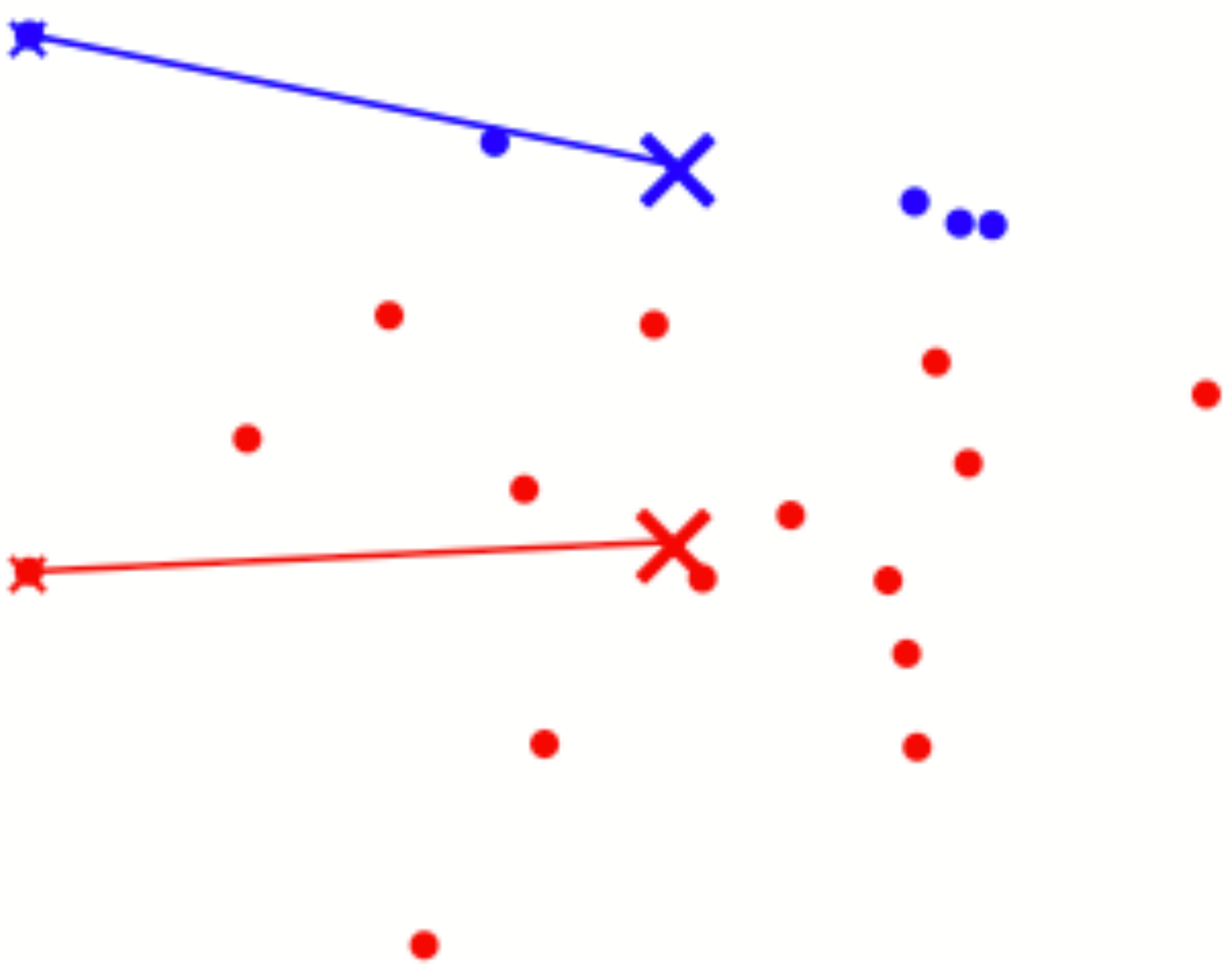
- k-means is really an optimisation problem
- We're rearranging cluster assignments to minimise an error function
- e.g. the within-cluster sum of squares: $E = \sum_{k=1}^{K} \sum_{x_i \in C_k} d(x_i, c_k)^2$
- This should decrease every step of Lloyd's algorithm
  - During assignment, each observation $x_i$ moves to a closer centroid
  - During centroid update, the centroid moves to minimise the average error

# k-means in action (data)

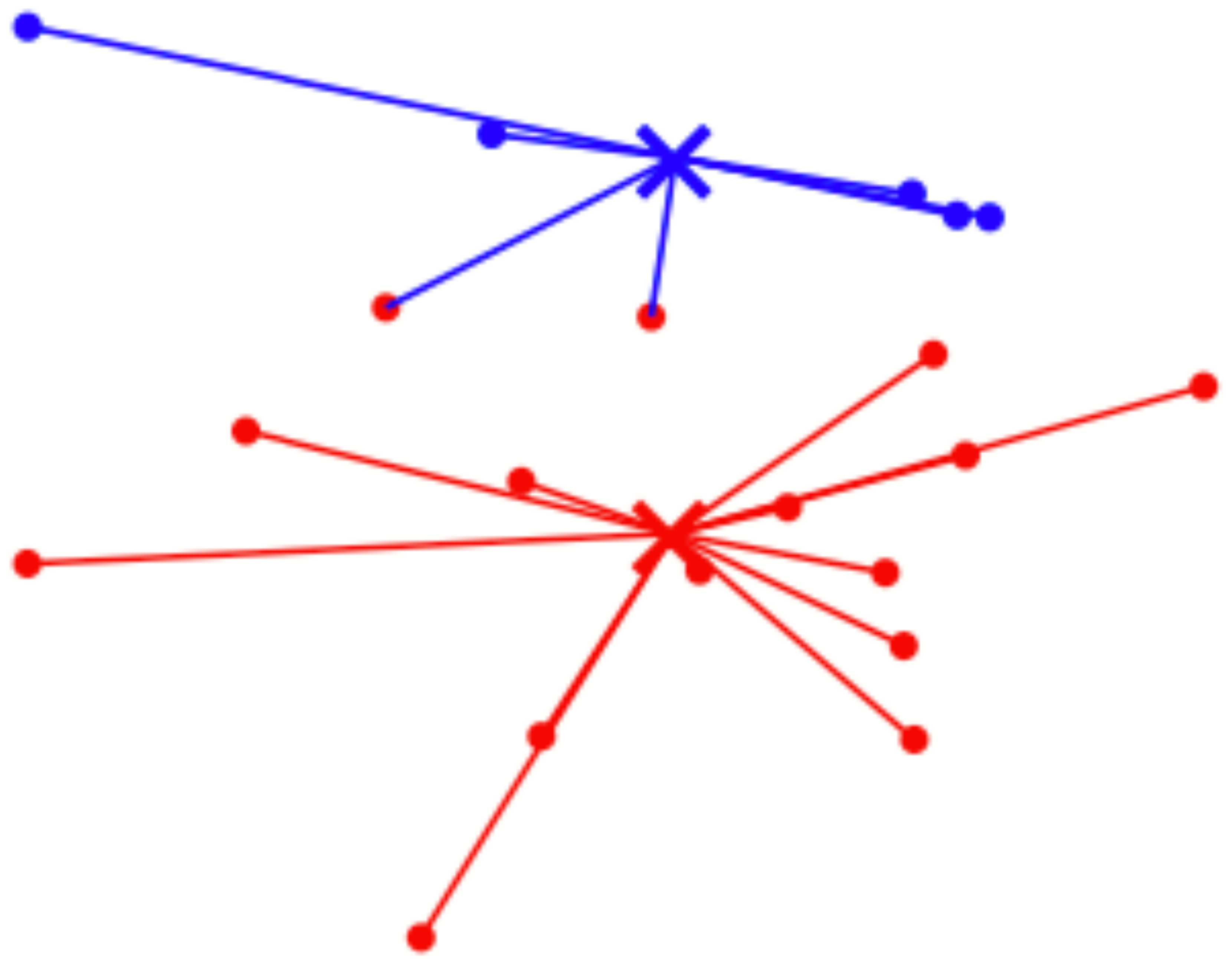
# k-means in action (initialisation + iteration 1 assign)



