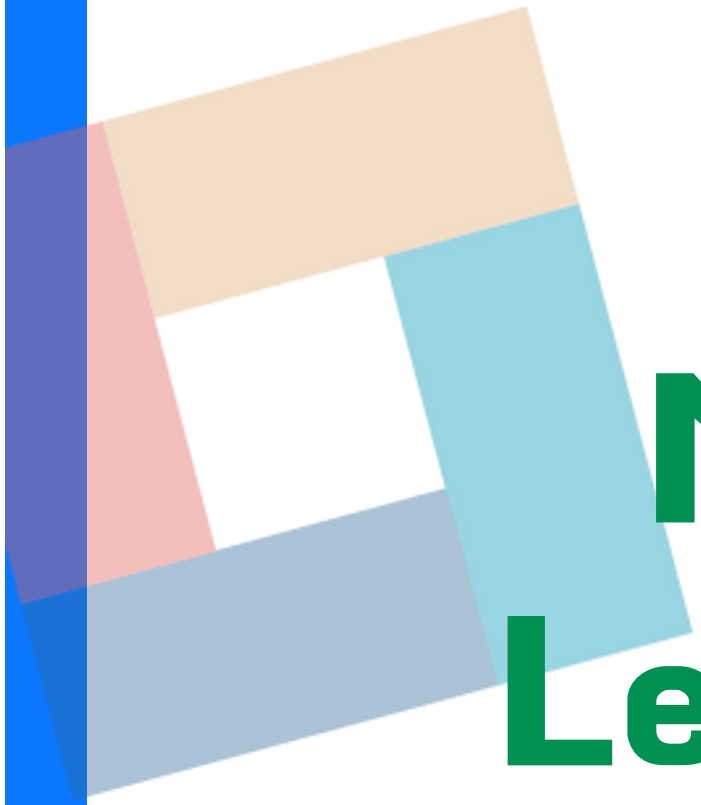


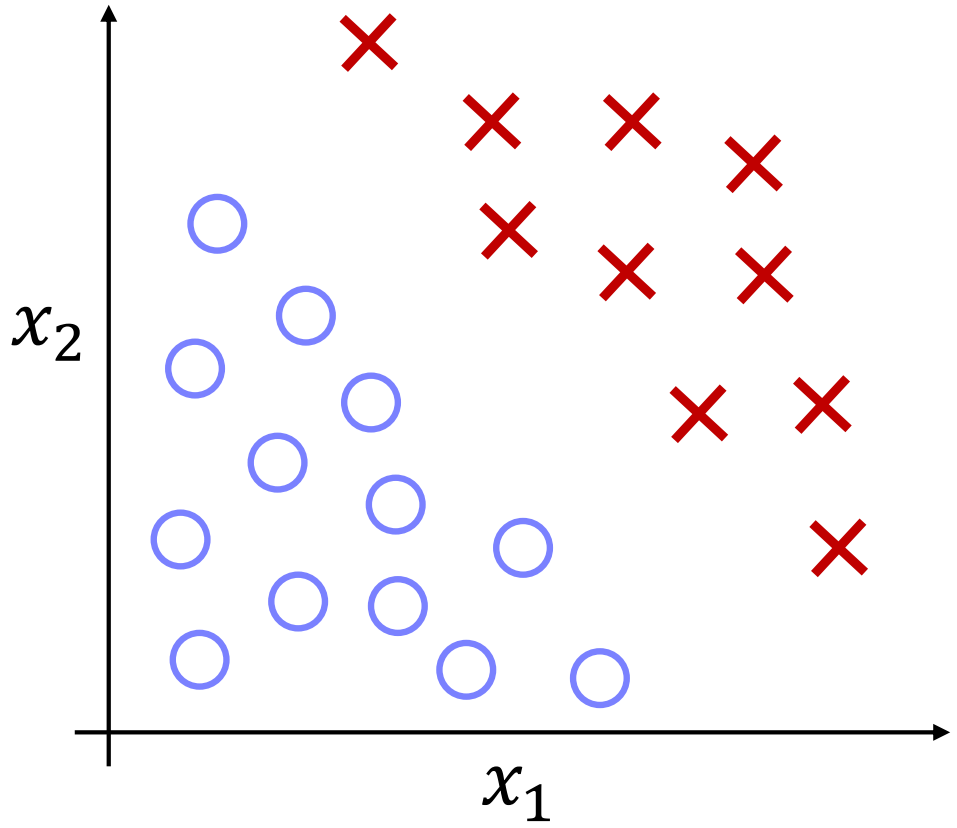
Machine Learning 11

Kihyun Shin
DMSE, HBNU

TheMAC

Clustering

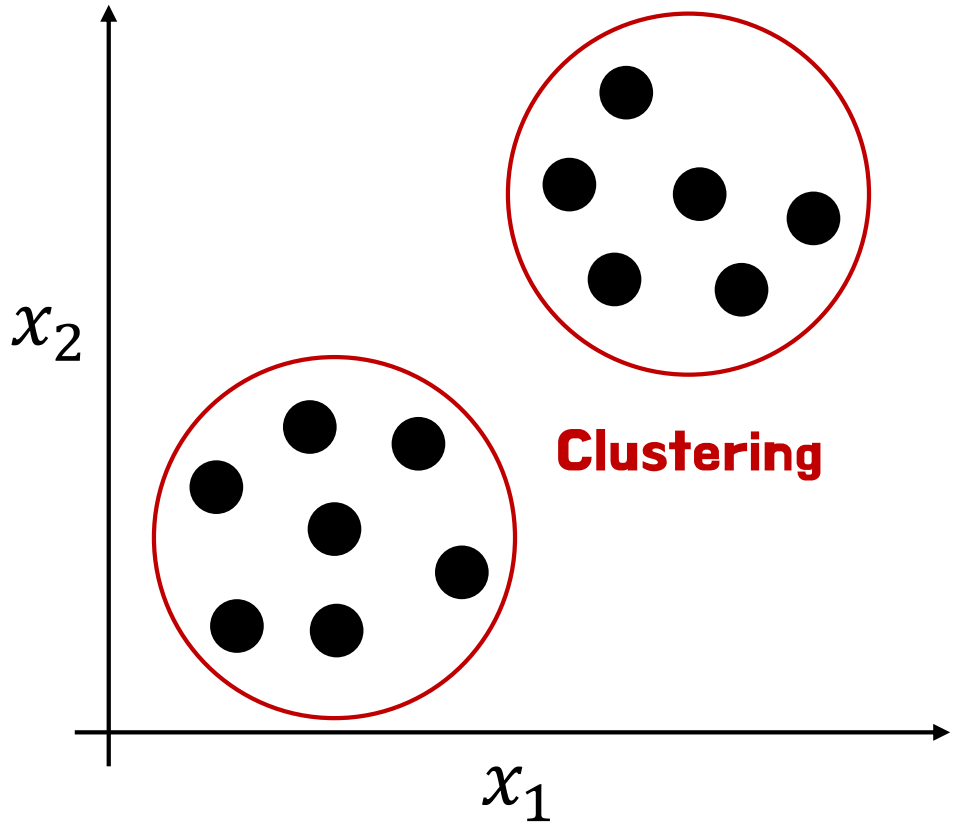
Supervised learning



Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), (x^{(3)}, y^{(3)}), \ldots, (x^{(m)}, y^{(m)})\}$

TheMAD

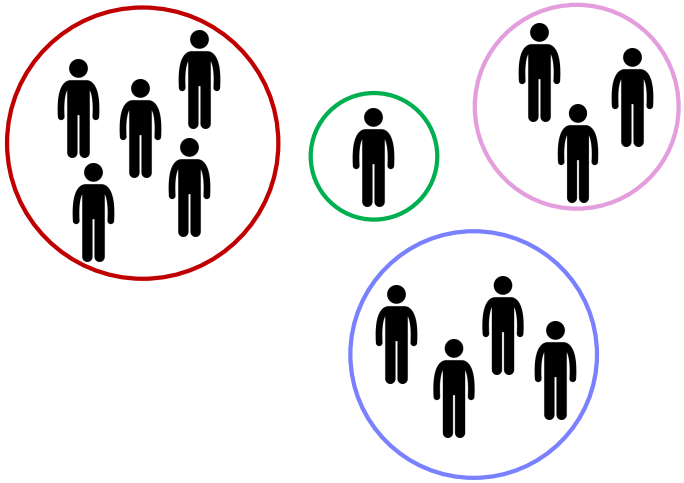
Unsupervised learning



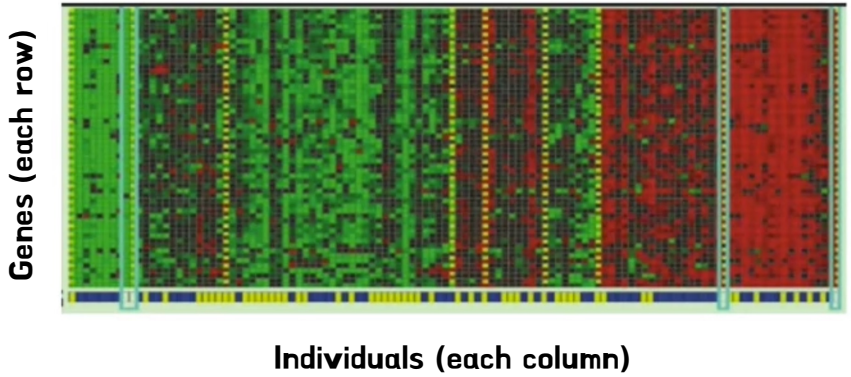
Training set: $\{x^{(1)}, x^{(2)}, x^{(3)}, \ldots, x^{(m)}\}$

