Big Data and Data Mining

Unsupervised Learning

Flavio Bertini

flavio.bertini@unipr.it

We have seen so far...

- Until now we have seen supervised methods, in which a human **training** (supervision) was needed
  - Training was given as a **large set of examples**
- Each example was an association from a set of **input variables** (X) to an **output variable** (y)

Price	Rooms	SQM
700000	5	230

→

City
Rome

- Now we will see **unsupervised methods**, in which no training is involved They are used to:
  - Mine frequent patterns to make **associations** between examples
  - Group together similar objects creating **clusters**

Price	Rooms	SQM
700000	5	230

→

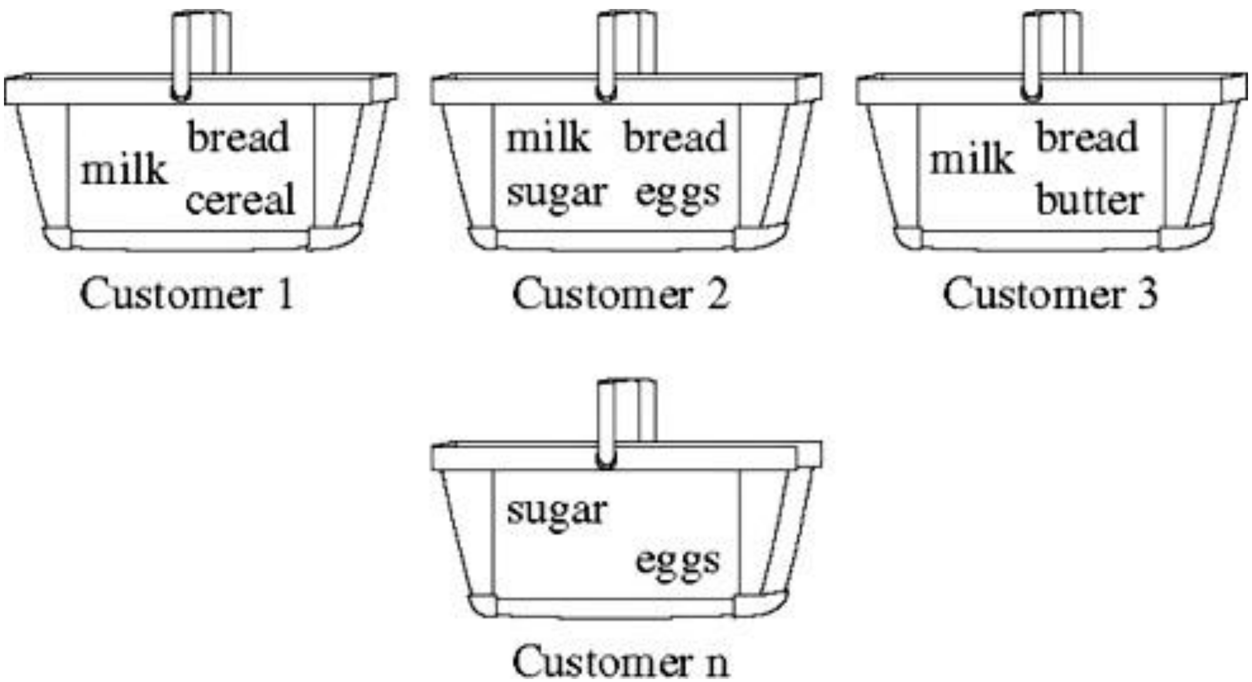
?

Mining frequent patterns

- Mine frequent patterns to make **associations** between examples allows to discover new relations from data
  - Frequent patterns are patterns that appear frequently in a data set
- This kind of technique can be applied in different scenarios, such as:
  - Marketing basket analysis (frequent patterns in products buying)
  - Bioinformatics (associating genes and diseases)
  - Intrusion detection systems (finding malicious activities)

Market basket analysis

- A typical example of frequent pattern mining is **market basket analysis**
- This process analyzes customer **buying habits** by finding associations between the different items that customers place in their "shopping baskets"
  - For instance, if customers are buying milk, how likely are they **to buy also** bread (and what kind of bread) on the same trip to the supermarket?



Example: AllElectronics

- As a manager of an *AllElectronics* branch you wonder: “Which groups or sets of items are customers likely to purchase on a given trip to the store?”
  - To answer your question, **market basket analysis** may be performed on the data of customer transactions at your store
- You can then use the results to design a smart store layout
  - Items that are frequently purchased together can be placed in proximity to encourage the combined sale of such items
  - Example: if customers who purchase computers also tend to buy antivirus software at the same time, then placing the hardware display close to the software display may help increase the sales of one or both items

Association Rules

- Transactions data can be analyzed for buying patterns that reflect items that are frequently associated (purchased together)
- These patterns can be represented in the form of **association rules**
  - For example, the information that *customers who purchase computers also tend to buy antivirus software at the same time* is represented in the following **association rule**:

```
computer => antivirus_software [support = 2%, confidence = 60%]
```

- *[support = 2%]* means that in the 2% of all the transactions, computer and antivirus software are bought together
- *[confidence = 60%]* means that 60% of the people who bought a computer also bought an antivirus software at the same time

Support & Confidence

- We can now have a more formally look at the two measures
  - Let *A* be a set of items (can be a singlet {*computer*})
  - Let *B* be a set of items (can be a singlet {*antivirus_software*})
- The rule $A \Rightarrow B$ holds in the transactions set with **support** *s*, where *s* is the percentage of all transactions, that **contains** $A \cup B$

$$support(A \Rightarrow B) = P(A \cup B)$$

- The rule $A \Rightarrow B$ holds in the transactions set with **confidence** *c*, where *c* is the percentage of transactions containing *A*, that also contain B

$$confidence(A \Rightarrow B) = P(B|A)$$

Mining strong association rules

- Association rules are considered **strong** if they satisfy **both**:
  1. A **minimum support threshold**
  2. A **minimum confidence threshold**
- These thresholds can be set by users or domain experts
- In general, **association rule mining** can be viewed as a two-step process:
  1. **Find all frequent itemsets** *A*: by definition, each of these itemsets will occur at least as frequently as a predetermined minimum support (if *A* doesn't satisfy the threshold, neither will $A \cup B$: if milk is not among the transactions, neither is milk and coffee)
  2. **Generate strong association rules** from the frequent itemsets *A* to *B*: by definition, these rules must satisfy minimum support and minimum confidence

Example: AllElectronics data

- Consider the following transactions data for *AllElectronics*:

Transaction ID	Item set
T100	{Item1, Item2, Item5}
T200	{Item2, Item4}
T300	{Item2, Item3}
T400	{Item1, Item2, Item4}
T500	{Item1, Item3}
T600	{Item2, Item3}
T700	{Item1, Item3}
T800	{Item1, Item3}
T900	{Item1, Item2, Item3, Item5}
T1000	{Item1, Item2, Item3}

- We set minimum support 30% and confidence 70%

Example: iteration 1

- We first consider the *support* of itemsets with cardinality 1:

1-Itemsets	Support
{Item1}	70%
{Item2}	70%
{Item3}	70%
{Item4}	20%
{Item5}	20%

- With support 30% (and confidence 70%)
  - The itemsets that **does not** satisfy minimum support are pruned
  - The itemsets that **does** satisfy minimum support are taken in the next iteration

Example: iteration 2

- We now consider the *support* of itemsets with cardinality 2:

2-Itemsets	Support
{Item1, Item2}	40%
{Item1, Item3}	50%
{Item2, Item3}	40%

- All itemsets satisfy minimum support threshold and are taken in the next iteration

Example: iteration 3

- We now consider the *support* of itemsets with cardinality 3:

3-Itemsets	Support
{Item1, Item2, Item3}	20%

- The itemset **does not** satisfy minimum support are pruned
- We stop the iterations as there can be no larger itemsets
- We have therefore found the **largest frequent itemsets** as:

2-Itemsets	Support
{Item1, Item2}	40%
{Item1, Item3}	50%
{Item2, Item3}	40%

Example: association rules

- We found the frequent itemsets {Item1, Item2}, {Item1, Item3}, {Item2, Item3}
- What are the **association rules** that can be generated from the above frequent itemsets?
  - `{Item1} ⇒ {Item2} [support=40%]`
  - `{Item2} ⇒ {Item1} [support=40%]`
  - `{Item1} ⇒ {Item3} [support=50%]`
  - `{Item3} ⇒ {Item1} [support=50%]`
  - `{Item2} ⇒ {Item3} [support=40%]`
  - `{Item3} ⇒ {Item2} [support=40%]`
- We already computed their support in the search for frequent itemsets, therefore we know that minimum support is satisfied
- We now go back to the data to compute their confidence and check if they are **strong** association rules (confidence ≥ 70%)

Example: confidence

Trans. ID	Item set
T100	{Item1, Item2, Item5}
T200	{Item2, Item4}
T300	{Item2, Item3}
T400	{Item1, Item2, Item4}
T500	{Item1, Item3}
T600	{Item2, Item3}
T700	{Item1, Item3}
T800	{Item1, Item3}
T900	{Item1, Item2, Item3, Item5}
T1000	{Item1, Item2, Item3}

- We can compute confidence with the support values we already computed

$$confidence(A \Rightarrow B) = P(B|A) = \frac{support(A \cup B)}{support(A)}$$

- Goal: support 30% and confidence 70%

- {Item1} ⇒ {Item2} [support=40%, confidence=4/7= 57%]
- {Item2} ⇒ {Item1} [support=40%, confidence=4/7= 57%]
- {Item1} ⇒ {Item3} [support=50%, confidence=5/7= 71%]
- {Item3} ⇒ {Item1} [support=50%, confidence=5/7= 71%]
- {Item2} ⇒ {Item3} [support=40%, confidence=4/7= 57%]
- {Item3} ⇒ {Item2} [support=40%, confidence=4/7= 57%]

