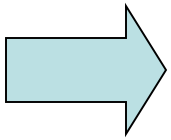
Unsupervised learning, Clustering

CS434

Unsupervised learning and pattern discovery

So far, our data has been in this form:

$x_1^1, x_2^1, x_3^1, \ldots, x_m^1$	y^1
$x_1^2, x_2^2, x_3^2, \ldots, x_m^2$	y^2
…	
…	
$x_1^n, x_2^n, x_3^n, \ldots, x_m^n$	y^n



We will be looking at ***unlabeled data:***

$x_1^1, x_2^1, x_3^1, \ldots, x_m^1$
$x_1^2, x_2^2, x_3^2, \ldots, x_m^2$
…
…
$x_1^n, x_2^n, x_3^n, \ldots, x_m^n$

What do we expect to learn from such data?

I have tons of data (web documents, gene data, etc) and need to:

organize it better – e.g., find subgroups

understand it better – e.g., understand interrelationships

find regular trends in it – e.g., If A and B, then C

The web organized by topic into categories.

Arts
Movies, Music, Television, ...

Business
Companies, Finance, Jobs, ...

Computers
Internet, Hardware, Software, ...

Games
Board, Roleplaying, Video, ...

Health
Alternative, Fitness, Medicine, ...

Home
Consumers, Homeowners, Family, ...

Kids and Teens
Computers, Entertainment, School, ...

News
Media, Newspapers, Current Events, ...

Recreation
Food, Outdoors, Travel, ...

Reference
Education, Libraries, Maps, ...

Regional
Asia, Europe, North America, ...

Science
Biology, Psychology, Physics, ...

Shopping
Autos, Clothing, Gifts, ...

Society
Issues, People, Religion, ...

Sports
Basketball, Football, Soccer, ...

World
Deutsch, Español, Français, Italiano, Japanese, Korean, Nederlands, Polska, Svenska, ...

Hierarchical clustering as a way to organize web pages

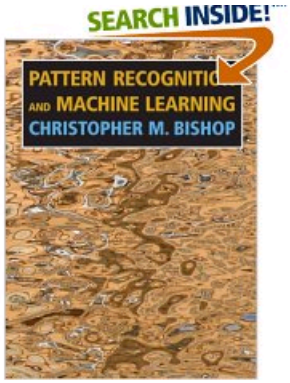
Music

Arts > Music

Categories

Anime (61)
Anti-Music (172)
Arranging (2)
Awards (52)
Bands and Artists (47326)
Business (5322)
Charts (109)
Chats and Forums (242)
Classifieds (37)
Clubs and Venues (435)
Collecting (220)
Comedians (128)
Composition (16836)
Computers (38)
Concerts and Events (507)
Directories (120)
Disabled (17)
DJs (665)
Education (346)
History (107)
Humor (99)
Instruments (11418)
Living History (56)
Lyrics (787)
Marching (1379)
Movies (335)
Museums (46)
Music Videos (129)
Musical Theatre (1831)
Musicology (264)
News and Media (197)
Organizations (127)
People (34)
Personal Pages (194)
Photography (136)
Radio (36)
Record Labels (2900)
Regional (11)
Resources (173)
Reviews (685)
Shopping (2935)
Songwriting (405)
Sound Files (1484)
Styles (31892)
Technology (96)
Television Shows (178)
Theory (174)
Trading (328)
Video Games (153)
Vocal (2952)
Weblogs (105)
Weddings (13)
Women in Music (1716)

Finding association patterns in data …



Share your own customer images
Search inside this book

Pattern Recognition and Machine Learning (Information Science and Statistics) (Hardcover)
by Christopher M. Bishop (Author)
★★★★☆ (22 customer reviews)

List Price: ~~$74.95~~
Price: **$50.11** & this item ships for **FREE with Super Saver Shipping**. Details
You Save: $24.84 (33%)

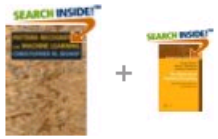
Upgrade this book for $14.99 more, and you can read, search, and annotate every page online. See details
Availability: In Stock. Ships from and sold by **Amazon.com**. Gift-wrap available.

Want it delivered Monday, October 29? Order it in the next 0 hours and 33 minutes, and choose **One-Day Shipping** at checkout. See details

58 used & new available from $43.59

Better Together

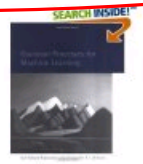
Buy this book with The Elements of Statistical Learning by T. Hastie today!



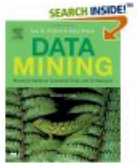
Buy Together Today: $118.84
Buy both now!

