SC4/SM8 Advanced Topics in Statistical Machine Learning

Unsupervised Learning Basics

Dino Sejdinovic
Department of Statistics
Oxford

Slides and other materials available at:
http://www.stats.ox.ac.uk/~sejdinov/atsml/

Course Structure

- MMath Part C & MSc in Applied Statistics

Lectures:

- Tuesdays 2pm, LG.01.
- Thursdays 4pm, LG.01.

MSc:

- 4 problem sheets: classes Mon 11am weeks 3,5,7,8, LG.01.

Part C:

- 4 problem sheets, **solutions due Mon 10am in weeks 3,5,7,8**.
- Class Tutor: Leonard Hasenclever.
- Teaching Assistants: Sam Davenport.
- Check the course website for class times and locations.

Lecture notes and slides are available at the course website:
http://www.stats.ox.ac.uk/~sejdinov/atsml/

Course Aims

1. Have ability to identify and use appropriate methods and models for given data and task.
2. Have ability to use the relevant software packages to analyse data, interpret results, and evaluate methods.
3. Understand the statistical theory framing statistical machine learning.
4. Able to construct appropriate models and derive machine learning algorithms for a given data and task.

What is Machine Learning?

Arthur Samuel, 1959

Field of study that gives computers the ability to **learn** without being explicitly programmed.

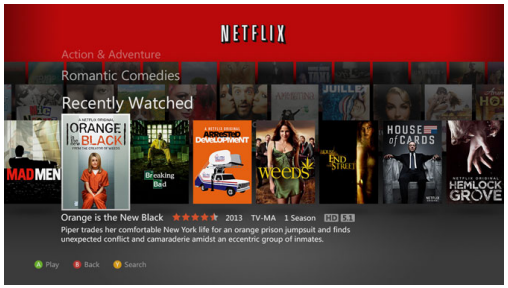
Tom Mitchell, 1997

Any computer program that **improves its performance** at some task **through experience**.

Kevin Murphy, 2012

To develop methods that can **automatically** detect **patterns in data**, and then to use the uncovered patterns to **predict** future data or other outcomes of interest.

What is Machine Learning?



recommender systems

machine translation

self-driving cars

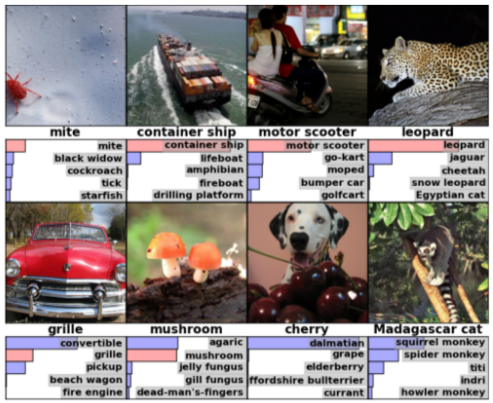
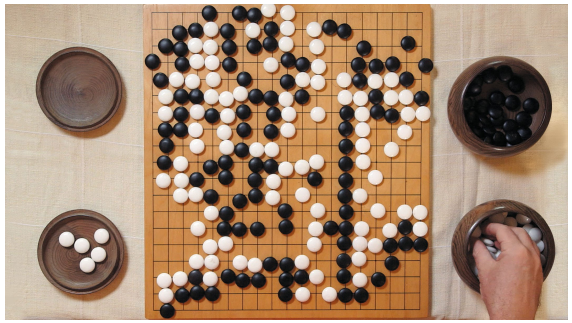


image recognition

DQN Atari games



AlphaGo

Types of Machine Learning

Supervised learning

- Data contains "labels": every example is an input-output pair
- classification, regression
- Goal: **prediction on new examples**

Unsupervised learning

- Extract key features of the "unlabelled" data
- clustering, signal separation, density estimation
- Goal: **representation, hypothesis generation, visualization**

Types of Machine Learning

Semi-supervised Learning

A database of examples, only a small subset of which are labelled.

Multi-task Learning

A database of examples, each of which has multiple labels corresponding to different prediction tasks.

Reinforcement Learning

An agent acting in an environment, given rewards for performing appropriate actions, learns to maximize their reward.

Software

- R
- Python: scikit-learn, mlpy, Theano
- TensorFlow, Torch, Keras, Shogun, Weka.
- Matlab/Octave

Unsupervised Learning

Goals:

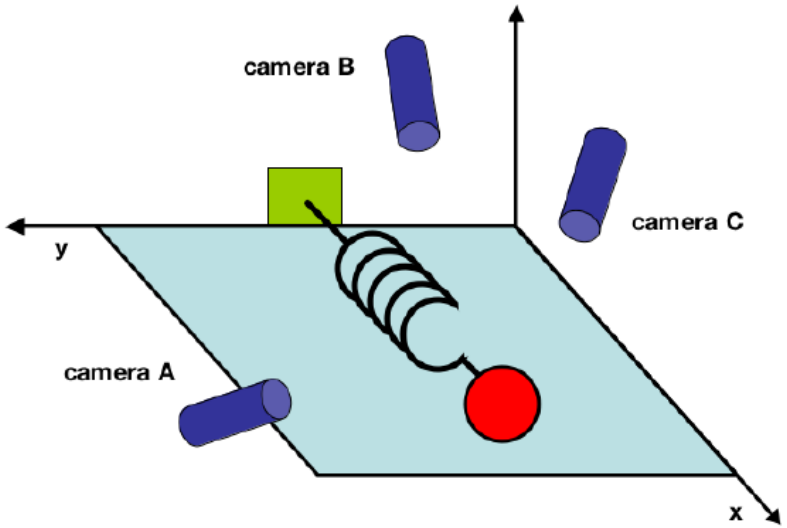
- Find the variables that summarise the data / capture relevant information.
- Discover informative ways to visualise the data.
- Discover the subgroups among the observations.

It is often much easier to obtain unlabeled data than labeled data!

Dimensionality Reduction

Dimensionality reduction

- deceptively many variables to measure, many of them redundant / correlated to each other (large p)
- often, there is a simple but unknown underlying relationship hiding
- example: ball on a frictionless spring recorded by three different cameras
  - our imperfect measurements obfuscate the true underlying dynamics
  - are our coordinates meaningful or do they simply reflect the method of data gathering?



J. Shlens, A Tutorial on Principal Component Analysis, 2005

Principal Components Analysis (PCA)

- PCA considers interesting directions to be those with greatest **variance**.
- A **linear** dimensionality reduction technique: looks for a **new basis** to represent a noisy dataset.
- Workhorse for many different types of data analysis (often used for data preprocessing before supervised techniques are applied).
- Often the first thing to run on high-dimensional data.
- PCA in R: `princomp, prcomp`.

Data Matrix notation

Notation

- Data consists of p variables (features/attributes/dimensions) on n examples (items/observations).
- $\mathbf{X} = (x_{ij})$ is a $n \times p$-matrix with $x_{ij} :=$ the j-th variable for the i-th example

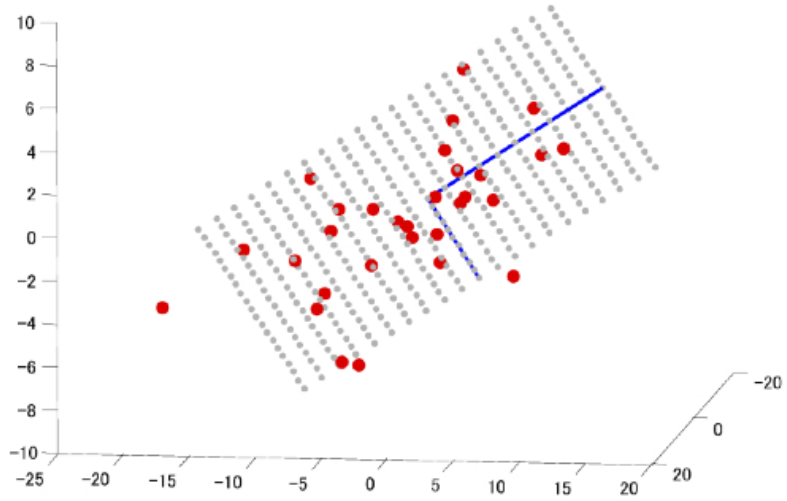
$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1j} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2j} & \dots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i1} & x_{i2} & \dots & x_{ij} & \dots & x_{ip} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nj} & \dots & x_{np} \end{bmatrix}.$$

- Denote the i-th data item by $x_i \in \mathbb{R}^p$ (we will treat it as a column vector: it is the transpose of the i-th row of $\mathbf{X}$).
- Assume $x_1, \dots, x_n$ are **independently and identically distributed** samples of a **random vector** X over $\mathbb{R}^p$. The j-th dimension of X will be denoted $X^{(j)}$.