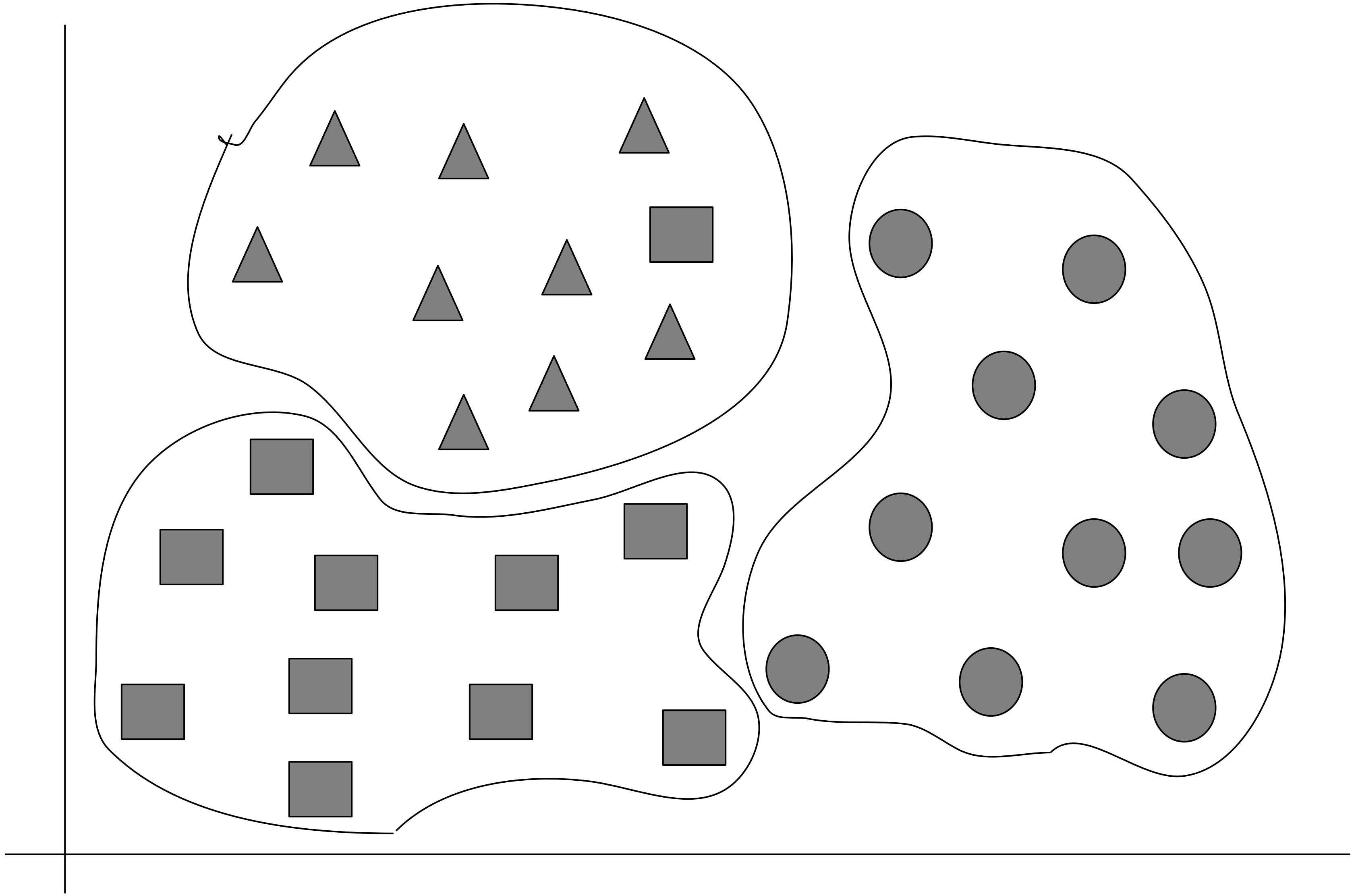
Mining association rules: recap

- We have seen how to iteratively find frequent itemsets and filter them by minimum support and minimum confidence rules
- The algorithm we used to find strong association rules is called ***a priori* algorithm**
  - The *a priori* property says that ***if a set cannot pass a test, all of its supersets will fail the same test as well*.** This allowed us to prune and reduce the search space
- **Association rules mining** of itemsets is an example of ***unsupervised learning* task**
- We will now see how items (or, more generally, objects) can be *clustered* together based on their common features
  - This method is called **clustering**

Clustering

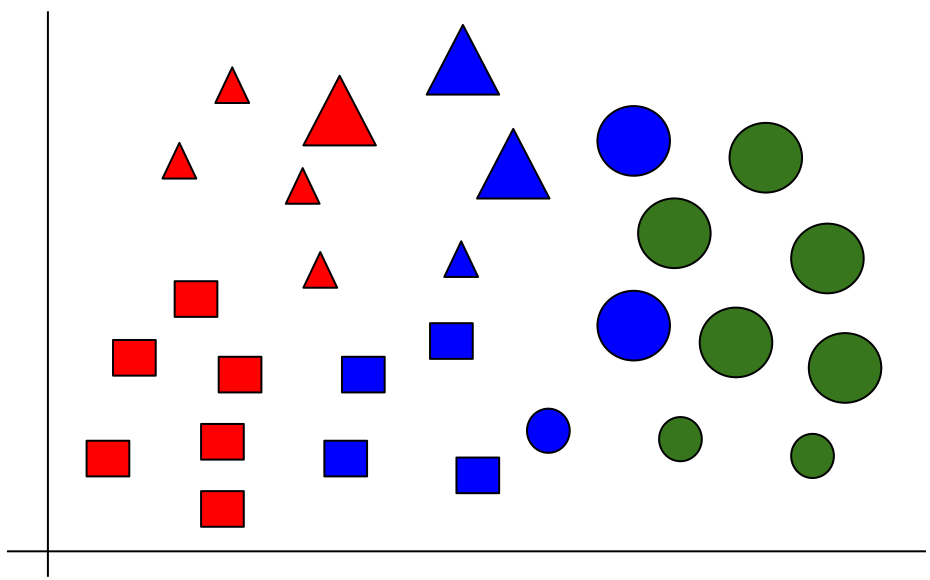
- Goal of clustering is to group together similar objects (documents, patients, houses) into “clusters”, depending on their features:
  - usually, objects are **grouped together if they belong to the same “category”** but...
  - **categories** are **not known**!
- Clustering has no “*ground truth*” since:
  - it’s an unsupervised technique: there is no labeled data
  - evaluation depends on perspective, may be different for different people
  - being without supervision restrictions, it may find new, unknown common features shared by objects: *pattern discovery*

Clustering example



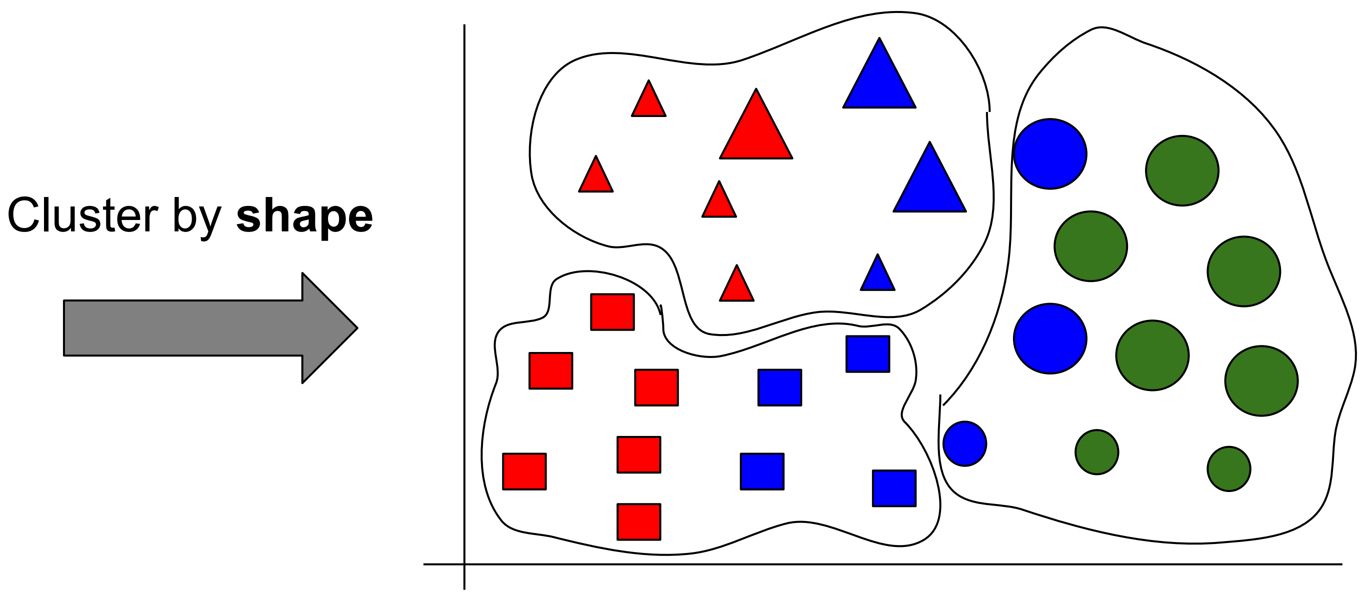
Clustering Bias (1)

- It's common to say that clusters are "in the eye of the beholder":
  - are the terms "horse" and "car" similar?
  - in the previous example the clustering bias was the shape of each object, it was obvious
  - with more features it may be less obvious