Customers Who Bought This Item Also Bought

Pattern Classification (2nd Edition) by



Gaussian Processes for Machine Learning by

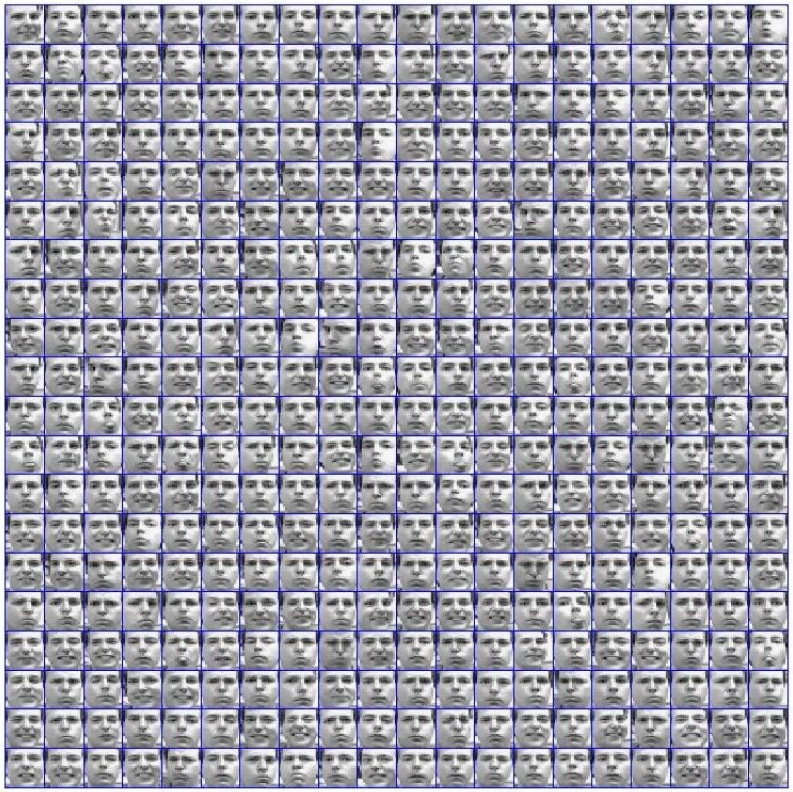


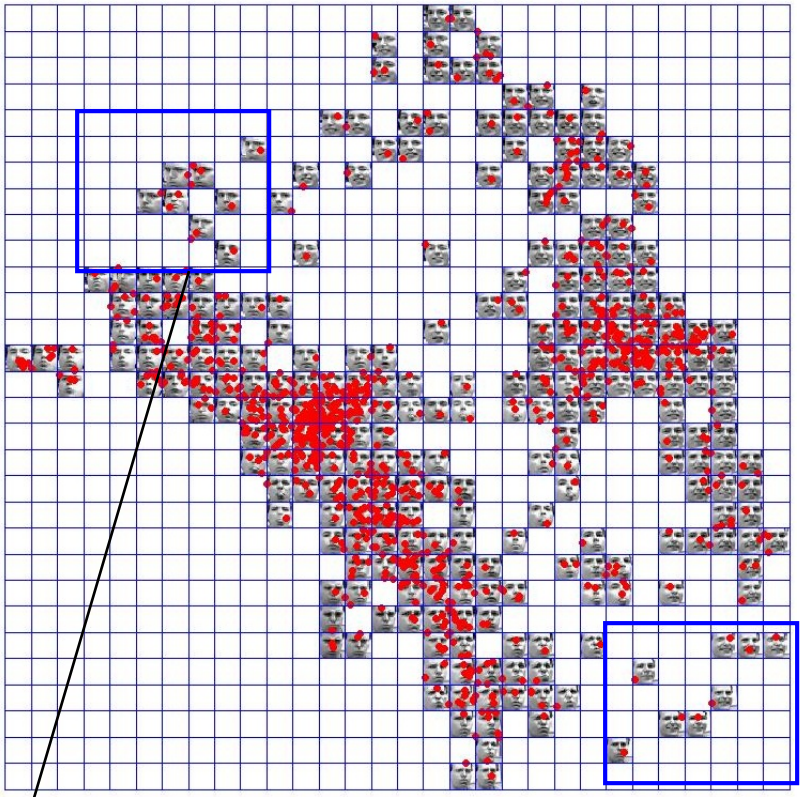
Data Mining by Ian H. Witten

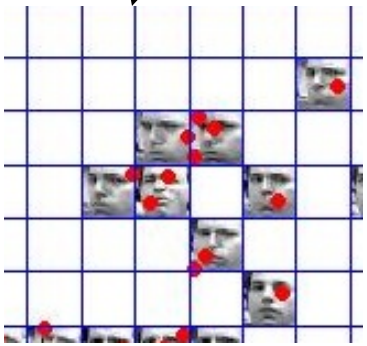
Machine Learning by Tom M. Mitchell

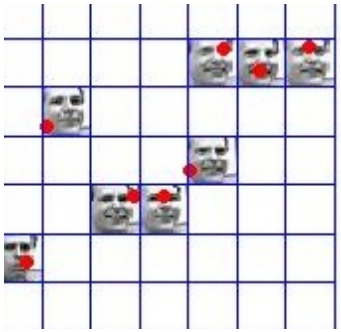
Computer Manual in MATLAB to Accompany

Dimension reduction



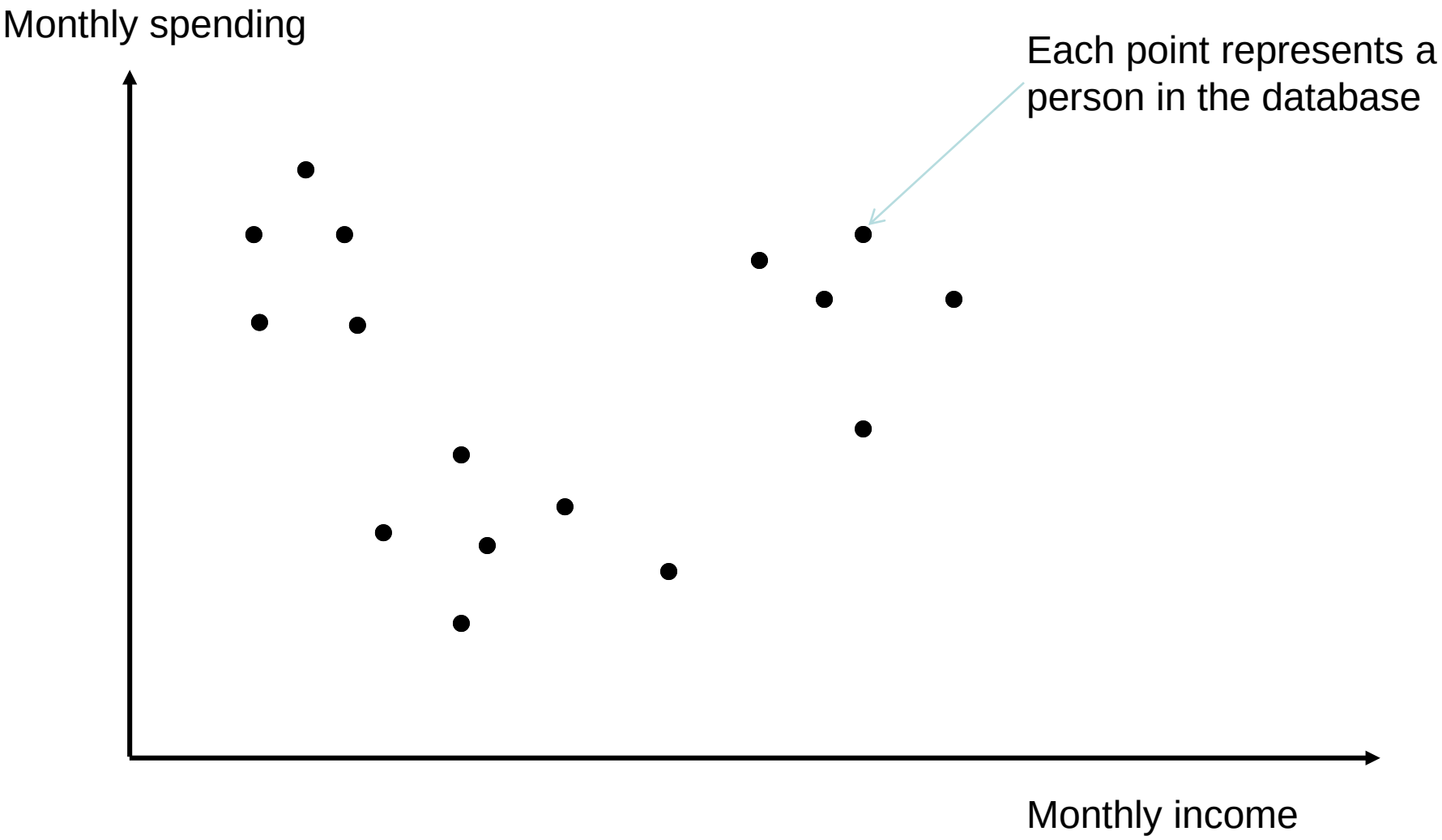




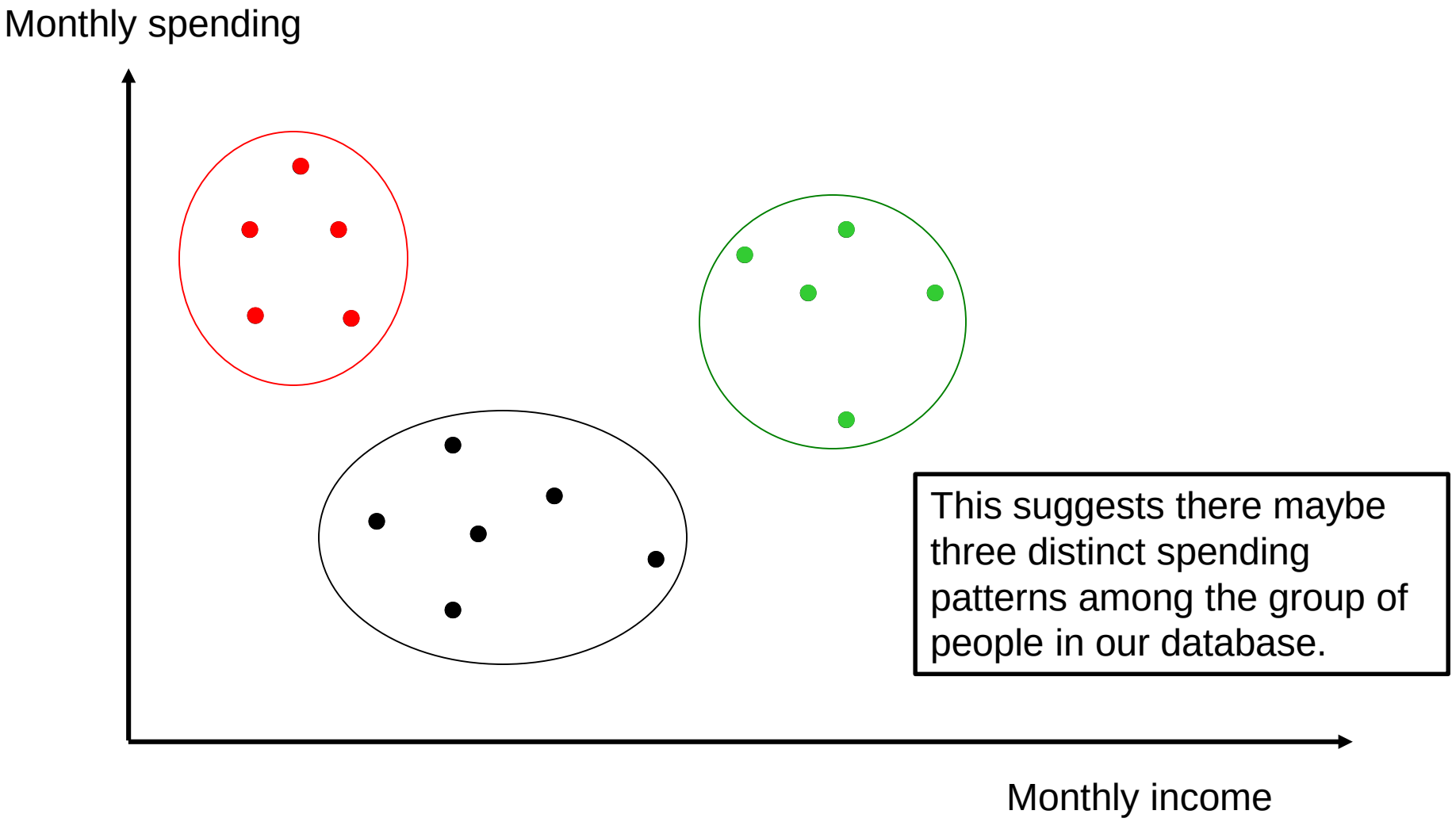


clustering

A hypothetical clustering example



A hypothetical clustering example



What is Clustering

- In general clustering can be viewed as an exploratory procedure for finding interesting subgroups in given data
- A large portion of the efforts focus on a special kind of clustering:
  - Group all given examples (i.e., ***exhaustive***) into disjoint clusters (i.e., ***partitional***), such that:
    - Examples within a cluster are (very) similar
    - Examples in different clusters are (very) different

Some example applications

- Information retrieval: cluster retrieved documents to present more organized and understandable results
- Consumer market analysis: cluster consumers into different interest groups so that marketing plans can be specifically designed for each individual group
- Image segmentation: decompose an image into regions with coherent color and texture
- Vector quantization for data (i.e., image) compression: group vectors into similar groups, and use group mean to represent group members
- Computational biology: group gene into co-expressed families based on their expression profile using different tissue samples and different experimental conditions

Image compression: Vector quantization

Group all pixels into self-similar groups, instead of storing all pixel values, store the means of each group

701,554 bytes → 127,292 bytes

Important components in clustering

- Distance/similarity measure
  - How to measure the similarity between two objects?
- Clustering algorithm
  - How to find the clusters based on the distance/similarity measures
- Evaluation of results
  - How do we know that our clustering result is good

Distance/similarity Measures

- One of the most important question in unsupervised learning, often more important than the choice of clustering algorithms
- What is similarity?
  - Similarity is hard to define, but we know it when we see it

  - More pragmatically, there are many mathematical definitions of distance/similarity

Common distance/similarity measures

- Euclidean distance $L_2(\mathbf{x}, \mathbf{x}')^2 = \sum_{i=1}^{d} (x_i - x_i')^2$

  Straight line distance >=0

- Manhattan distance

$$L_1(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{d} |x_i - x_i'|$$

  City block distance >=0

- Cosine similarity

$$cos(\mathbf{x}, \mathbf{x}') = \frac{<\mathbf{x} \cdot \mathbf{x}'>}{|\mathbf{x}| \cdot |\mathbf{x}'|}$$

  Angle between two vectors, commonly used for measuring document similarity

- More flexible measures: $D(\mathbf{x}, \mathbf{x}') = \sqrt{\sum_{i=1}^{d} w_i (x_i - x_i')^2}$

  Scale each feature differently using w_i's

one can learn the appropriate weights given user guidance

Note: We can always transform between distance and similarity using a monotonically decreasing function, for example $e^{-\alpha D(x_1, x_2)^2}$