Principal Components Analysis (PCA)

- Assume that the dataset is centred, i.e., $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i = 0$.
- Sample covariance:

$$S = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(x_i - \bar{x})^\top$$
$$= \frac{1}{n-1}\sum_{i=1}^{n} x_i x_i^\top = \frac{1}{n-1}\mathbf{X}^\top\mathbf{X}.$$



PCA

PCA recovers an orthonormal basis $v_1, v_2, \ldots, v_p$ in $\mathbb{R}^p$ – vectors v_i are **called principal components (PC) or loading vectors** – such that:

- The first principal component (PC) v_1 is the **direction of greatest variance** of data.
- The j-th PC v_j is the **direction orthogonal to $v_1, v_2, \ldots, v_{j-1}$ of greatest variance**, for $j = 2, \ldots, p$.

Principal Components Analysis (PCA)

- The k-dimensional representation of data item x_i is the vector of projections of x_i onto first k PCs:

$$z_i = V_{1:k}^\top x_i = \left[v_1^\top x_i, \ldots, v_k^\top x_i\right]^\top \in \mathbb{R}^k,$$

  where $V_{1:k} = [v_1, \ldots, v_k]$.
- Transformed data matrix, also called the **scores matrix**

$$\mathbf{Z} = \mathbf{X} V_{1:k} \in \mathbb{R}^{n \times k}.$$

- Reconstruction of x_i:

$$\hat{x}_i = V_{1:k} V_{1:k}^\top x_i.$$

- PCA gives the **optimal linear reconstruction** of the original data based on a k-dimensional compression (problem sheets).

Deriving the First Principal Component

- Our data set is an i.i.d. sample $\{x_i\}_{i=1}^n$ of a random vector $X = \left[X^{(1)} \dots X^{(p)}\right]^\top$.
- For the 1^{st} PC, we seek a derived scalar variable of the form

$$Z^{(1)} = v_1^\top X = v_{11}X^{(1)} + v_{12}X^{(2)} + \cdots + v_{1p}X^{(p)}$$

where $v_1 = [v_{11}, \dots, v_{1p}]^\top \in \mathbb{R}^p$ are chosen to maximise the sample variance

$$\widehat{\mathsf{Var}}(Z^{(1)}) = v_1^\top \widehat{\mathsf{Cov}}(X) v_1 = v_1^\top S v_1.$$

- Optimisation problem

$$\max_{v_1} \; v_1^\top S v_1$$
$$\text{subject to: } v_1^\top v_1 = 1.$$

Deriving the First Principal Component

- Lagrangian of the problem is given by:

$$\mathcal{L}(v_1, \lambda_1) = v_1^\top S v_1 - \lambda_1 \left(v_1^\top v_1 - 1\right).$$

- The corresponding vector of partial derivatives is

$$\frac{\partial \mathcal{L}(v_1, \lambda_1)}{\partial v_1} = 2Sv_1 - 2\lambda_1 v_1.$$

- Setting this to zero reveals the eigenvector equation $Sv_1 = \lambda_1 v_1$, i.e. v_1 must be an eigenvector of S and the dual variable λ_1 is the corresponding eigenvalue.
- Since $v_1^\top S v_1 = \lambda_1 v_1^\top v_1 = \lambda_1$, the first PC must be the eigenvector associated with the largest eigenvalue of S.

PCA as eigendecomposition of the covariance matrix

- The 2^{nd} PC is chosen to be orthogonal with the 1^{st} and is computed in a similar way (see notes). It will have the largest variance in the remaining $p-1$ dimensions, etc.
- The eigenvalue decomposition of S is given by

$$S = V\Lambda V^\top$$

where Λ is a diagonal matrix with eigenvalues

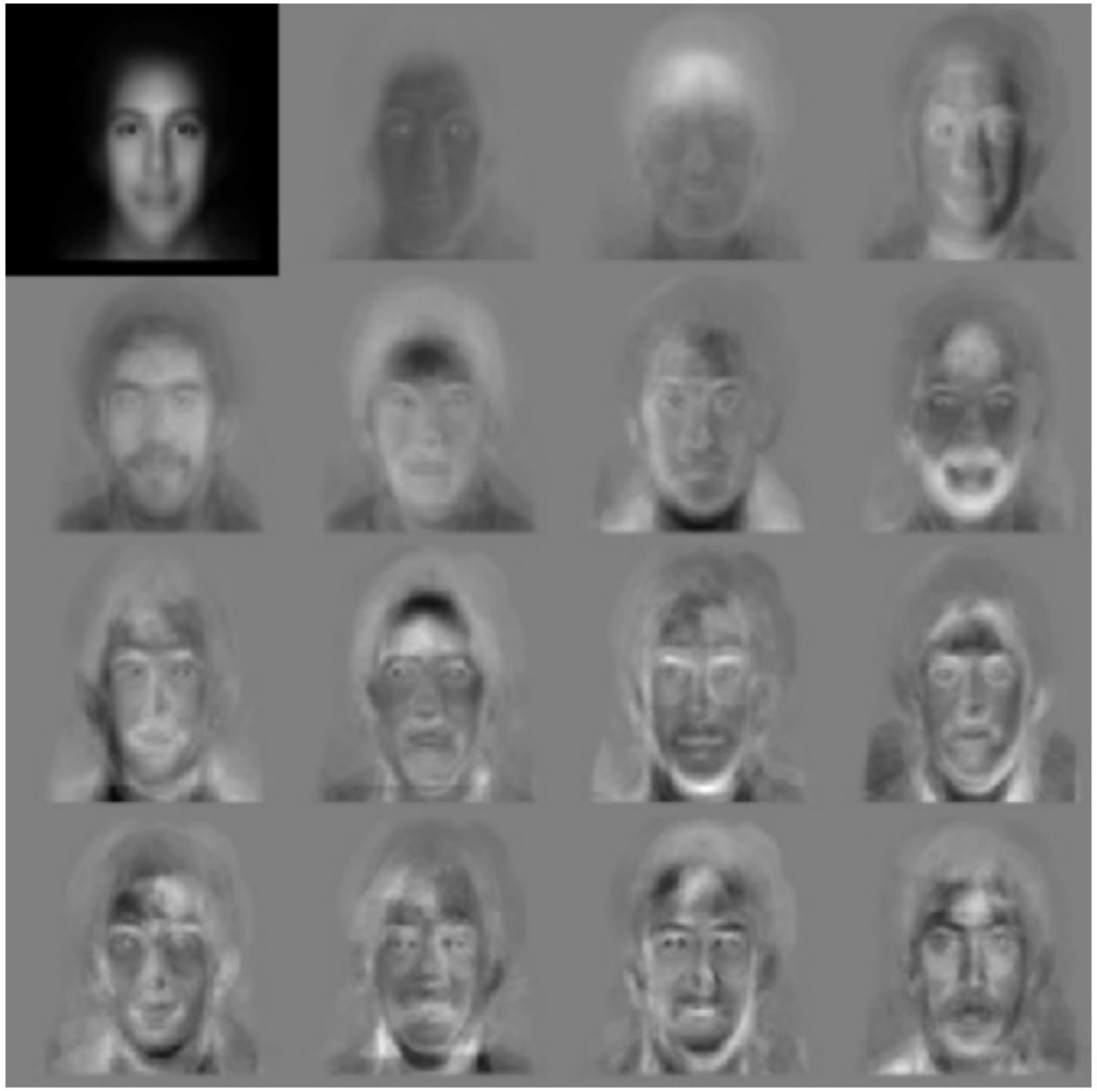
$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0$$

and V is a $p \times p$ orthogonal matrix whose columns are the p eigenvectors of S, i.e. the principal components $v_1, \ldots, v_p$.