Applications of clustering

Grouping similar news



Market segmentation

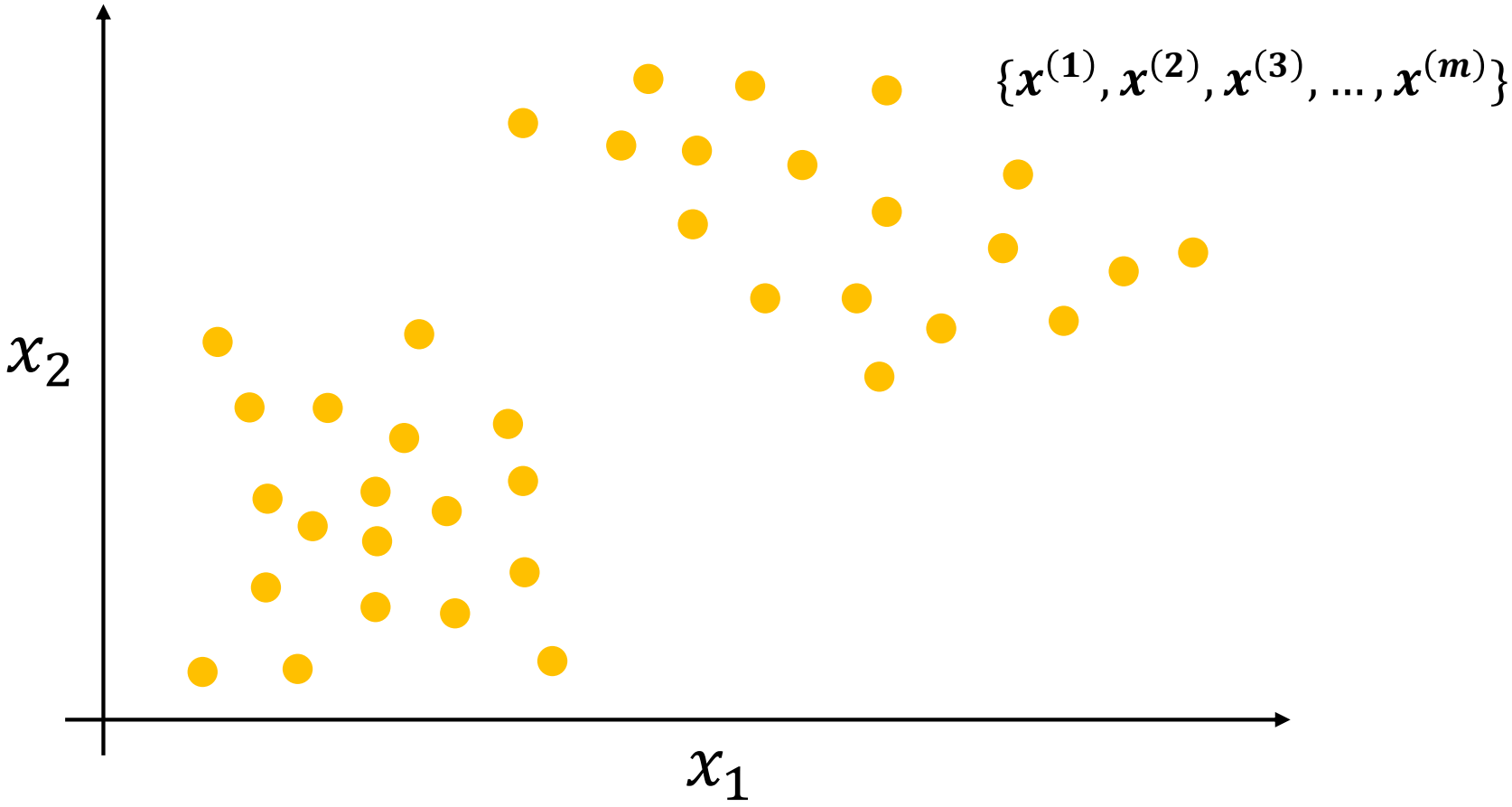


DNA analysis

Astronomical data analysis

TheMAD

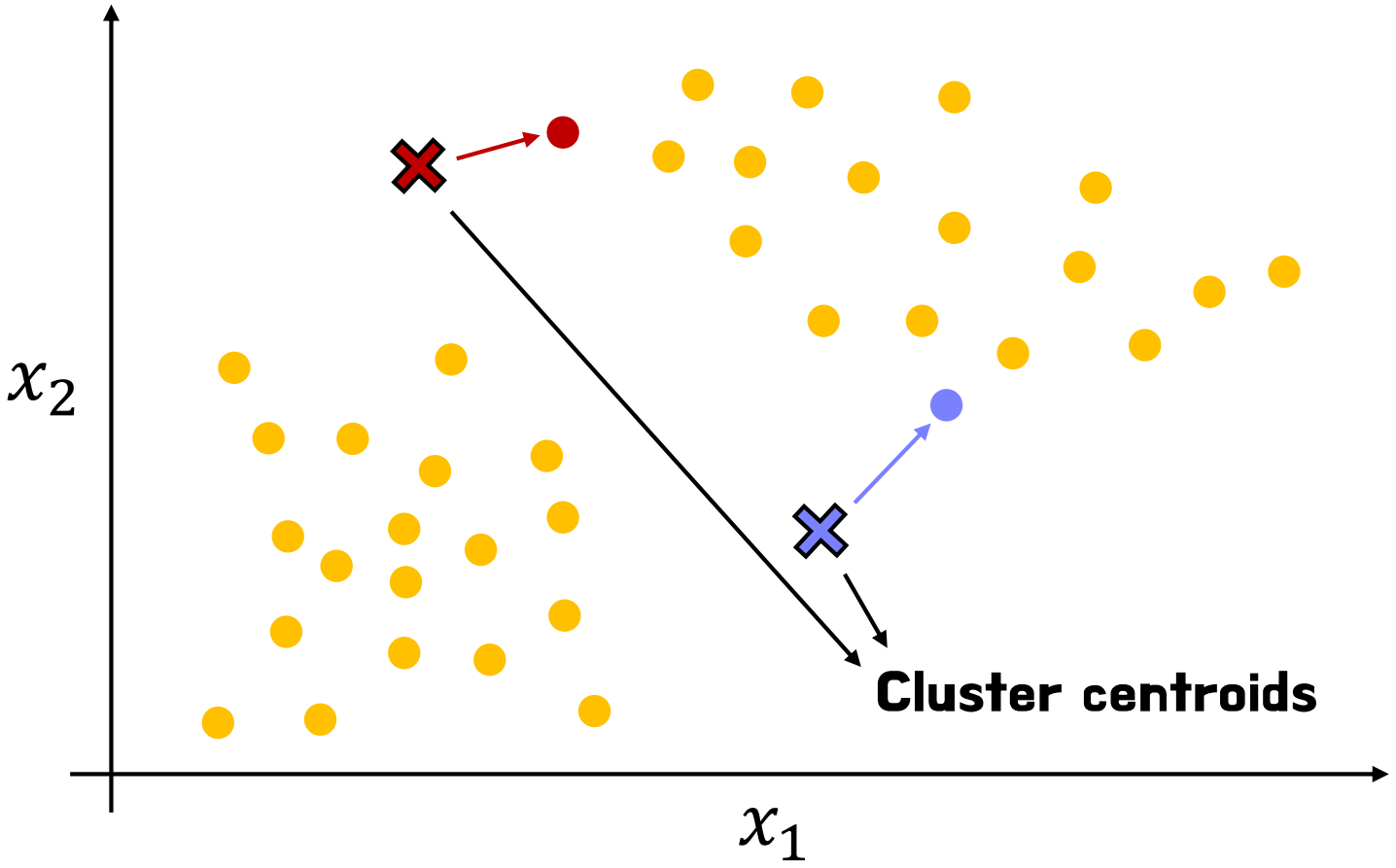
K-means intuition



TheMAD

K-means intuition

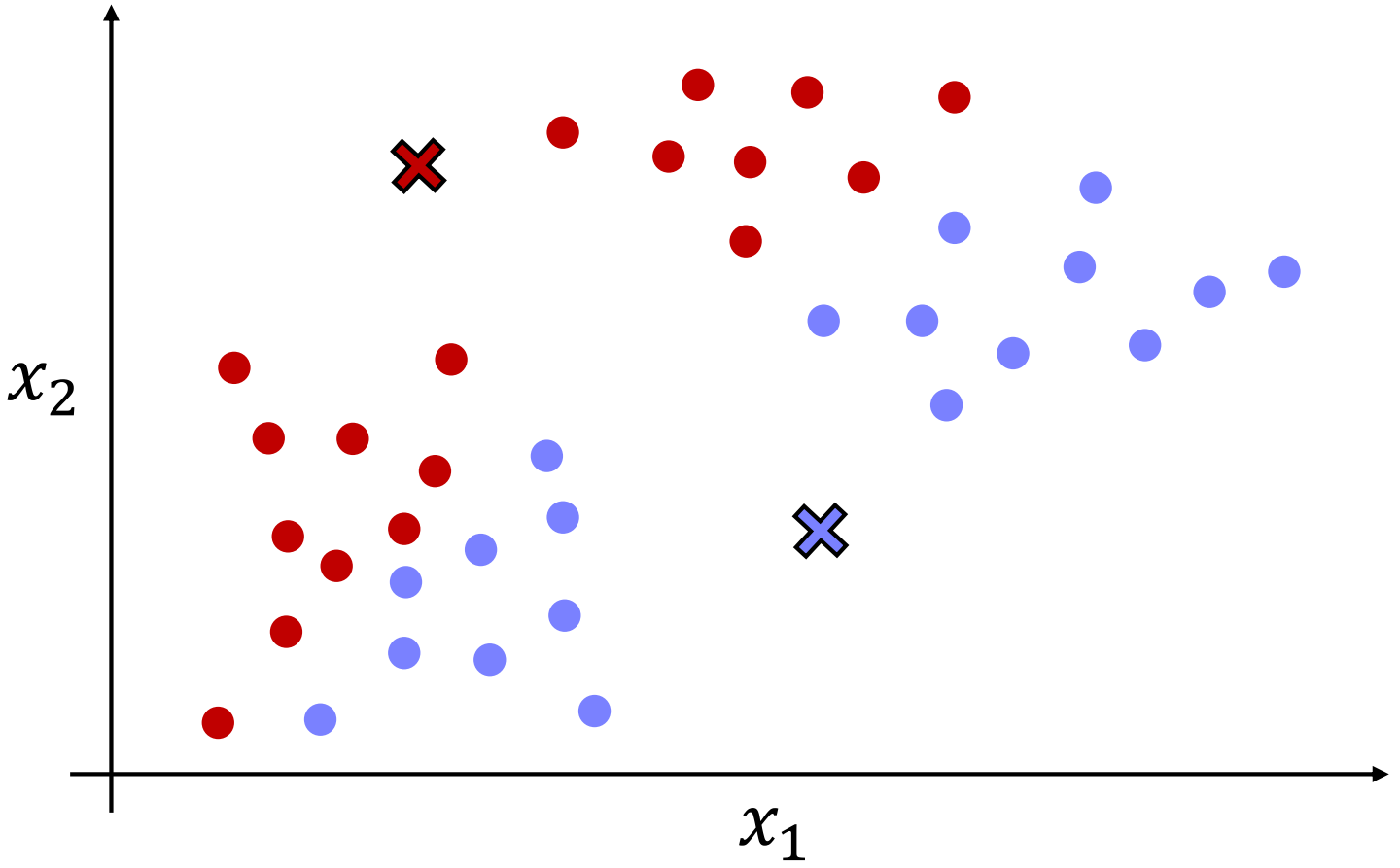
Step 1: Assign each point to its closest centroid



TheMAD

K-means intuition

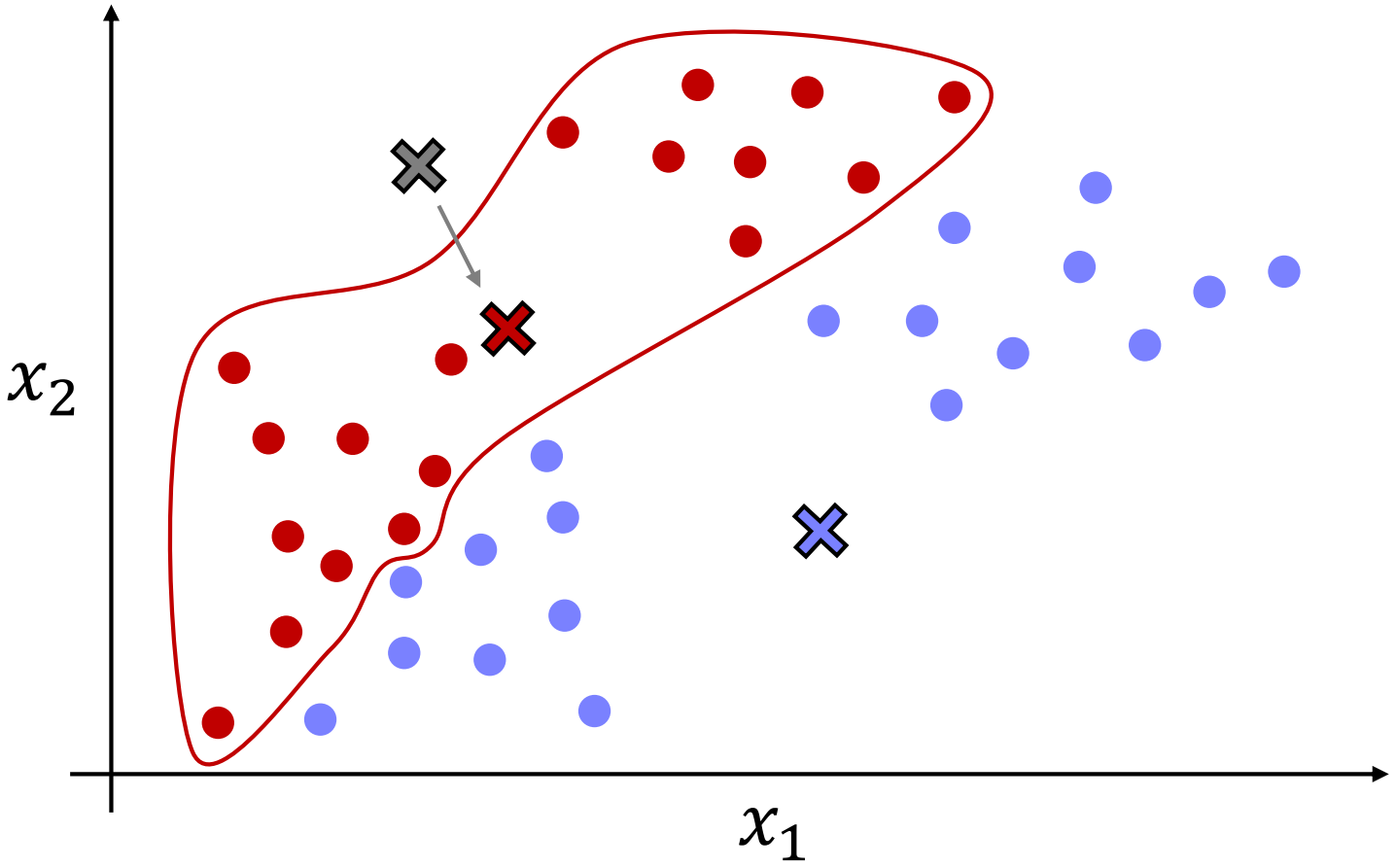
Step 1: Assign each point to its closest centroid



TheMAD

K-means intuition

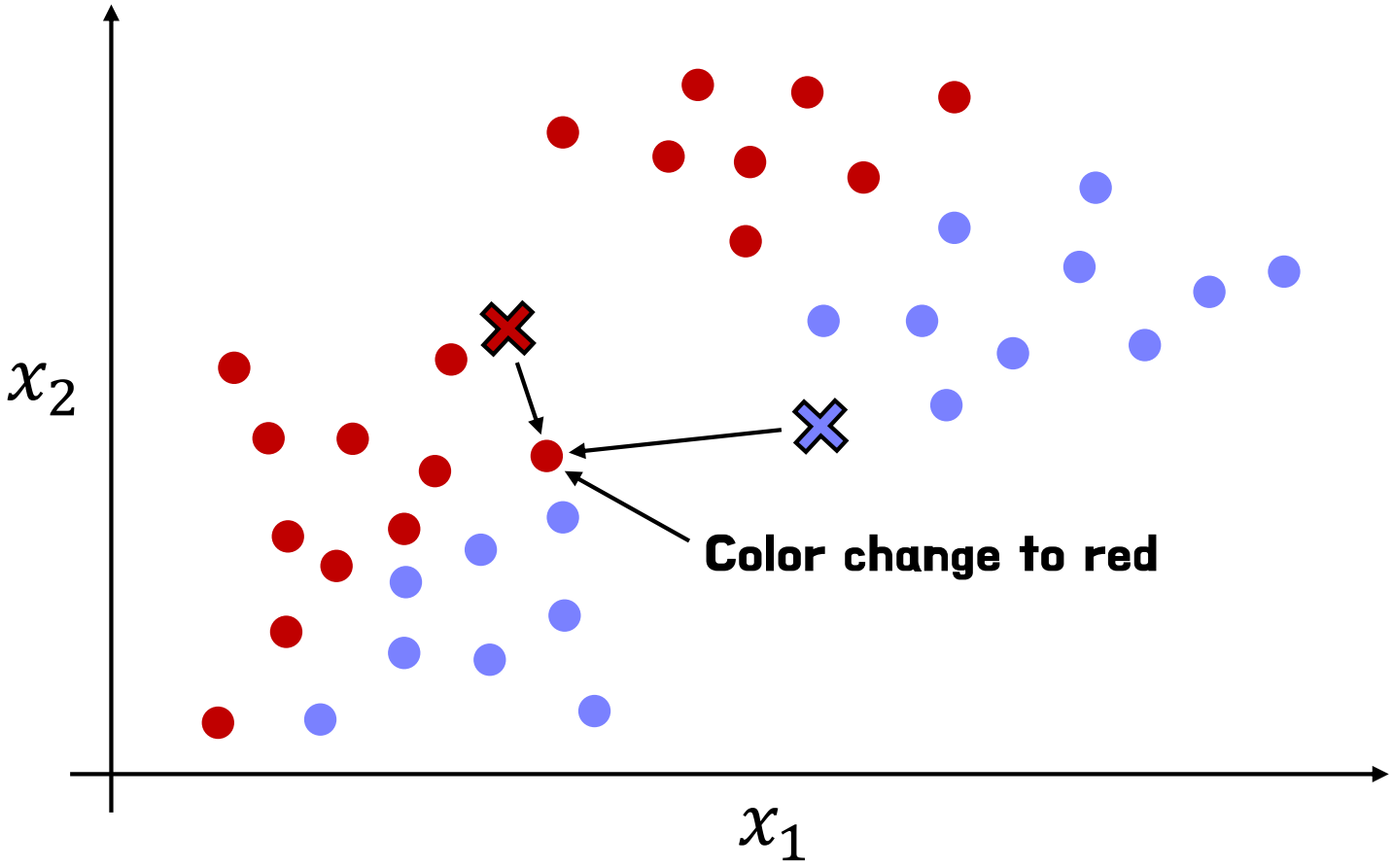
Step 2: Recompute the centroids



TheMAD

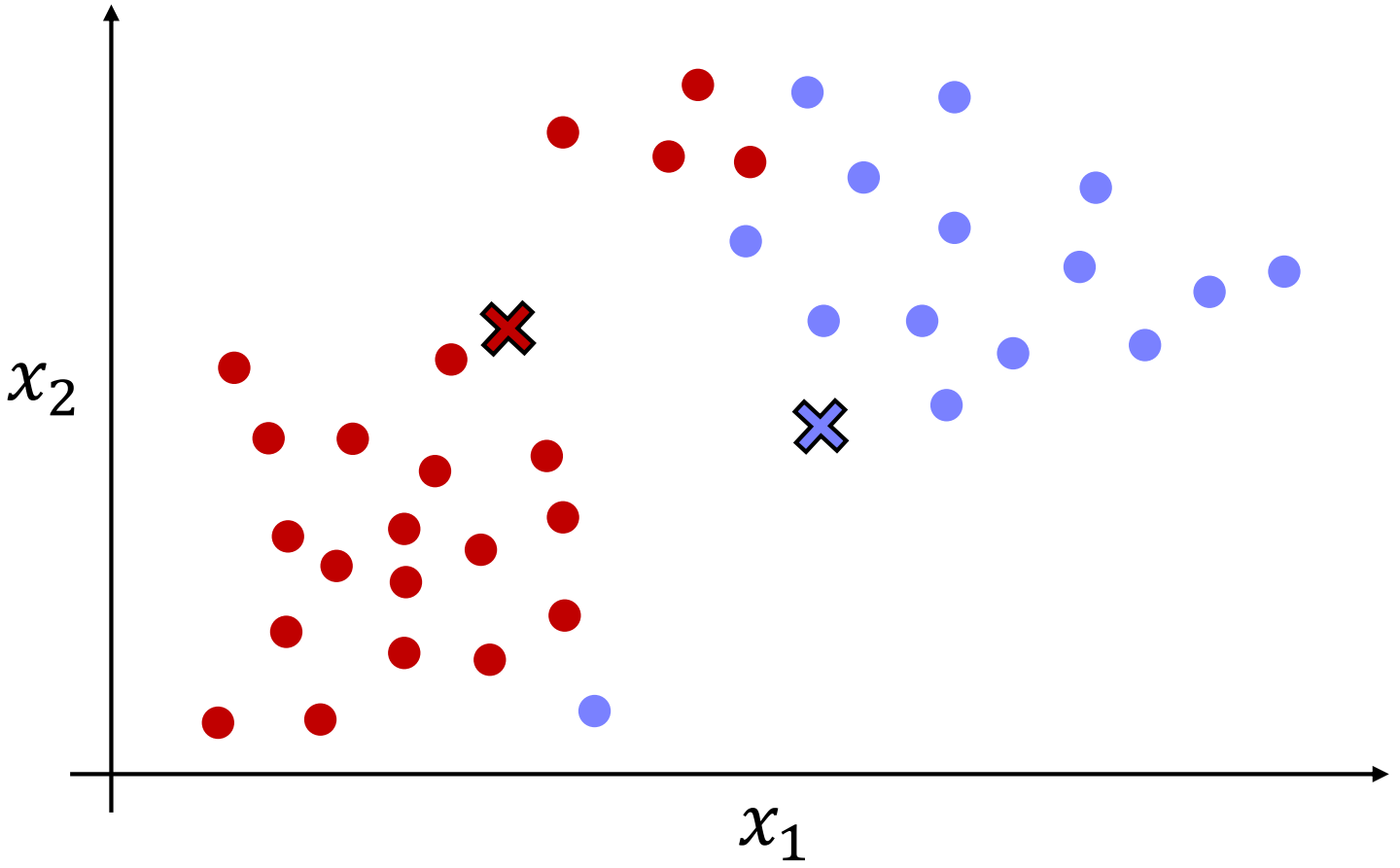
K-means intuition

Step 1: Assign each point to its closest centroid (new)



