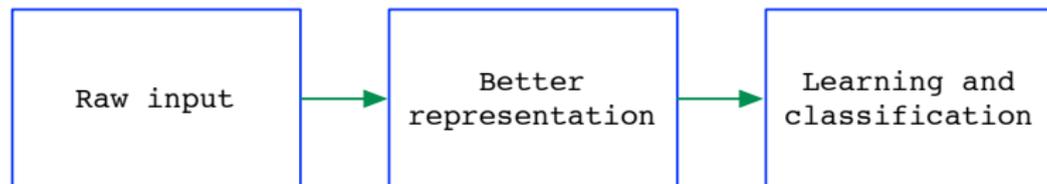


# Apprendimento della rappresentazione: Clustering e Principal Component Analysis

Vincenzo Bonifaci

IN550 – Machine Learning

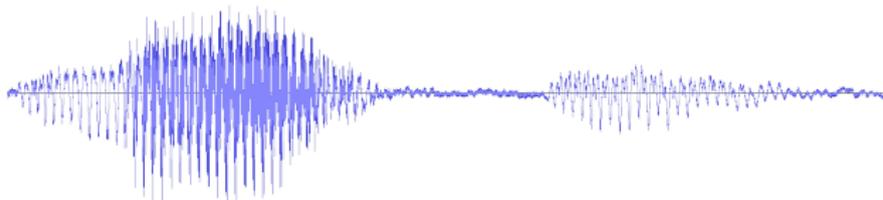
# Apprendimento della rappresentazione



Una buona rappresentazione semplifica l'apprendimento:

- Cattura correttamente i **gradi di libertà** presenti nei dati
- Cattura strutture rilevanti su **varie scale**
- Maschera informazioni **rumorose** o **irrilevanti**

# Gradi di libertà



Rappresentazione tipica del parlato:

- Si fa scorrere una finestra sul segnale audio
- Si calcolano svariati filtri su ogni finestra
- Molti filtri  $\Rightarrow$  alto numero di dimensioni

Eppure, l'input proviene da un sistema fisico con **pochi** gradi di libertà

# Struttura multiscala



A vari livelli ci sono strutture ricorrenti

# Obiettivi dell'apprendimento della rappresentazione

**Obiettivo** (informale): apprendere i gradi di libertà e la struttura multiscala di una distribuzione partendo da campioni di dati **non etichettati**

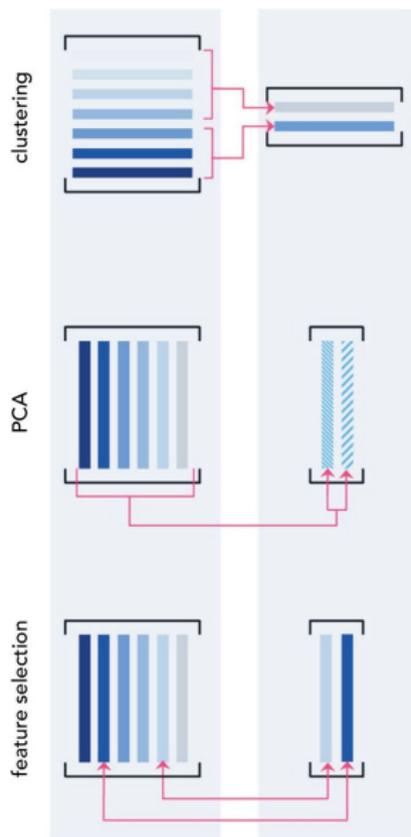
Esploreremo i seguenti approcci:

- Clustering
- Proiezioni lineari

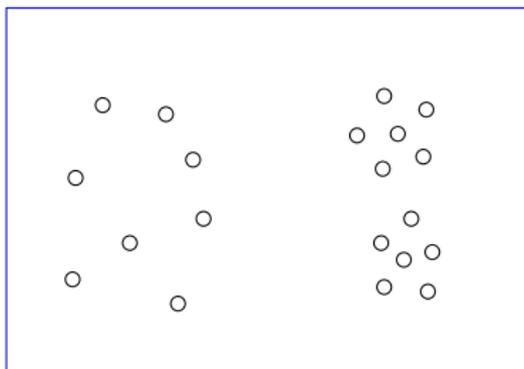
È una tipologia di apprendimento **non supervisionato** perché non ci sono variabili di uscita (etichette), né predizioni