Properties of the Principal Components

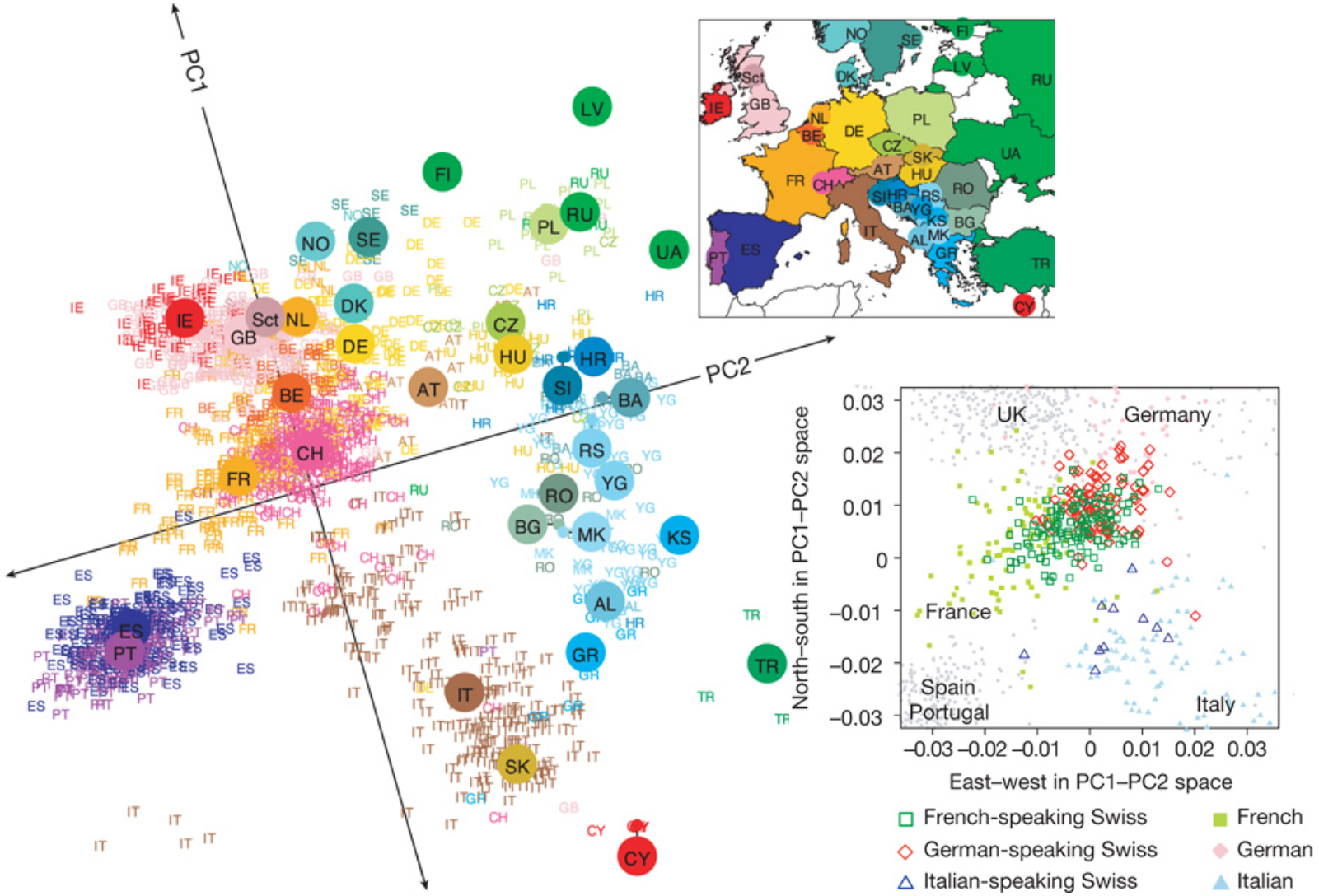
- Derived scalar variable (projection to the j-th principal component) $Z^{(j)} = v_j^\top X$ has sample variance λ_j, for $j = 1, \ldots, p$
- S is a real symmetric matrix, so eigenvectors (principal components) are orthogonal.
- Projections to principal components are **uncorrelated**: $\widehat{\mathrm{Cov}}(Z^{(i)}, Z^{(j)}) \approx v_i^\top S v_j = \lambda_j v_i^\top v_j = 0$, for $i \neq j$.
- The **total sample variance** is given by $\mathrm{Tr}(S) = \sum_{i=1}^p S_{ii} = \lambda_1 + \ldots + \lambda_p$, so the **proportion of total variance explained** by the j^{th} PC is $\frac{\lambda_j}{\lambda_1 + \lambda_2 + \ldots + \lambda_p}$

PCA on Face Images: Eigenfaces



Turk and Pentland, CVPR 1995

PCA on European Genetic Variation



Genes mirror geography within Europe, Nature 2008

PCA: summary

PCA

Find an orthogonal basis $\{v_1, v_2, \ldots, v_p\}$ for the data space such that:

- The first principal component (PC) v_1 is the **direction of greatest variance** of data.
- The j-th PC v_j is the **direction orthogonal to $v_1, v_2, \ldots, v_{j-1}$ of greatest variance**, for $j = 2, \ldots, p$.

- Eigendecomposition of the sample covariance matrix $S = \frac{1}{n-1} \sum_{i=1}^{n} x_i x_i^\top$.

$$S = V \Lambda V^\top.$$

  - Λ is a diagonal matrix with eigenvalues (variances along each principal component) $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0$
  - V is a $p \times p$ orthogonal matrix whose columns are the p eigenvectors of S, i.e. the principal components $v_1, \ldots, v_p$
- Dimensionality reduction by projecting $x_i \in \mathbb{R}^p$ onto first k principal components:

$$z_i = \left[v_1^\top x_i, \ldots, v_k^\top x_i\right]^\top \in \mathbb{R}^k.$$

Comments on the use of PCA

- PCA commonly used to project data X onto the first k PCs giving the k-dimensional view of the data that best preserves **the first two moments**.
- Although PCs are uncorrelated, scatterplots sometimes reveal structures in the data other than linear correlation.
- Emphasis on variance is where the weaknesses of PCA stem from:
  - Assuming large variances are meaningful (high signal-to-noise ratio)
  - The PCs depend heavily on the units measurement. Where the data matrix contains measurements of vastly differing orders of magnitude, the PC will be greatly biased in the direction of larger measurement. In these cases, it is recommended to calculate PCs from $\mathrm{Corr}(X)$ instead of $\mathrm{Cov}(X)$ (`cor=True` in the call of `princomp`).
  - Lack of robustness to outliers: variance is affected by outliers and so are PCs.

Eigendecomposition and PCA

$$S = \frac{1}{n-1}\sum_{i=1}^{n} x_i x_i^\top = \frac{1}{n-1}\mathbf{X}^\top\mathbf{X}.$$

- S is a **real and symmetric** matrix, so there exist p eigenvectors $v_1, \ldots, v_p$ that are pairwise orthogonal and p associated eigenvalues $\lambda_1, \ldots, \lambda_p$ which satisfy the eigenvalue equation $Sv_i = \lambda_i v_i$. In particular, V is an orthogonal matrix:
$$VV^\top = V^\top V = I_p.$$

- S is a **positive-semidefinite** matrix, so the eigenvalues are non-negative:
$$\lambda_i \geq 0, \ \forall i.$$

Why is S symmetric? Why is S positive-semidefinite?
Reminder: A symmetric $p \times p$ matrix R is said to be positive-semidefinite if
$$\forall a \in \mathbb{R}^p, a^\top R a \geq 0.$$

Singular Value Decomposition (SVD)

SVD

Any real-valued $n \times p$ matrix $\mathbf{X}$ can be written as $\mathbf{X} = UDV^\top$ where

- U is an $n \times n$ orthogonal matrix: $UU^\top = U^\top U = I_n$
- D is a $n \times p$ matrix with decreasing **non-negative** elements on the diagonal (the singular values) and zero off-diagonal elements.
- V is a $p \times p$ orthogonal matrix: $VV^\top = V^\top V = I_p$

- SVD **always** exists, even for non-square matrices.
- Fast and numerically stable algorithms for SVD are available in most packages. The relevant R command is `svd`.