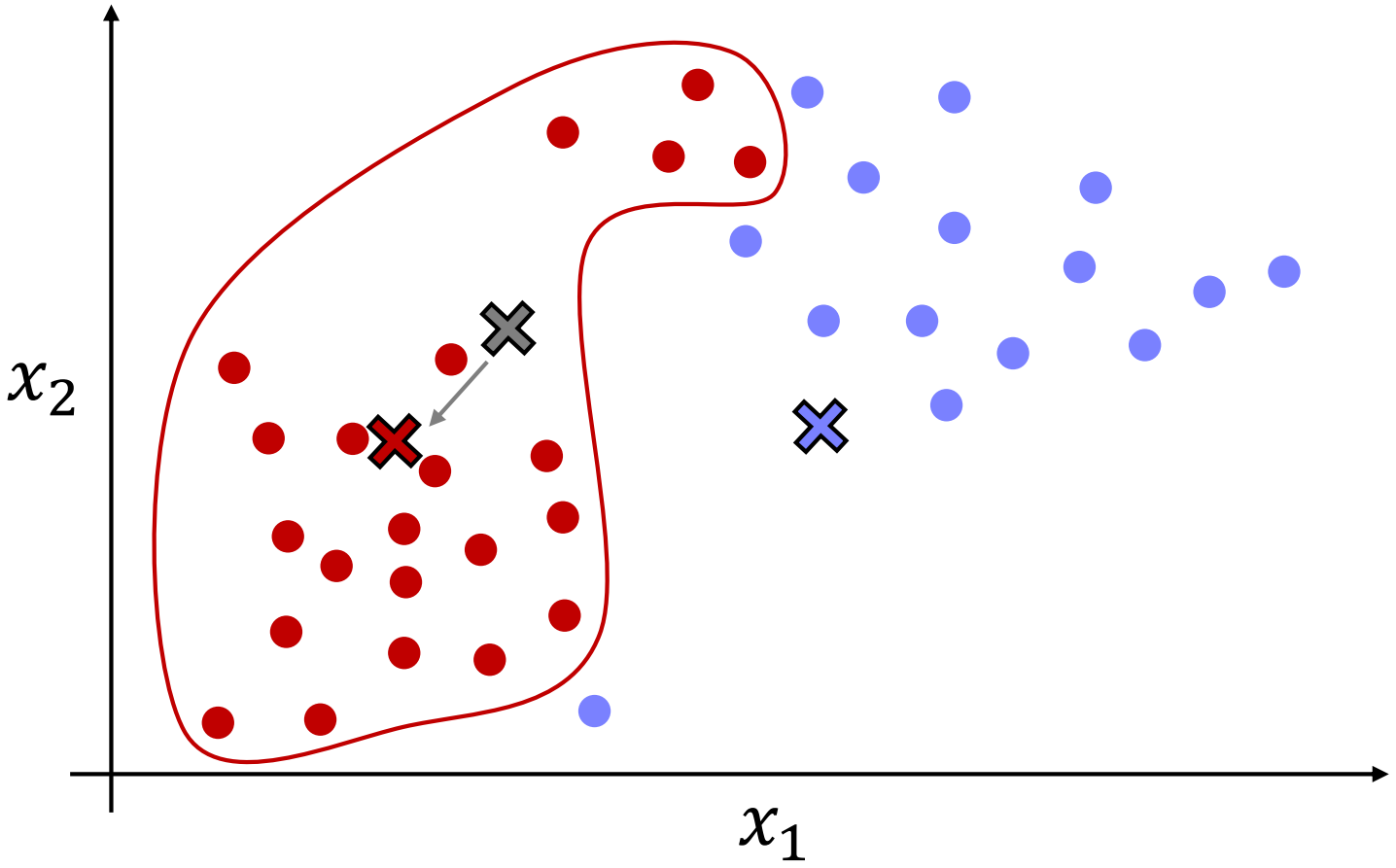
K-means intuition

Step 1: Assign each point to its closest centroid (new)



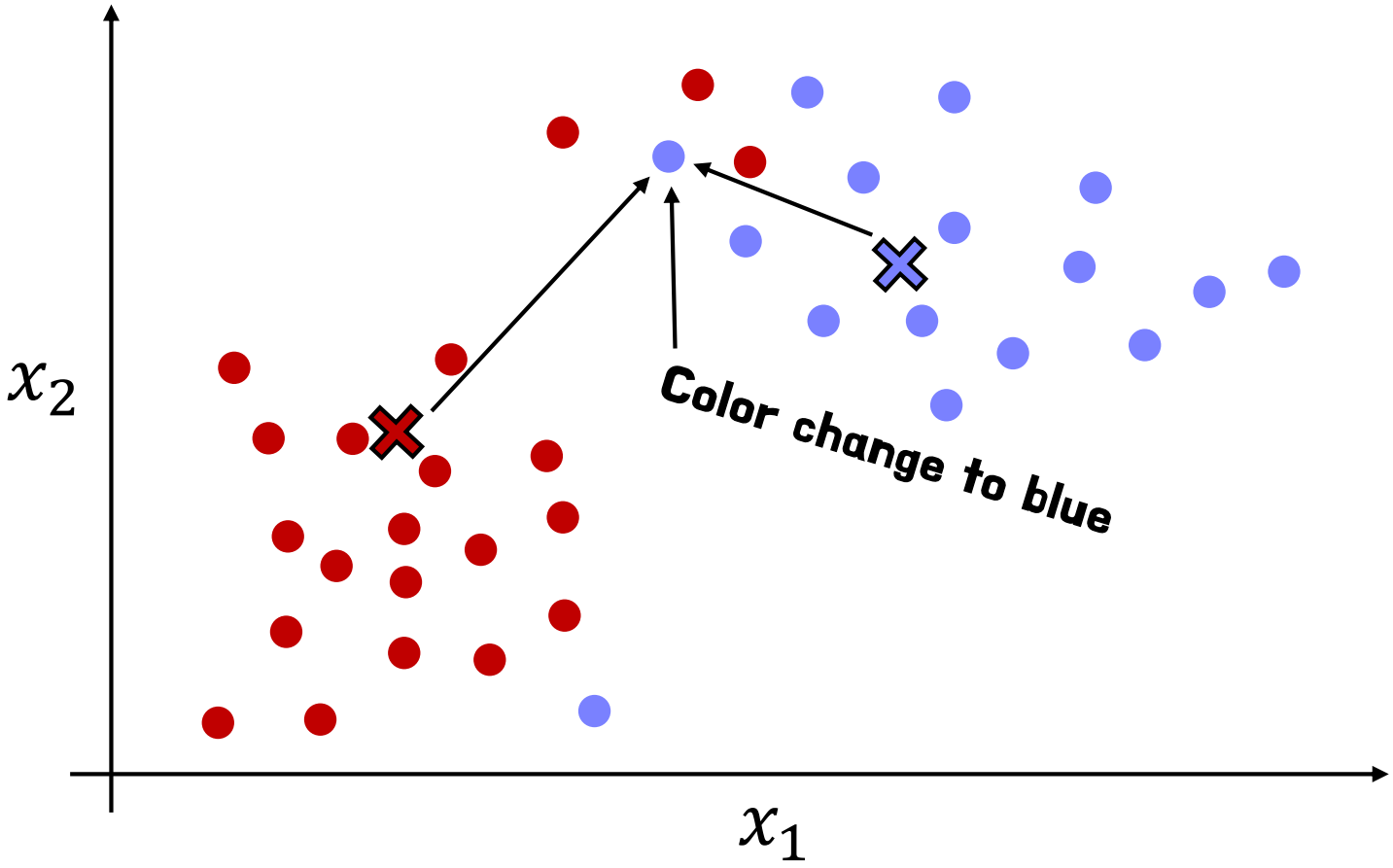
K-means intuition

Step 2: Recompute the centroids



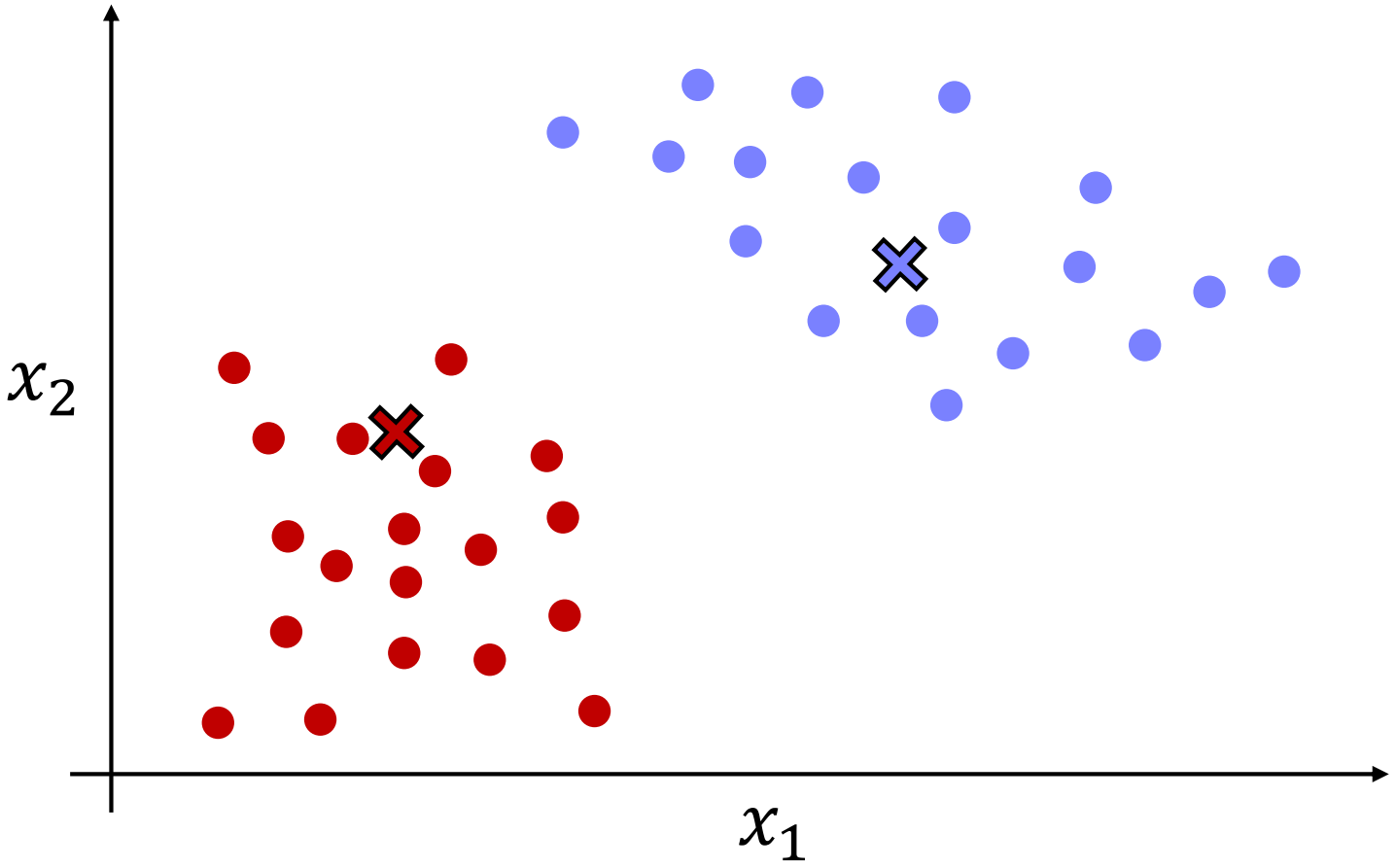
K-means intuition

Step 1: Assign each point to its closest centroid (new)



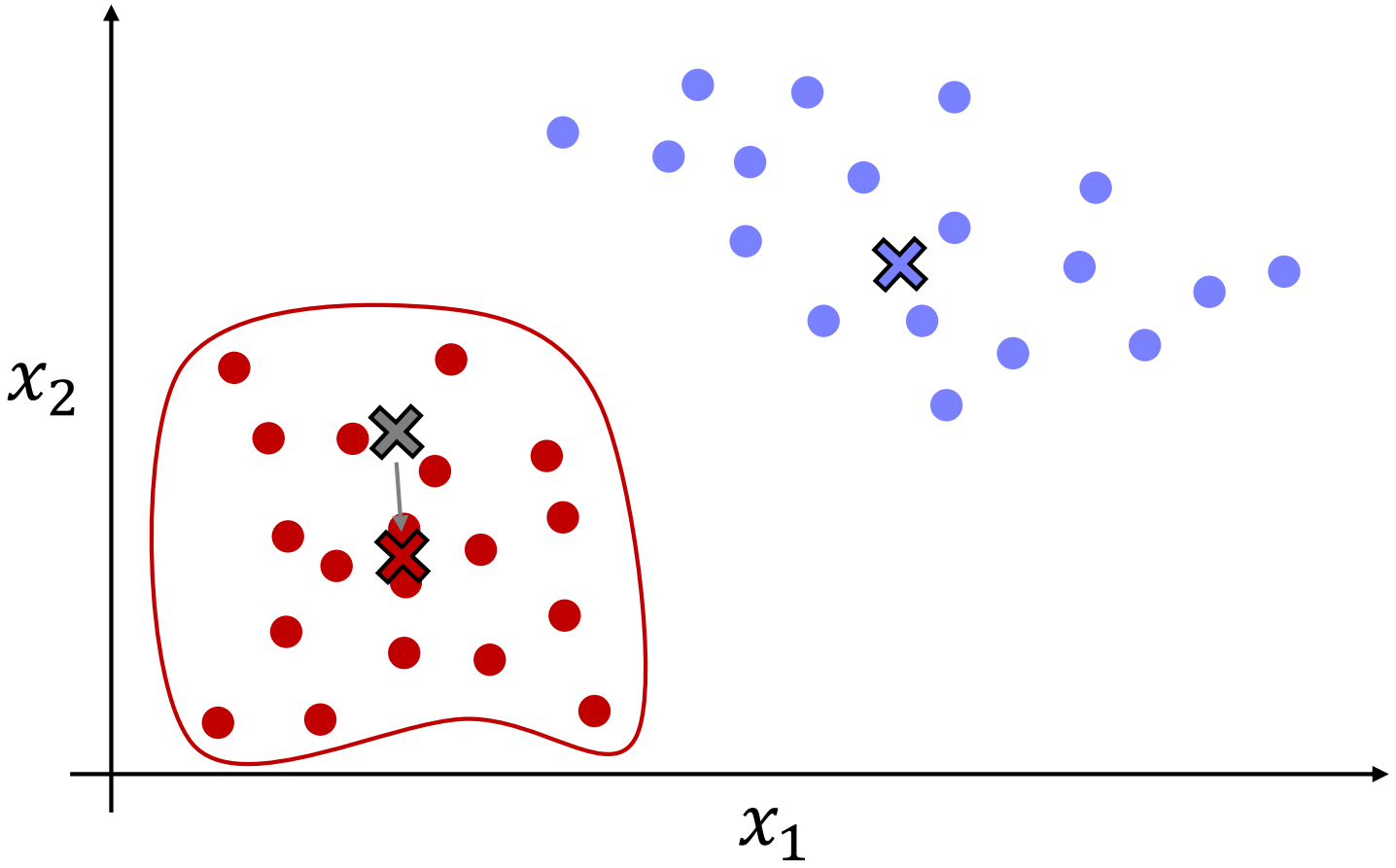
K-means intuition

Step 1: Assign each point to its closest centroid (new)



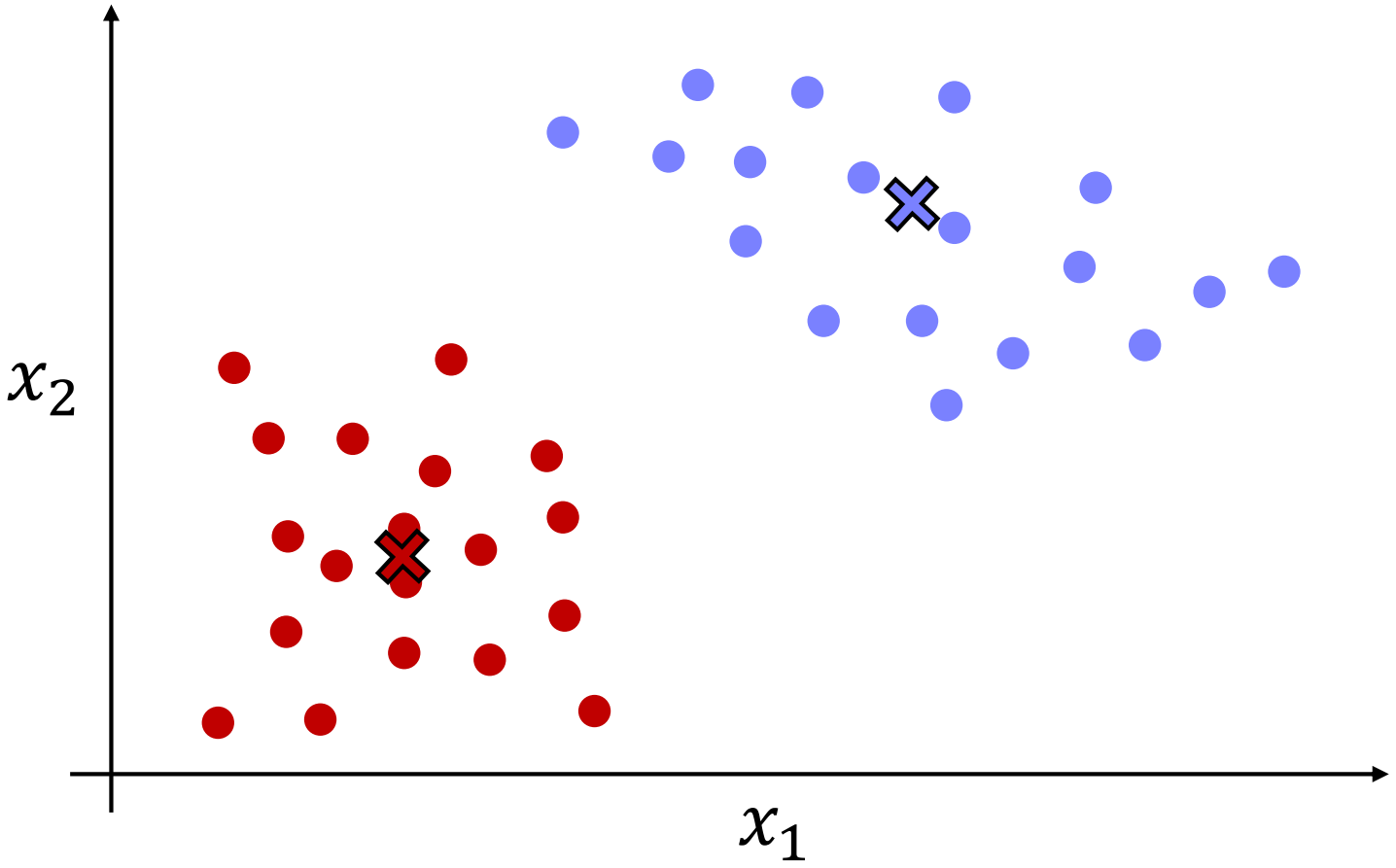
K-means intuition

Step 2: Recompute the centroids



K-means intuition

Step 1: Assign each point to its closest centroid (new)