Clustering Bias (2)

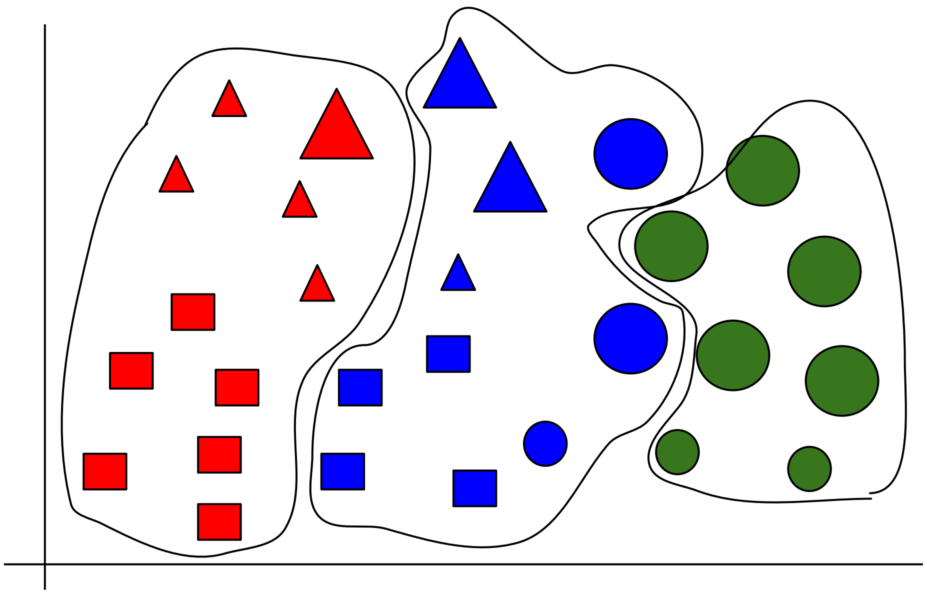
- It's common to say that clusters are "in the eye of the beholder":
  - are the terms "horse" and "car" similar?
  - in the previous example the clustering bias was the shape of each object, it was obvious
  - with more features it may be less obvious



Clustering Bias (3)

- It's common to say that clusters are "in the eye of the beholder":
  - are the terms "horse" and "car" similar?
  - in the previous example the clustering bias was the shape of each object, it was obvious
  - with more features it may be less obvious

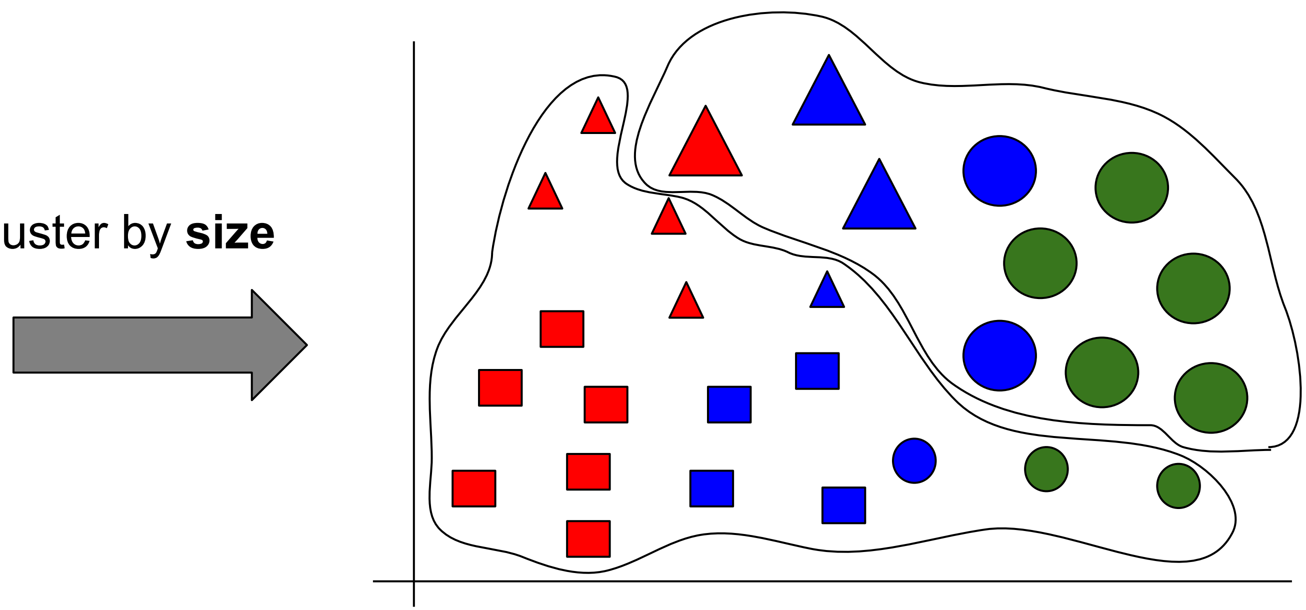
Cluster by **color**



Clustering Bias (4)

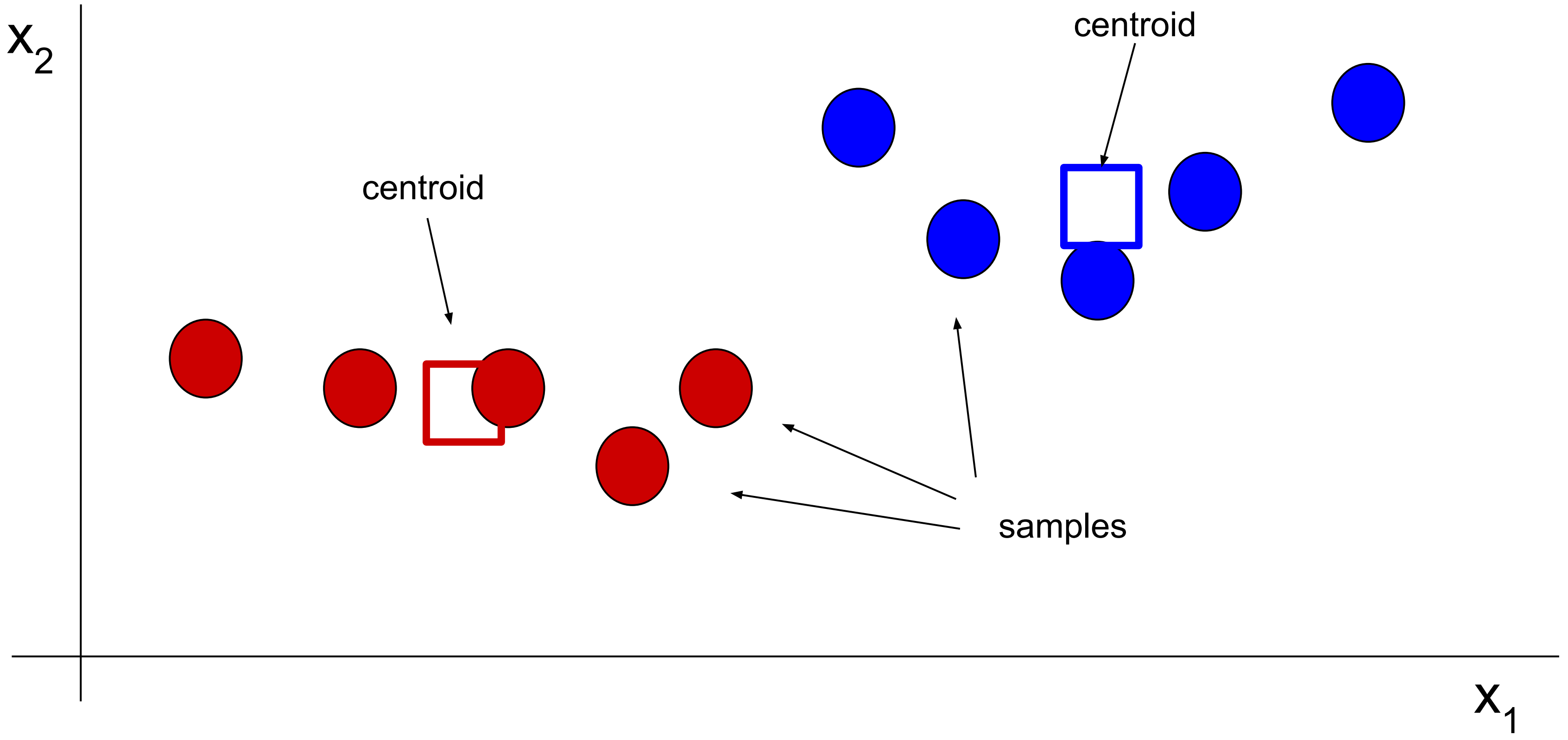
- It's common to say that clusters are "in the eye of the beholder":
  - are the terms "horse" and "car" similar?
  - in the previous example the clustering bias was the shape of each object, it was obvious
  - with more features it may be less obvious

Cluster by **size**



k-means

- k-means clustering aims to partition n samples into k clusters in which each sample **belongs to the cluster with the nearest mean**, serving as a prototype of the cluster



k-means

- Algorithm for k-means starts with a “weak” clustering that is **refined** in many rounds **until nothing changes**
- The parameter k is user-defined (number of clusters)