SVD and PCA

- Let $\mathbf{X} = UDV^\top$ be the SVD of the $n \times p$ data matrix $\mathbf{X}$.
- Note that

$$(n-1)S = \mathbf{X}^\top\mathbf{X} = (UDV^\top)^\top(UDV^\top) = VD^\top U^\top UDV^\top = VD^\top DV^\top,$$

  using orthogonality ($U^\top U = I_n$) of U.
- The eigenvalues of S are thus the diagonal entries of $\Lambda = \frac{1}{n-1}D^\top D$.
- We also have

$$\mathbf{X}\mathbf{X}^\top = (UDV^\top)(UDV^\top)^\top = UDV^\top VD^\top U^\top = UDD^\top U^\top,$$

  using orthogonality ($V^\top V = I_p$) of V.

Gram matrix

$\mathbf{K} = \mathbf{X}\mathbf{X}^\top$, $\mathbf{K}_{ij} = x_i^\top x_j$ is called the Gram matrix of dataset $\mathbf{X}$.
$\mathbf{K}$ and $(n-1)S = \mathbf{X}^\top\mathbf{X}$ have the same nonzero eigenvalues, equal to the non-zero squared singular values of $\mathbf{X}$.

PCA projections from Gram matrix

If we consider projections to **all principal components**, the transformed data matrix is

$$\mathbf{Z} = \mathbf{X}V = UDV^\top V = UD, \tag{1}$$

If $p \leq n$ this means

$$z_i = [U_{i1}D_{11}, \ldots, U_{ip}D_{pp}]^\top, \tag{2}$$

and if $p > n$ only the first n projections are defined (sample covariance will have rank at most n):

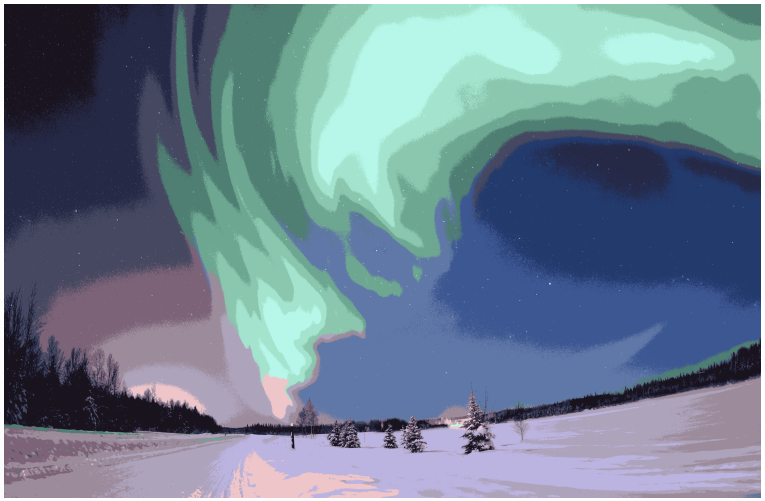
$$z_i = [U_{i1}D_{11}, \ldots, U_{in}D_{nn}, 0, \ldots, 0]^\top. \tag{3}$$

Thus, $\mathbf{Z}$ can be obtained from the eigendecomposition of Gram matrix $\mathbf{K}$. When $p \gg n$, eigendecomposition of $\mathbf{K}$ requires much less computation, $O(n^3)$, than the eigendecomposition of the covariance matrix, $O(p^3)$, so is the preferred method for PCA in that case.

Clustering

Clustering

- Many datasets consist of multiple heterogeneous subsets.
- **Cluster analysis**: Given an unlabelled data, want algorithms that automatically group the datapoints into coherent subsets/clusters. Examples:
  - market segmentation of shoppers based on browsing and purchase histories
  - different types of breast cancer based on the gene expression measurements
  - discovering communities in social networks
  - image segmentation



Types of Clustering

- **Model-free** clustering:
  - Defined by **similarity/dissimilarity** among instances within clusters.
- **Model-based** clustering:
  - Each cluster is described using a probability model.

Model-free clustering

- notion of similarity/dissimilarity between data items is central: many ways to define and the choice will depend on the dataset being analyzed and dictated by domain specific knowledge
- most common approach is **partition-based** clustering: one divides n data items into K clusters $C_1, \ldots, C_K$ where for all $k, k' \in \{1, \ldots, K\}$,

$$C_k \subset \{1, \ldots, n\}, \qquad C_k \cap C_{k'} = \emptyset \ \ \forall k \neq k', \qquad \bigcup_{k=1}^{K} C_k = \{1, \ldots, n\}.$$

- Intuitively, clustering aims to group similar items together and to place separate dissimilar items into different groups
- two objectives can contradict each other (similarity is not a transitive relation, while being in the same cluster is an equivalence relation)

Axiomatic approach

Clustering method is a map $\mathcal{F} : (\mathcal{D} = \{x_i\}_{i=1}^n, \rho) \mapsto \{C_1, \ldots, C_K\}$ which takes as an input dataset $\mathcal{D}$ and a dissimilarity function ρ and returns a partition of $\mathcal{D}$. Three basic properties required

- **Scale invariance.** For any $\alpha > 0$, $\mathcal{F}(\mathcal{D}, \alpha\rho) = \mathcal{F}(\mathcal{D}, \rho)$.
- **Richness.** For any partition $C = \{C_1, \ldots, C_K\}$ of $\mathcal{D}$, there exists dissimilarity ρ, such that $\mathcal{F}(\mathcal{D}, \rho) = C$.
- **Consistency.** If ρ and ρ' are two dissimilarities such that for all $x_i, x_j \in \mathcal{D}$ the following holds:

$$
\begin{aligned}
x_i, x_j \text{ belong to the same cluster in } \mathcal{F}(\mathcal{D}, \rho) &\implies \rho'(x_i, x_j) \leq \rho(x_i, x_j) \\
x_i, x_j \text{ belong to different clusters in } \mathcal{F}(\mathcal{D}, \rho) &\implies \rho'(x_i, x_j) \geq \rho(x_i, x_j),
\end{aligned}
$$

  then $\mathcal{F}(\mathcal{D}, \rho') = \mathcal{F}(\mathcal{D}, \rho)$.

Kleinberg (2003) proves that there exists no clustering method that satisfies all three properties!

Examples of Model-free Clustering

- **K-means clustering**: a partition-based method into K clusters. Finds groups such that variation within each group is small. The number of clusters K is usually fixed beforehand or various values of K are investigated as a part of the analysis.
- **Spectral clustering**: Similarity/dissimilarity between data items defines a graph. Find a partition of vertices which does not “cut" many edges. Can be interpreted as nonlinear dimensionality reduction followed by K-means.
- **Hierarchical clustering**: nearby data items are joined into clusters, then clusters into super-clusters forming a hierarchy. Typically, the hierarchy forms a binary tree (a **dendrogram**) where each cluster has two “children” clusters. Dendrogram allows to view the clusterings for each possible number of clusters, from 1 to n (number of data items).

K-means

Goal: divide data items into a **pre-assigned number K of clusters** $C_1, \ldots, C_K$ where for all $k, k' \in \{1, \ldots, K\}$,

$$C_k \subset \{1, \ldots, n\}, \qquad C_k \cap C_{k'} = \emptyset \ \ \forall k \neq k', \qquad \bigcup_{k=1}^{K} C_k = \{1, \ldots, n\}.$$

Each cluster is represented using a **prototype** or **cluster centroid** μ_k.
We can measure the quality of a cluster with its **within-cluster deviance**

$$W(C_k, \mu_k) = \sum_{i \in C_k} \|x_i - \mu_k\|_2^2.$$

The overall quality of the clustering is given by the total within-cluster deviance:

$$W = \sum_{k=1}^{K} W(C_k, \mu_k).$$

W is the overall objective function used to select both the cluster centroids and the assignment of points to clusters.

K-means

$$W = \sum_{k=1}^{K} \sum_{i \in C_k} \|x_i - \mu_k\|_2^2 = \sum_{i=1}^{n} \|x_i - \mu_{c_i}\|_2^2$$

where $c_i = k$ if and only if $i \in C_k$.

- Given partition $\{C_k\}$, we can find the optimal prototypes easily by differentiating W with respect to μ_k:

$$\frac{\partial W}{\partial \mu_k} = 2 \sum_{i \in C_k} (x_i - \mu_k) = 0 \qquad \Rightarrow \mu_k = \frac{1}{|C_k|} \sum_{i \in C_k} x_i$$

- Given prototypes, we can easily find the optimal partition by assigning each data point to the closest cluster prototype:

$$c_i = \operatorname*{argmin}_k \|x_i - \mu_k\|_2^2$$

But joint minimization over both is computationally difficult.