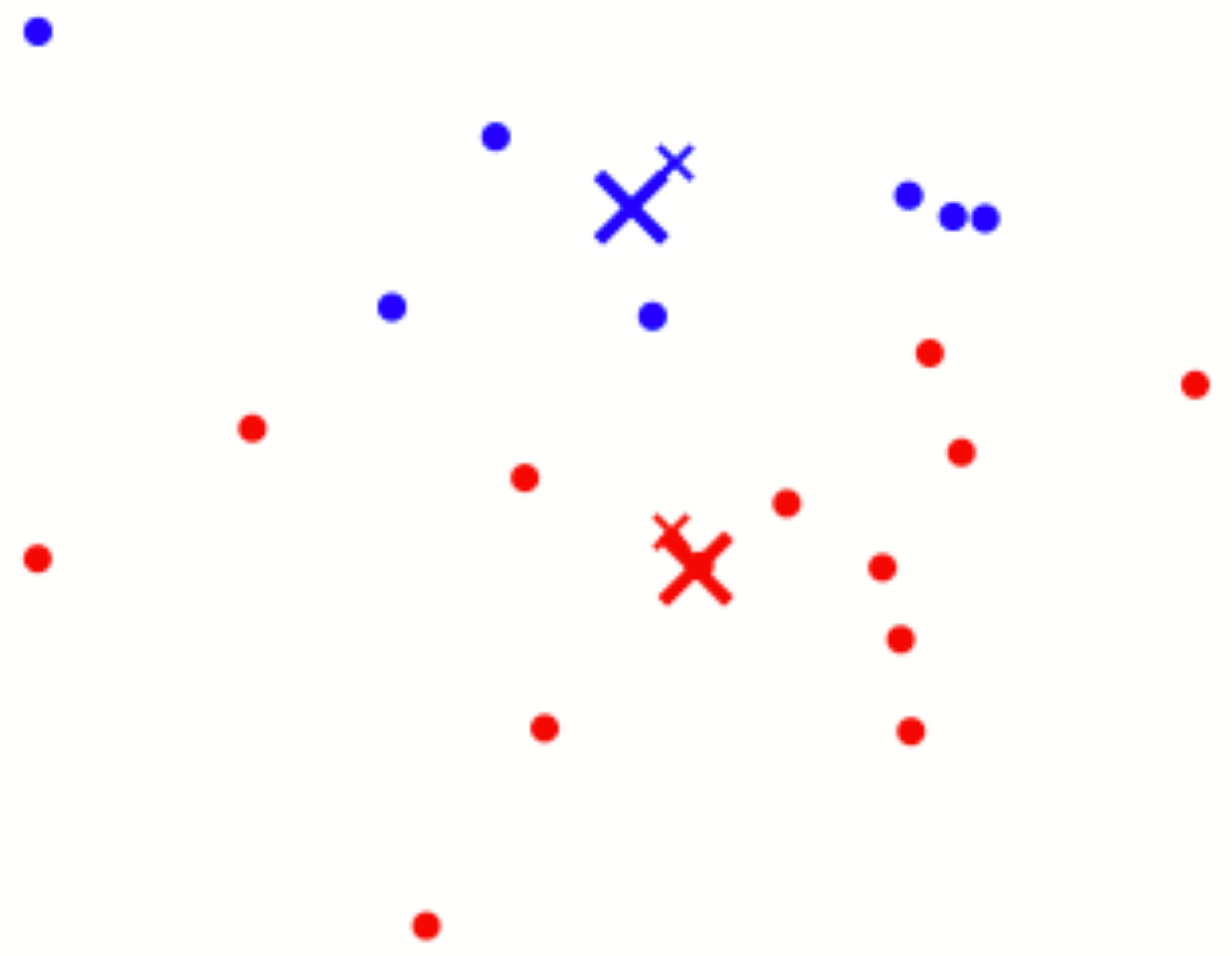
# k-means in action (iteration 1 update centroids)



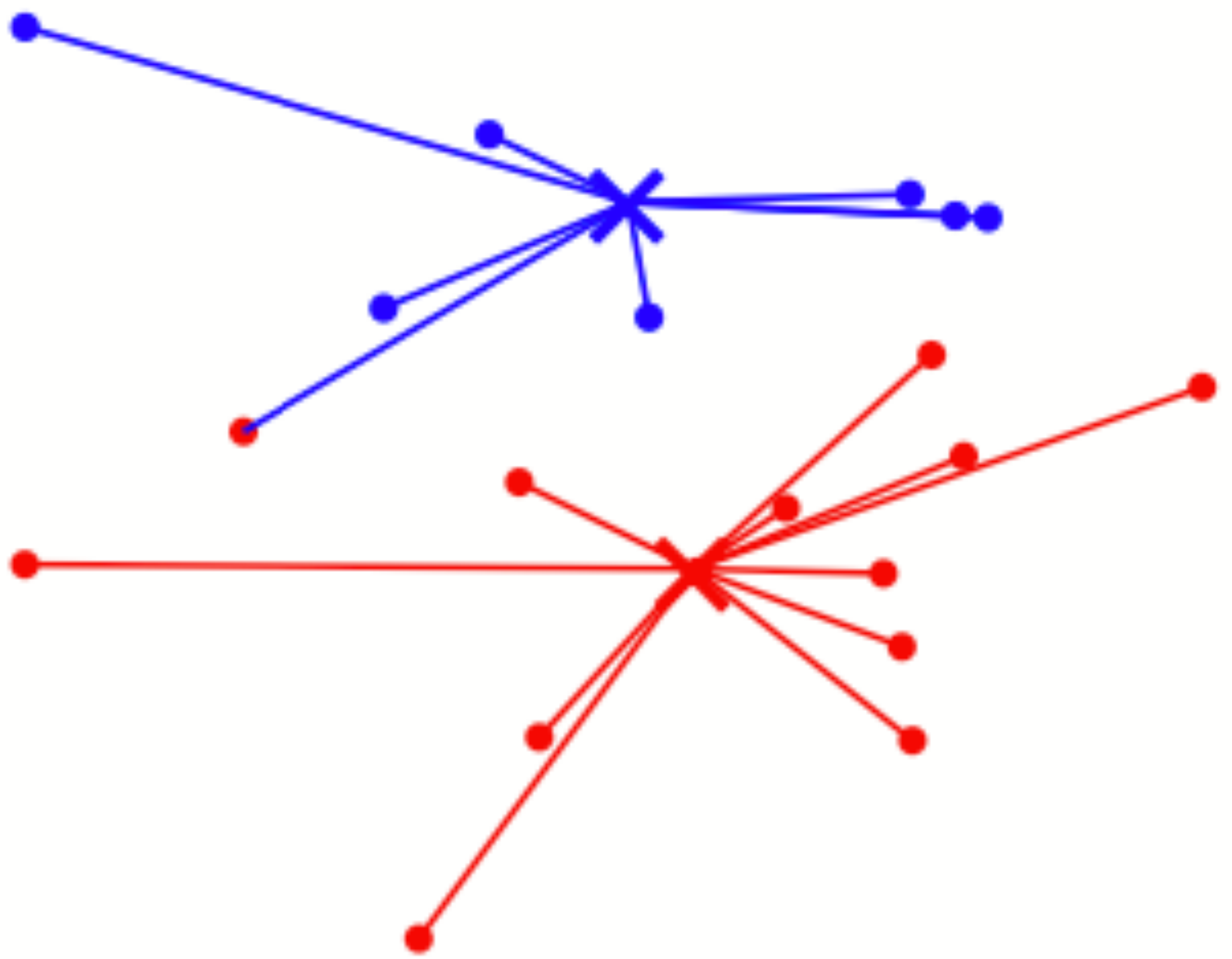
# k-means in action (iteration 2 assignment)