How to decide which to use?

- It is application dependent
- Consider your application domain, you need to ask questions such as:
  - What does it mean for two consumers to be similar to each other? Or for two genes to be similar to each other?
- For example, for text domain, we typically use cosine similarity
- This may or may not give you the answer you want depends on your existing knowledge of the domain
  - When domain knowledge is not enough, one could consider learning based approach

A learning approach

- Ideally we'd like to learn a distance function from user's inputs
  - Ask users to provide things like object A is similar to object B, dissimilar to object C
  - For example, if a user want to group the marbles based on their pattern

  - Learn a distance function to correctly reflect these relationships
    - E.g. weigh pattern-related features more than color features
  - This is a more advanced topic that we will not cover in this class, but nonetheless very important

- When we can not afford to learn a distance measure, and don’t have a clue about which distance function is appropriate, what should we do?

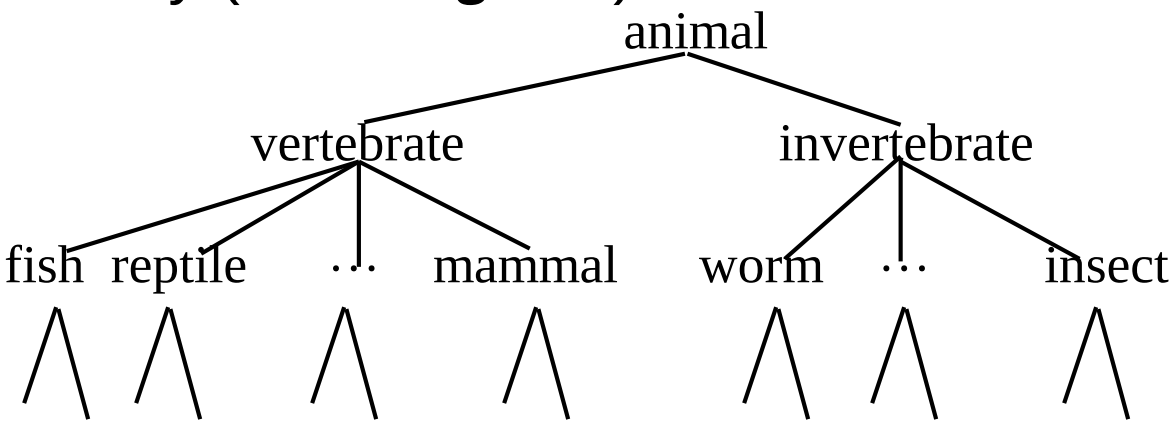
- Clustering is an exploratory procedure and it is important to explore – i.e., try different options and see what makes sense

Clustering

- Now we have briefly discussed distances or similarities among objects, how can we use them to group the objects into clusters?
- Many different approaches are available and they can be roughly categorized into two distinct types

Hierarchical and non-hierarchical Clustering

- **Hierarchical clustering** builds a tree-based hierarchical taxonomy (***dendrogram***)



- **A non-hierarchical (flat) clustering** produces a single partition of the unlabeled data
  - Generates a single partition of the data without resorting to the hierarchical procedure
  - Representative example for flat clustering: K-means clustering

Goal of Clustering

- Given a data set $D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$
- We need to partition D into k disjoint clusters, such that
  - Examples are self-similar in the same cluster
- How do we quantify this?

Objective: Sum of Squared Errors

- Given a partition of the data into k clusters, we can compute the center (i.e., mean, center of mass) of each cluster

$$\mu_i = \frac{1}{n_i} \sum_{\boldsymbol{x} \in C_i} \boldsymbol{x}$$

- For a well formed cluster, its points should be close to its center. We measure this with sum of squared error (SSE), and formulate our objective to find a partition $\mathbb{C}^*$ that minimizes sum of squared error:

$$\mathbb{C}^* = \underset{\mathbb{C}=\{C_1,\ldots,C_k\}}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in C_i} \|\boldsymbol{x} - \boldsymbol{\mu}_i\|^2$$

- This is a combinatorial optimization problem
  - NP-hard
  - K-Means finds a local optimal solution using an iterative approach

Basic idea

- We assume the number of desired clusters, *k*, is given
- Randomly choose *k* examples as *seeds*, one for each cluster.
- Form initial clusters based on these seeds.
- Iterate by repeatedly re-allocating instances to different clusters to improve the overall clustering.
- Stop when clustering converges or after a fixed number of iterations.