ALGORITHM

```
samples = [(x11,x12),...,(xm1,xm2)]   # our dataset of m samples in 2-dimension space
centroids = pick_random(samples,k) # pick k random samples as centroids

for each sample in samples:         # first weak clustering
     cluster[sample] = nearest(sample,centroids) # assign to nearest centroids

anychange = True

while (anychange):
     anychange = False
     centroids = [mean(cluster1),...,mean(clusterk)] # recompute centroids
     for each sample in samples:
          if cluster[sample] != nearest(sample,centroids): # we should move
               cluster[sample] = nearest(sample,centroids)
               anychange = True
```

k-means convergence

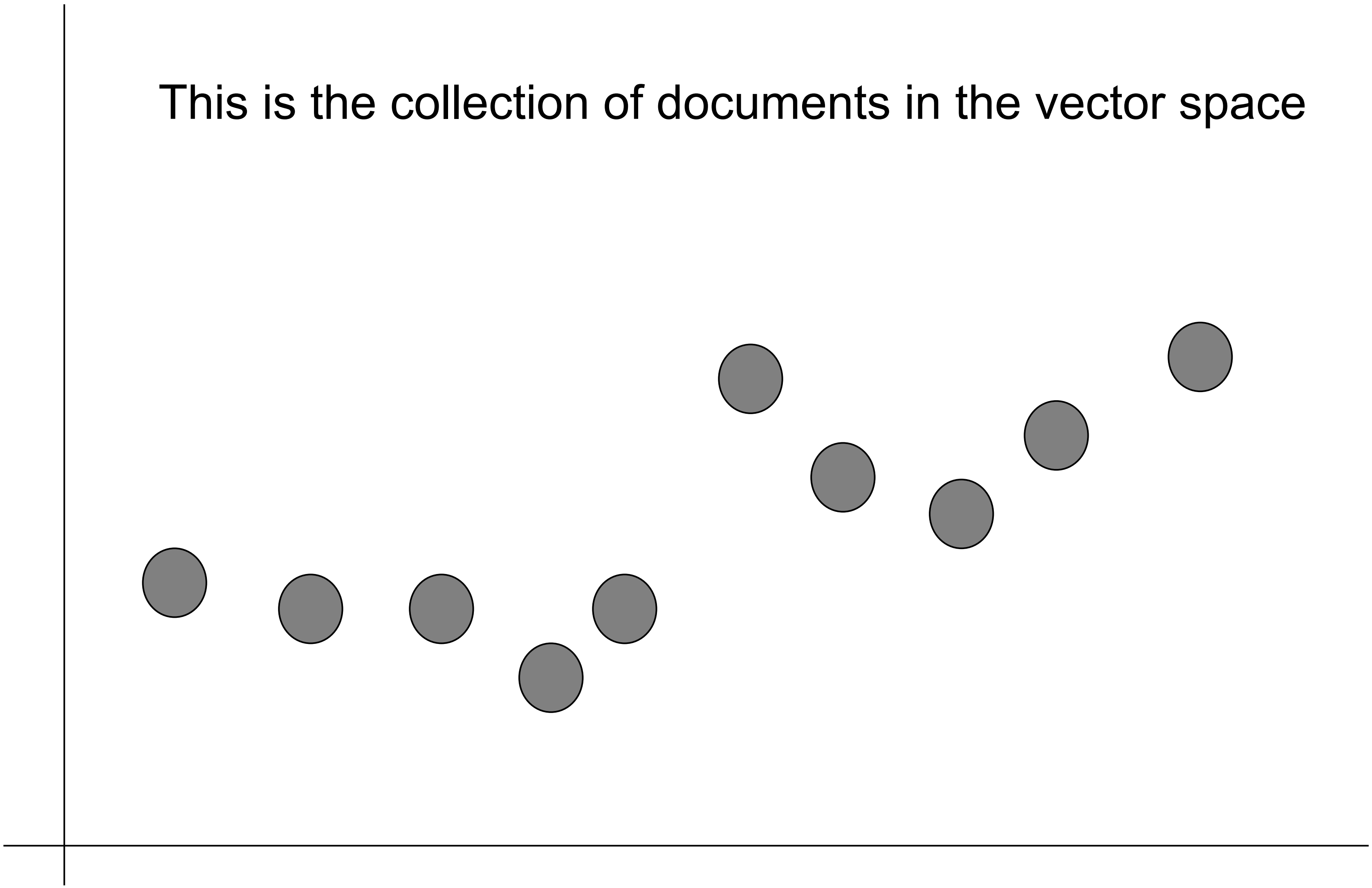
- The **Stirling numbers of the Second kind** gives the number of subdivisions of a set of n objects into k cluster. It is a large but finite number

$$\left\{ \begin{matrix} n \\ k \end{matrix} \right\} = \frac{1}{k!} \sum_{j=1}^{k} (-1)^{k-j} \binom{k}{j} j^n$$

- For each iteration of the algorithm, we produce a new clustering based only on the old clustering
- At each round there can be one of two cases:
  1. The old clustering is the same as the new → the iteration stops
  2. The new clustering is different → the distance between the samples and the centroids is lower
- If the old clustering is the same as the new, then the next clustering will again be the same
- If the new clustering is different from the old then the newer one has a lower cost

k-means: example

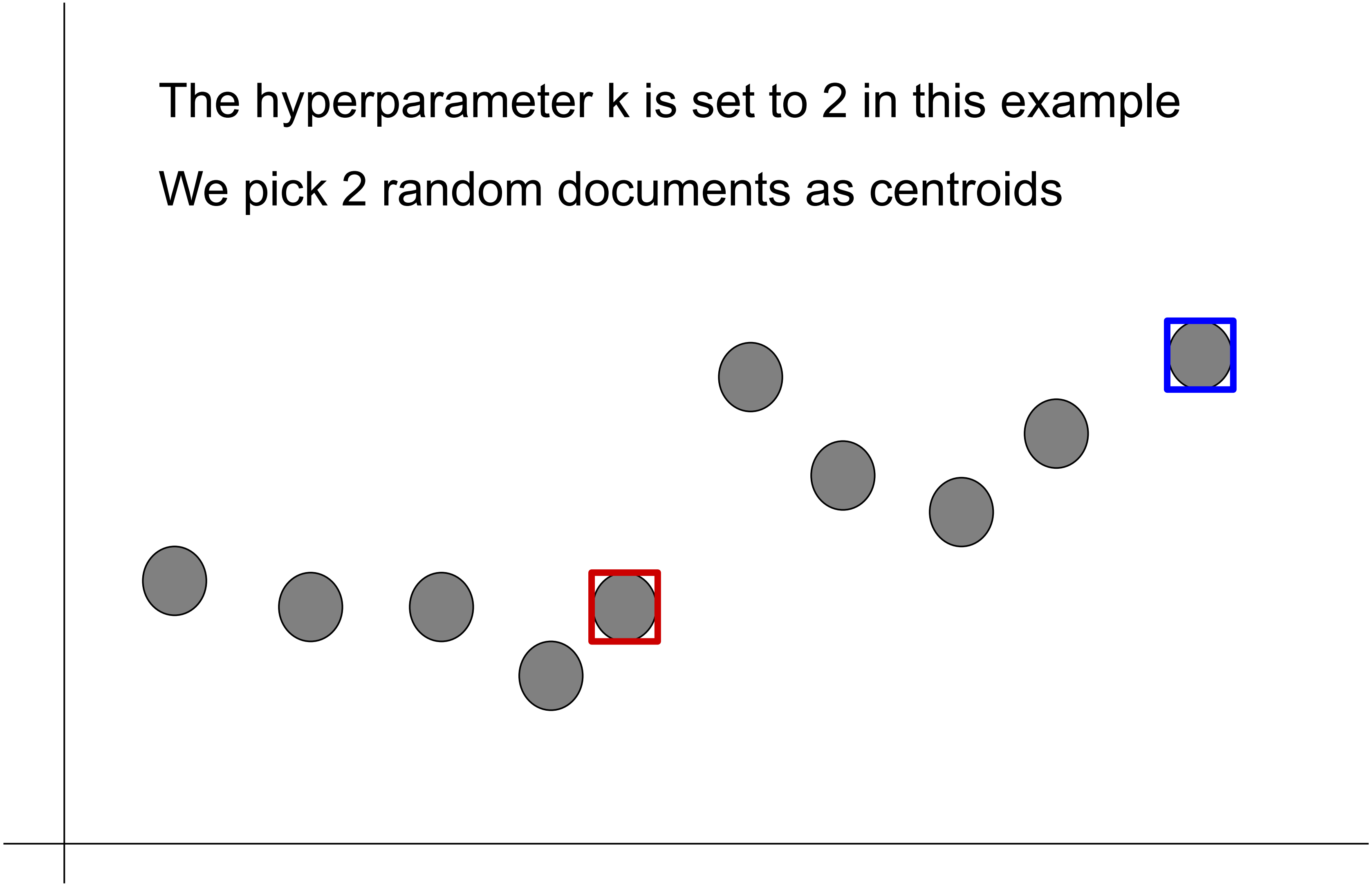
This is the collection of documents in the vector space