K-means

The K-means algorithm is a widely used method that returns a **local optimum** of the objective function W, using iterative and alternating minimization.

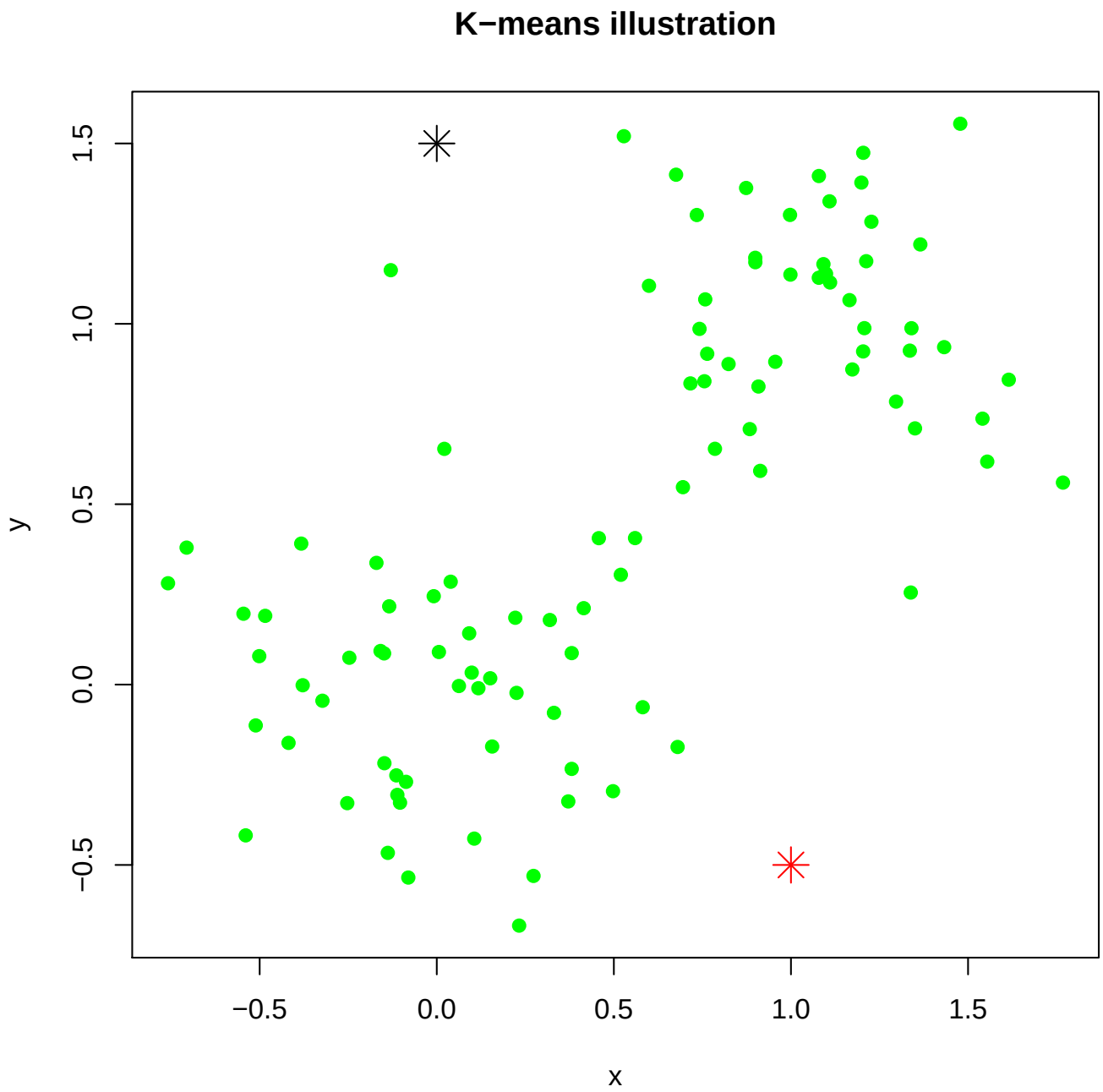
1. Randomly initialize K cluster centroids $\mu_1, \ldots, \mu_K$.
2. **Cluster assignment:** For each $i = 1, \ldots, n$, assign each x_i to the cluster with the nearest centroid,

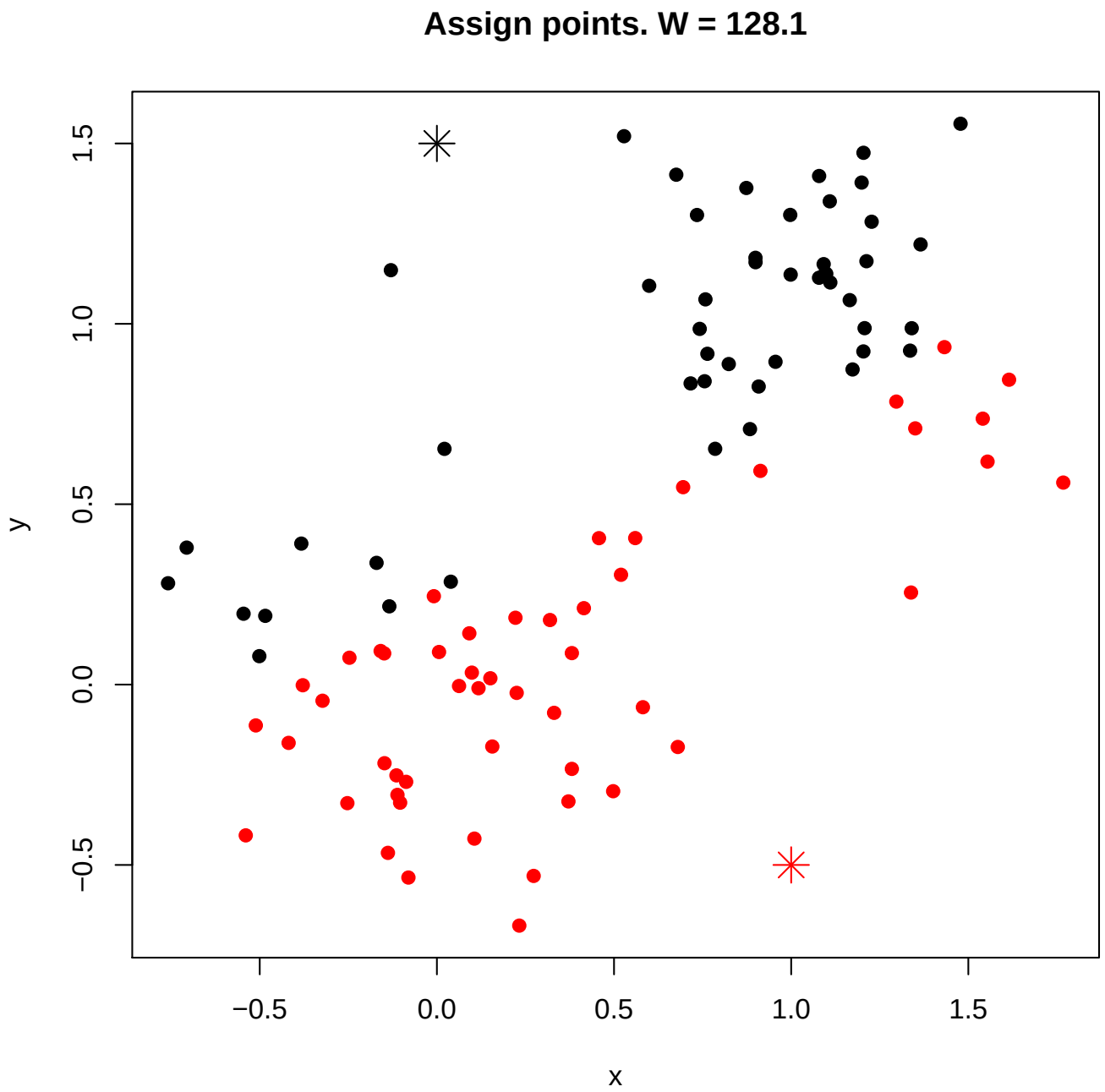
$$c_i := \underset{k}{\operatorname{argmin}} \|x_i - \mu_k\|_2^2$$