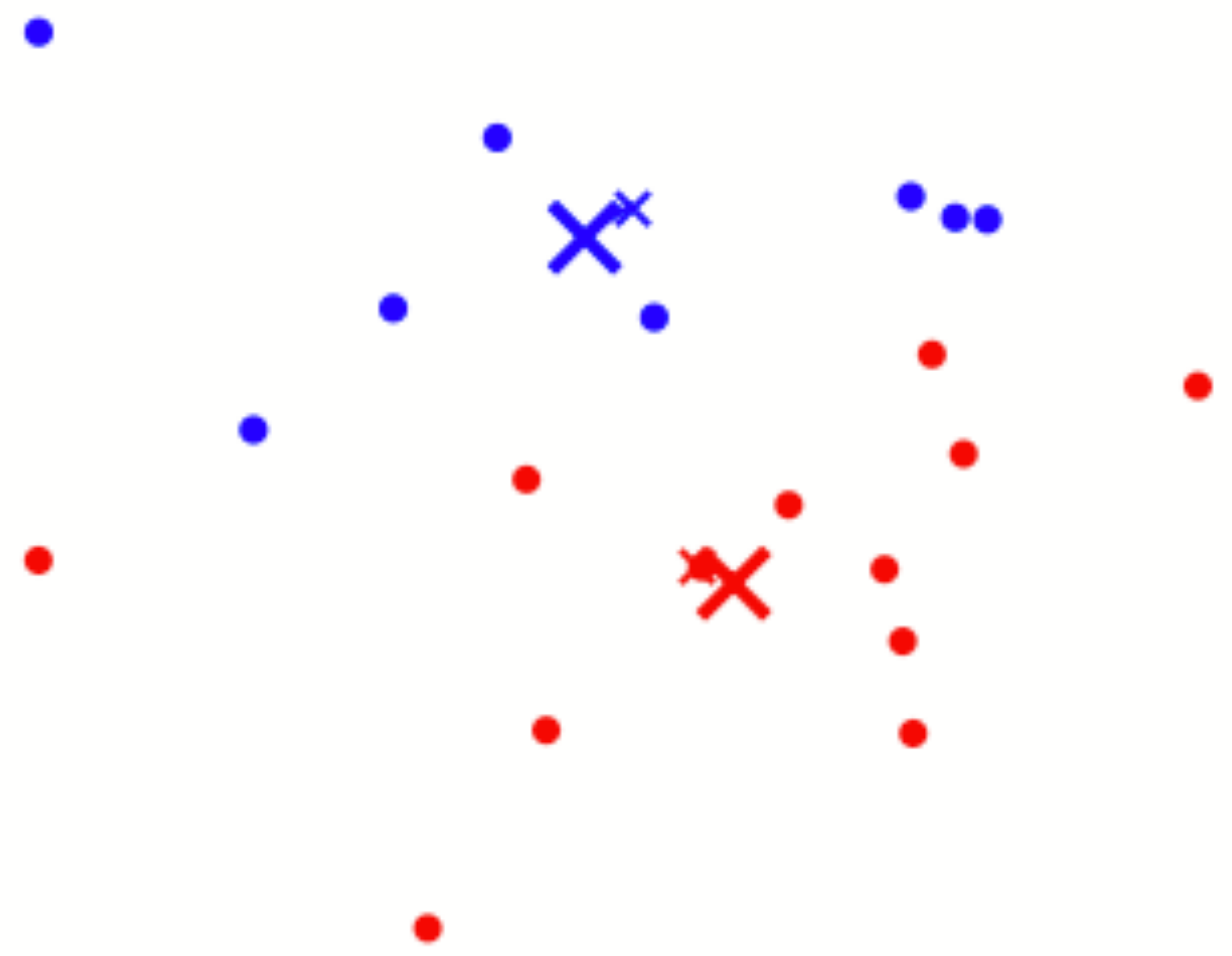
# k-means in action (iteration 2 update centroids)



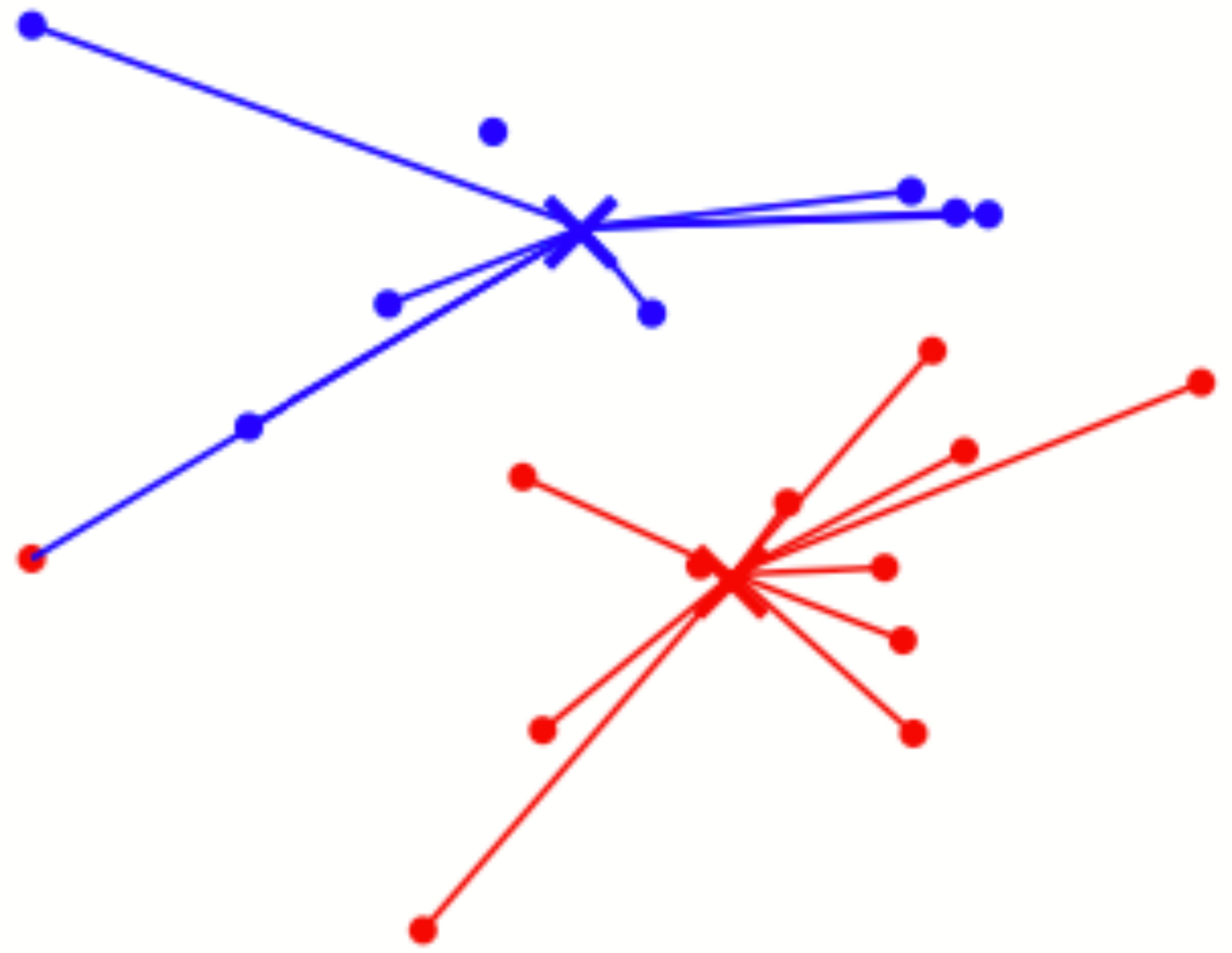
# k-means in action (iteration 3 assignment)