L'apprendimento della rappresentazione può essere usato prima di applicare un metodo supervisionato, per migliorarne i risultati, o semplicemente come modo di **esplorare i dati**

# Feature selection, proiezioni lineari e clustering a confronto



# Clustering in $\mathbb{R}^d$



Due comuni utilizzi del clustering:

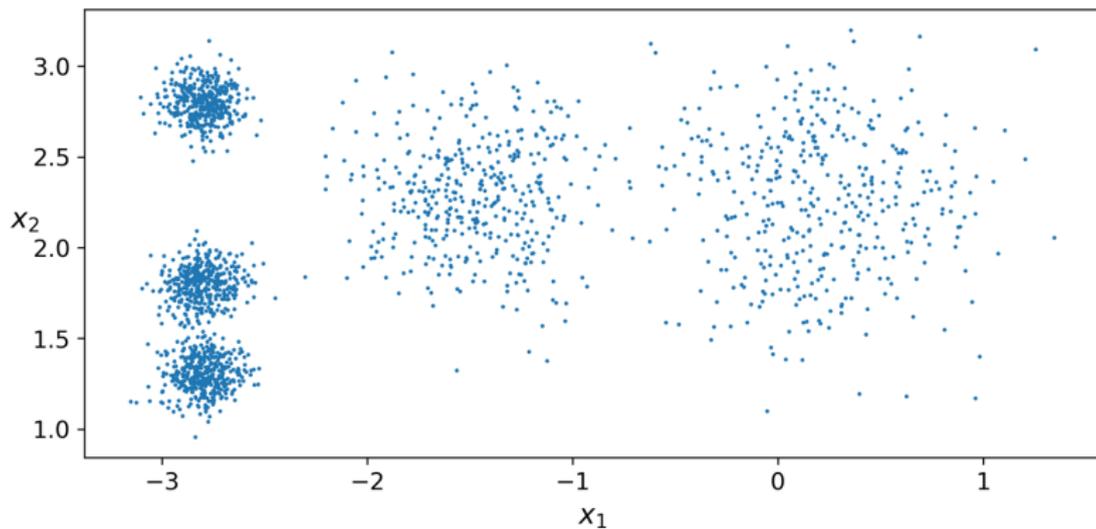
- *Quantizzazione vettoriale:*

Trovare un insieme finito di rappresentanti che “coprono bene” dei dati altamente multidimensionali

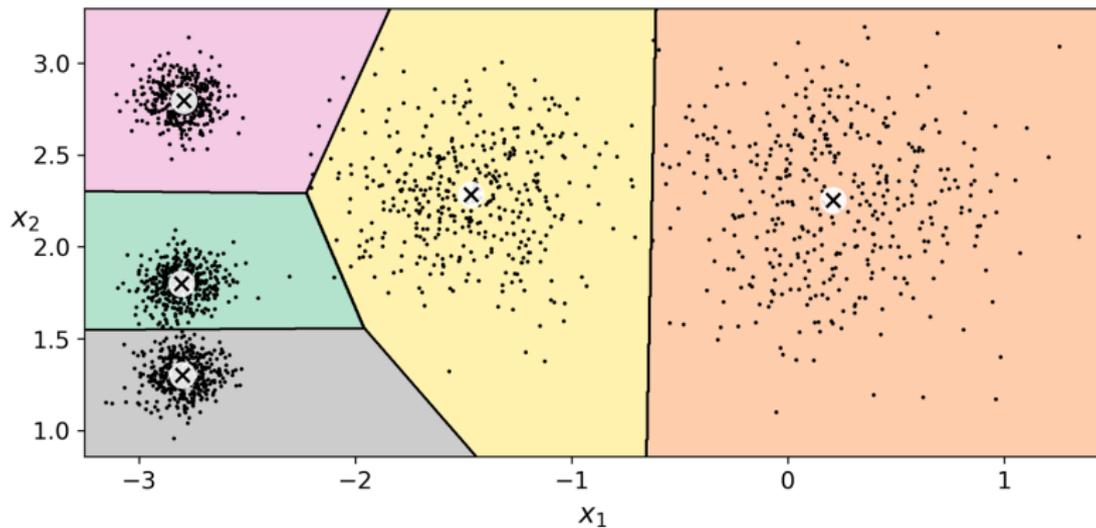
- *Ricerca di struttura significativa nei dati:*