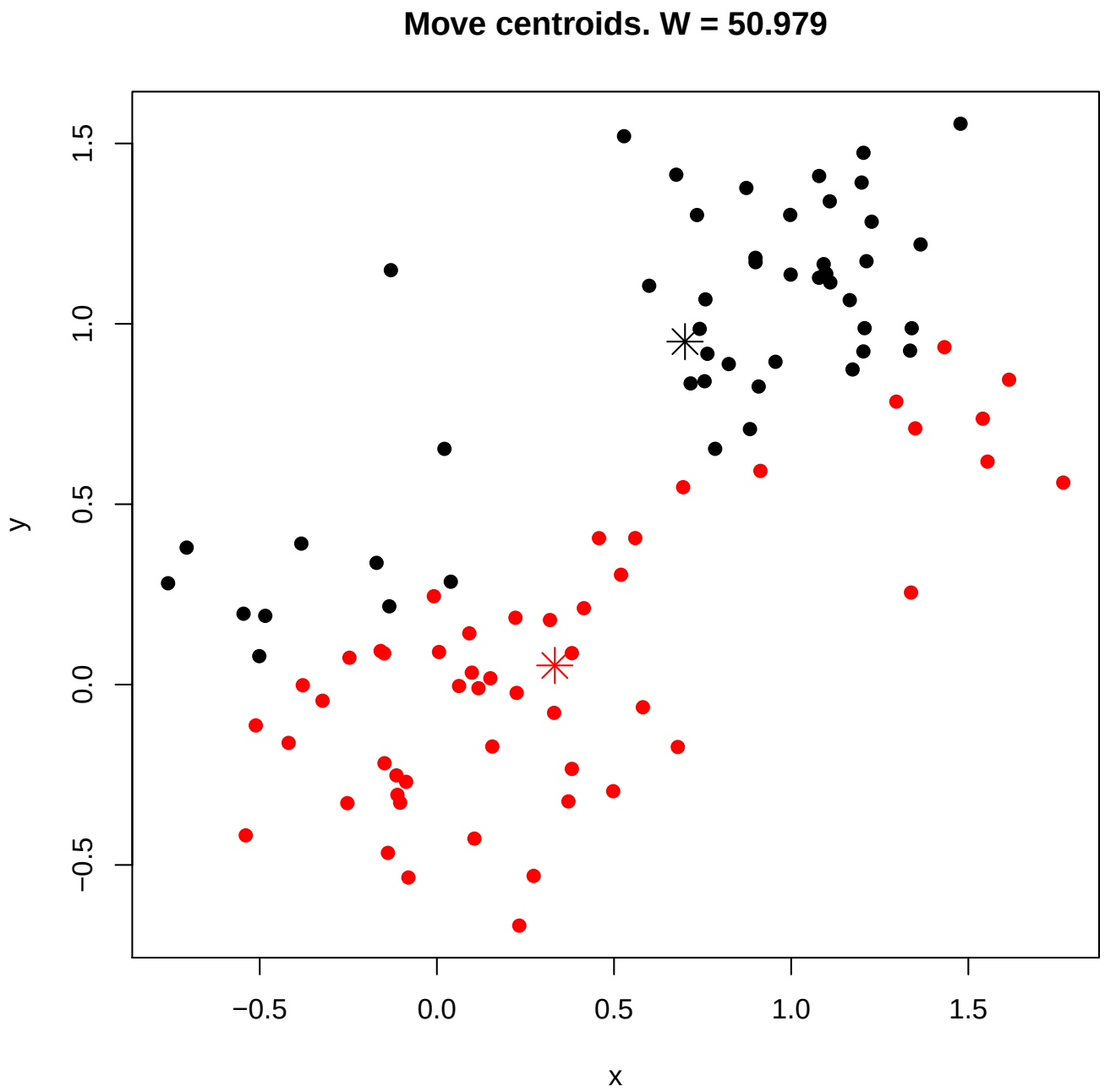
   Set $C_k := \{i : c_i = k\}$ for each k.
3. **Move centroids:** Set $\mu_1, \ldots, \mu_K$ to the averages of the new clusters:

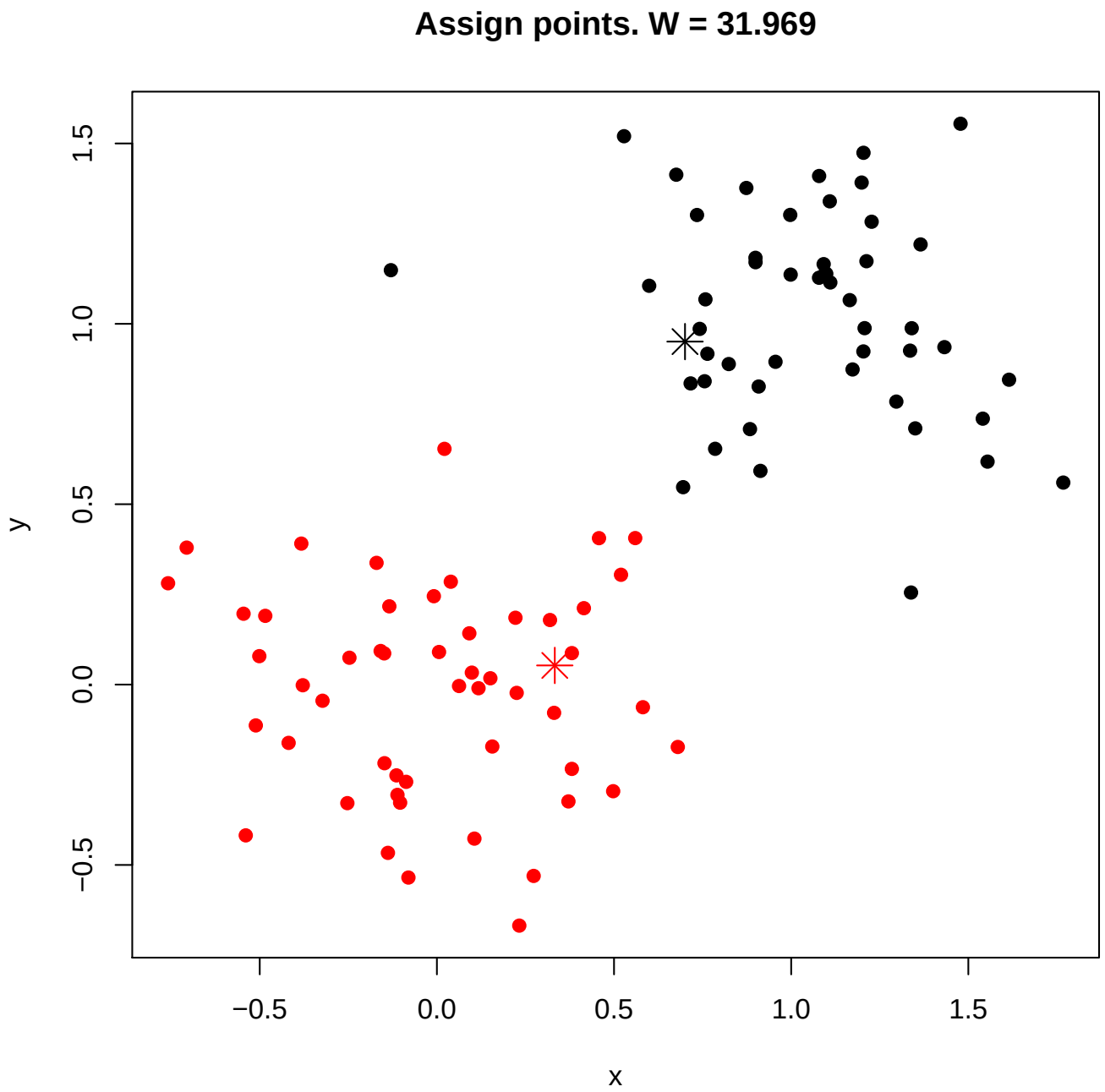
$$\mu_k := \frac{1}{|C_k|} \sum_{i \in C_k} x_i$$