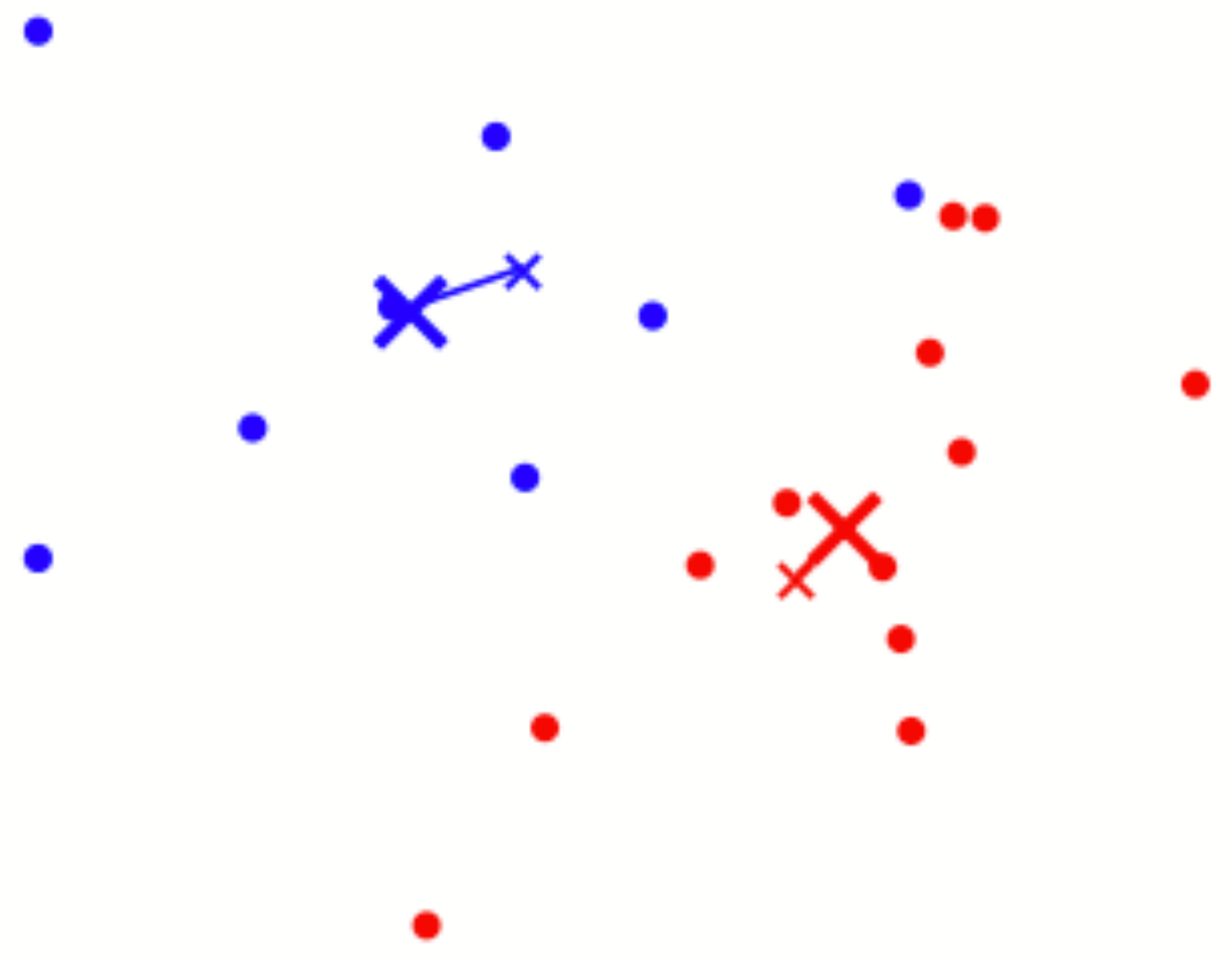
# k-means in action (iteration 3 update centroids)



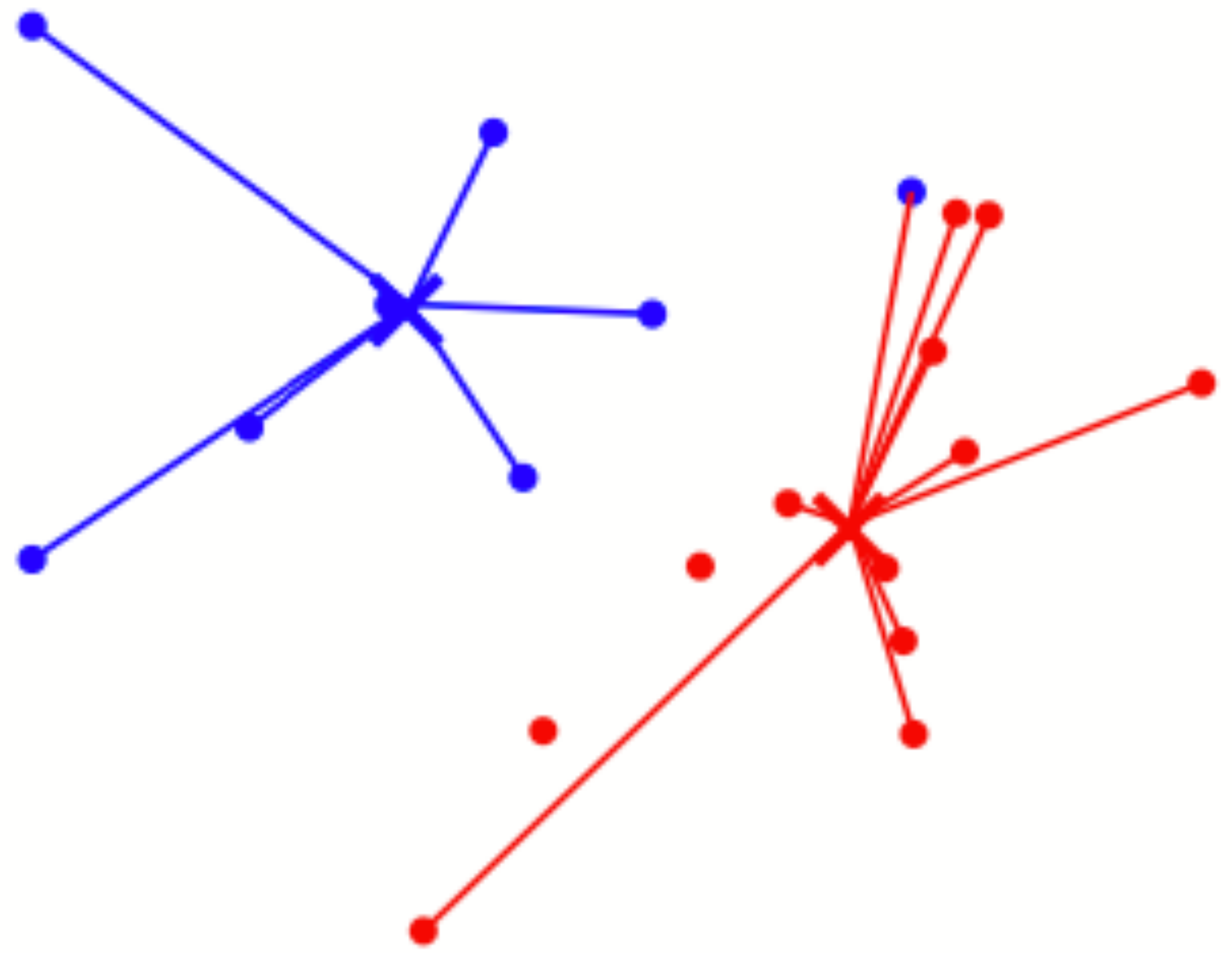
# k-means in action (iteration 4 assignment)



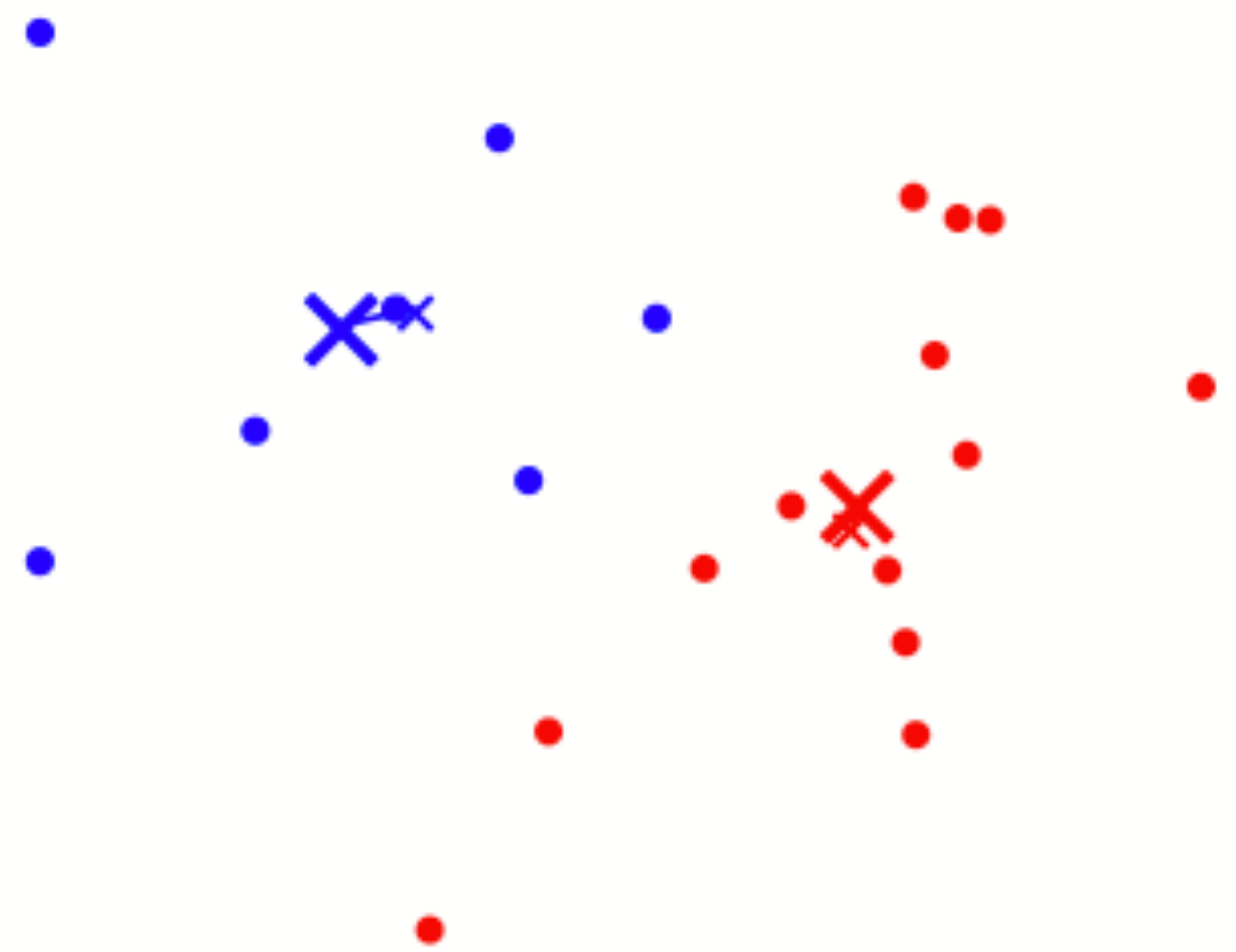
# k-means in action (iteration 4 update centroids)