The Standard K-means algorithm

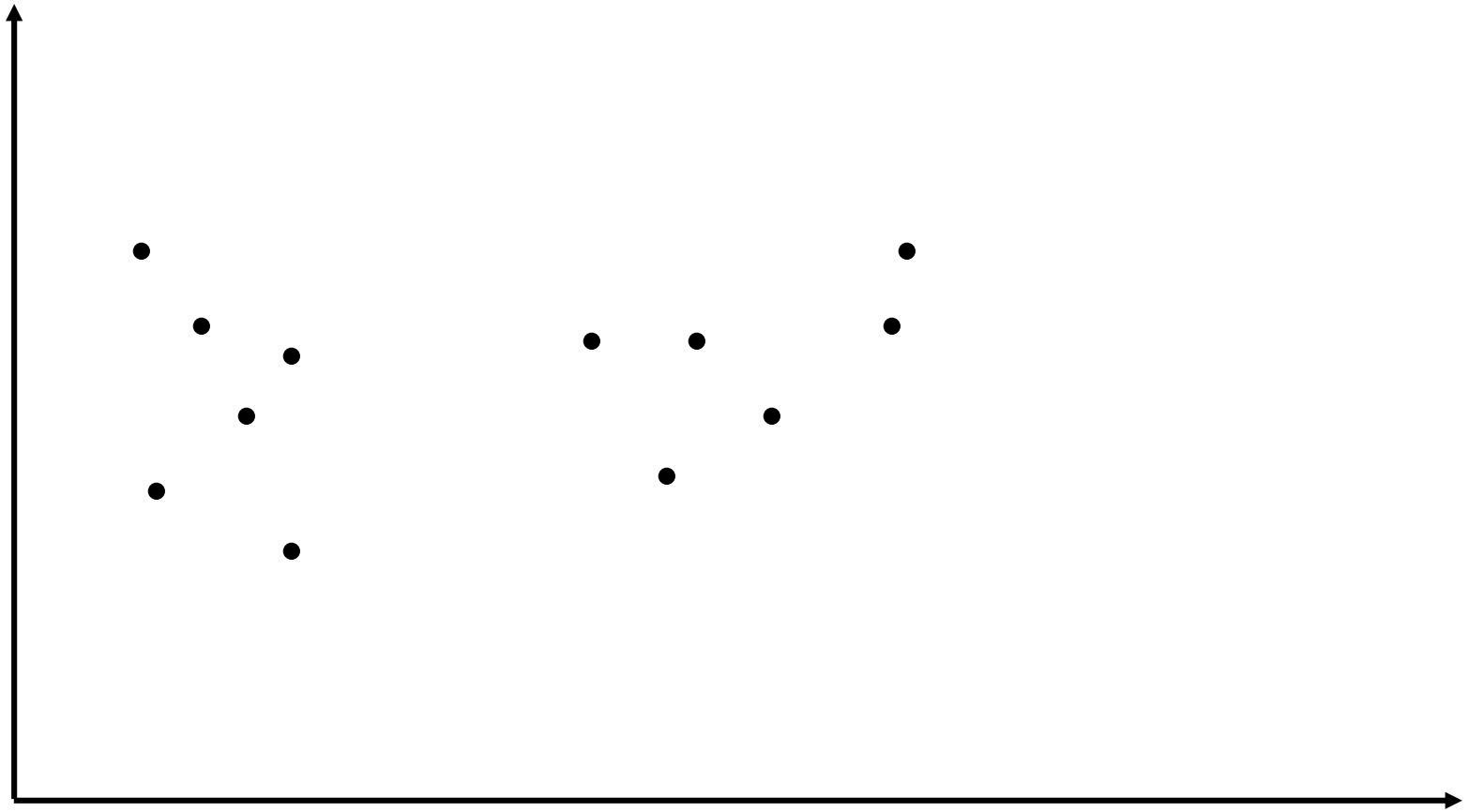
Input: $D = \{x_1\ x_2\ \ldots x_n\}$ and desired number of clusters k

Output: a partition of D into k disjoint clusters $c_1\ \ldots\ c_k$ (s.t. D = $c_1 \cup c_2 \cup \cdots \cup c_k$)

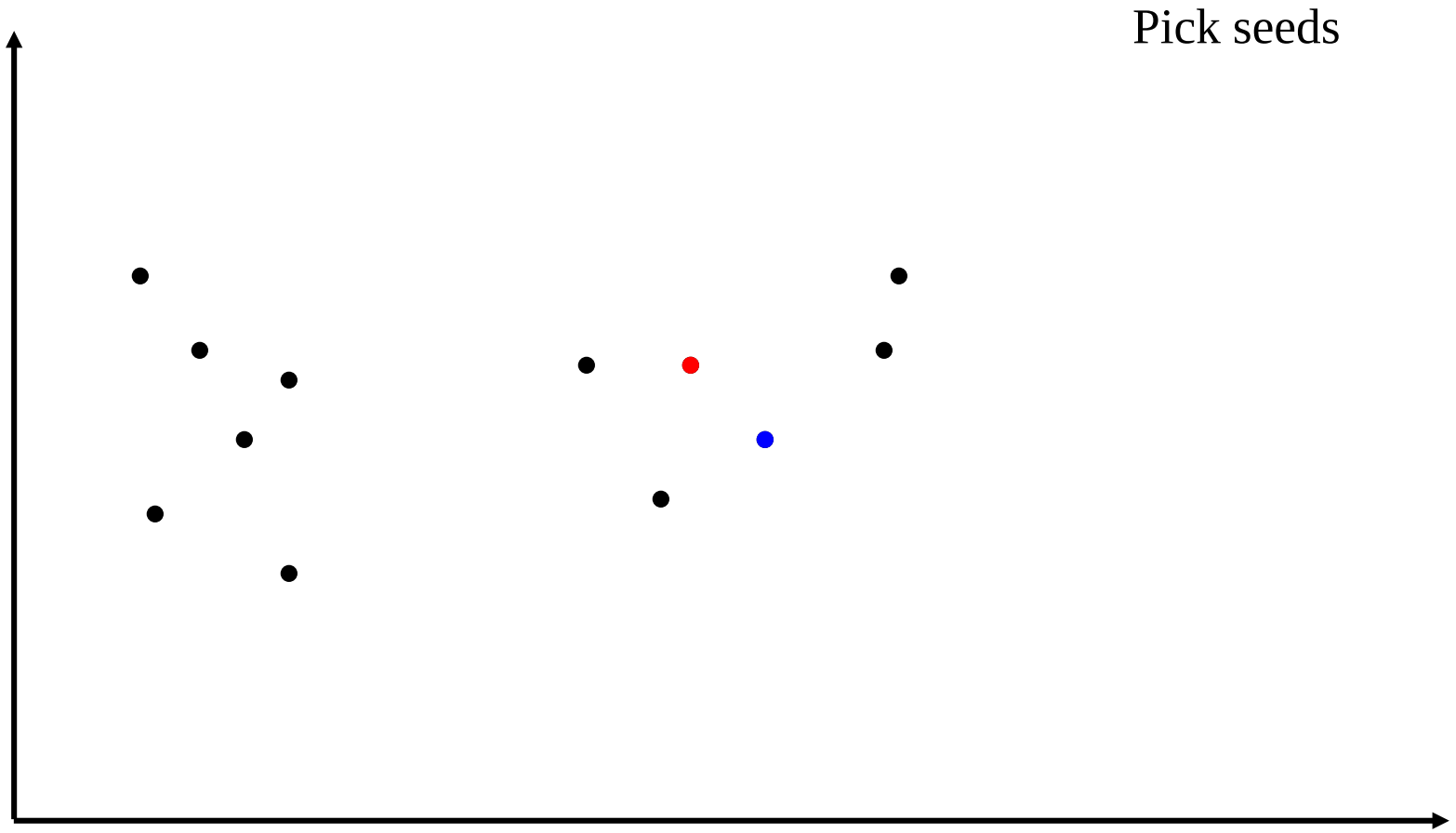
Let $\boldsymbol{d}$ be the distance function between examples

1. Select k random samples from D as centers $\{\mu_1\ \ldots \mu_k\}$ //**Initialization**
2. Do
3. for each example $\boldsymbol{x}_i$,
4. assign $\boldsymbol{x}_i$ to $\boldsymbol{c}_j$ such that $\boldsymbol{d}(\boldsymbol{\mu}_j, \boldsymbol{x}_1)$ is minimized // **the Assignment step**
5. for each cluster j, update cluster center
6. $$\mu_j = \frac{1}{|c_j|}\sum_{\mathbf{x}\in c_j}\mathbf{x}$$ // **the update step**
7. Until convergence