K-means algorithm

Randomly initialize K cluster centroids $(\mu_1, \mu_2, \dots, \mu_K)$

Repeat {

Assign points to cluster centroids

$for\ i = 1\ to\ m$

$c^{(i)} := index\ (from\ 1\ to\ K) of\ cluster\ centroid\ closest\ to\ x^{(i)}$

$\longrightarrow\ min_k \left\| x^{(i)} - \mu_k \right\|^2$

TheMAD

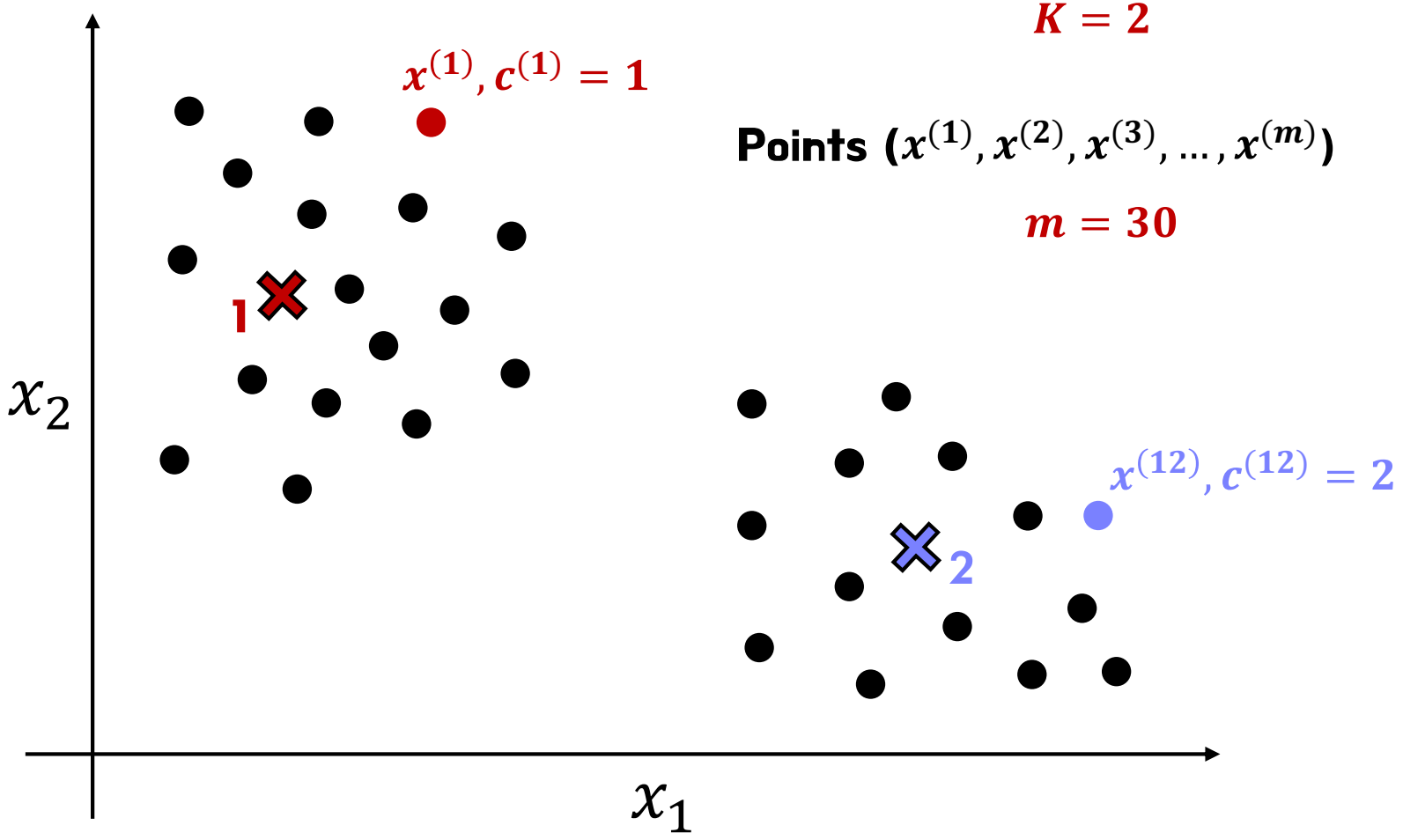
K-means algorithm

K cluster centroids $(\mu_1, \mu_2, \dots, \mu_K)$

$K = 2$

Points $(x^{(1)}, x^{(2)}, x^{(3)}, \dots, x^{(m)})$

$m = 30$



K-means algorithm

Randomly initialize K cluster centroids $(\mu_1, \mu_2, \dots, \mu_K)$

Repeat {

 # Assign points to cluster centroids

$for\ i = 1\ to\ m$

$c^{(i)} := index\ (from\ 1\ to\ K) of\ cluster\ centroid\ closest\ to\ x^{(i)}$

$\rightarrow\ min_k \left\| x^{(i)} - \mu_k \right\|^2$

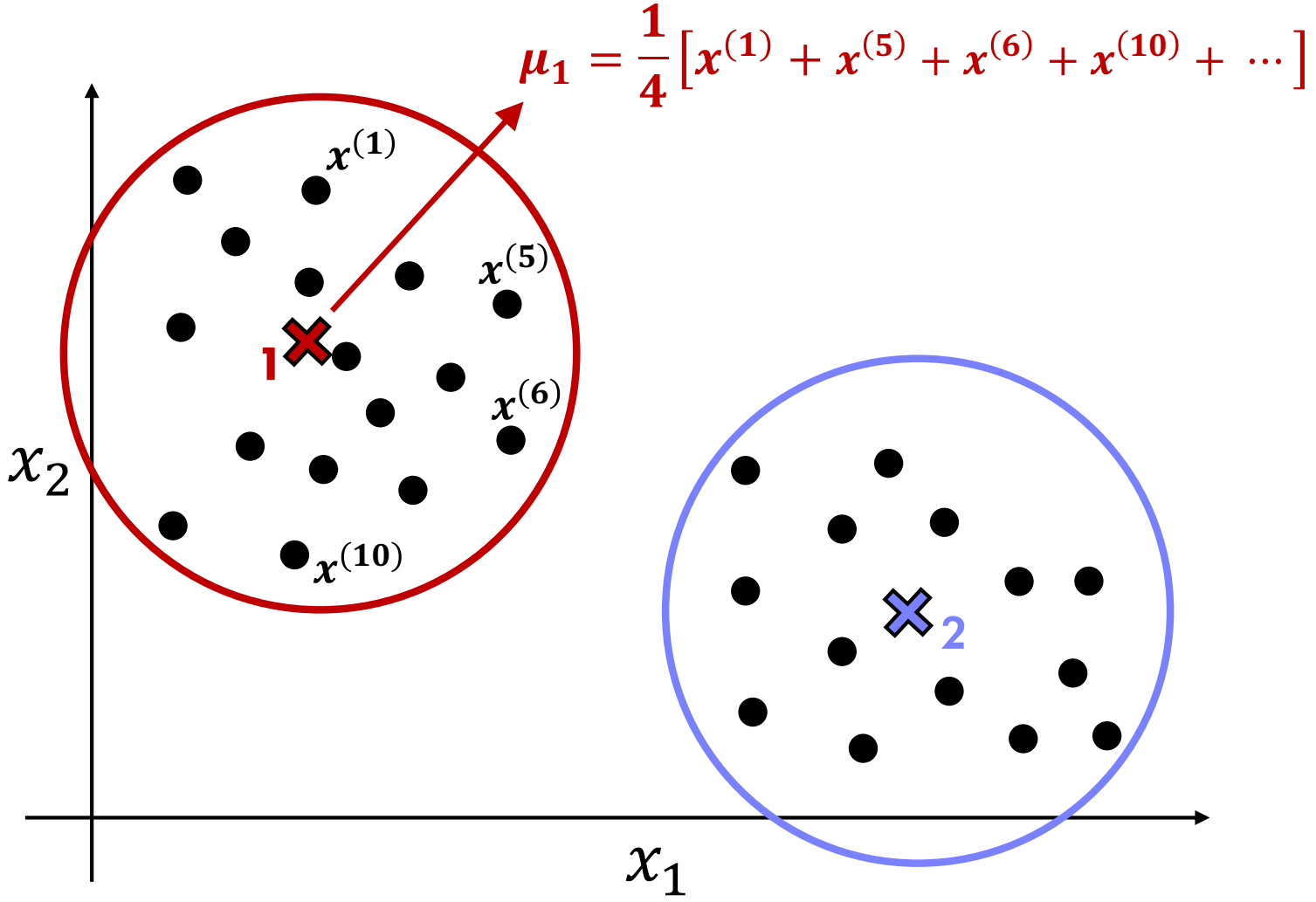
 # Move cluster centroids

$for\ k = 1\ to\ K$

$\mu_k := average\ (mean) of\ points\ assigned\ to\ cluster\ k$

$\rightarrow\ \mu_1 = \frac{1}{4}\left[x^{(1)} + x^{(5)} + x^{(6)} + x^{(10)} + \cdots\right]$

K-means algorithm

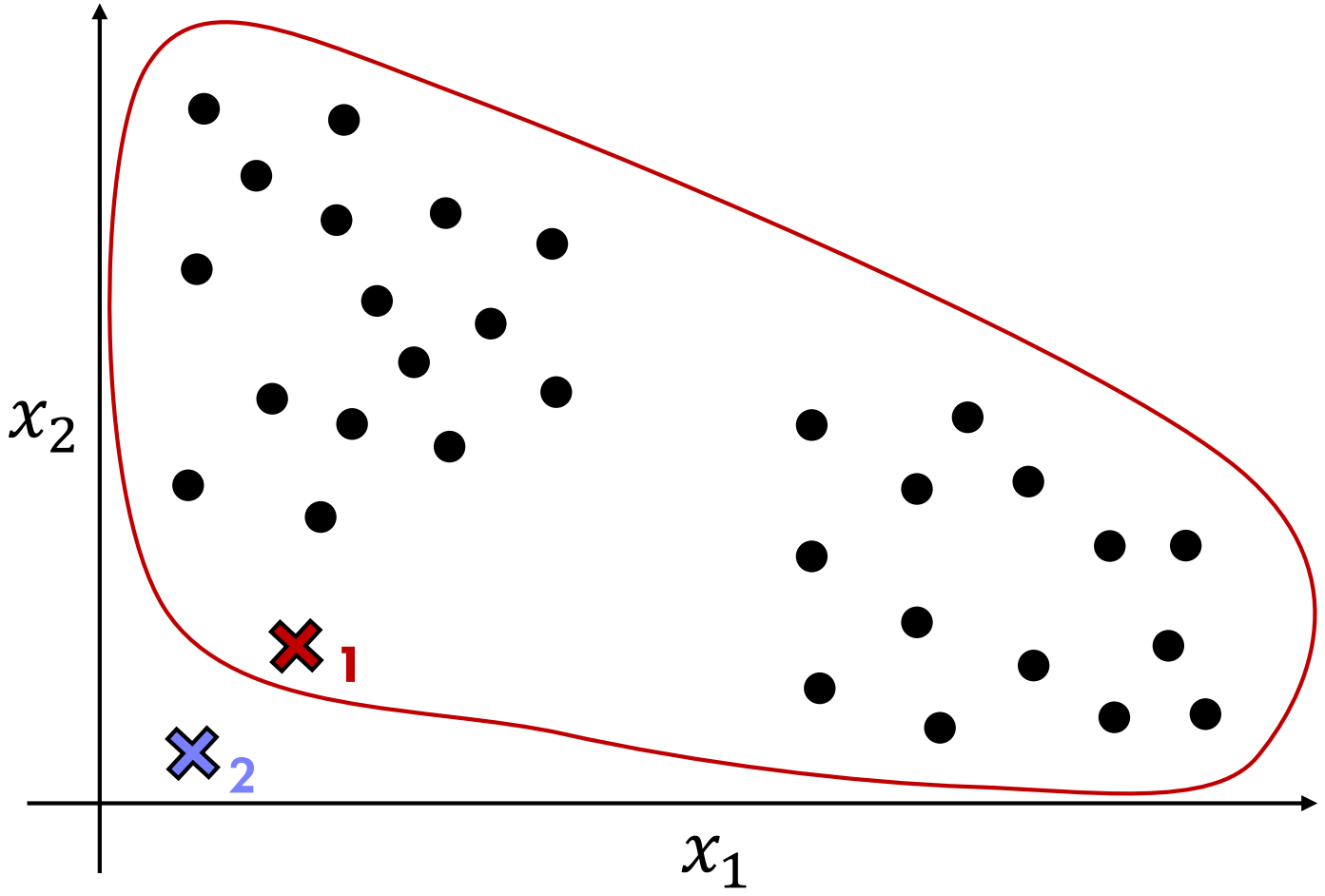


K-means algorithm

In case, one of the centroids doesn't have any point

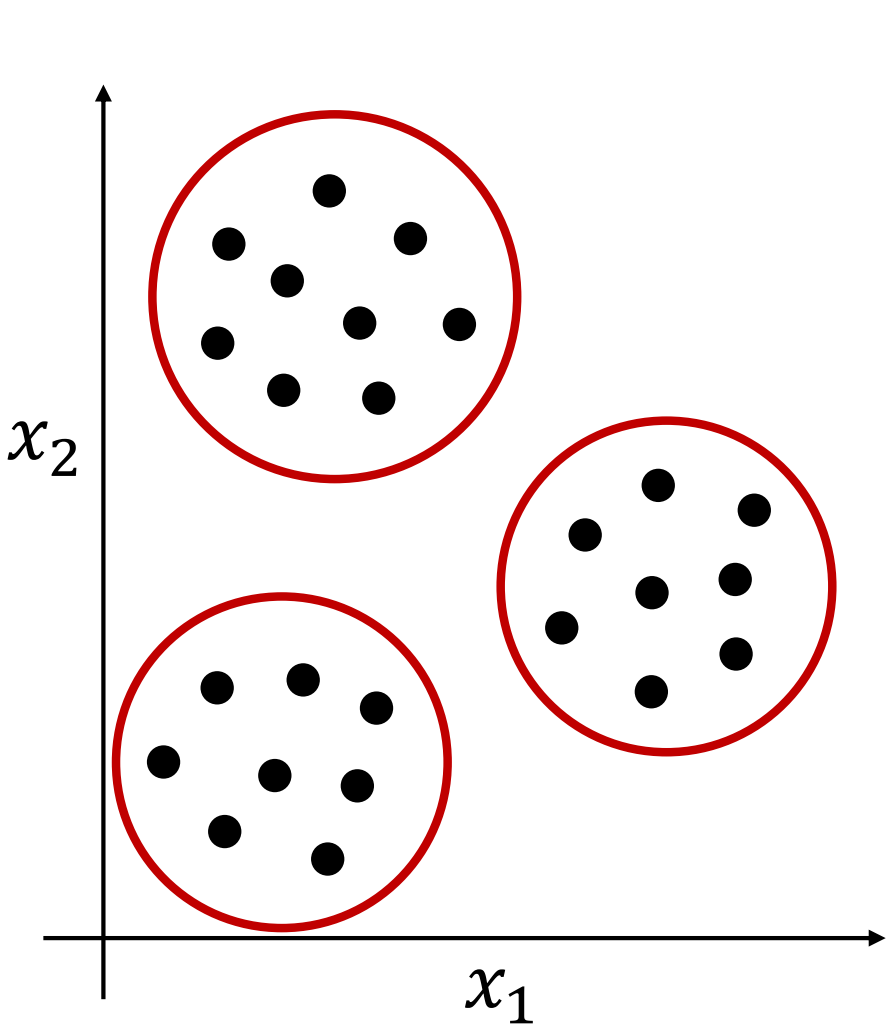
→ simply remove one centroid ($K-1$)