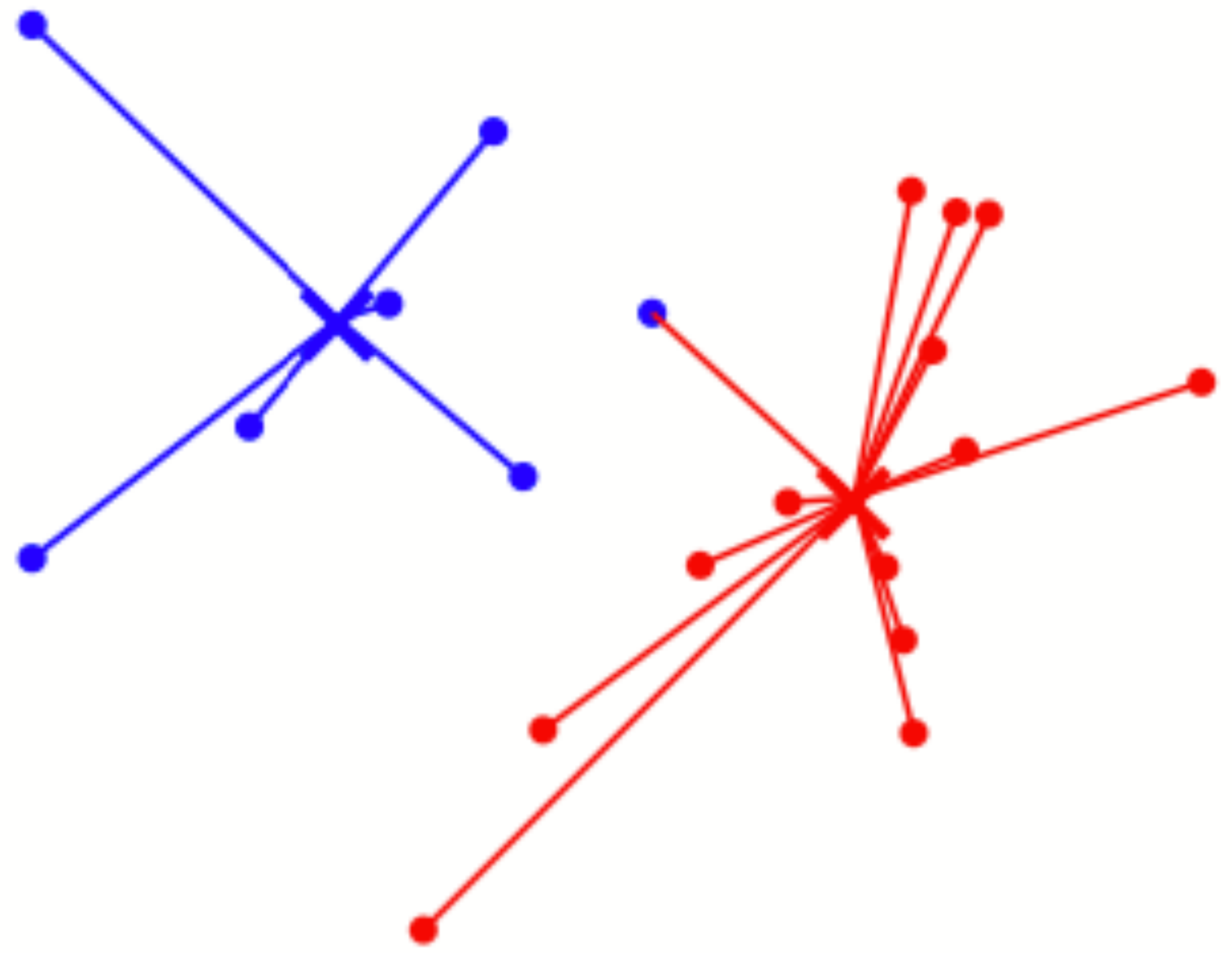
# k-means in action (iteration 5 assignment)



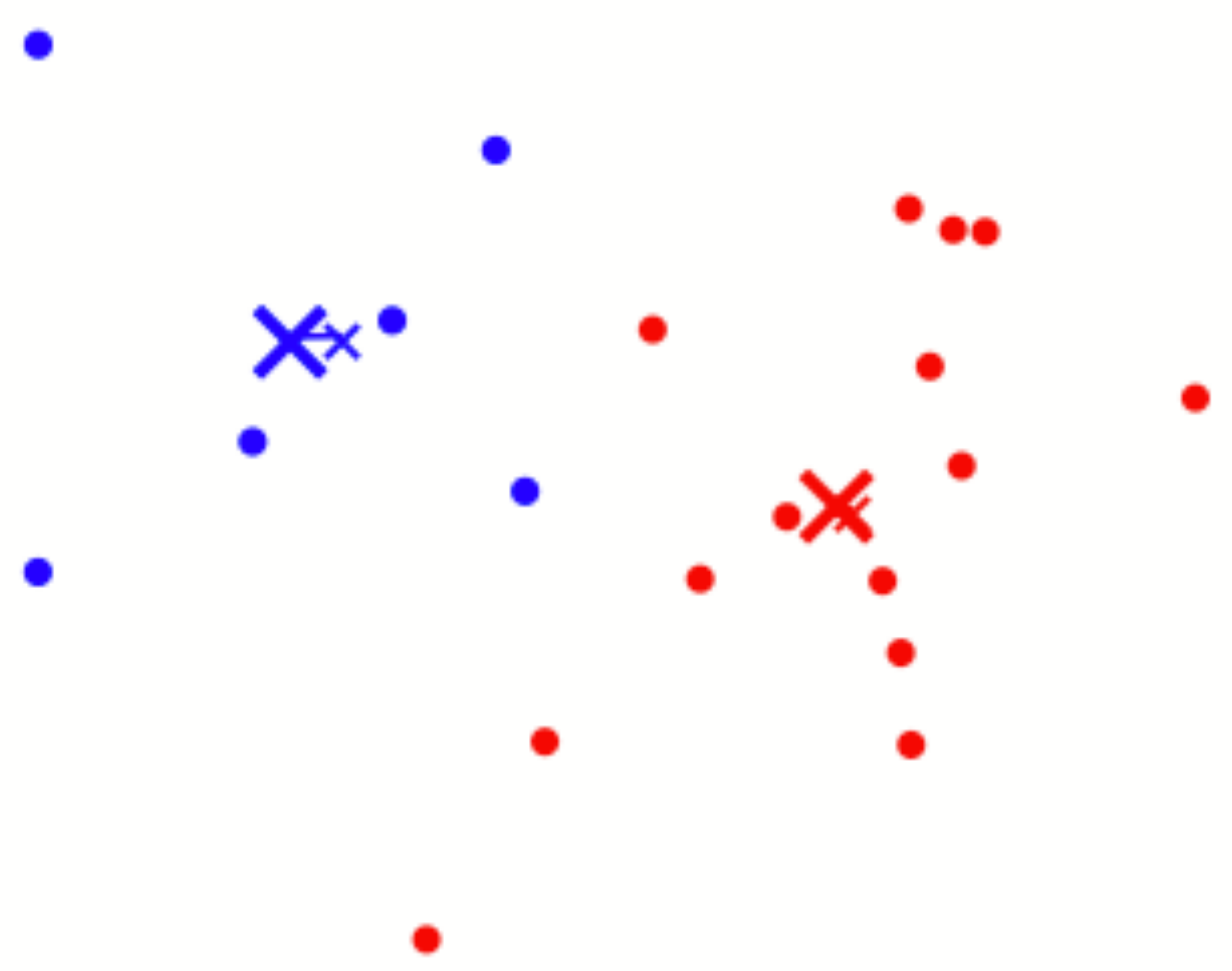
# k-means in action (iteration 5 update centroids)