Identificare raggruppamenti significativi nei dati

# Esempio



# Esempio



# Due approcci al clustering

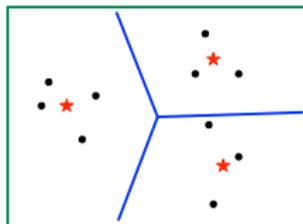
Qui discuteremo due approcci al clustering:

- Clustering  $k$ -means
- Clustering gerarchico

## Il problema di ottimizzazione $k$ -means

- Input: punti  $x^{(1)}, \dots, x^{(m)} \in \mathbb{R}^d$ ; intero  $k$
- Output: “*Centri*”, o rappresentanti,  $\mu^{(1)}, \dots, \mu^{(k)} \in \mathbb{R}^d$
- Obiettivo: minimizzare la distanza quadratica media tra i punti e i loro rappresentanti più vicini:

$$\text{costo}(\mu^{(1)}, \dots, \mu^{(k)}) = \sum_{i=1}^m \min_j \left\| x^{(i)} - \mu^{(j)} \right\|^2$$



I centri partizionano  $\mathbb{R}^d$  in  $k$  regioni convesse

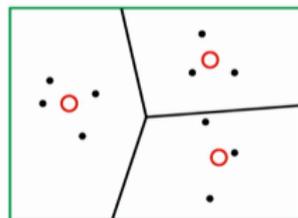
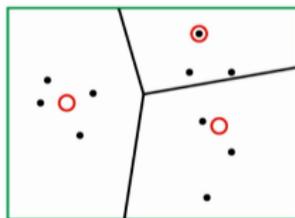
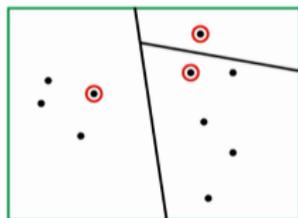
La regione  $j$  consiste di tutti i punti il cui centro più vicino è  $\mu^{(j)}$

# L'algoritmo di Lloyd per $k$ -means

Il problema del  $k$ -means è NP-arduo! L'**euristica** più usata è la seguente

## Algoritmo di Lloyd per $k$ -means

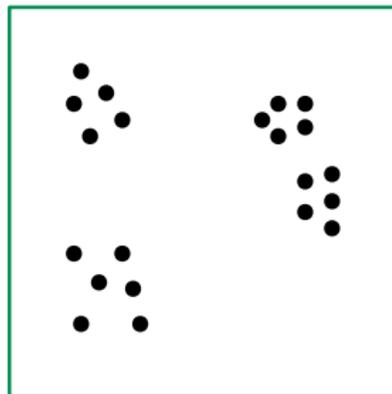
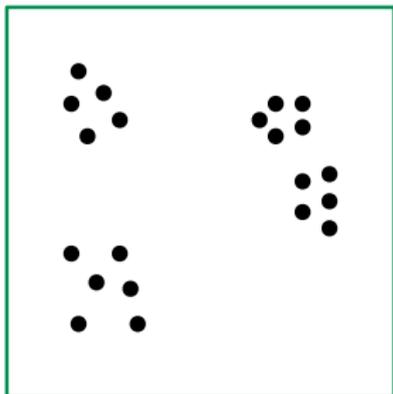
- Inizializza i centri  $\mu^{(1)}, \dots, \mu^{(k)}$  (in qualche modo)
- Ripeti fino ad avere convergenza:
  - Assegna ogni punto al suo centro **più vicino**
  - Aggiorna ciascun  $\mu^{(j)}$  al **baricentro** dei punti assegnati a  $\mu^{(j)}$