k-means: first round

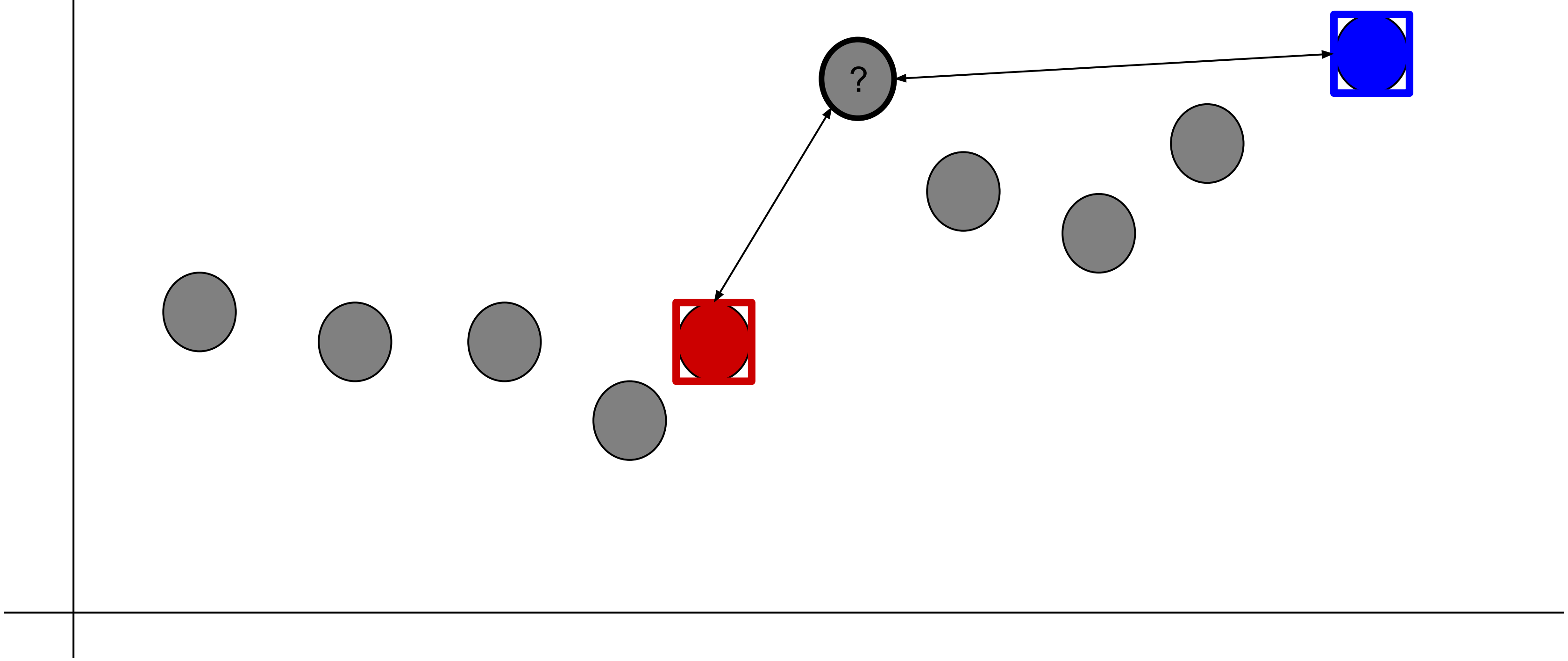
The hyperparameter k is set to 2 in this example