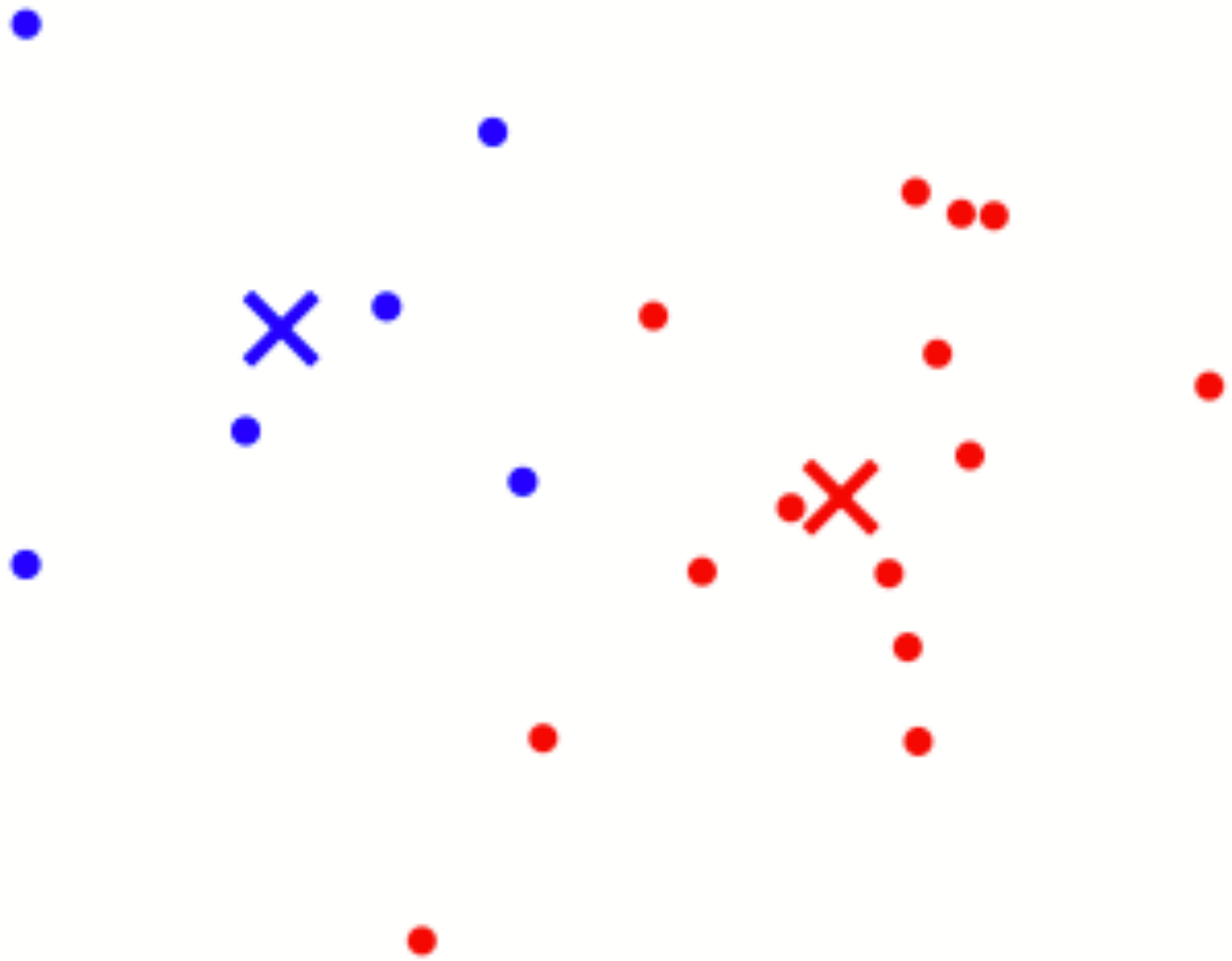
# k-means in action (iteration 6 assignment)



# k-means in action (iteration 6 update centroids)



# k-means in action (convergence)

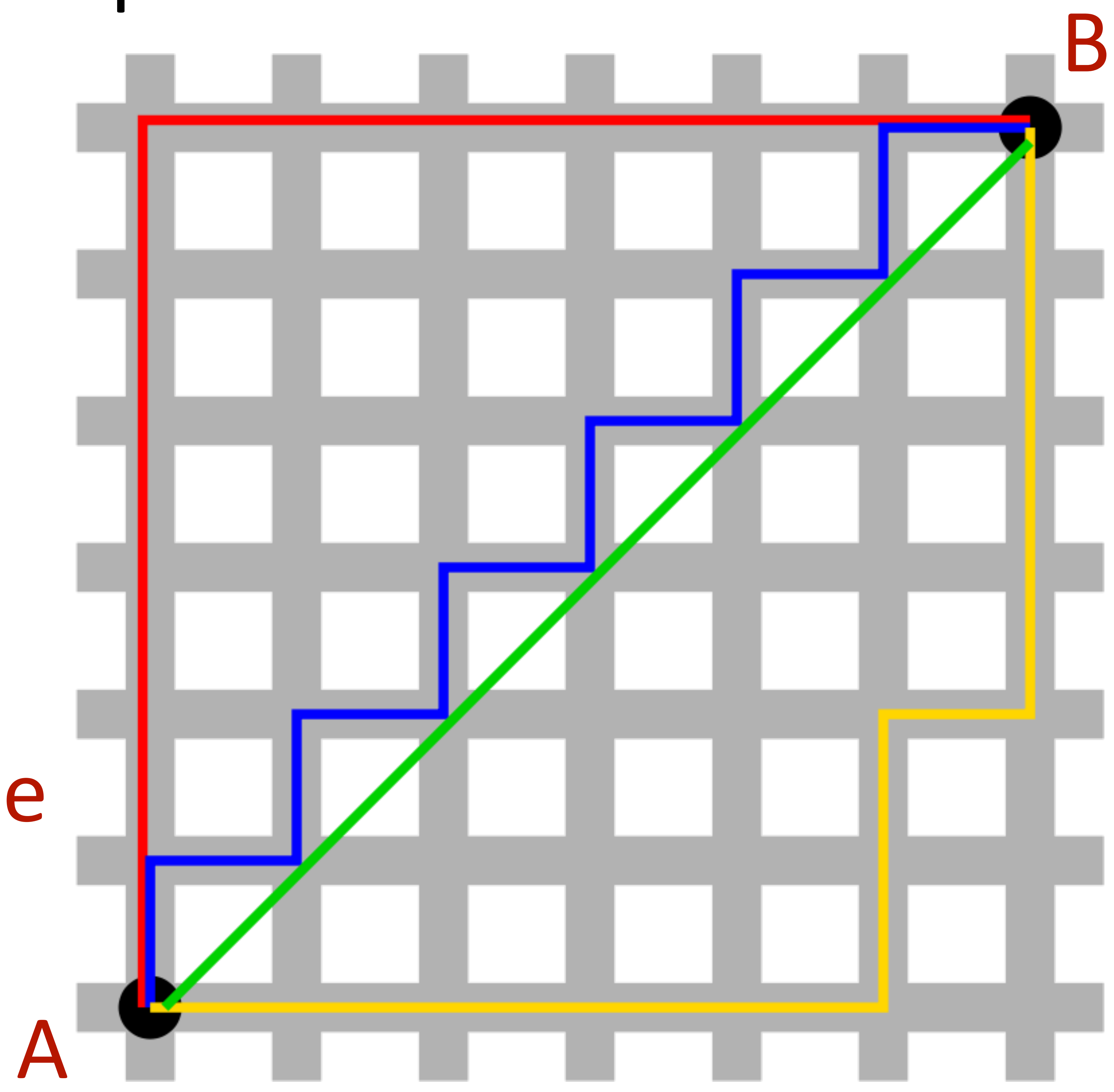


# Convergence

- If WSS decreases at every stage and there are only a finite number of clusterings, then the algorithm must converge to a minimum
- This is almost certainly a local minimum, k-means rarely finds the global optimal solution
- Results often depend on the initialisation and on the number of clusters $K$

# k-medians

- This uses the median of the observations (independently for each dimension) within each cluster as the centroid, instead of the mean
- Overall, it minimises the error over all clusters with respect to Manhattan distance instead of Euclidean distance
- It uses an alternating update like Lloyd's algorithm
- This sometimes works better than k-means. Why?