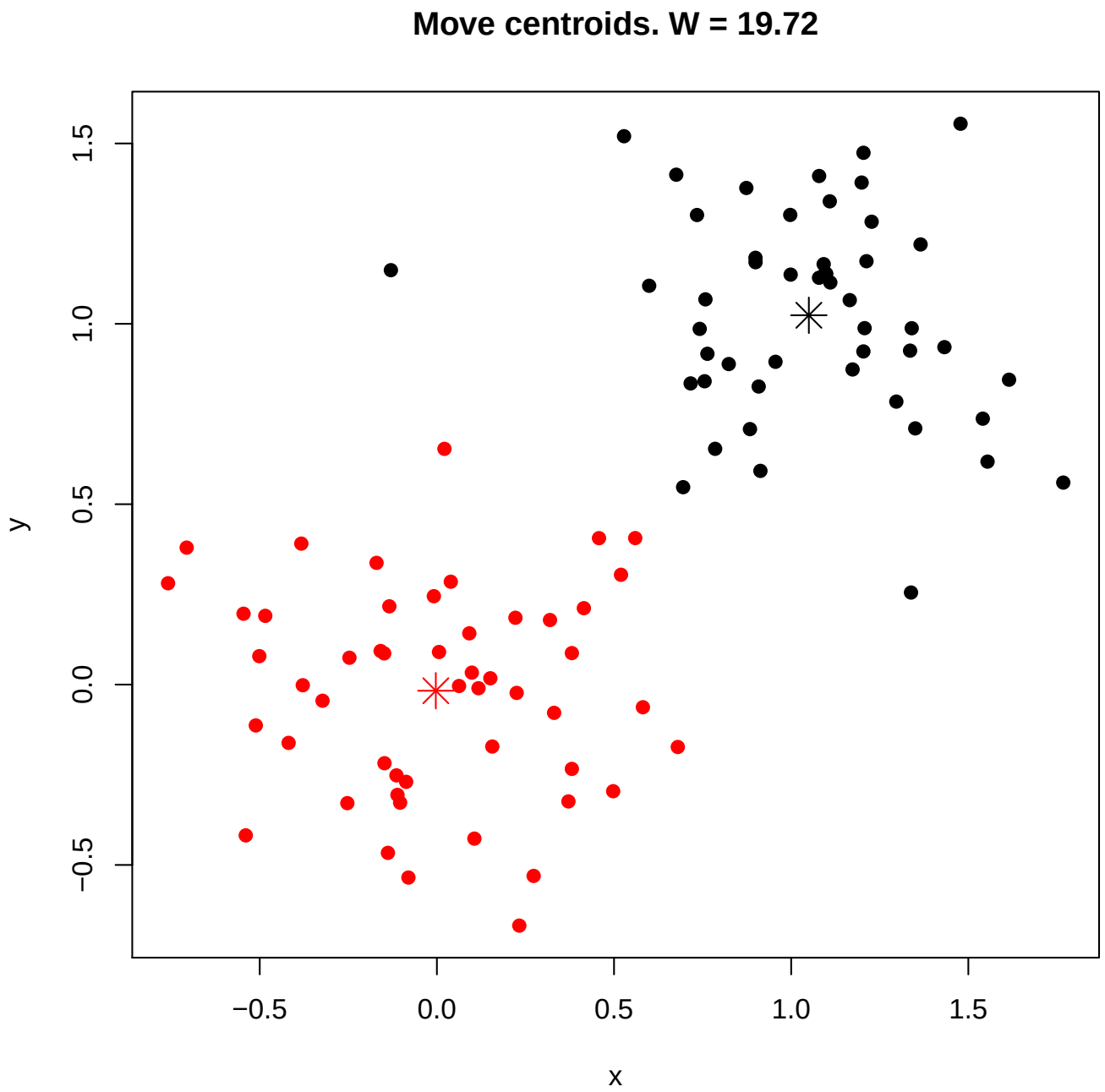
4. Repeat steps 2-3 until convergence.
5. Return the partition $\{C_1, \ldots, C_K\}$ and means $\mu_1, \ldots, \mu_K$.