→ or regenerate them randomly

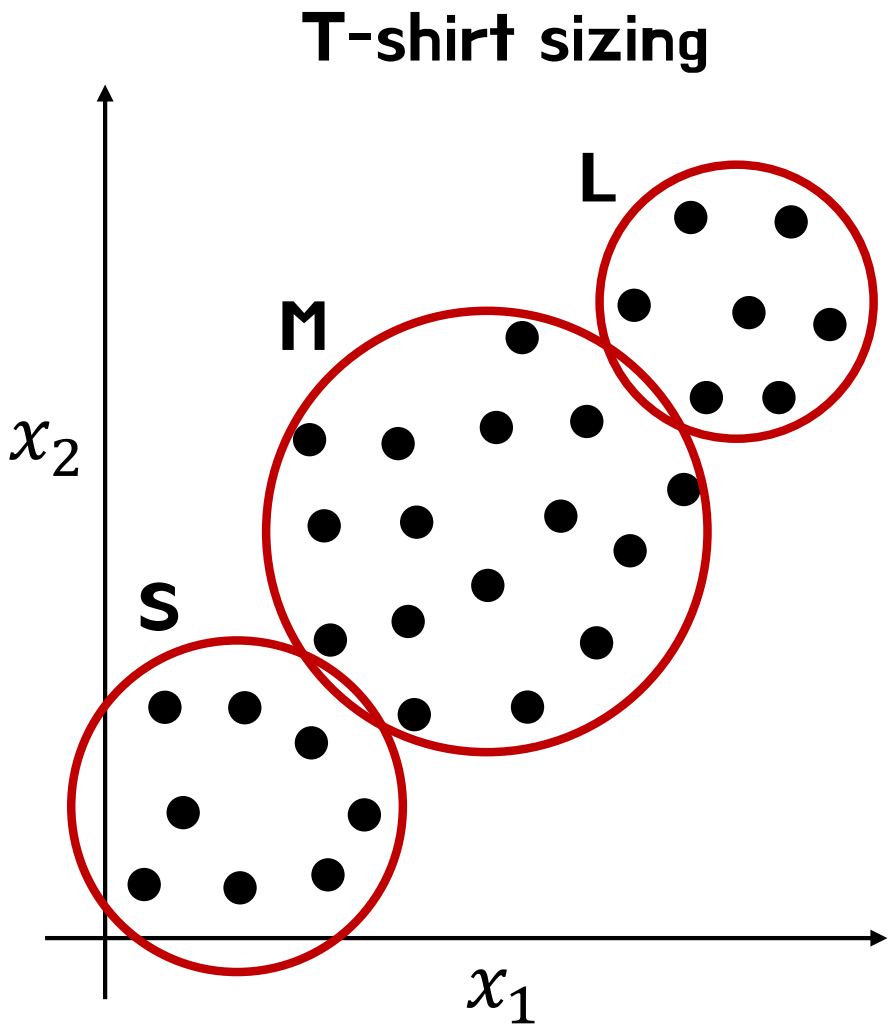


K-means algorithm

Working for points that are not well separated



T-shirt sizing



K-means optimization objective

$c^{(i)}$= index of cluster $(1,2,\ldots,K)$ to which example $x^{(i)}$ is currently assigned

μ_k= cluster centroid k

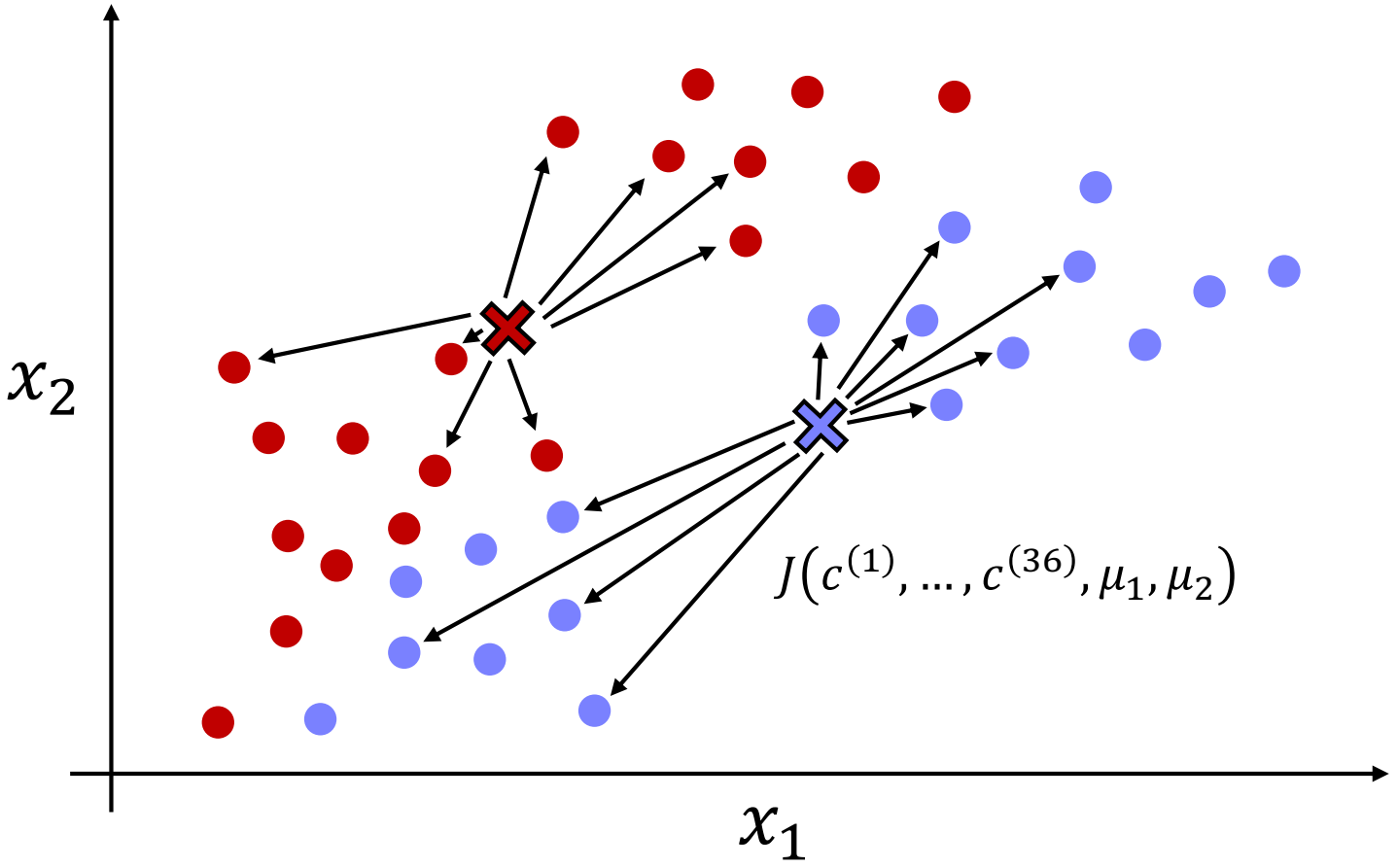
$\mu_{c^{(i)}}$= cluster centroid of cluster to which example $x^{(i)}$ has been assigned

Cost function (or distortion function)

$$J\left(c^{(i)},\ldots,c^{(m)},\mu_1,\ldots,\mu_k\right)=\frac{1}{m}\sum_{i=1}^{m}\left\|x^{(i)}-\mu_c(i)\right\|^2$$

$$\min_{C^{(i)},\ldots,c^{(m)},\mu_1,\ldots,\mu_k} J\left(C^{(i)},\ldots,c^{(m)},\mu_1,\ldots,\mu_k\right)$$

Cost function



Cost function for K-means

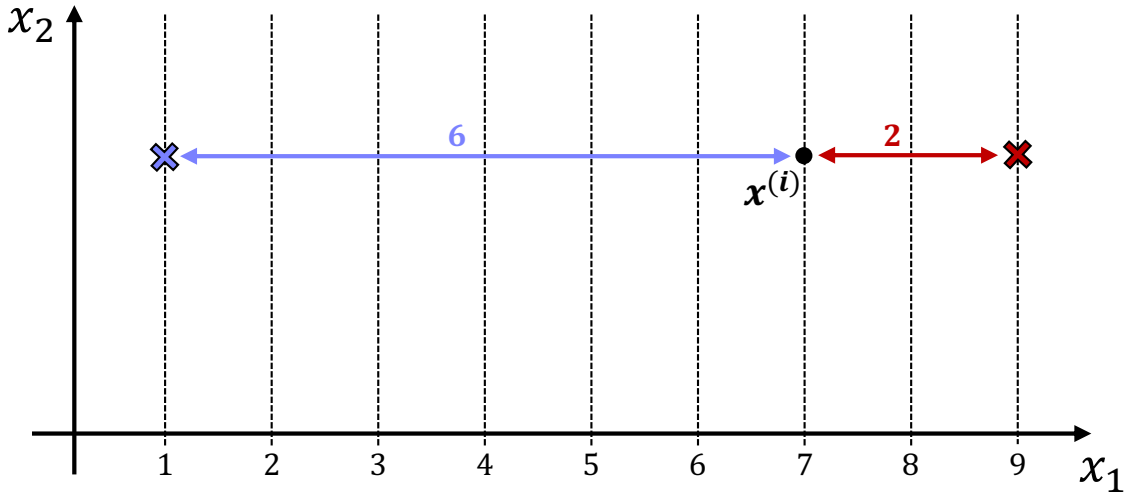
$$J(c^{(i)}, \dots, c^{(m)}, \mu_1, \dots, \mu_k) = \frac{1}{m}\sum_{i=1}^{m}\left\|x^{(i)} - \mu_c(i)\right\|^2$$

Repeat {

```
# Assign points to cluster centroids
for i = 1 to m
    c(i) := index (from 1 to K) of cluster centroid closest to x(i)

# Move cluster centroids
for k = 1 to K
    μk := average (mean) of points assigned to cluster k
```

}



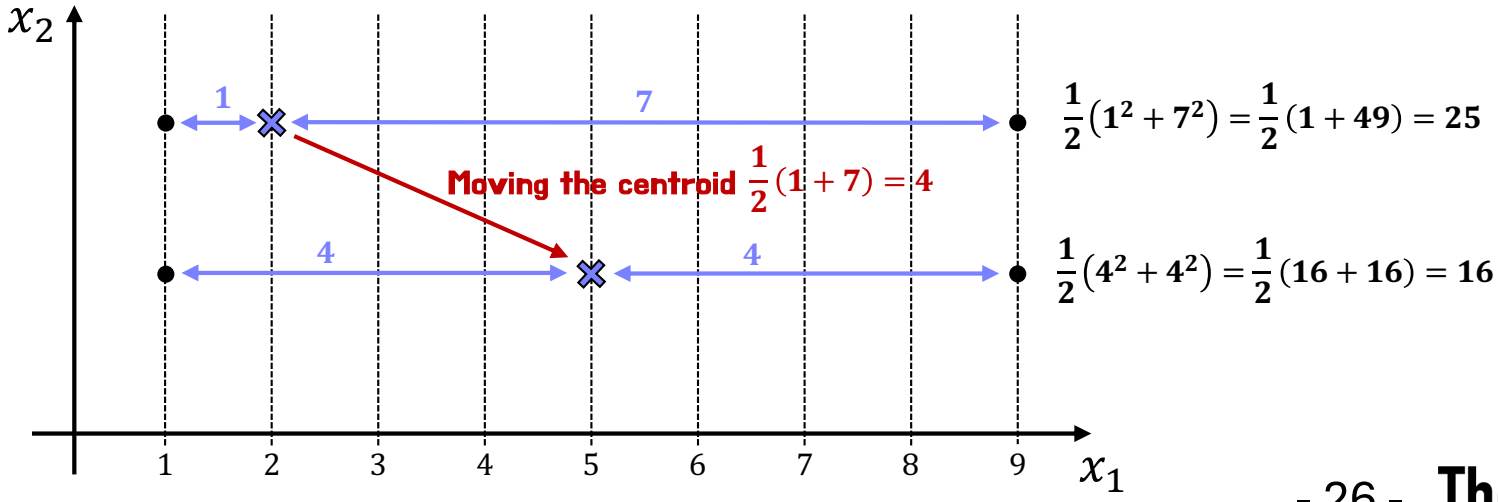
Cost function for K-means

$$J(c^{(i)}, \dots, c^{(m)}, \mu_1, \dots, \mu_k) = \frac{1}{m}\sum_{i=1}^{m}\left\|x^{(i)} - \mu_c(i)\right\|^2$$

Repeat {

```
# Assign points to cluster centroids
for i = 1 to m
    c(i) := index (from 1 to K) of cluster centroid closest to x(i)

# Move cluster centroids
for k = 1 to K
    μk := average (mean) of points assigned to cluster k
}
```



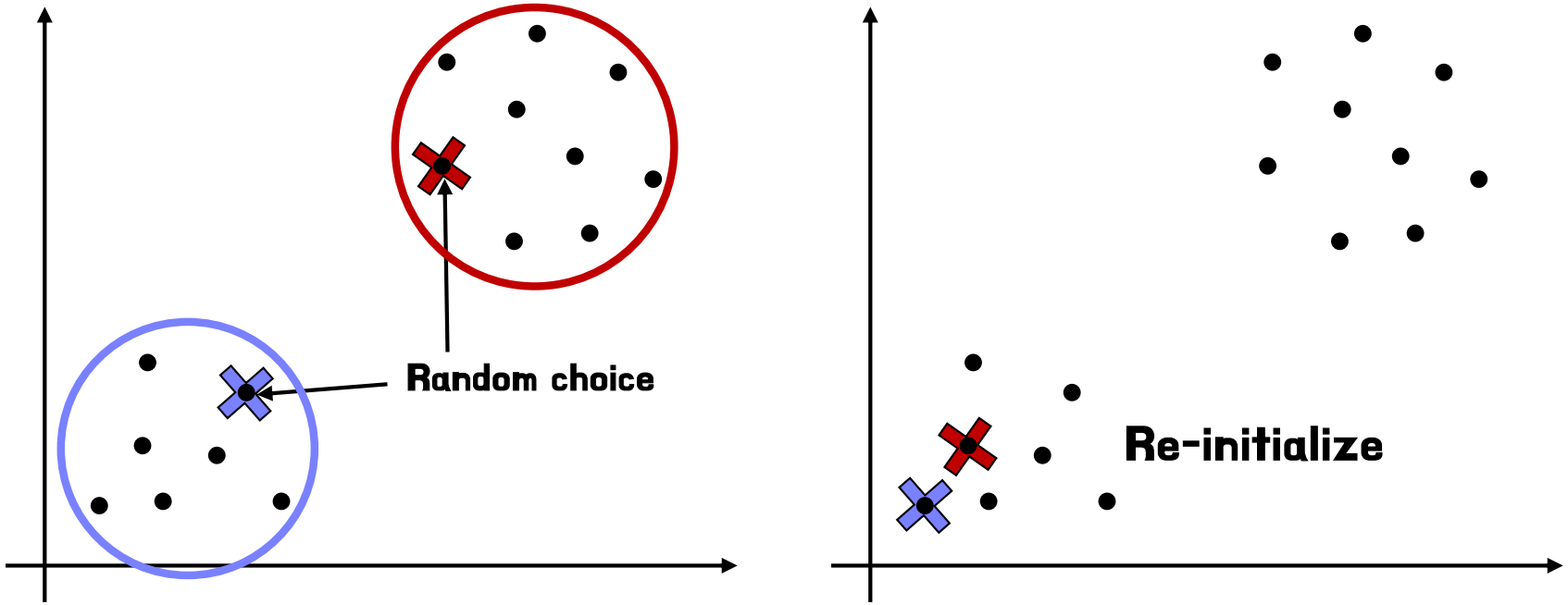
Random initialization

Choose $K < m$

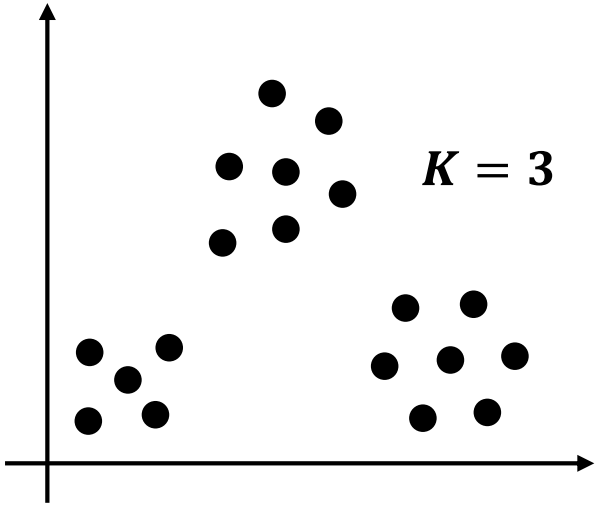
Randomly pick K training examples

Set $\mu_1, \mu_2, \ldots, \mu_k$ equal to these K examples

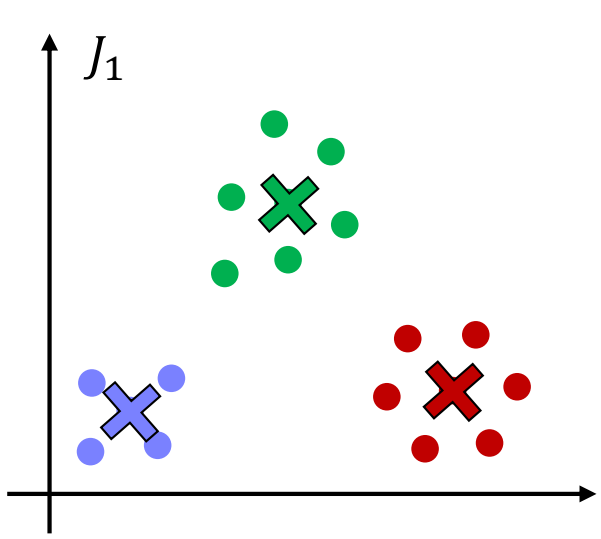
Common way to initialize the K-means optimization