We pick 2 random documents as centroids



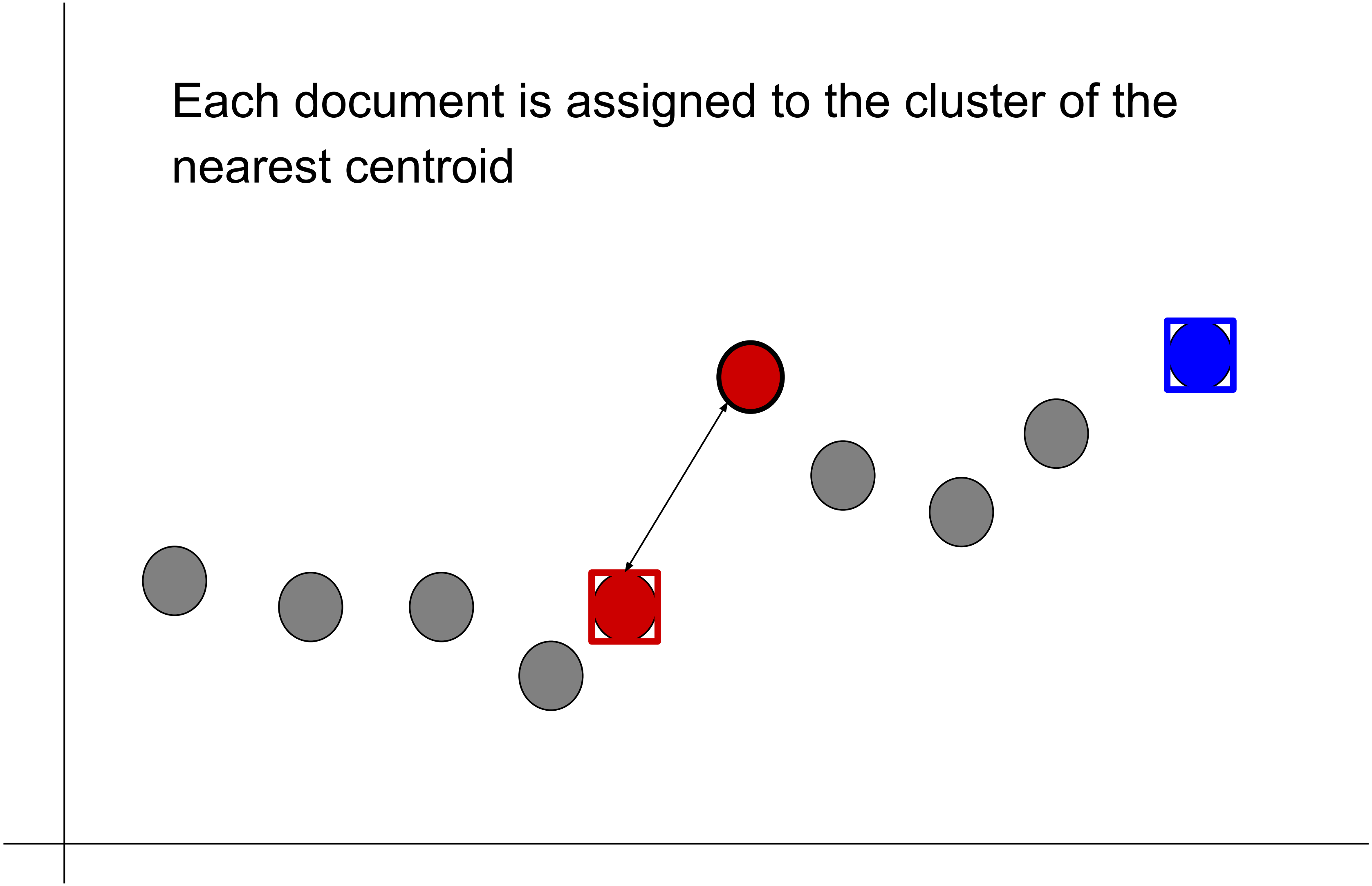
k-means: first round

Each document is assigned to the cluster of the nearest centroid



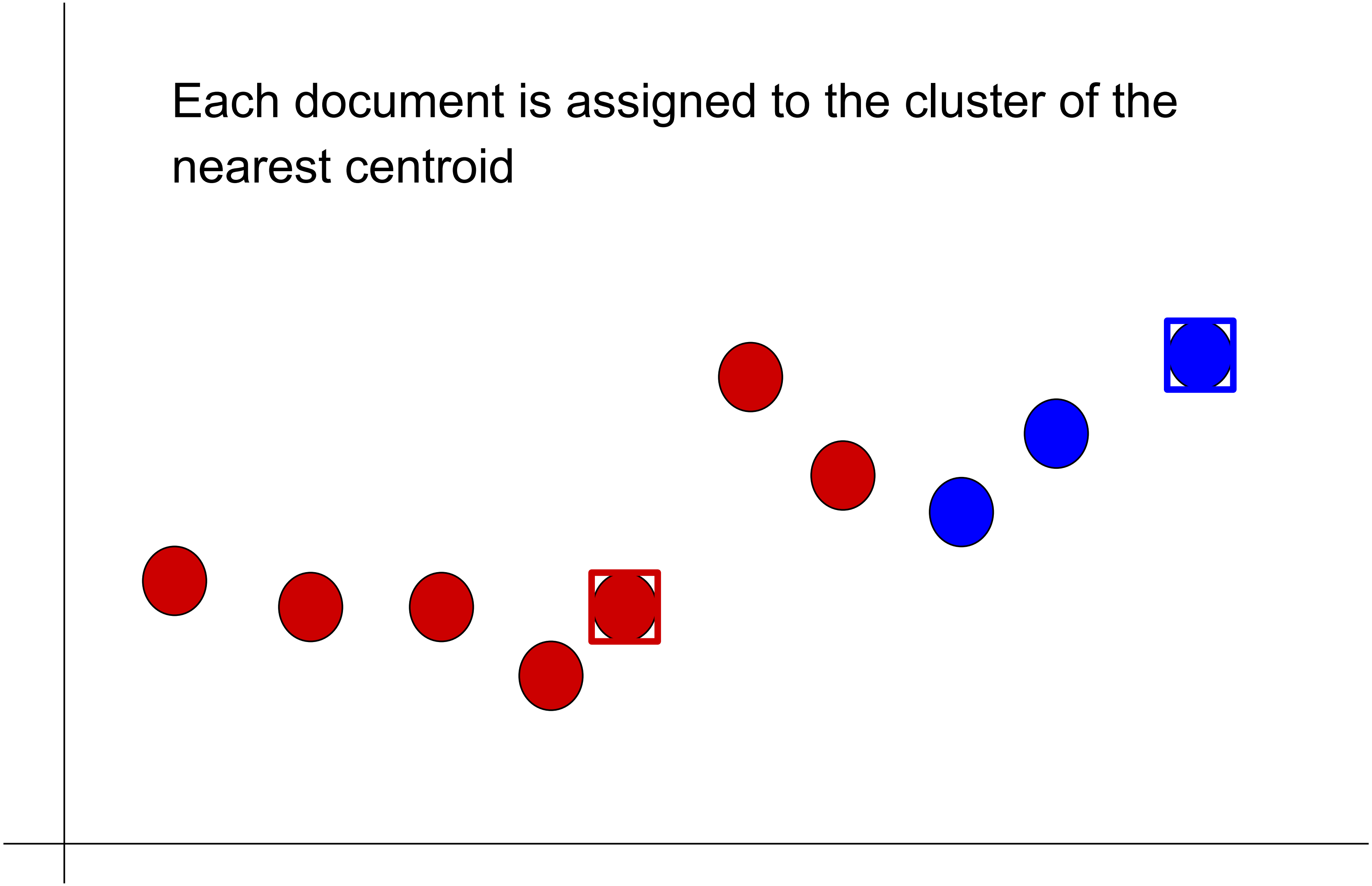
k-means: first round

Each document is assigned to the cluster of the nearest centroid



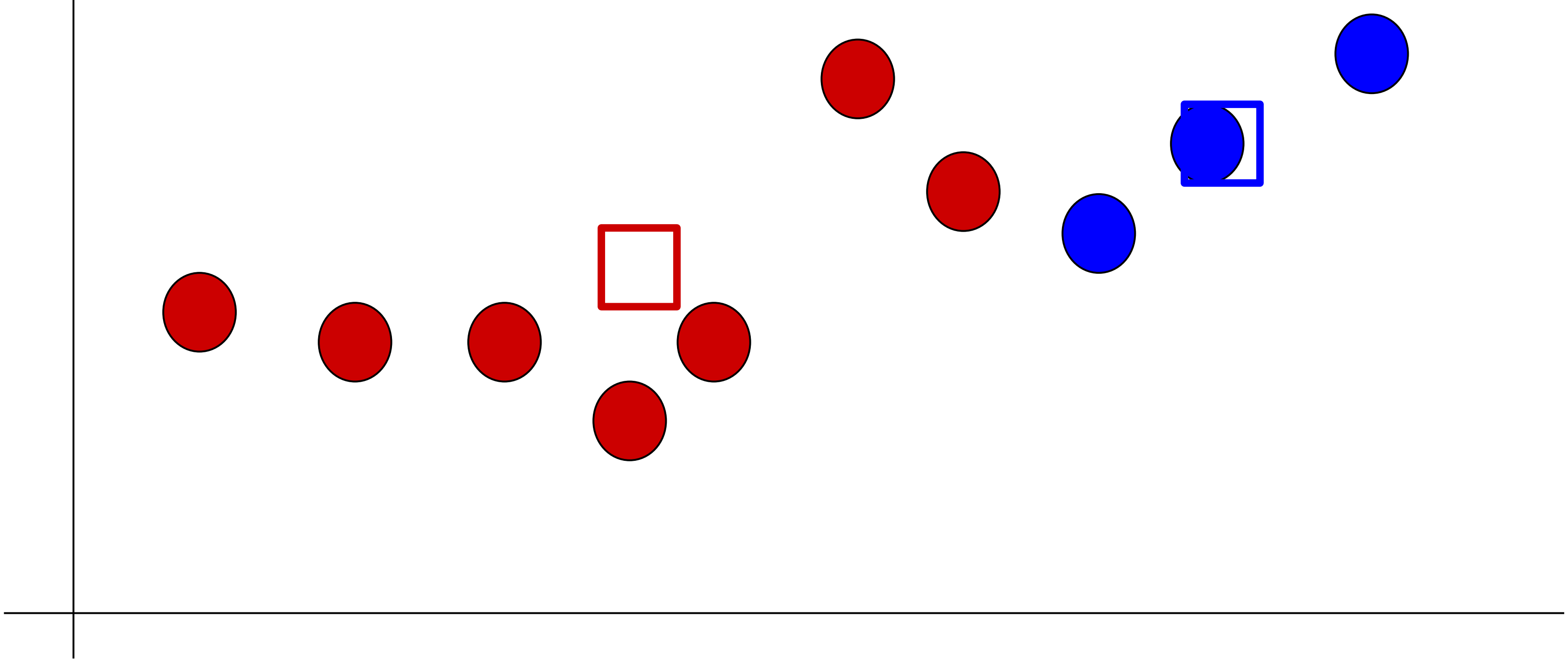
k-means: first round