Blue, yellow, red are all the minimum Manhattan distance

Green is minimum Euclidean distance

# k-medians

- This uses the median of the observations (independently for each dimension) within each cluster as the centroid, instead of the mean
- Overall, it minimises the error over all clusters with respect to Manhattan distance instead of Euclidean distance
- It uses an alternating update like Lloyd's algorithm
- This sometimes works better than k-means. Why?

# k-medoids

- Also similar to k-means
- The centroids are called medoids, these must be observations i.e. $\forall k, c_k = x_i$ for some $i = 1,\ldots,n$
- This can make the centroids easier to interpret (exemplar for each cluster)
- k-medoids uses any dissimilarity measure and minimises the sum of pairwise dissimilarities (more general than k-means, also more robust to noise/outliers)

# Initialisation

- k-means final clustering depends on the initial centroids
- There are various methods to find a good initialisation before running Lloyd's algorithm

# Initialisation: Forgy

- Choose $K$ observations randomly from the observations $x_i$, $i = 1, \ldots, n$
- These will be the initial centroids $c_k$, $k = 1, \ldots, K$
- In high-dimensions, this tends to spread the initial centroids out

# Initialisation: Random partition

- Randomly assign each observation to a cluster, then the centroids are the mean of these starting clusters
- In high-dimensions, this tends to put all of the centroids near the mean of the data

# Initialisation: k-means++

- An initialisation algorithm to 'seed' the centroids
- This aims to spread out the initial centroids

1. Choose one centroid uniformly at random from among the observations
2. For each observation $x_i$ that isn't a centroid, compute the distance $d_i$ between $x_i$ and the nearest centroid
3. Choose a new centroid at random, with weighted probability so that $x_i$ is chosen with probability proportional to $d_i^2$
4. Repeat 2. and 3. until there are $K$ centroids, then run standard k-means

# Other algorithms: Hartigan-Wong method

- A variation of k-means that proposes to move $x_i$ to move from cluster $C_k$ to cluster $C_j$ (for all $j = 1,\ldots,K$) with some acceptance strategy

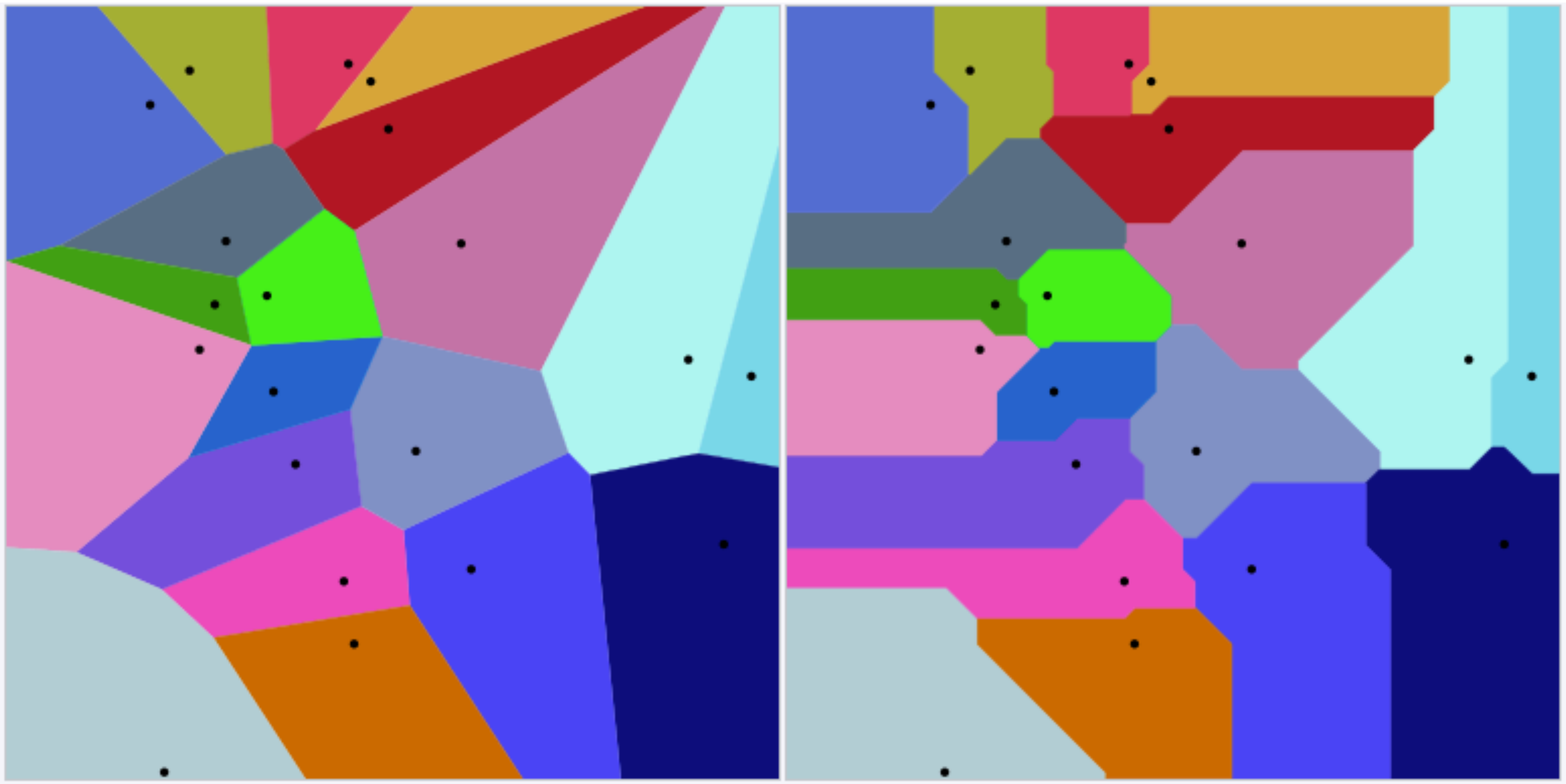
- E.g. cluster cost $\phi(C_k) = \sum_{x_i \in C_k} \|x_i - c_k\|^2$

- Change in cost $\Delta(C_k, C_j, x_i) = \phi(C_k) + \phi(C_j) - \phi(C_k \backslash \{x_i\}) - \phi(C_j \cup \{x_i\})$