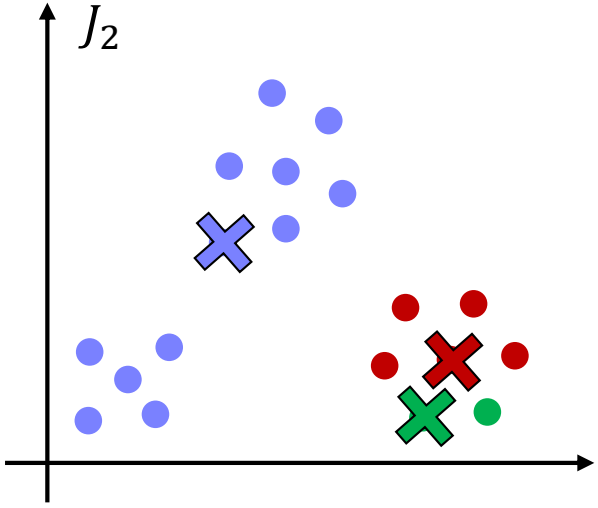


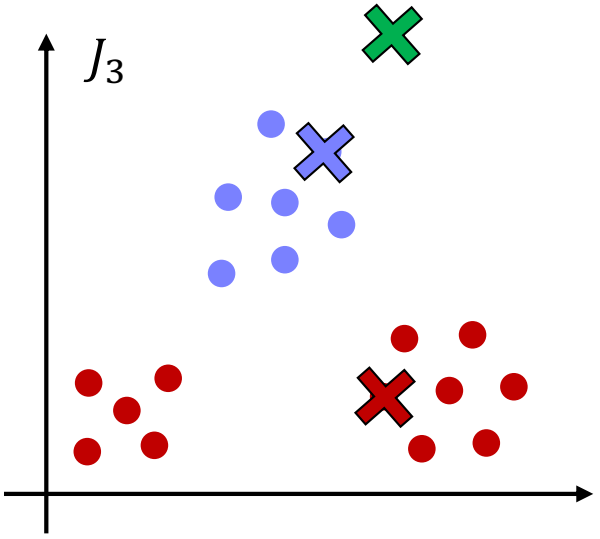
Random initialization



Compare the amount of cost function







Random initialization

$50 - 1,000$ (commonly used)

For $i = 1$ to 100 {

Randomly initialize K-means.

Run K-means. Get $c^{(i)}, \dots, c^{(m)}, \mu_1, \dots, \mu_k$

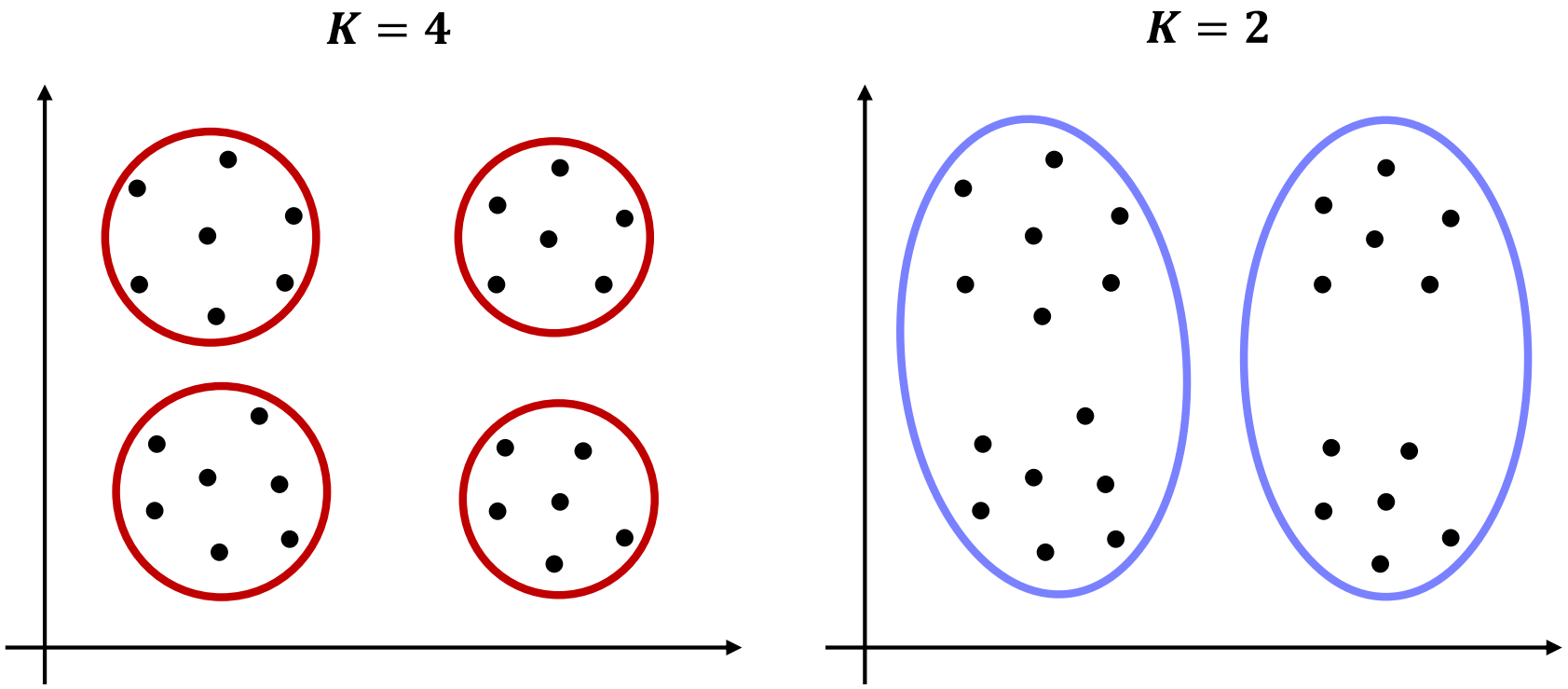
Computer cost function (distortion)

$J\left(c^{(i)}, \dots, c^{(m)}, \mu_1, \dots, \mu_k\right)$

}

Pick set of clusters that gave lowest cost J

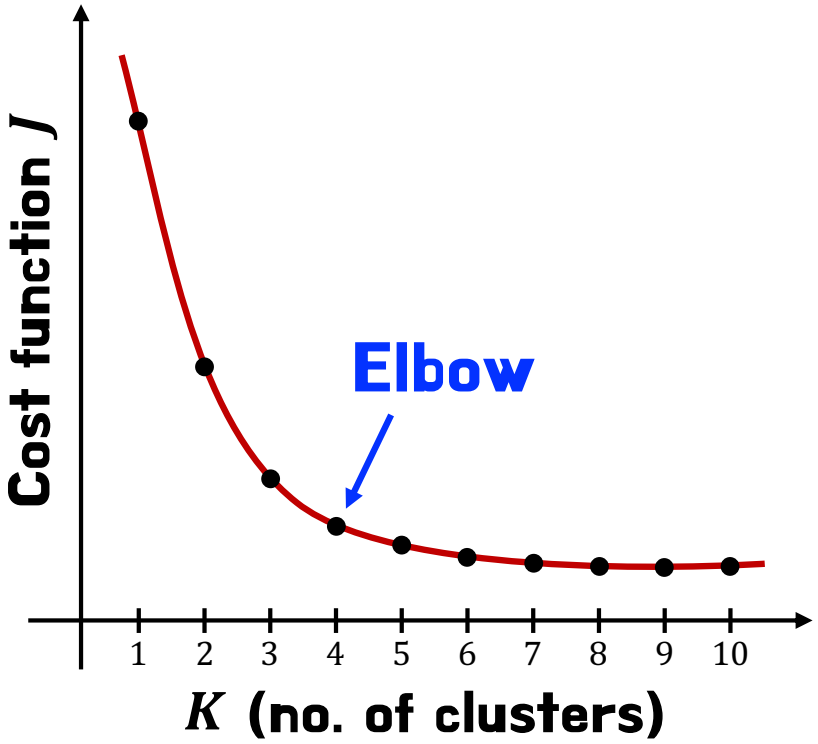
What is the right value of K?

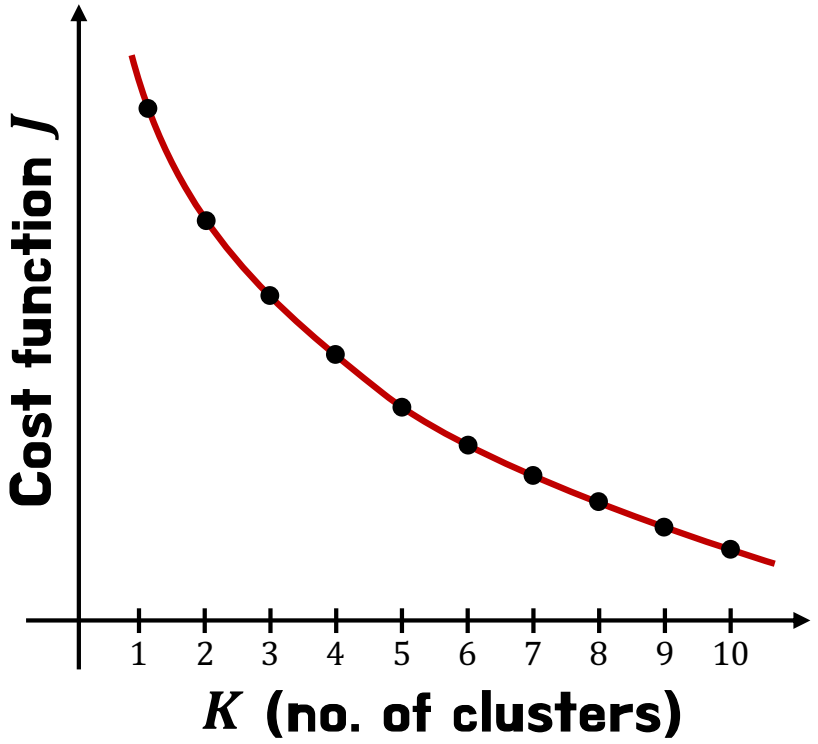


Choosing the value of K

Elbow method - usually not recommended

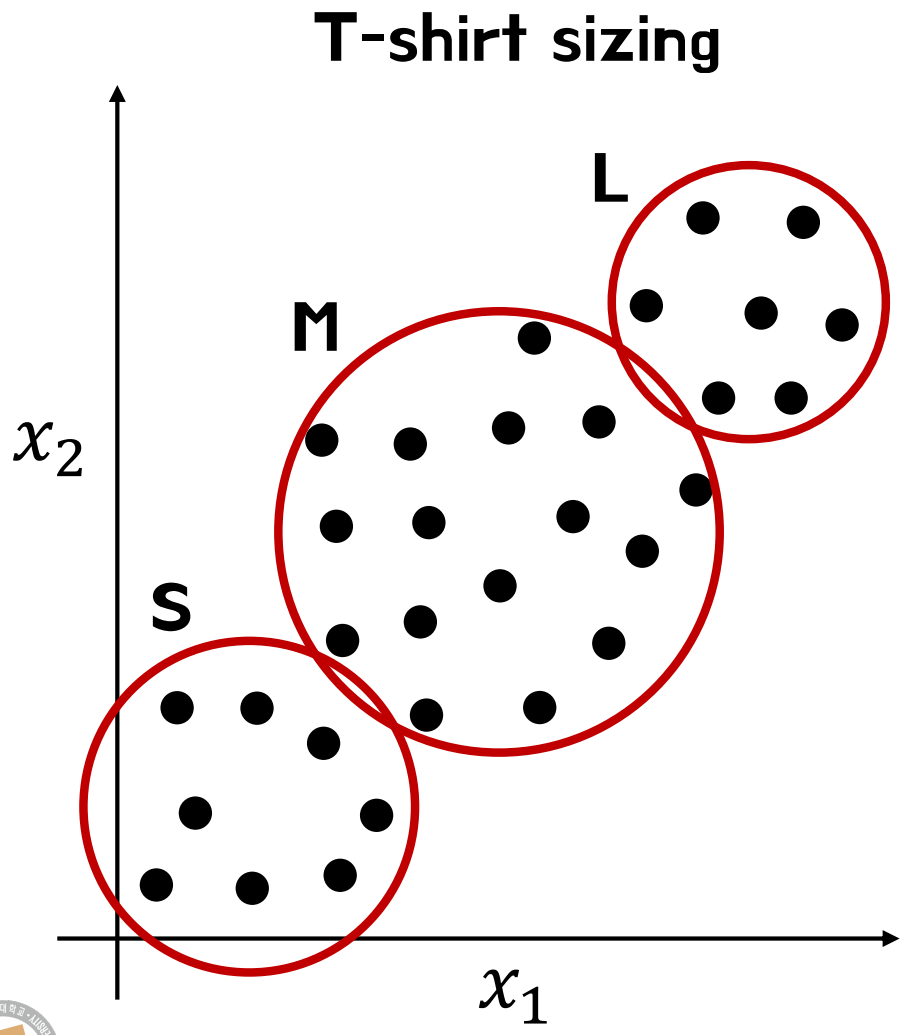
The right "K" is often ambiguous
Don't choose K just to minimize cost J

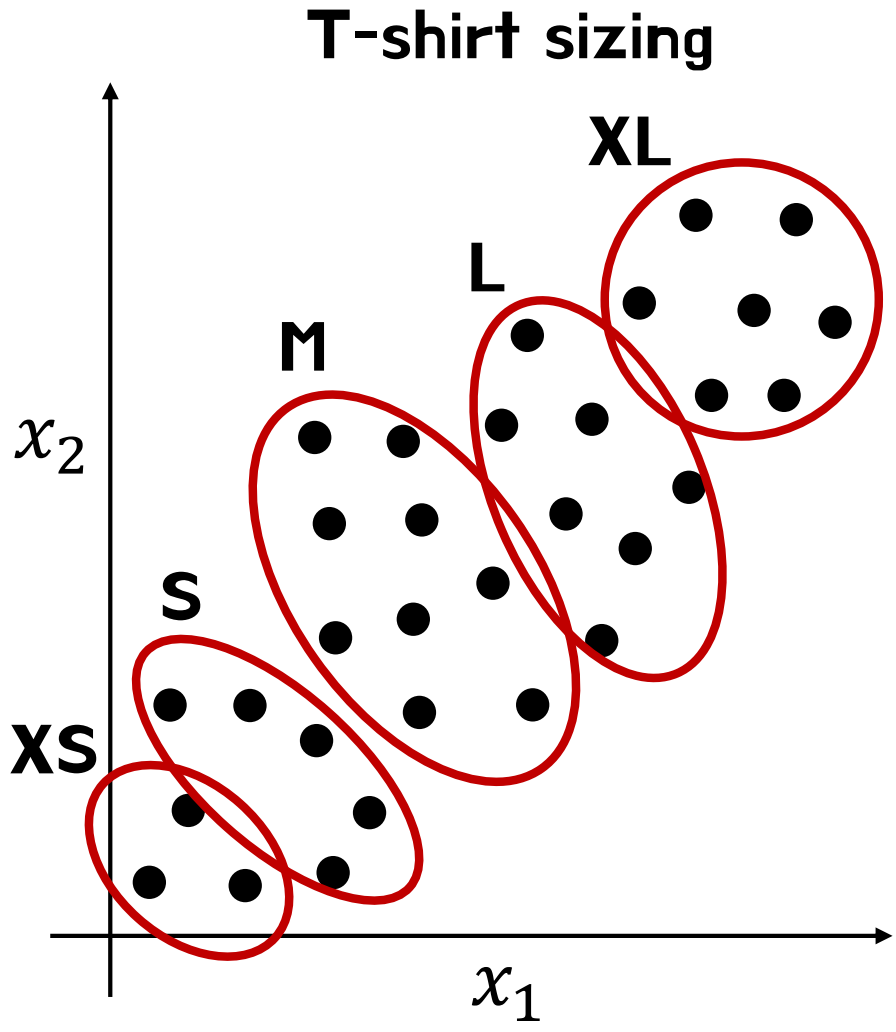




Choosing the value of K

Can be determined by later purpose





Anomaly detection

Anomaly detection example

Aircraft engine features:

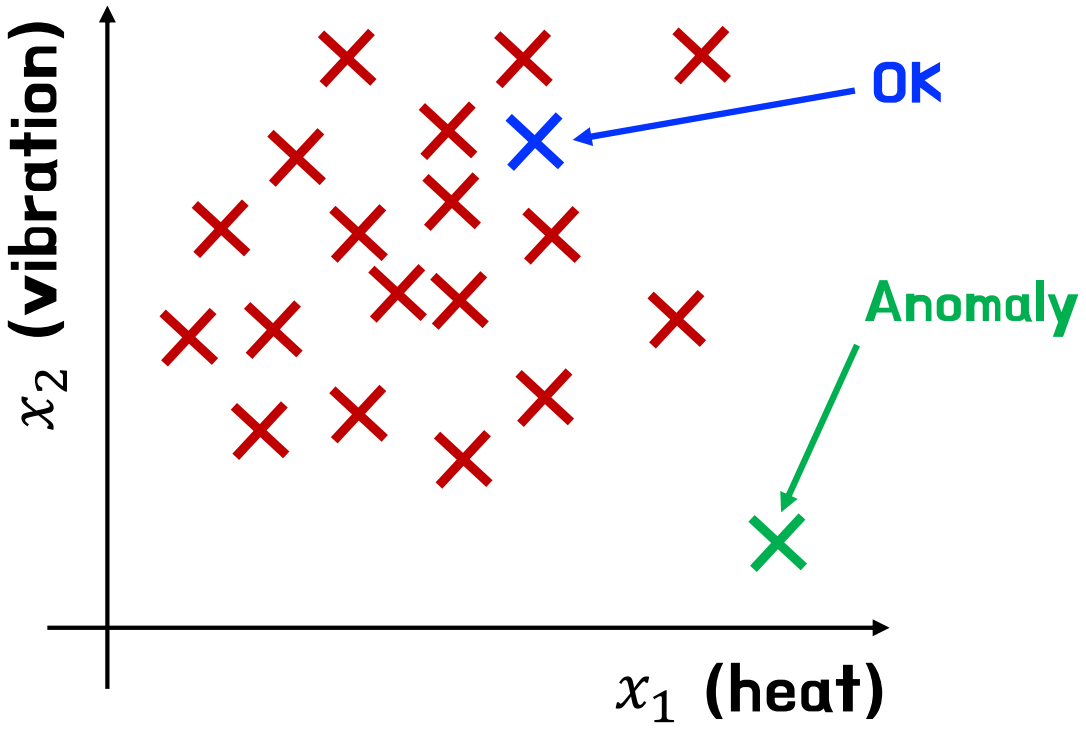
x_1 = heat generated

x_2 = vibration intensity

...

Dataset: $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$

New engine: x_{test}



TheMAC

Density estimation

Dataset: $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$

Probability of x being seen in dataset

Model $p(x)$

Is x_{test} **anomalous ?**

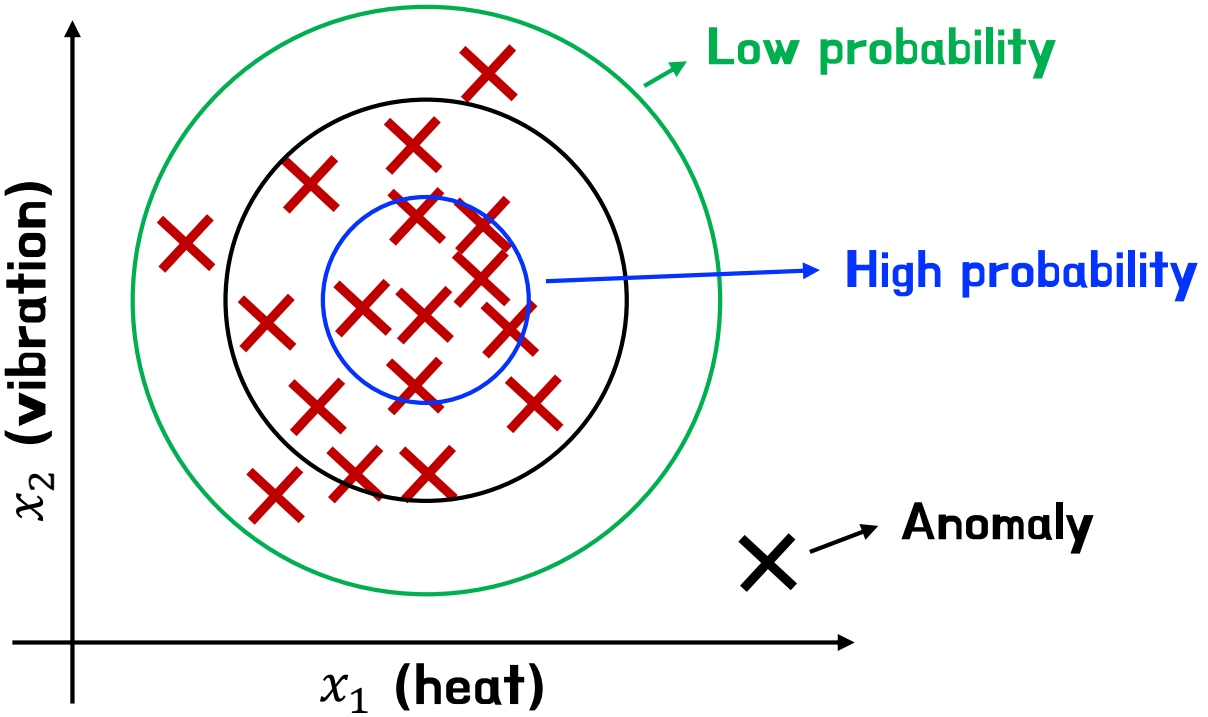
epsilon: small number

$p(x_{test}) \geq \varepsilon$

: **OK (normal)**

$p(x_{test}) < \varepsilon$

: **Anomaly**



Anomaly detection example

Fraud detection:

- $x^{(i)}$ = features of user i's activities
  - How often log in ?
  - How many webpages visited ?
  - Transactions ?
  - Posts ?
  - Typing speed ?
- Model $p(x)$ from data.
- Identify unusual users by checking which have $p(x) < \varepsilon$

Manufacturing:

- $x^{(i)}$ = features of product i
  - Airplane engine
  - Circuit board
  - Smartphone

Monitoring computers in a data center:

$x^{(i)}$ = features of product i

- x_1 = memory use,
- x_2 = number of disk accesses/sec,
- x_3 = CPU load,
- x_4 = CPU load/network traffic,

Gaussian (Normal) distribution

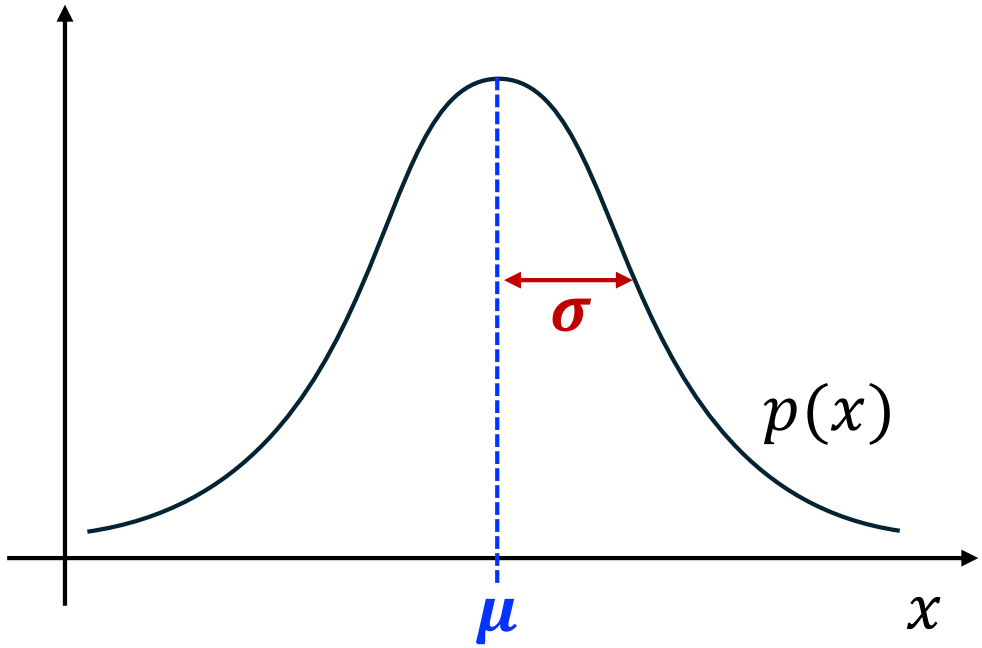
Say x is a number.

Probability of x is determined by a Gaussian with mean μ, variance σ^2.

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

σ : standard deviation

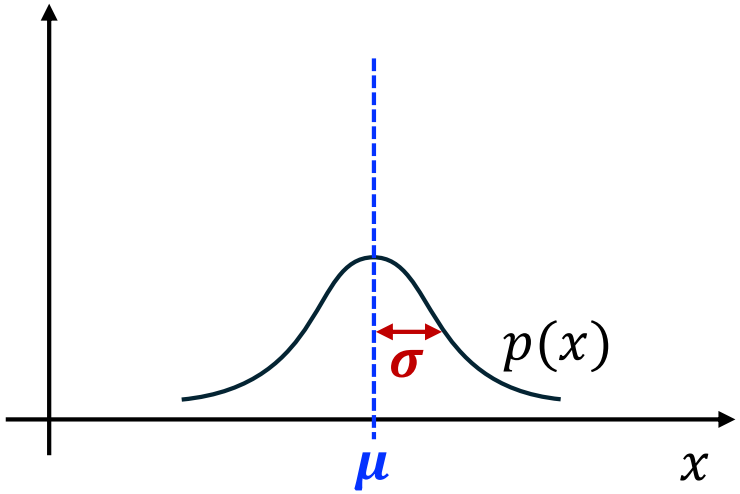
σ^2 : variance



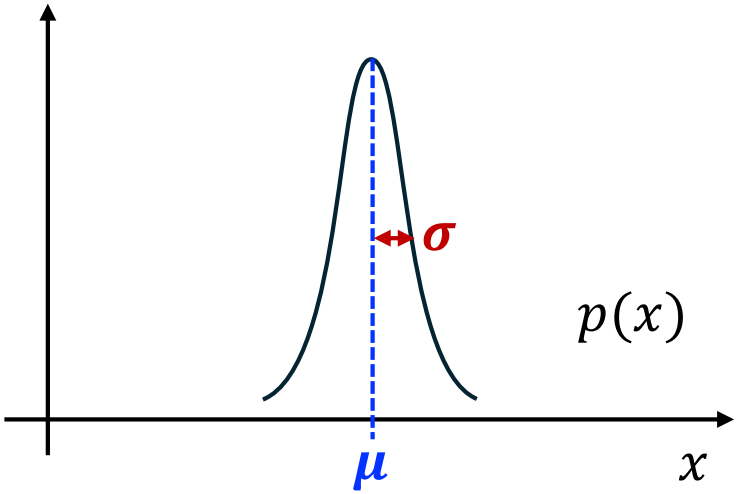
TheMAC

Gaussian distribution example

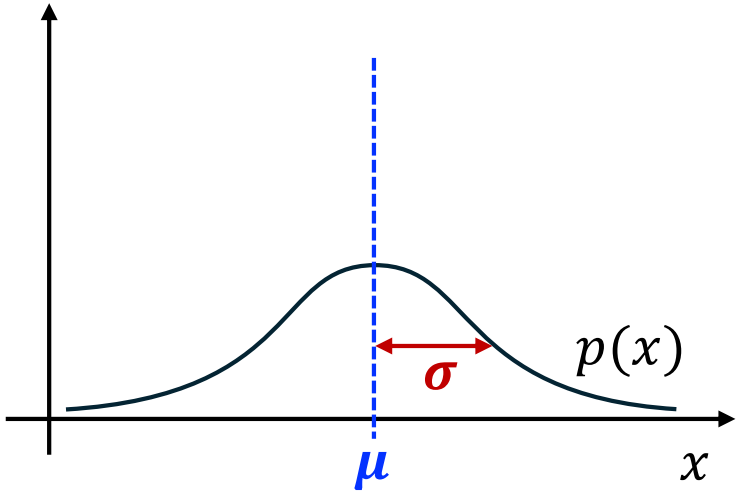
$\mu = 0, \sigma = 1$



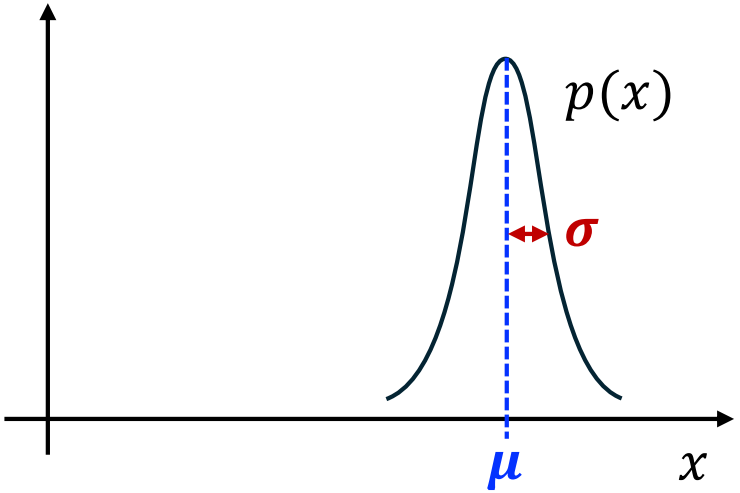
$\mu = 0, \sigma = 0.5$



$\mu = 0, \sigma = 2$



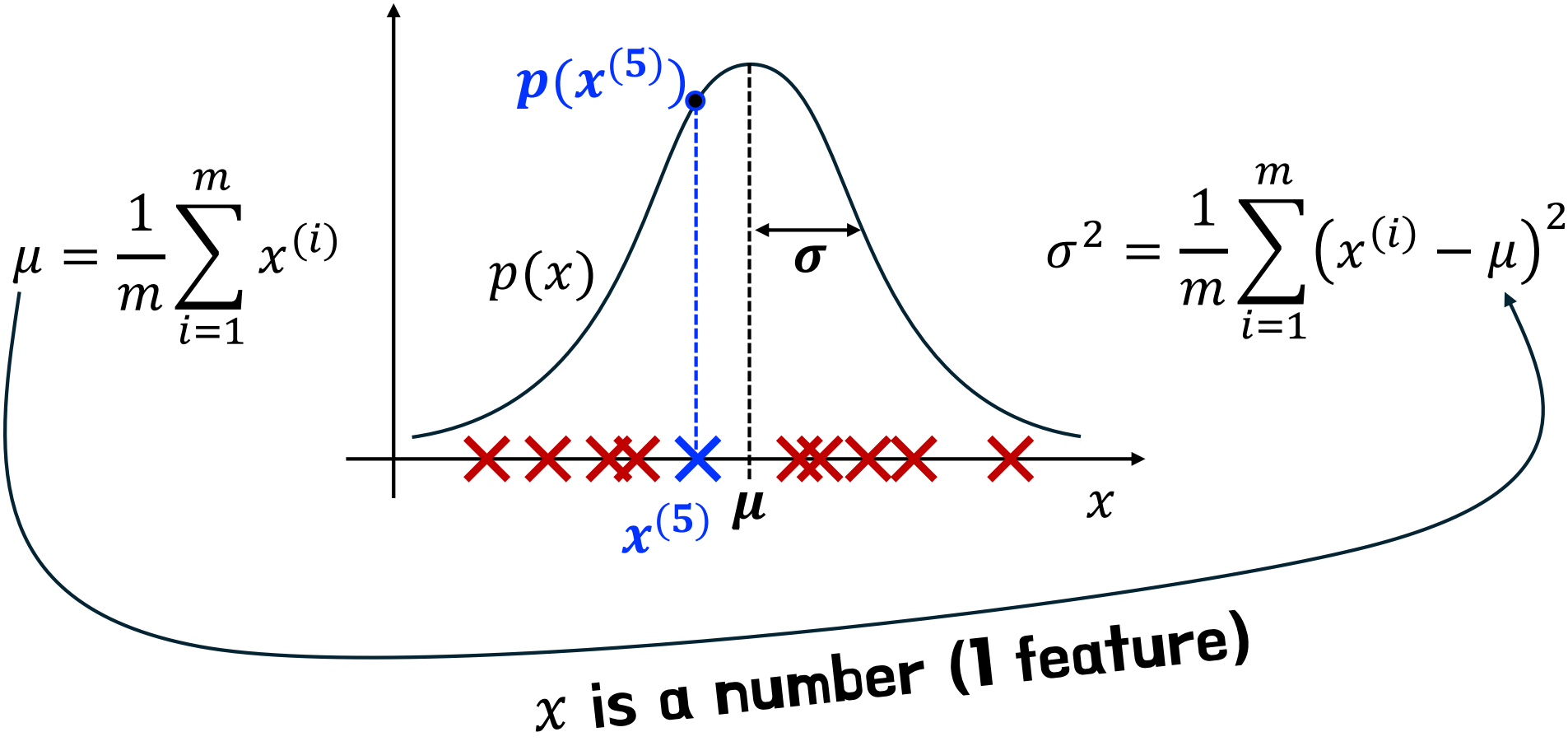
$\mu = 3, \sigma = 0.5$



TheMAC

Parameter estimation

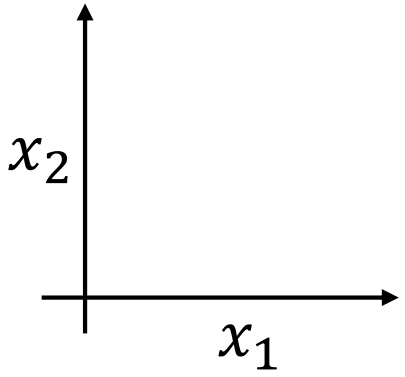
Dataset: $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$