Each document is assigned to the cluster of the nearest centroid



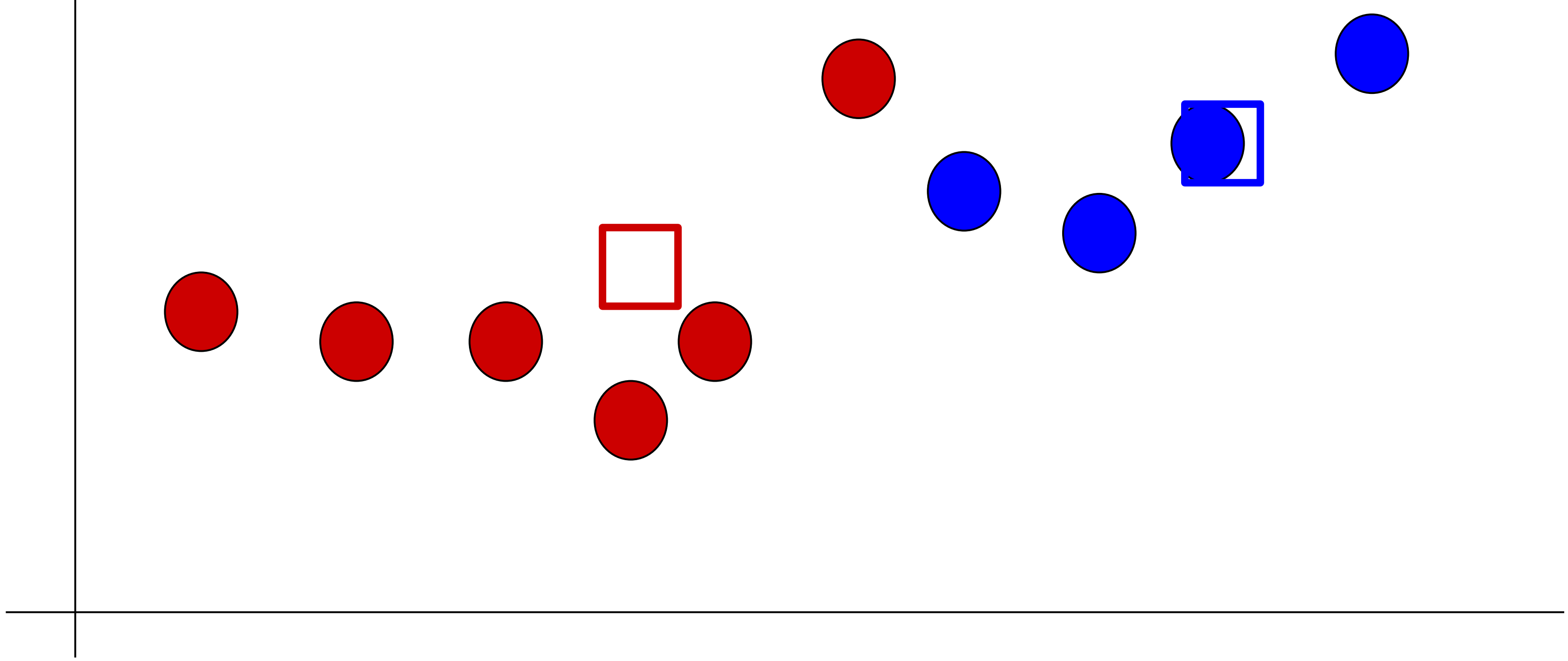
k-means: second round

Centroids are updated: real centroid of the assigned samples

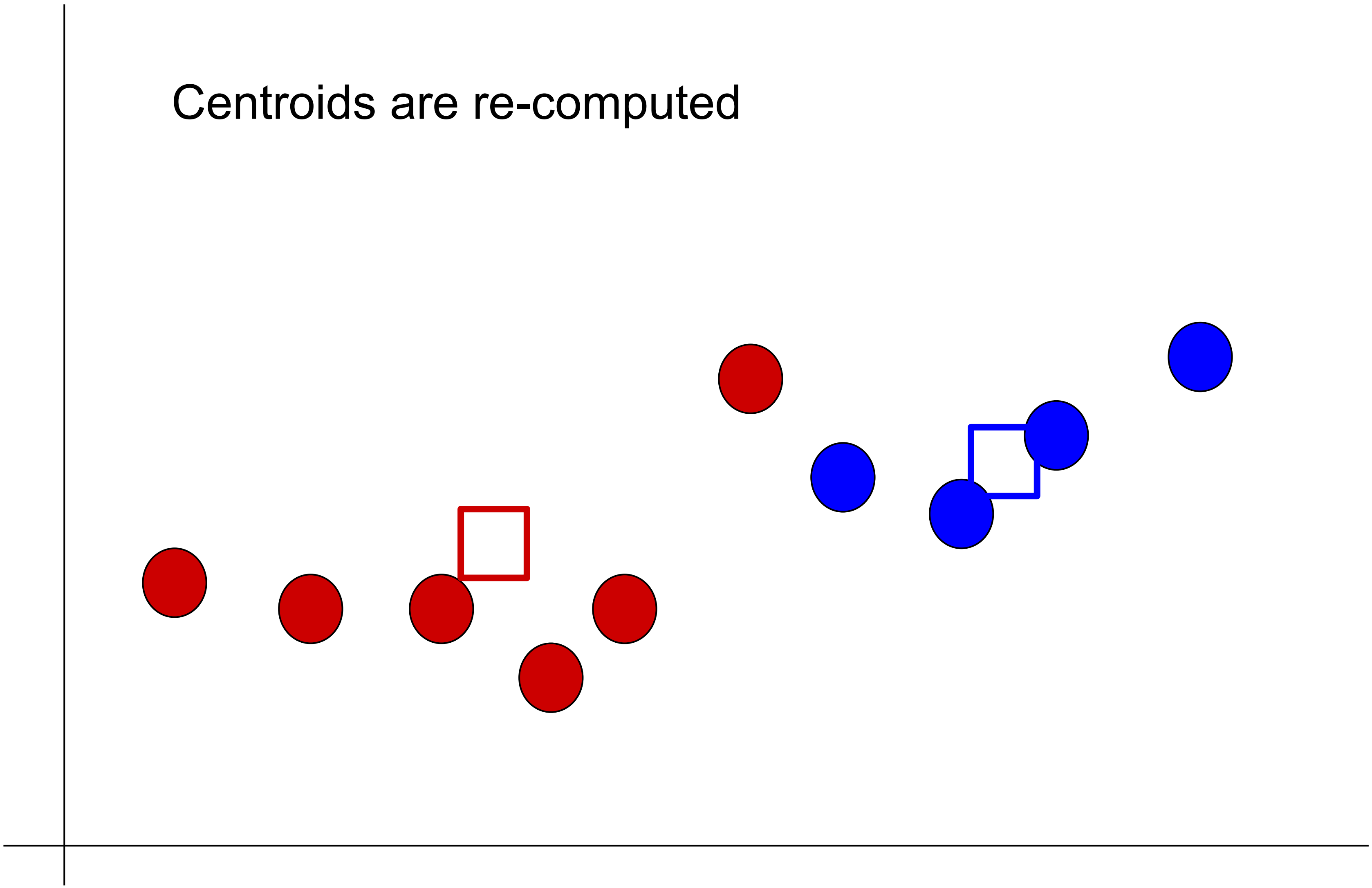


k-means: second round

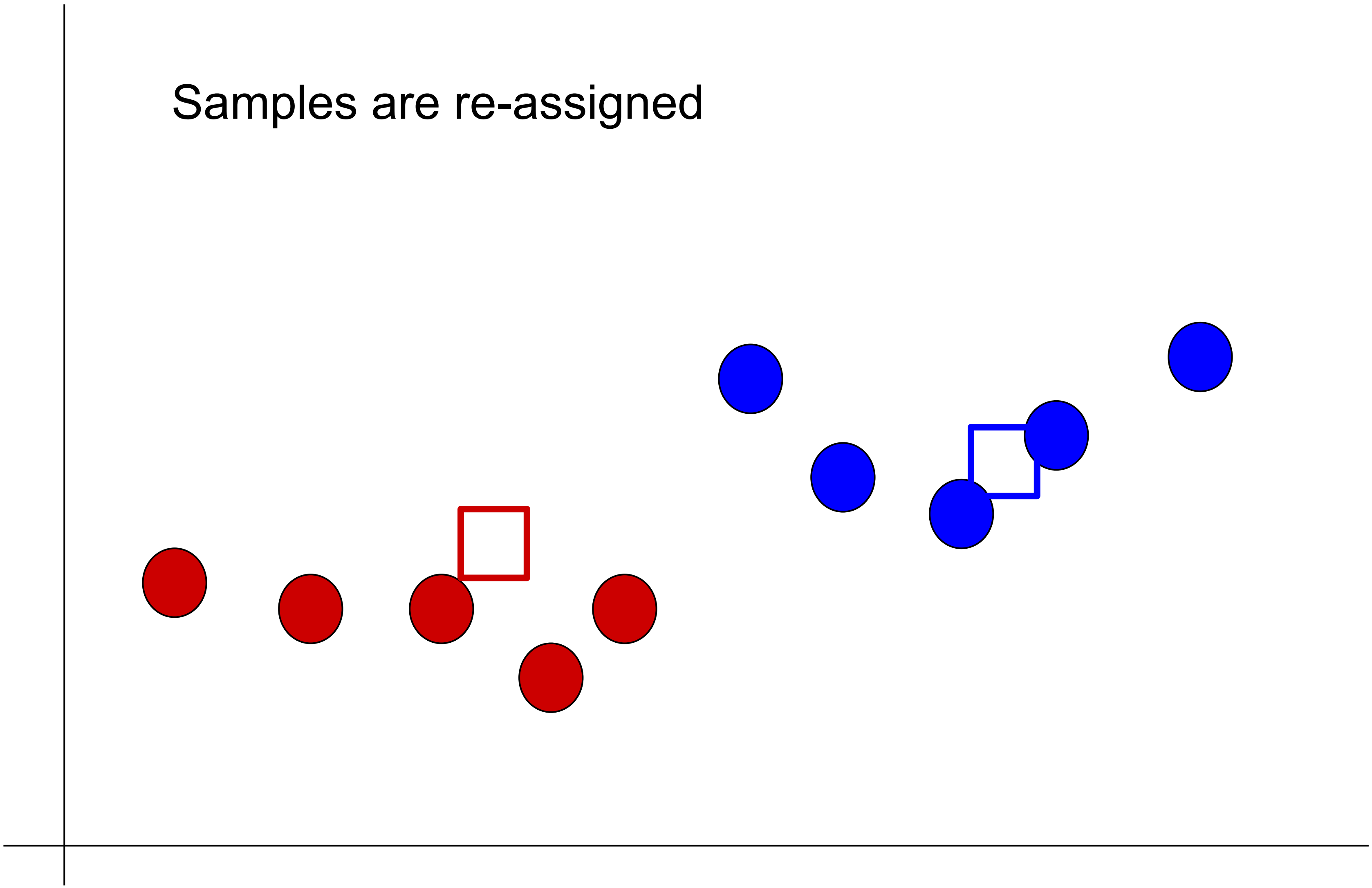
Samples are re-assigned on the basis of the new centroids