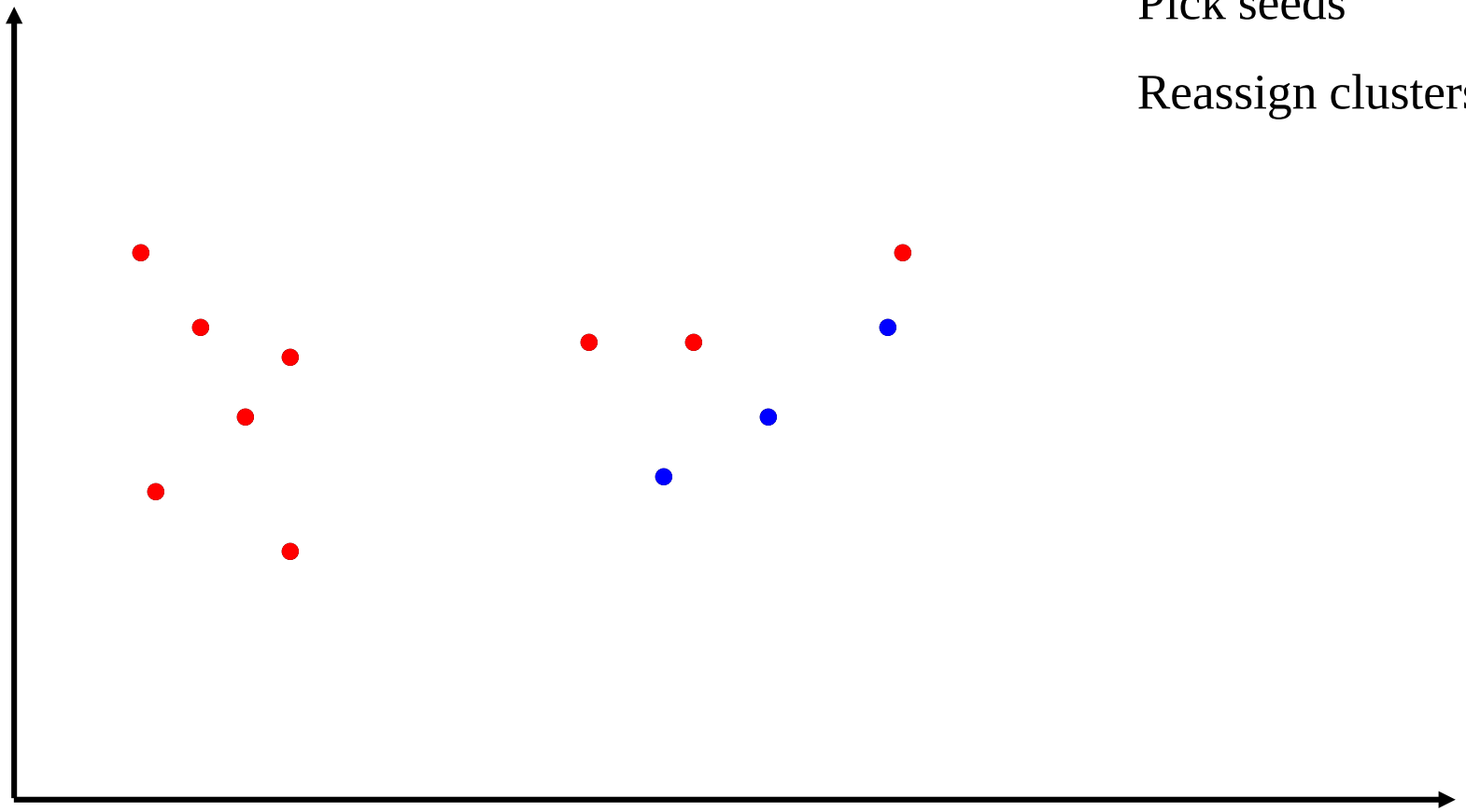
K-Means Example (K=2)



K-Means Example (K=2)



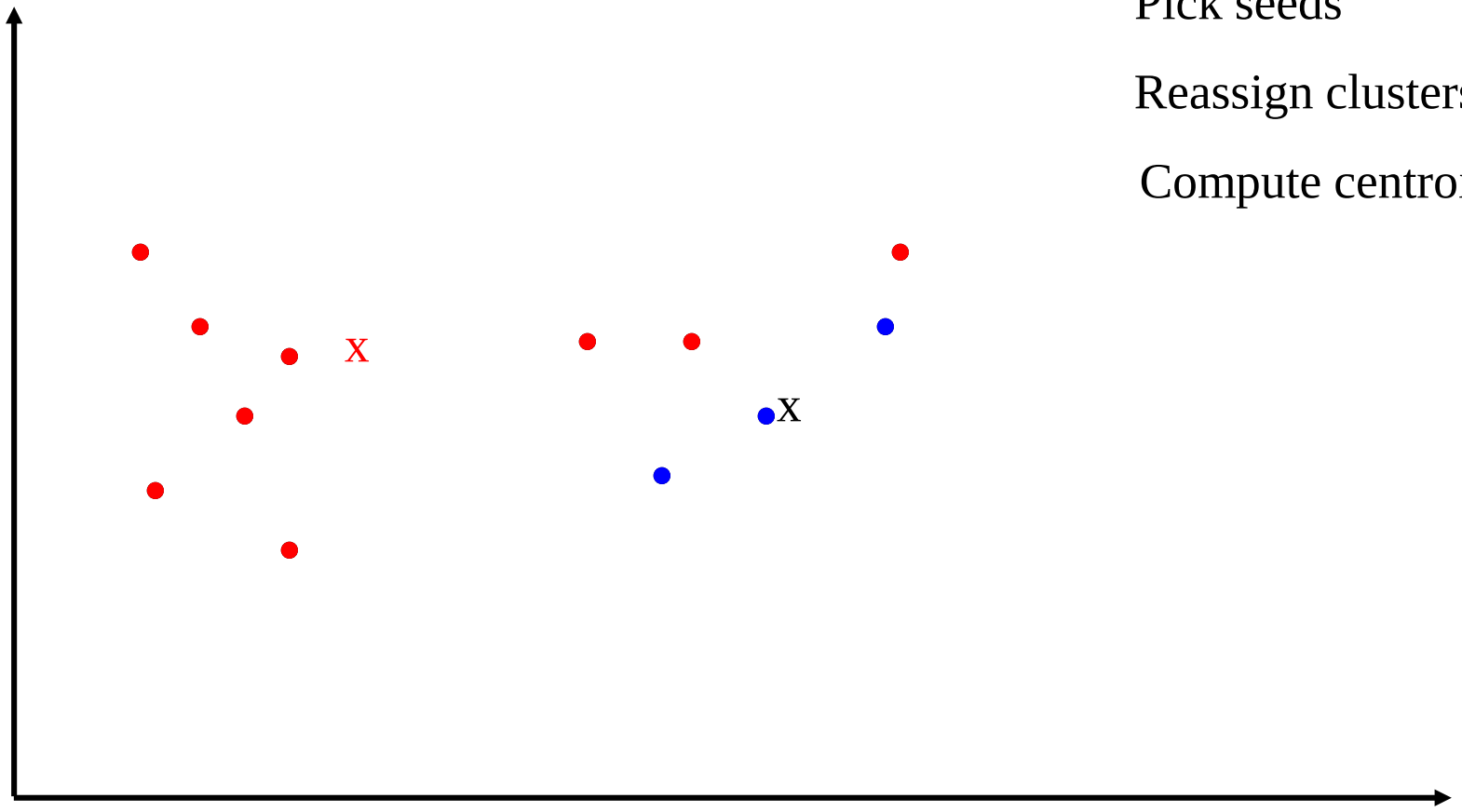
K-Means Example (K=2)



Pick seeds

Reassign clusters

K-Means Example (K=2)