Density estimation

Training set: $\{\vec{x}^{(1)}, \vec{x}^{(2)}, \ldots, \vec{x}^{(m)}\}$

Each example $\vec{x}^{(i)}$ **has** n **features**



$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$p(\vec{x}) = p(x_1; \mu_1, \sigma_1^2) * p(x_2; \mu_2, \sigma_2^2) * p(x_3; \mu_3, \sigma_3^2) * \cdots * p(x_n; \mu_n, \sigma_b^2)$$

$$= \prod_{j=1}^{n} p(x_j; \mu_j, \sigma_j^2)$$

$p(x_1 = high\ temp) =$ **1/10**

$p(x_2 = high\ vib.) =$ **1/20**

$p(\vec{x}) = p(x_1) * p(x_2) =$ **1/200**

Anomaly detection algorithm

1. Choose n features x_i that you think might be indicative of anomalous examples

2. Fit parameters $\mu_1, \dots, \mu_n, \sigma_1^2, \dots, \sigma_n^2$

$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)} \qquad \sigma_j^2 = \frac{1}{m}\sum_{i=1}^{m}\left(x_j^{(i)} - \mu_j\right)^2$$

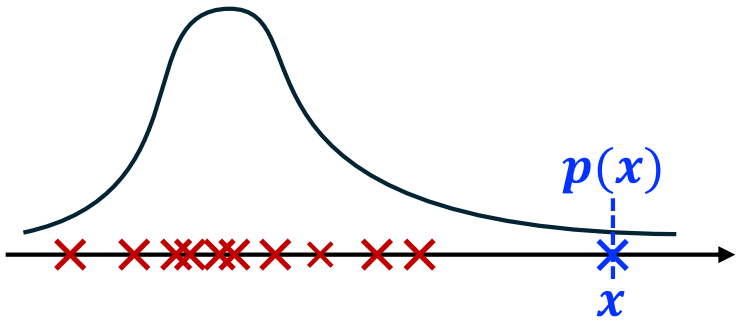
Vectorized formula

$$\vec{\mu} = \frac{1}{m}\sum_{i=1}^{m} \vec{x}^{(i)} \qquad \vec{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{bmatrix}$$

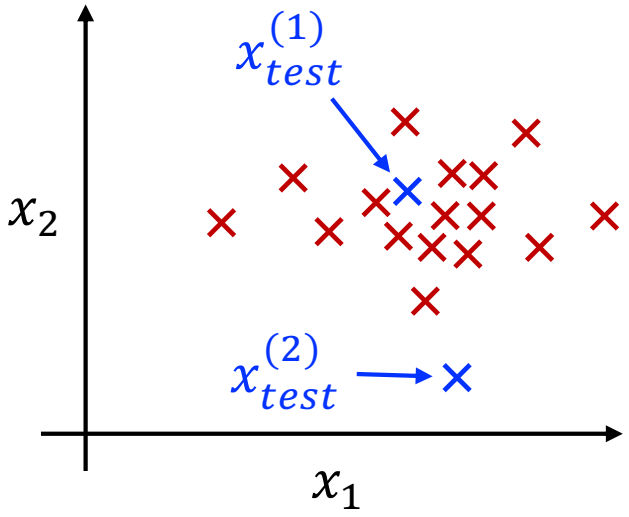
3. Given new example x, compute $p(x)$:

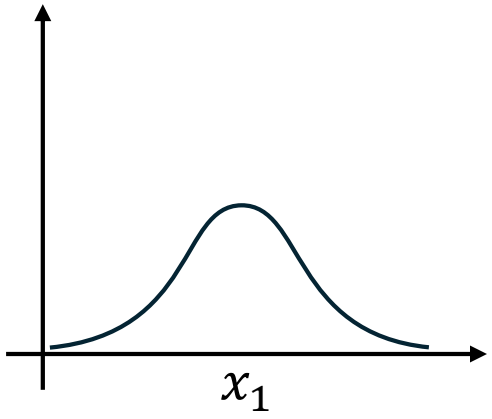
$$p(x) = \prod_{j=1}^{n} p\left(x_j; \mu_j, \sigma_j^2\right) = \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_j} e^{\frac{(x_j-\mu_j)^2}{2\sigma_j^2}}$$

Anomaly if $p(x) < \varepsilon$

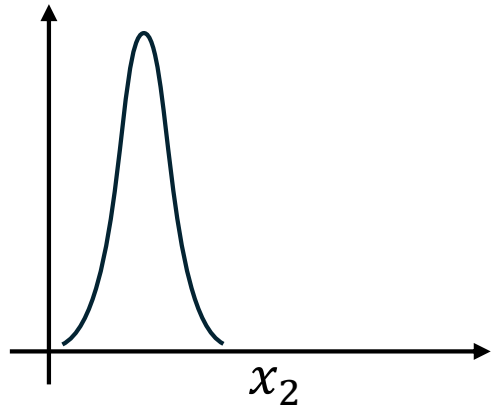


Anomaly detection example

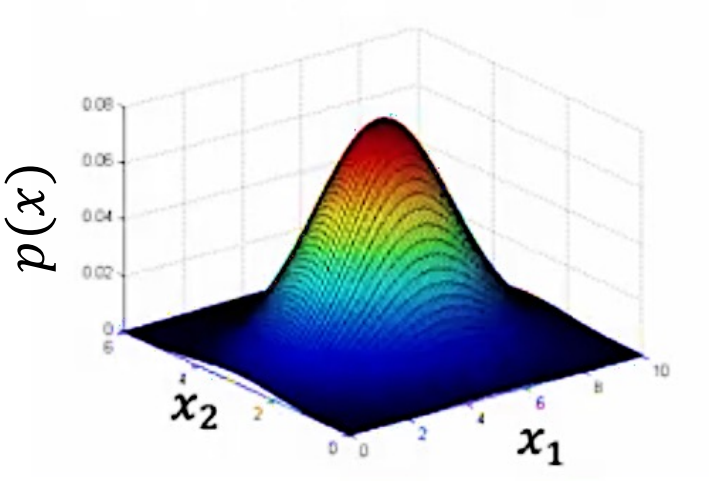




$\mu_1 = 5, \sigma_1 = 2$



$\mu_2 = 3, \sigma_2 = 1$



$\varepsilon = 0.02$

$p\left(x_{test}^{(1)}\right) = 0.0426$ **OK**

$p\left(x_{test}^{(2)}\right) = 0.0021$ **Anomaly**

The importance of real-number evaluation

- When developing a learning algorithm (choosing features, etc.), making decision is much easier if we have a way of evaluating our learning algorithm.

- Assume we have some labeled data, of anomalous and non-anomalous examples.
  $y = 1$ $y = 0$

- Training set: $x^{(1)}, x^{(2)}, \dots, x^{(m)}$ (assume normal examples/not anomalous)
  $y = 0$ for all trining examples

- Cross validation set: $\left(x_{cv}^{(1)}, y_{cv}^{(1)}\right), \dots, \left(x_{cv}^{(m_{cv})}, y_{cv}^{(m_{cv})}\right)$
- Test set: $\left(x_{test}^{(1)}, y_{test}^{(1)}\right), \dots, \left(x_{test}^{(m_{test})}, y_{test}^{(m_{test})}\right)$

Include a few anomalous examples ($y = 1$)

Mostly normal examples ($y = 0$)

Aircraft engines monitoring example

10,000 : Good (normal) engines

20 : Flawed engines (anomalous)

Training set : 6,000 good engines ($y = 0$) → Train algorithm on training set

CV : 2,000 good engines ($y = 0$), 10 anomalous ($y = 1$) → Tune ε and x_j

Test : 2,000 good engines ($y = 0$), 10 anomalous ($y = 1$)

If very few labeled anomalous examples,

Alternative: No test set

Training set : 6,000 good engines ($y = 0$)

CV : 4,000 good engines ($y = 0$), 2 anomalous ($y = 1$)

But, higher risk of overfitting

Algorithm evaluation

- Fit model $p(x)$ on training set $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$
- On a cross validation/test example x, predict

$$y = \begin{cases} 1 & if\ p(x) < \varepsilon\ (anomaly) \\ 0 & if\ p(x) \geq \varepsilon\ (normal) \end{cases}$$

- Possible evaluation metrics:
    - True positive, false positive, false negative, true negative
    - Precision/Recall
    - F_1-score

- Use cross validation set to choose parameter ε

Anomaly detection vs. supervised learning

Anomaly detection

- Very small number of positive examples ($y = 1$). (0-20 is common)
- Large number of negative examples ($y = 0$).
- Many different "types" of anomalies.
- Hard for any algorithm to learn from positive examples what the anomalies look like; future anomalies may look nothing like any of the anomalous examples we've seen so far. Fraud

Supervised learning

- Large number of positive and negative examples. (20 positive examples)
- Enough positive examples for algorithm to get a sense of what positive examples are like, future positive examples likely to be similar to ones in training set. Spam

Anomaly detection vs. supervised learning

Anomaly detection

- Fraud detection
- Manufacturing - finding new previously unseen defects in manufacturing.

  (e. g. aircraft engines)
- Monitoring machines in a data center

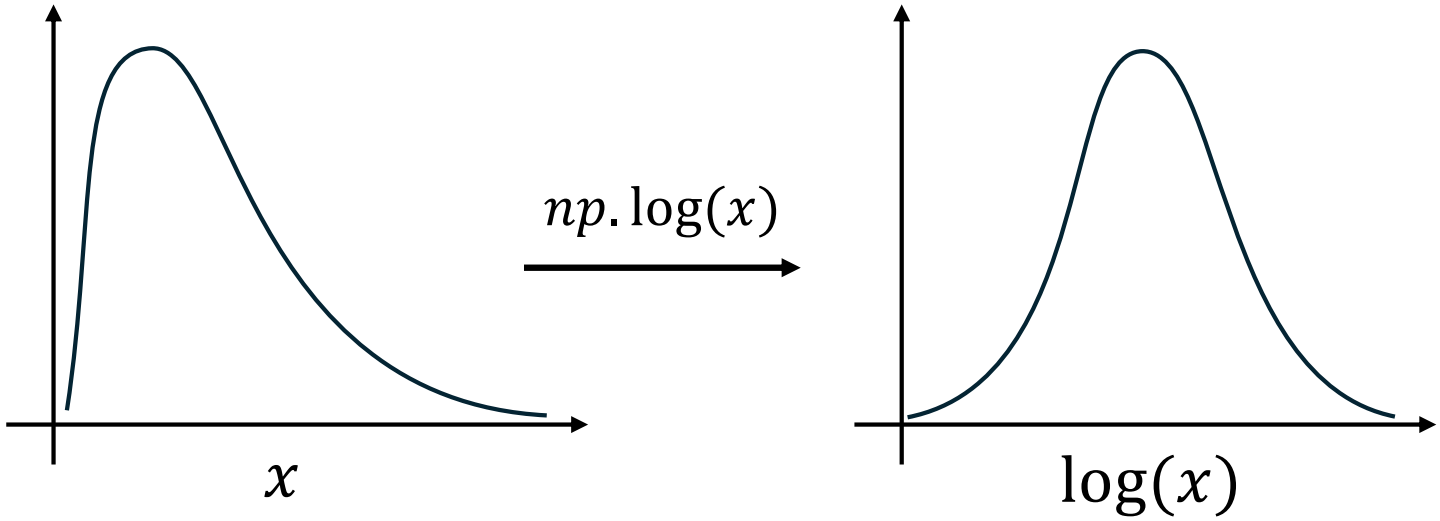
⋮

Supervised learning

- E-mail spam classification
- Manufacturing - finding known, previously seen defects
- Weather prediction (sunny/rainy/etc.)
- Diseases classification

⋮

Non-gaussian features



Other ways

$$
\begin{cases}
x_1 \longrightarrow \log(x_1) \\
x_2 \longrightarrow \log(x_2 + C) \\
x_3 \longrightarrow \sqrt{x_3} \\
x_4 \longrightarrow x_4^{1/3}
\end{cases}
$$

Error analysis for anomaly detection

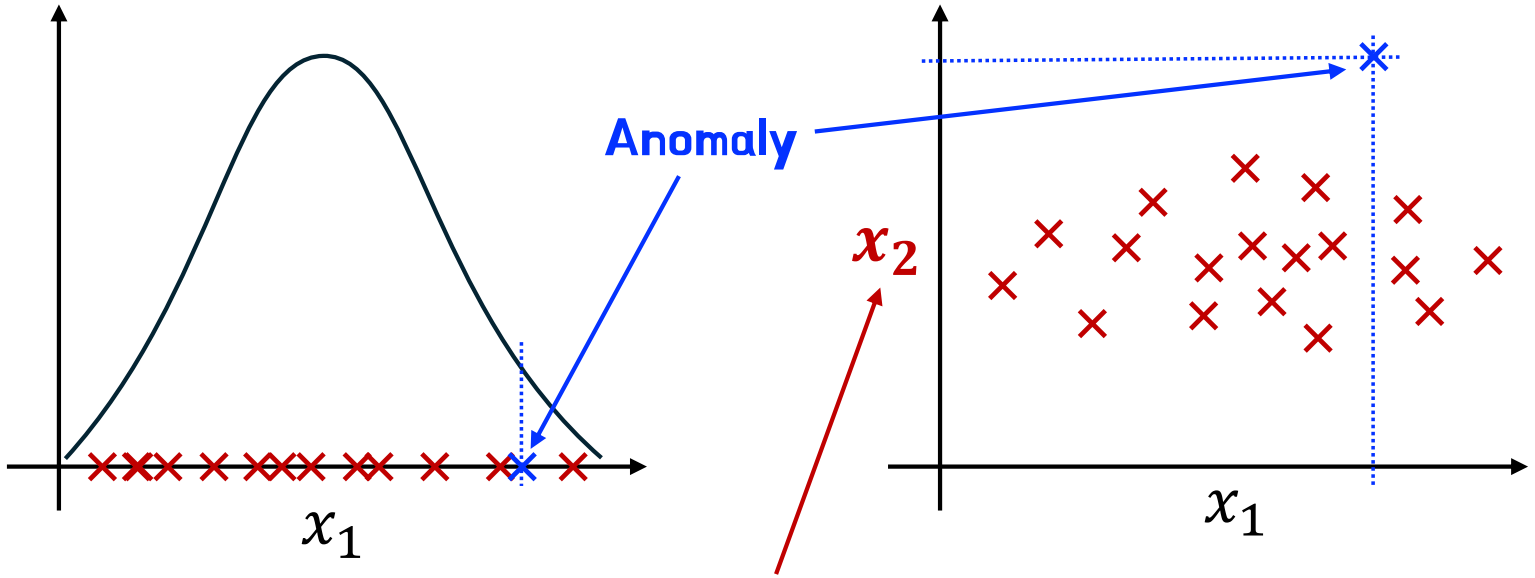
Want $p(x) > \varepsilon$: large for normal examples x.

$p(x) \leq \varepsilon$: small for anomalous examples x.

Most common problem:

$p(x)$ is comparable for normal and anomalous examples.

($p(x)$ is large for both)



Need to find the better feature

TheMAC

Monitoring computers in a data center

- Choose features that might take on unusually large or small values in the event of an anomaly.

$x_1 =$ memory use of computer

$x_2 =$ number of disk accesses/sec

$x_3 =$ CPU load

$x_4 =$ network traffic

$x_5 =$ CPU load / network traffic

$x_6 =$ (CPU load)^2 / network traffic

- Deciding feature choice based on $p(x)$
  - Large for normal examples;
  - Becomes small for anomaly in the cross validation set

Summary

- Clustering (Unsupervised Learning)

  Group unlabeled data into meaningful clusters based on similarity.

- K-means Algorithm

  Iteratively assign points to nearest centroids and update centroids to minimize distortion (cost function).

- Choosing K & Initialization

  Proper K selection depends on purpose; random initialization requires multiple runs.

- Anomaly Detection

  Model normal data distribution and flag low-probability examples as anomalies.

- Anomaly Detection vs. Supervised Learning

  Use anomaly detection when anomalies are rare or unseen; use supervised learning when sufficient labeled data exists.