- Find the $C_j$ that maximises this, and re-assign $x_i$ to it

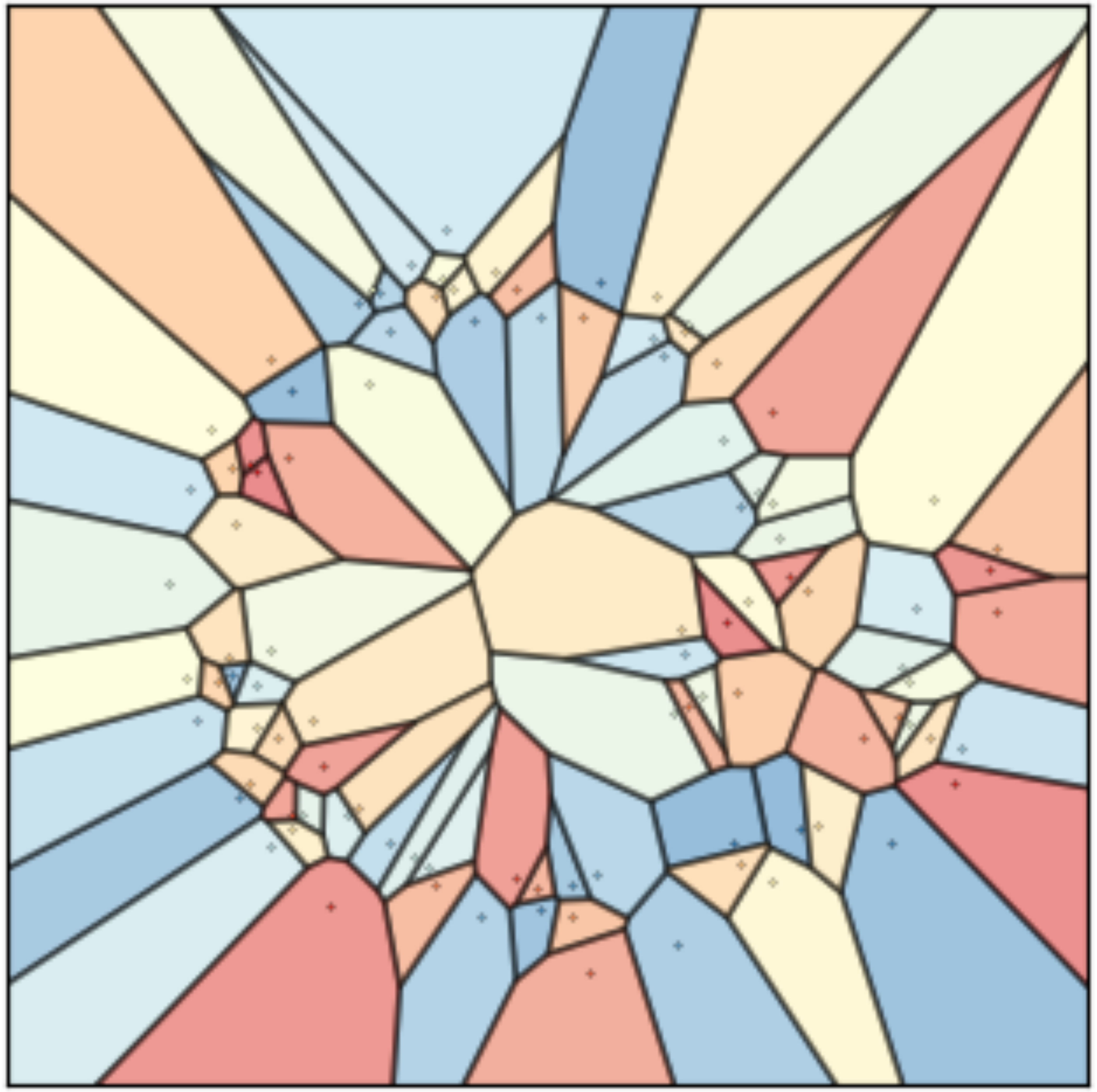
# Voronoi cells look pretty

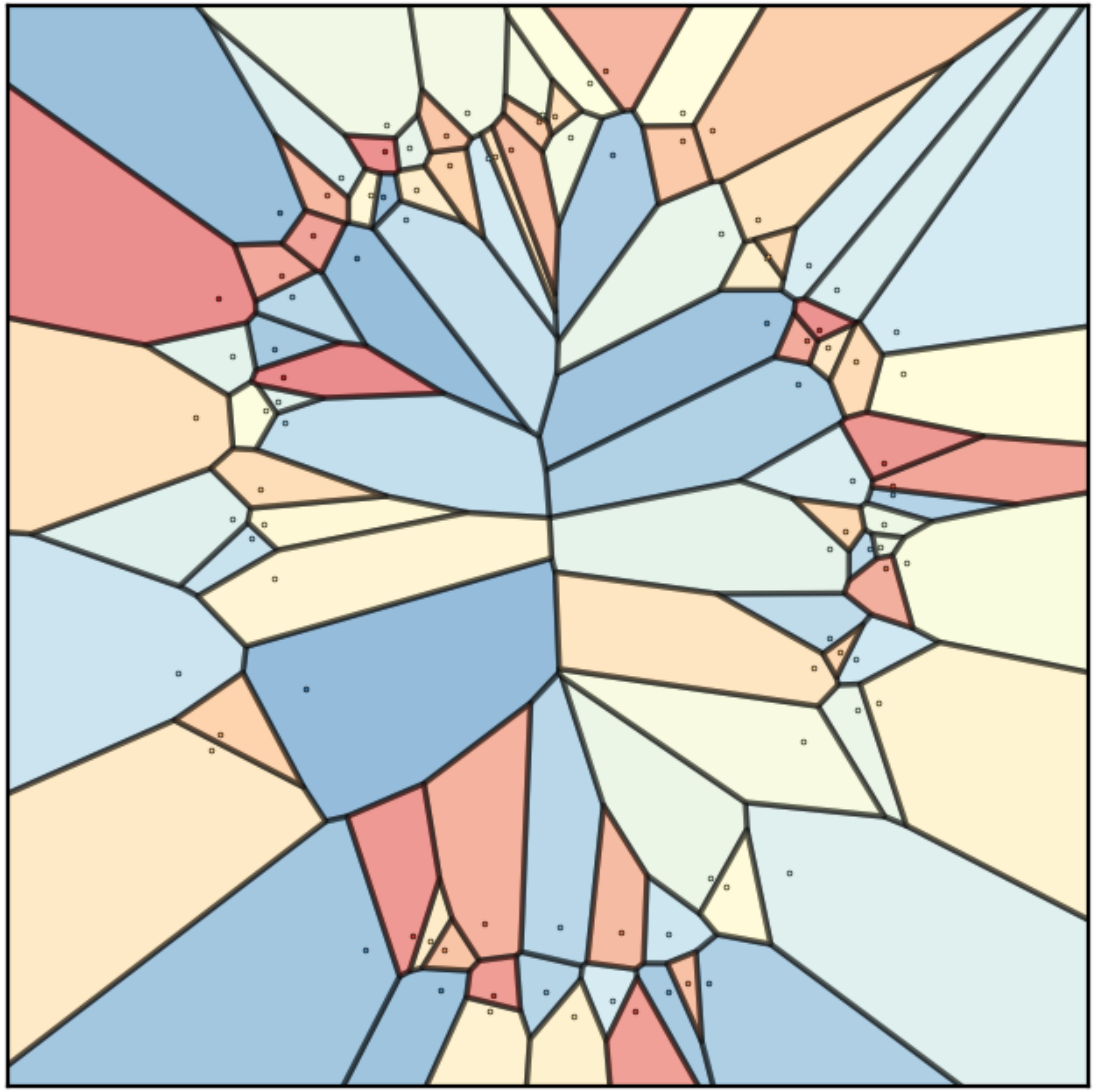
- A set of points and a distance metric partition the space into regions called Voronoi cells
- The region $R_k = \{x : d(x, c_k) \leq d(x, c_j),\ \forall j \neq k\}, x_i \in R_k \iff x_i \in C_k$



Euclidean distance | Manhattan distance

# Voronoi cells (spiral, 0)





# Fuzzy c-means

- Fuzzy or soft clustering assigns each observation a weight of belonging to each cluster, so it can belong to multiple clusters

- Cluster membership is graded to indicate the degree to which each observation belongs to a cluster (i.e. an observation in the middle of cluster is in the cluster to a greater degree than one at the edge of the cluster)

- Cluster membership matrix $W$ (size $n \times K$) with $w_{ik} \in [0,1]$ the degree to which observation $x_i$ belongs to cluster $k$

# Fuzzy c-means

- Two steps (as with k-means)
  1. Multiple cluster assignment
     - $x_i$ has cluster assignment $w_{ik}$, with $w_{ik}^{-1} = \sum_{j=1}^{K} \left( \frac{\|x_i - c_k\|}{\|x_i - c_j\|} \right)^{\frac{2}{m-1}}$
  2. Centroid update
     - New cluster centroids are $c_k = \frac{\sum_{i=1}^{n} w_{ik}^m x_i}{\sum_{i=1}^{n} w_{ik}^m}$
- This minimises the weighted mean squared error $E = \sum_{i=1}^{n} \sum_{k=1}^{K} w_{ik}^m \|x_i - c_k\|^2$
- $m$ is a fuzziness hyper parameter

# Questions?

- Feel free to email me at te269@cam.ac.uk

# Next time

- Clustering
  - Hierarchical clustering