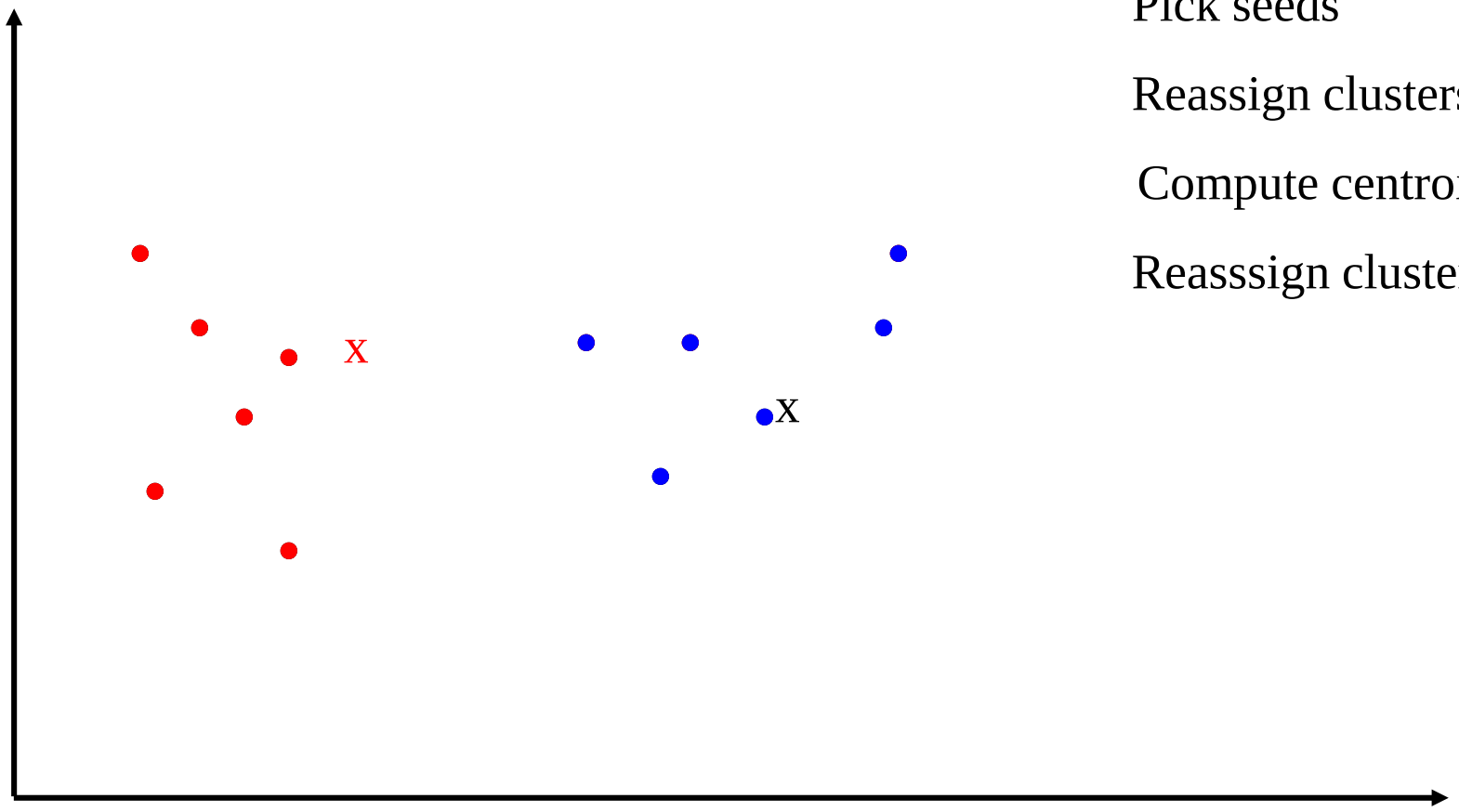


Pick seeds

Reassign clusters

Compute centroids

K-Means Example (K=2)



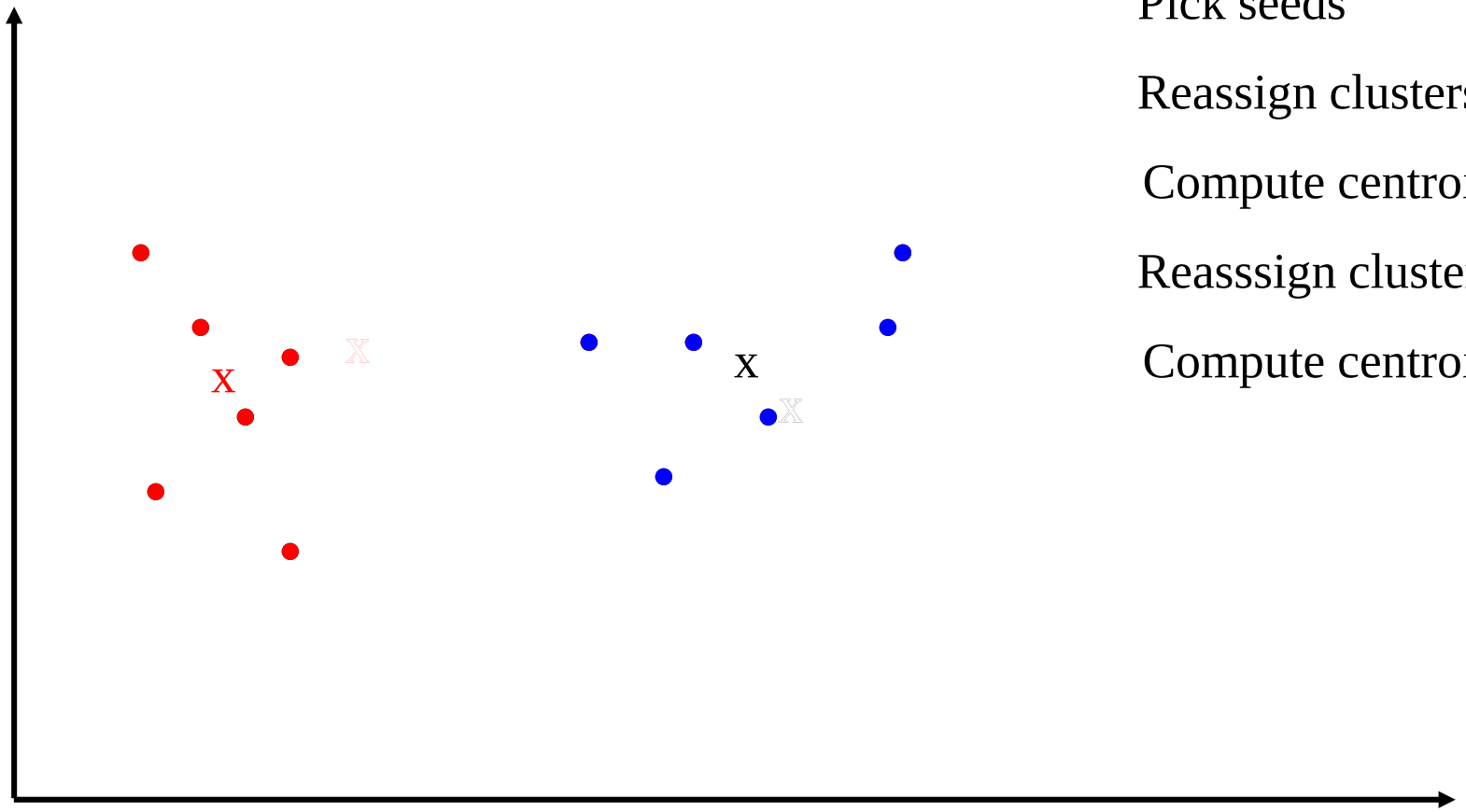
Pick seeds

Reassign clusters

Compute centroids

Reasssign clusters

K-Means Example (K=2)