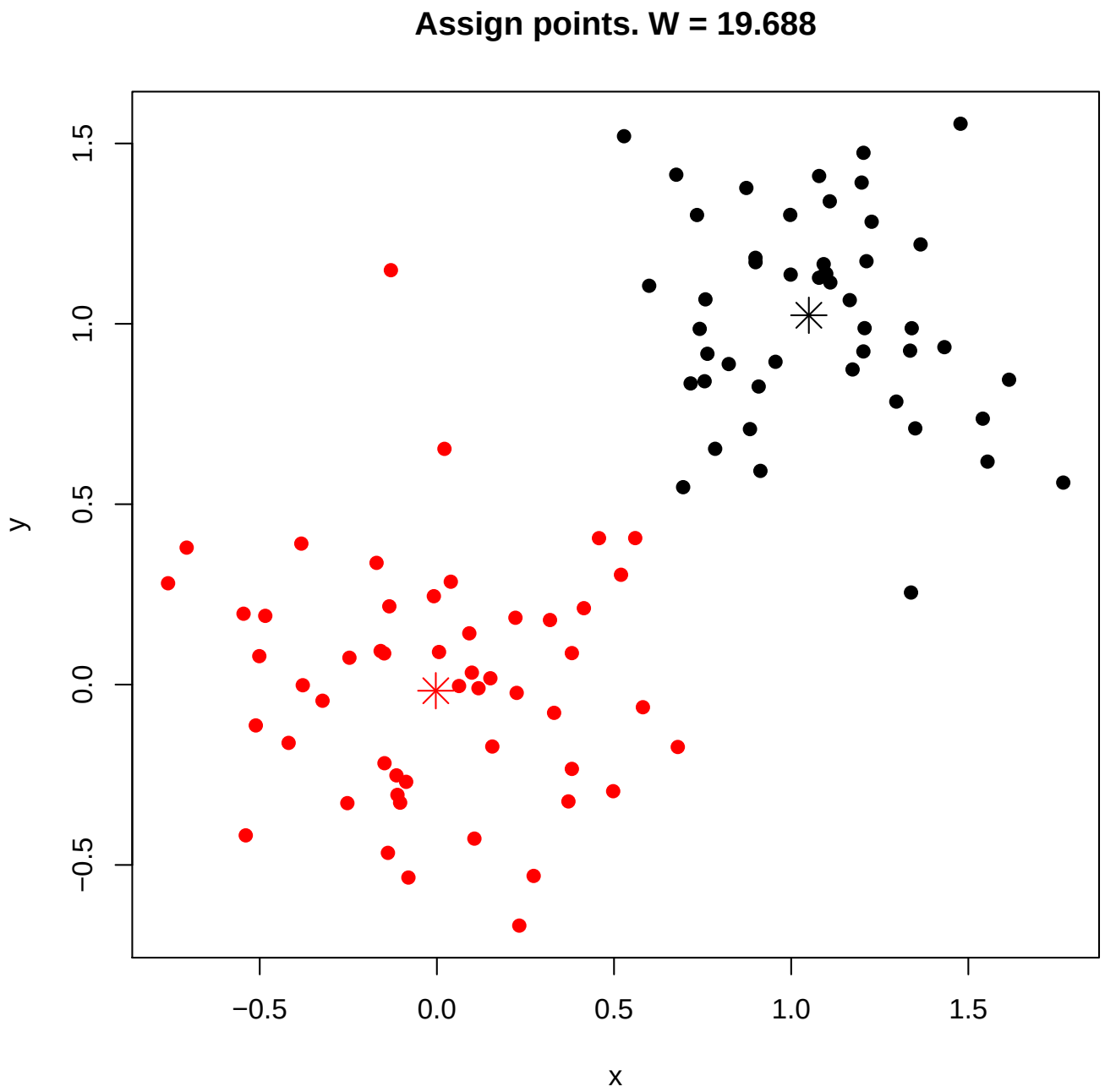

K−means illustration
y
x
1.5
1.0
0.5
0.0
−0.5
−0.5
0.0
0.5
1.0
1.5

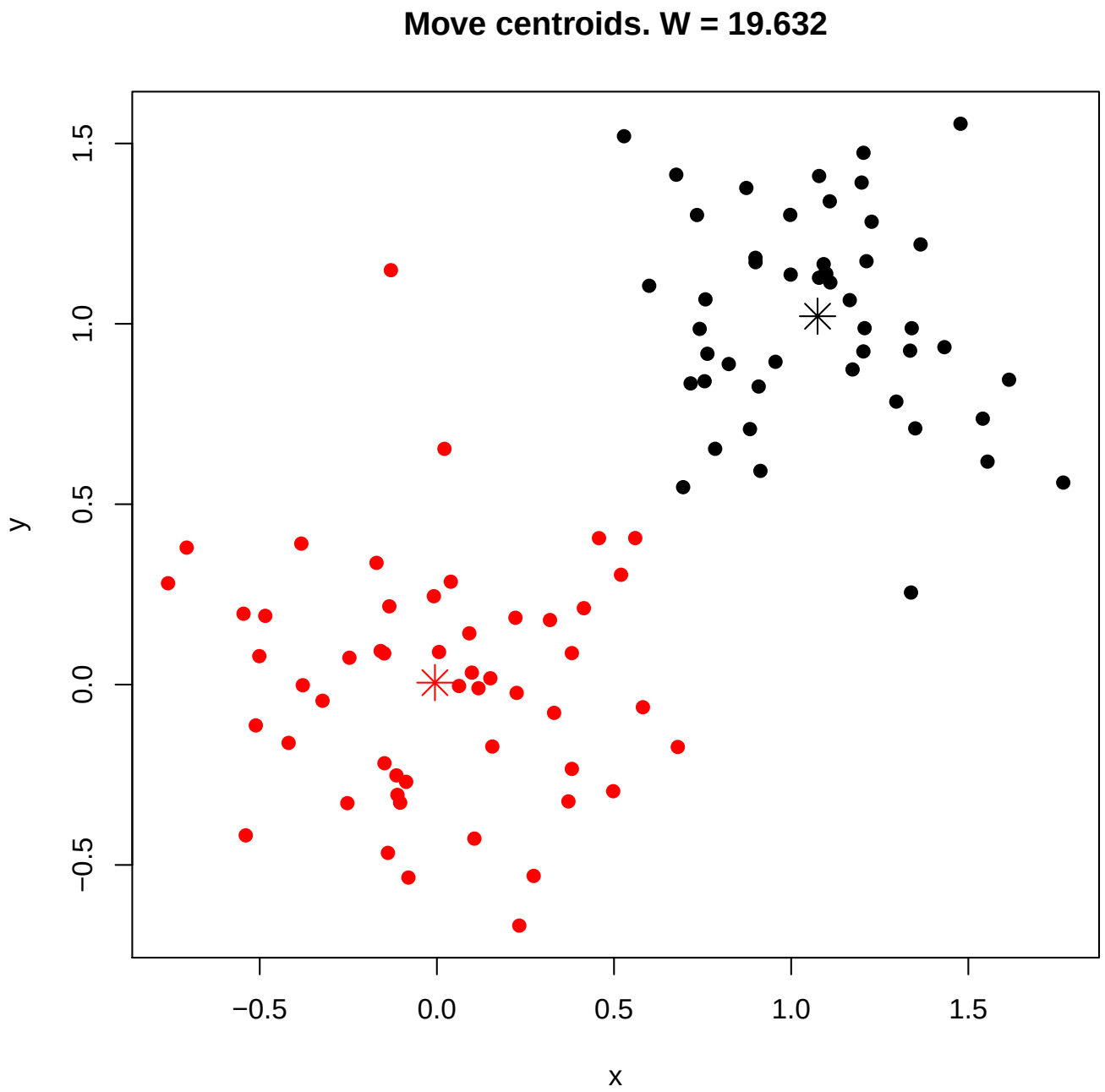

Assign points. W = 128.1
y
x
1.5
1.0
0.5
0.0
−0.5
−0.5
0.0
0.5
1.0
1.5


Move centroids. W = 50.979
y
x
−0.5
0.0
0.5
1.0
1.5


Assign points. W = 31.969
y
x
1.5
1.0
0.5
0.0
−0.5
−0.5
0.0
0.5
1.0
1.5

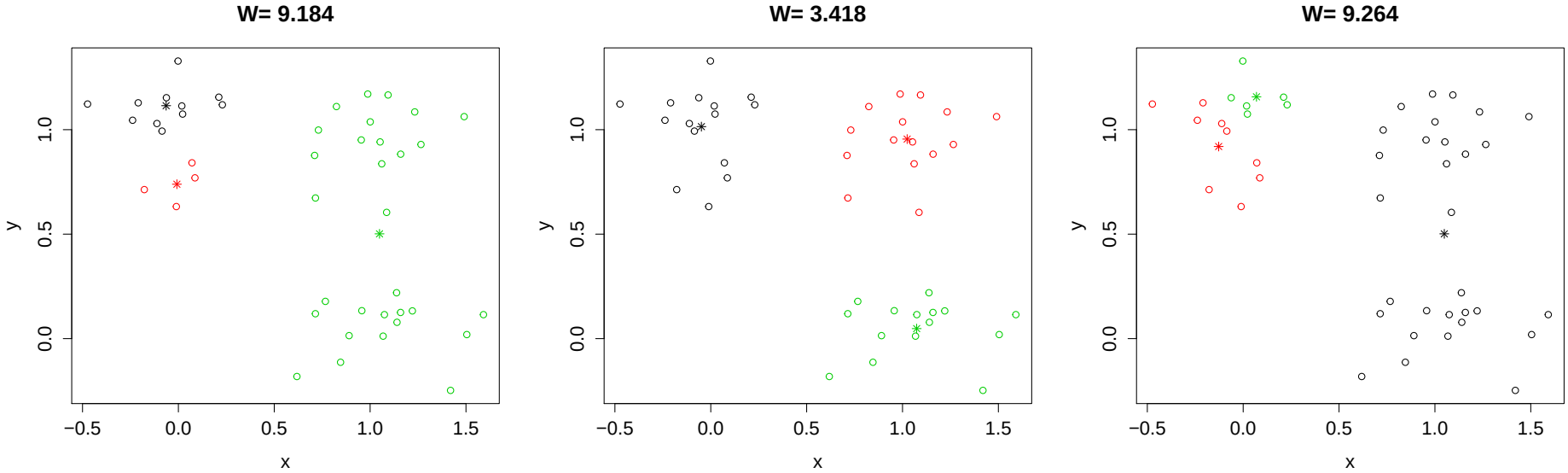

Move centroids. W = 19.72
y
x
1.5
1.0
0.5
0.0
−0.5
−0.5
0.0
0.5
1.0
1.5


Assign points. W = 19.688
y
x
1.5
1.0
0.5
0.0
−0.5
−0.5
0.0
0.5
1.0
1.5


Move centroids. W = 19.632
y
x
1.5
1.0
0.5
0.0
−0.5
−0.5
0.0
0.5
1.0
1.5

K-means

- **The algorithm stops in a finite number of iterations.** Between steps 2 and 3, W either stays constant or it decreases, this implies that we never revisit the same partition. As there are only finitely many partitions, the number of iterations cannot exceed this.
- **The K-means algorithm need not converge to global optimum.** K-means is a heuristic search algorithm so it can get stuck at suboptimal configurations. The result depends on the starting configuration. Typically perform a number of runs from different configurations, and pick the end result with minimum W.



K-means Additional Comments

- **Good practice initialization.** Set centroids $\mu_1, \mu_2, \ldots, \mu_K$ equal to a subset of training examples (chosen without replacement). Initialization using weighted sampling of training examples (**K-means++**) has precise theoretical guarantees[1]
- **Sensitivity to distance measure.** Euclidean distance can be greatly affected by measurement unit and by strong correlations. Can use Mahalanobis distance instead:

$$\|x - y\|_M = \sqrt{(x-y)^\top M^{-1}(x-y)}$$

  where M is positive semi-definite matrix, e.g. sample covariance.
- **Determination of K.** The K-means objective will always improve with larger number of clusters K. Determination of K requires an additional **regularization** criterion. E.g., in **DP-means**[2], use

$$W = \sum_{k=1}^{K} \sum_{i \in C_k} \|x_i - \mu_k\|_2^2 + \lambda K$$

[1] Arthur & Vassilvitskii, 2007

[2] Kulis & Jordan, 2012

Other partition based methods

Other partition-based methods with related ideas:

- **K-medoids**: requires cluster centroids μ_i to be an observation x_j
- **K-medians**: cluster centroids represented by a median in each dimension
- **K-modes**: cluster centroids represented by a mode estimated from a cluster