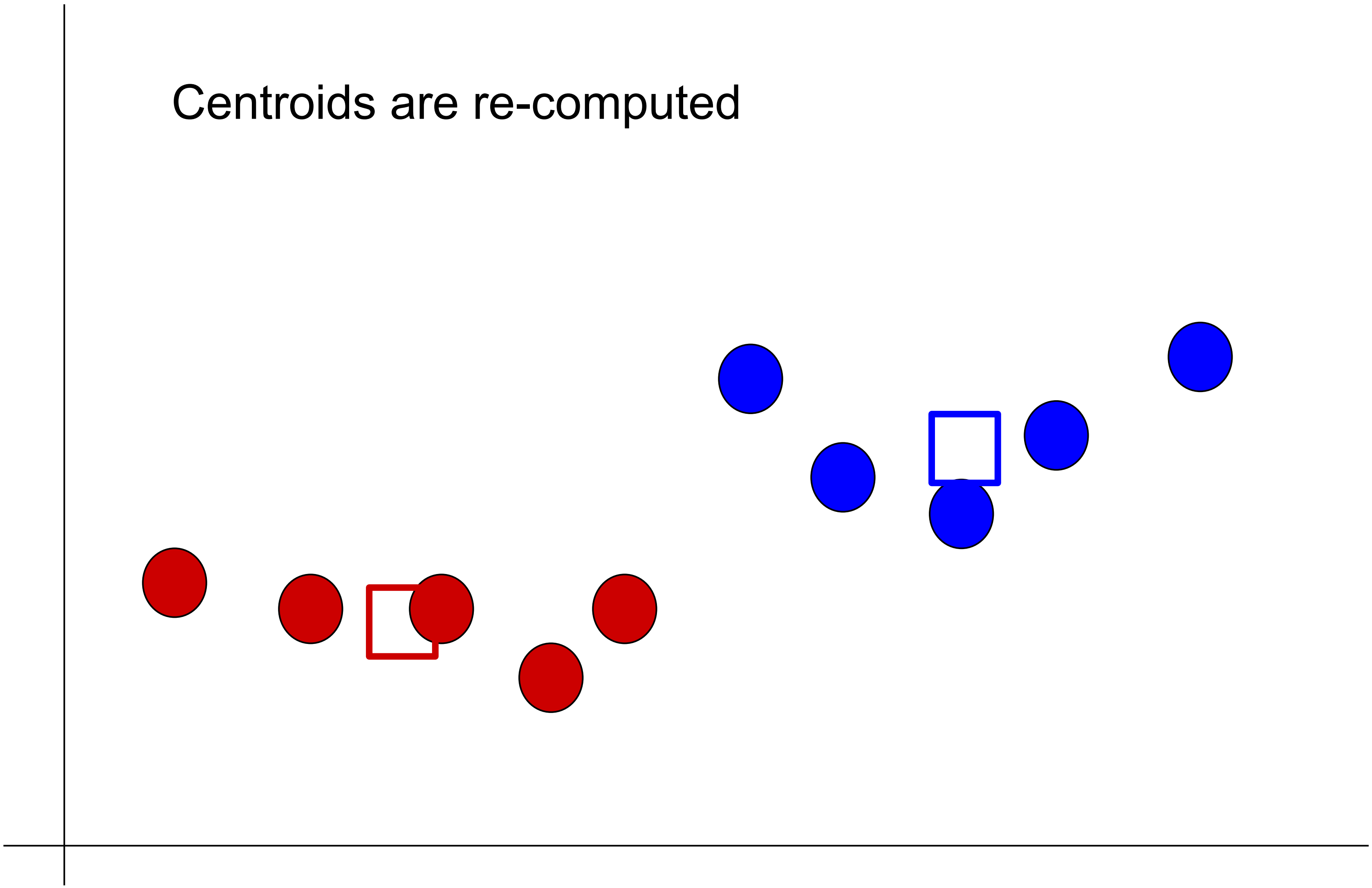
k-means: third round



k-means: third round



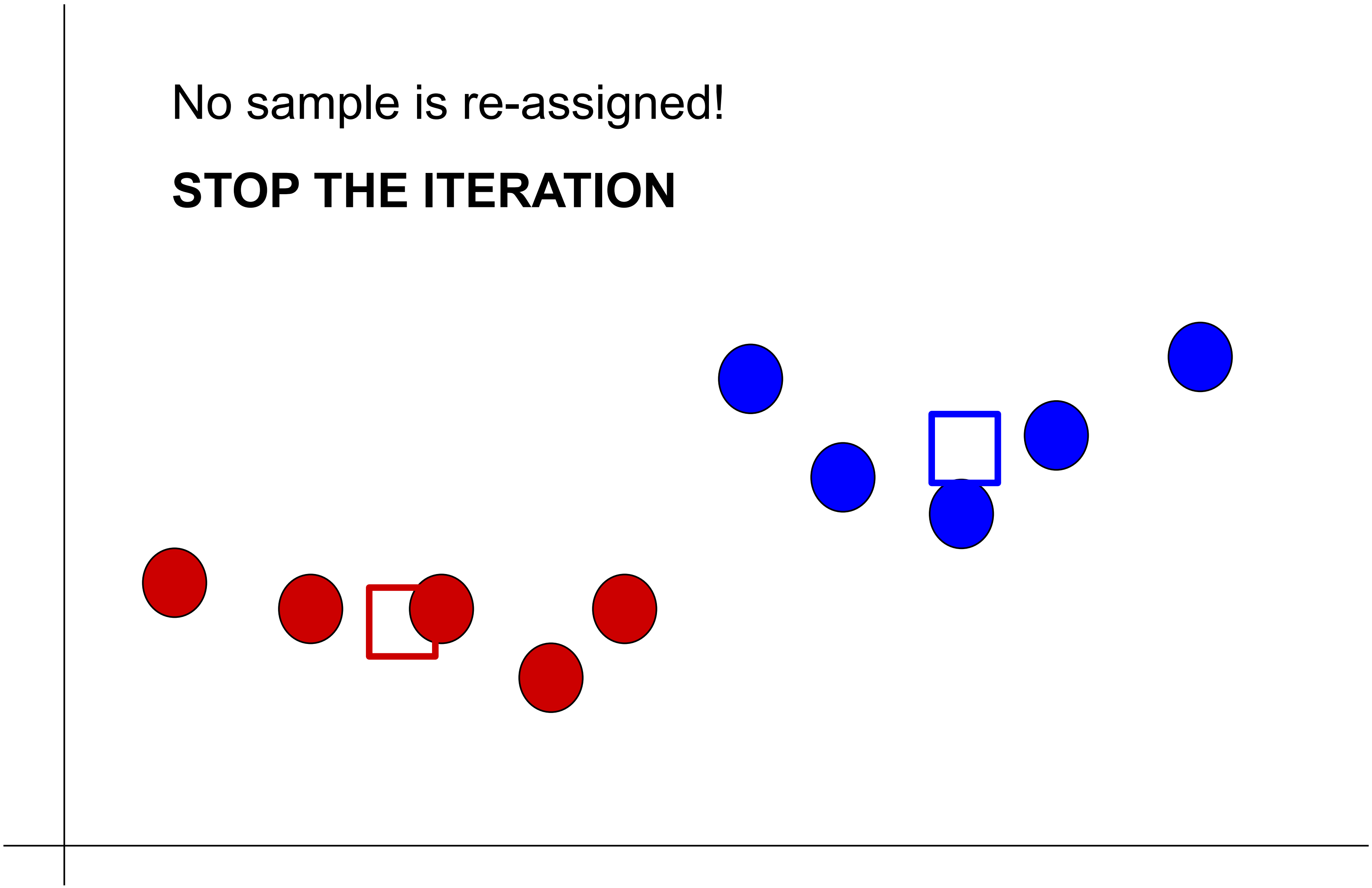
k-means: fourth round



k-means: fourth and last round

No sample is re-assigned!

STOP THE ITERATION



Picking first k points

- The way how we choose the k points can make the k-means algorithm to **converge faster**
  - The assignment loop (internal loop) is linear on the number of points for each cluster… but number of rounds (external loop) is unknown!
  - Usually rounds << m but the worst case reaches 2^m rounds: **exponential time**
- Good way to choose first points is to pick **dispersed** set of points:
  - pick first point at random
  - pick as next point the one with the **largest minimum distance** from the already selected points
  - repeat until we have k points

Largest minimum distance?

- When we have selected only 1 centroid, picking the next is easy, because the above becomes:

$$\mathbf{max}(d(p_1,c_1),\ldots,d(p_m,c_1))$$

- Let's say we already selected *s* centroids *c* and we want to pick up the next one among the *m* points *p*

$$\mathbf{max}(\ \mathbf{min}(d(p_1,c_1),\ldots,d(p_1,c_s))\ ,\ \ldots\ ,\ \mathbf{min}(d(p_m,c_1),\ldots,d(p_m,c_s))\ )$$

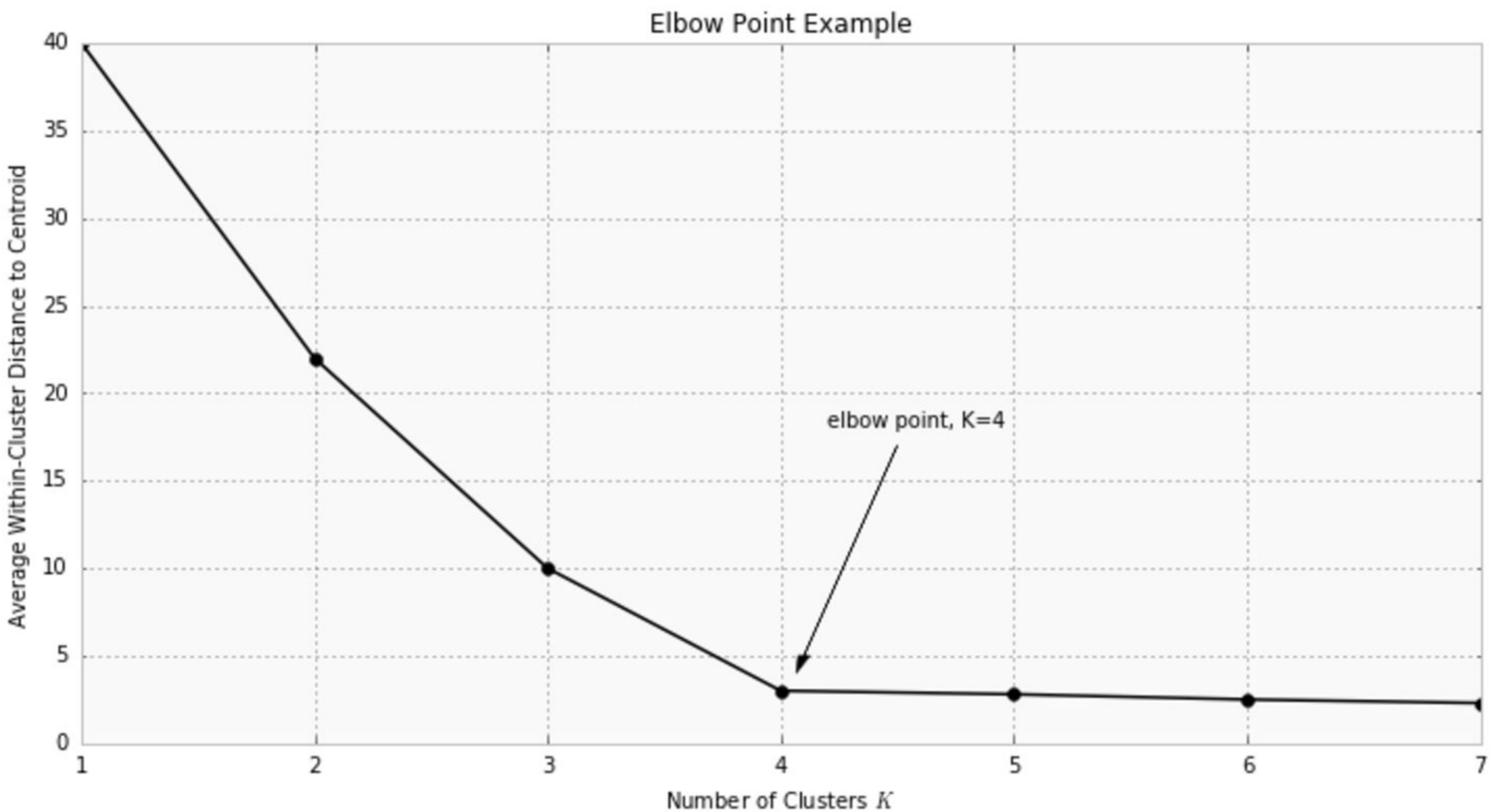
- When we already have 2 or more centroids, for each point we
  - first search the nearest centroid (minimum)
  - then use that value to pick the point farther from any centroid (largest)

Choosing k

- Choosing into how many clusters we should group the documents depends on the task, there's no good automatic way
- However, some measures can help:
  - usually the average distance of all documents from the cluster centroid is used
  - k-means is then ran over different values of k
  - increasing k **will always** decrease the average distance
    - We should stop when the improvement begin to be small: **elbow point**

Elbow method

- The elbow method is a visual method to roughly find the best value for k
- When the line chart looks like an arm, then the "elbow" on the arm is the value of k that is the best



References

[Data Mining](#)

[Concepts and Techniques](#)

Authors: Jiawei Han, Micheline Kamber, Jian Pei