Pick seeds

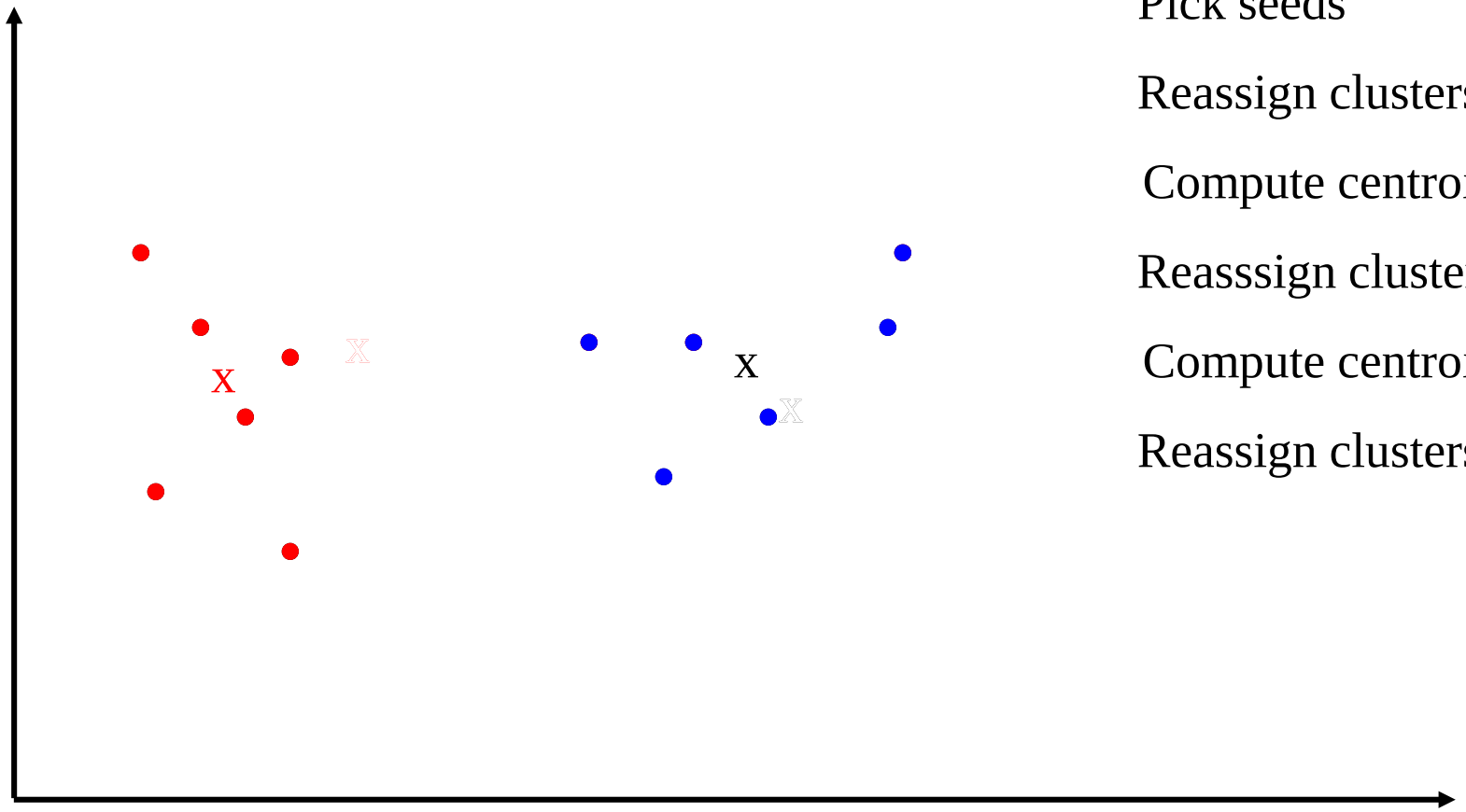
Reassign clusters

Compute centroids

Reasssign clusters

Compute centroids

K-Means Example (K=2)



Pick seeds

Reassign clusters

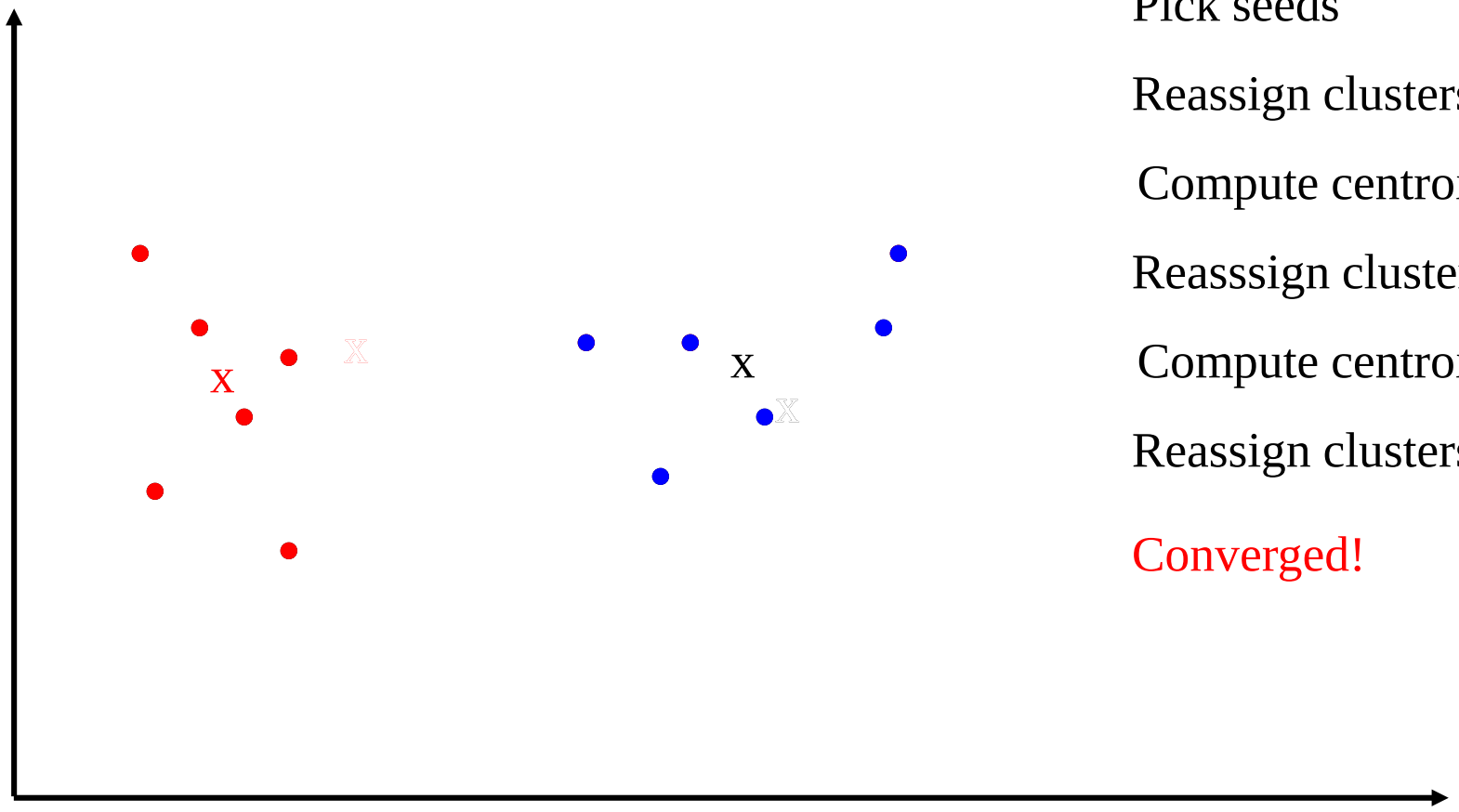
Compute centroids

Reasssign clusters

Compute centroids

Reassign clusters

K-Means Example (K=2)



Pick seeds

Reassign clusters

Compute centroids

Reasssign clusters

Compute centroids

Reassign clusters

Converged!

Monotonicity of K-means

- **Monotonicity Property:** Each iteration of K-means strictly decreases the SSE until convergence
- The following lemma is key to the proof:
  - **Lemma:** Given a finite set C of data points the value of μ that minimizes the SSE:

$$J = \sum_{\mathrm{x} \in C} \|\mathrm{x} - \mu\|^2$$

is:

$$\mu = \frac{1}{|C|} \sum_{\mathbf{x} \in C} \mathbf{x}$$

Proof of monoticity

- Given a current set of clusters with their means, the SSE is given by :

$$J_e = \sum_{i=1}^{k} \sum_{\mathrm{x} \in C_i} \|\mathrm{x} - \mu_i\|^2$$

- Consider the assignment step:
  - Since each point is only reassigned if it is closer to some other cluster than its current cluster, so we know the reassignment step will only decrease SSE
- Consider the re-center step:
  - From our lemma we know the updated μ_i minimizes the SSE of c_i, which implies that the resulting SSE again is decreased.
- Combine the above two, we know K-means always decreases the SSE