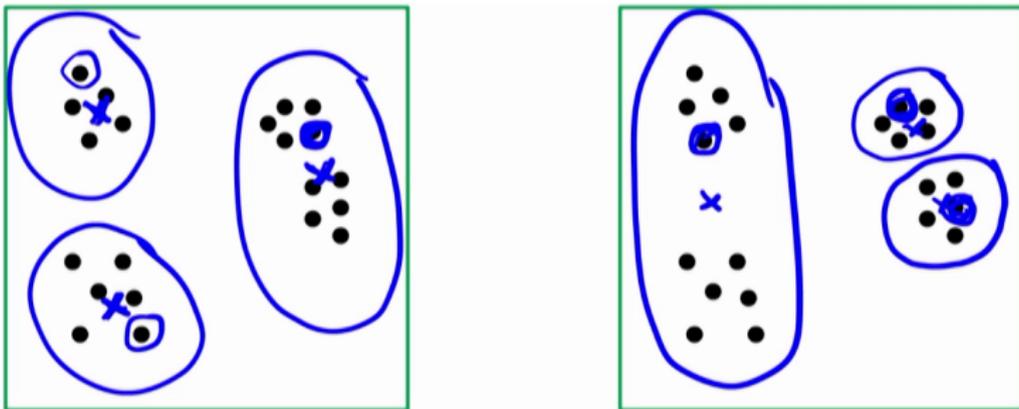
Si può dimostrare che ogni iterazione riduce il costo

Quindi si ha convergenza ad un **ottimo locale** della funzione costo

# L'inizializzazione può avere un grosso impatto



# L'inizializzazione può avere un grosso impatto



## Inizializzazione dell'algoritmo $k$ -means

Metodo spesso utilizzato:  $k$  centri iniziali sono scelti a caso dai dati

Trucco ulteriore: si inizia con dei centri aggiuntivi, per poi rimuoverli alla fine

Un'inizializzazione particolarmente buona:  *$k$ -means++*

- Scegli un esempio  $x$  a caso come primo centro
- Sia  $C = \{x\}$  (insieme dei centri scelti finora)
- Ripeti fino ad avere il numero desiderato di centri:
  - Scegli un esempio  $x$  a caso con la seguente distribuzione di probabilità:

$$\Pr(x) \propto \text{dist}(x, C)^2,$$

dove  $\text{dist}(x, C) = \min_{z \in C} \|x - z\|^2$

- Aggiungi  $x$  a  $C$

# Due esempi di utilizzo del clustering $k$ -means

- *Quantizzazione vettoriale:*  
Trovare un insieme finito di rappresentanti che “coprono bene” dei dati altamente multidimensionali
- *Ricerca di struttura significativa nei dati:*  
Identificare raggruppamenti significativi nei dati

## Es. 1: Rappresentazione di immagini con codifica $k$ -means

Come rappresentare una **collezione di immagini** con vettori di lunghezza fissa  $k$ ?



- Forma tutti blocchi  $\ell \times \ell$  in **tutte** le immagini. Estraine le feature.
- Applica  $k$ -means all'intera collezione di blocchi, ottenendo  $k$  centri
- Ora associa ad ogni blocco dell'immagine il suo centro più vicino
- Rappresenta l'immagine tramite un istogramma sull'insieme  $\{1, 2, \dots, k\}$

## Esempio 2: Ricerca di raggruppamenti naturali

Dataset su animali con vari attributi

- 50 animali: antilope, orso grizzly, castoro, dalmata, tigre...
- 85 attributi: ha il collo lungo, ha la coda, è un nuotatore, è notturno, è erbivoro, abita nel deserto, abita nella savana...
- Ogni animale ha un punteggio numerico per ogni attributo
- 50 punti dati in  $\mathbb{R}^{85}$