K-means properties

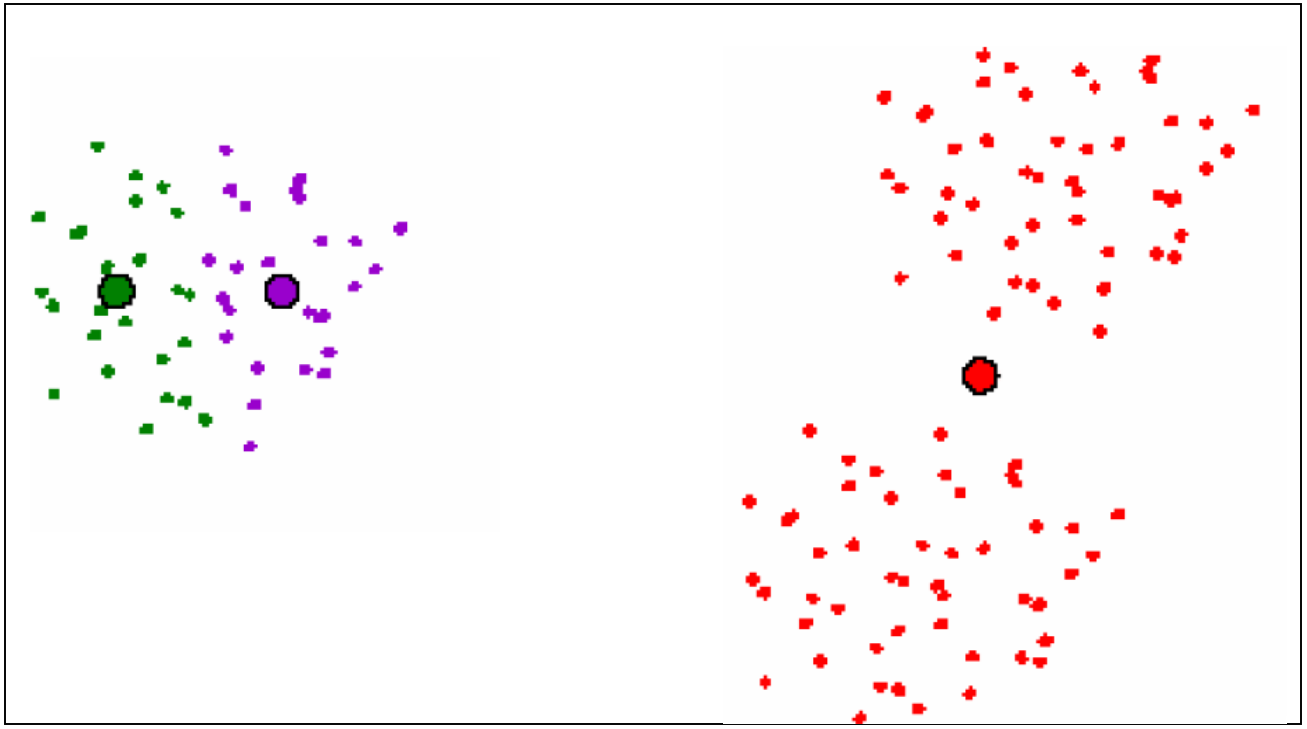
Good news!

- K-means always converges in a finite number of steps
  - Typically converges very fast (in fewer iterations than $\mathrm{n} = |D|$)
- Time complexity:
  - Assume computing distance between two instances is $O(d)$ where d is the dimensionality of the vectors.
  - Reassigning clusters: $O(kn)$ distance computations, or $O(knd)$.
  - Computing centers: Each instance vector gets added once to some center: $O(nd)$.
  - Assume these two steps are each done once for l iterations: $O(lknd)$.
  - Linear in all relevant factors, assuming a fixed number of iterations

More Comments

Bad news!

- Highly sensitive to the initial seeds



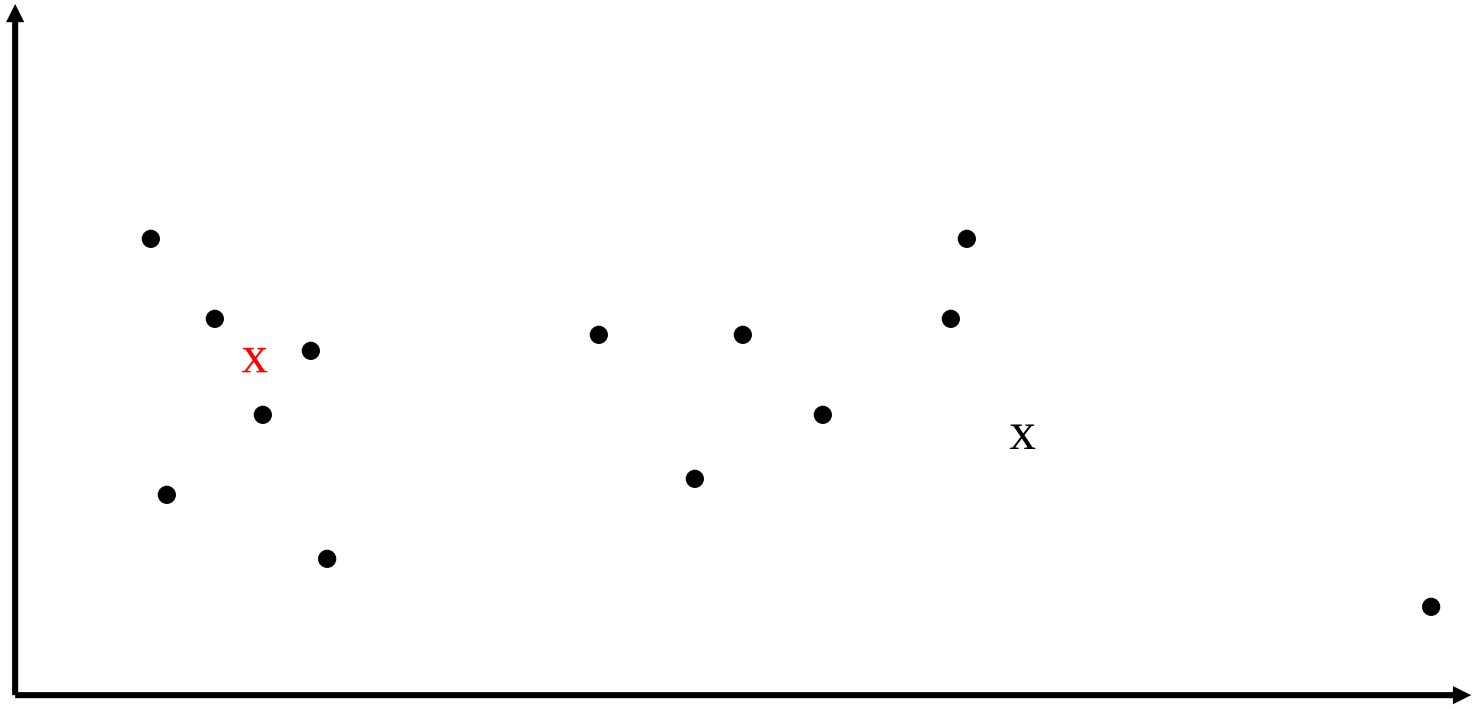
- This is because SSE has many local minimal solutions, i.e., solutions that can not be improved by local reassignments of any particular points

Solutions

- Run multiple trials and choose the one with the best SSE
  - This is typically done in practice
- Heuristics: try to choose initial centers to be far apart
  - Using furthest-first traversal
    - Start with a random initial center
    - Set the second center to be furthest from the first center
    - The third center to be furthest from the first two centers
    - …

Even more comments

- K-Means is exhaustive:
  - Cluster every data point, no notion of outlier
- Outliers may cause problems, why?



K-medoids

- Outliers strongly impact the cluster centers
- K-medoids – use the medoid for each cluster, i.e., the data point that is closest to other points in the cluster

$$\mu = \frac{1}{|C|}\sum_{\mathbf{x}\in C}\mathbf{x} \quad \Rightarrow \quad \mu = \underset{\mathrm{x}\in C}{\operatorname{argmin}} \sum_{\mathrm{z}\in C} \|\mathrm{x}-\mathrm{z}\|^2$$

- K-medoids is computationally more expensive but more robust to outliers

Deciding ***k*** – a model selection problem

- What if we don't know how many clusters there are in the data?
- Can we use SSE to decide k by choosing k that gives the smallest SSE?
  - We will always favor larger k values
- Any quick solutions?
  - heuristic: find the knee