Applichiamo  $k$ -means con  $k = 15$

## Esempio 2: Ricerca di raggruppamenti naturali

- |   |  |
|---|--|
| <b>1</b> scimmia ragno, gorilla, scimpanzé  | <b>1</b> castoro, lontra   |
| <b>2</b> talpa, criceto, coniglio, chihuahua, ratto, topo                         | <b>2</b> scoiattolo  |
| <b>3</b> antilope, cavallo, alce, giraffa, zebra, cervo                           | <b>3</b> talpa, criceto, topo  |
| <b>4</b> puzzola, procione  | <b>4</b> antilope, cavallo, alce, pecora, giraffa, zebra, cervo, vacca                         |
| <b>5</b> orso grizzly, pastore tedesco, lupo, orso polare                         | <b>5</b> orso grizzly  |
| <b>6</b> pipistrello  | <b>6</b> pipistrello, ratto, donnola   |
| <b>7</b> scoiattolo, pastore scozzese   | <b>7</b> puzzola, procione   |
| <b>8</b> gatto persiano, gatto siamese  | <b>8</b> ippopotamo, elefante, bue, rinoceronte, bufalo, maiale                                |
| <b>9</b> panda gigante  | <b>9</b> panda gigante   |
| <b>10</b> orca assassina, balenottera azzurra, megattera, foca, tricheco, delfino | <b>10</b> coniglio   |
| <b>11</b> volpe, donnola, lince   | <b>11</b> tigre, leopardo, volpe, lupo, lince, leone   |
| <b>12</b> dalmata   | <b>12</b> orso polare  |
| <b>13</b> tigre, leopardo, leone  | <b>13</b> dalmata, gatto persiano, pastore tedesco, gatto siamese, chihuahua, pastore scozzese |
| <b>14</b> castoro, lontra   | <b>14</b> scimmia ragno, gorilla, scimpanzé  |
| <b>15</b> ippopotamo, elefante, bue, pecora, rinoceronte, bufalo, maiale, vacca   | <b>15</b> orca assassina, balenottera azzurra, megattera, foca, tricheco, delfino              |

# Clustering $k$ -means: pregi e difetti

Pregi:

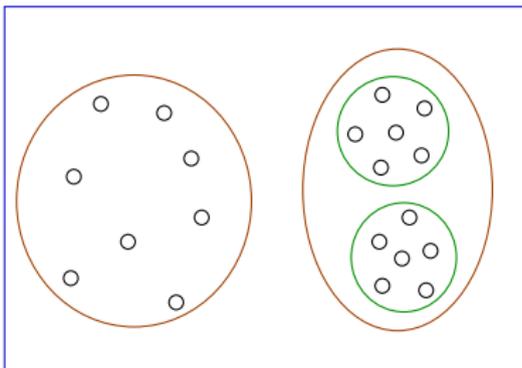
- Rapido e semplice
- Approccio efficace alla quantizzazione vettoriale

Difetti:

- Pensato soprattutto per cluster all'incirca **sferici** e di raggio abbastanza simile
- Il numero di cluster va specificato

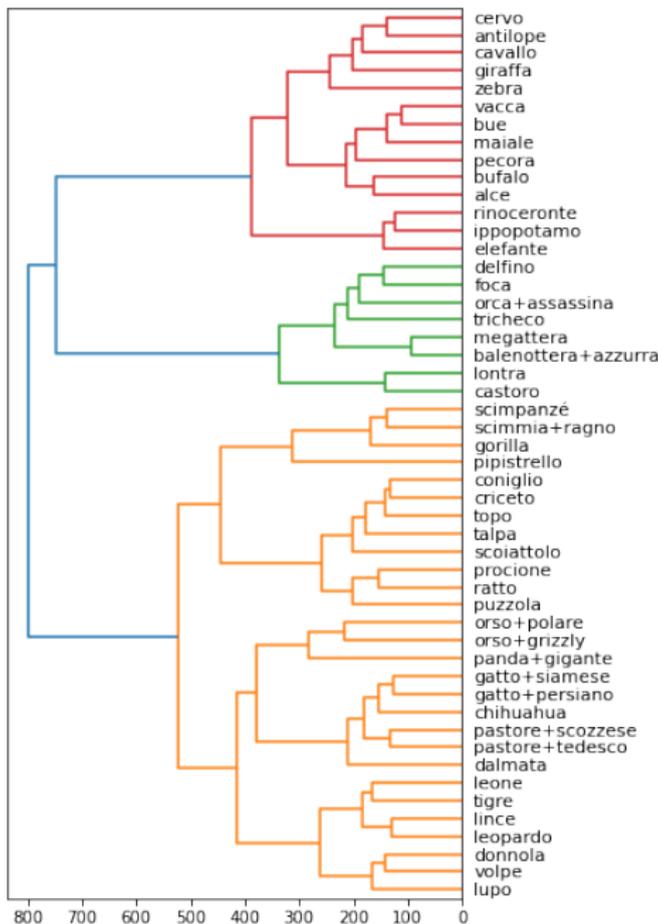
# Clustering gerarchico

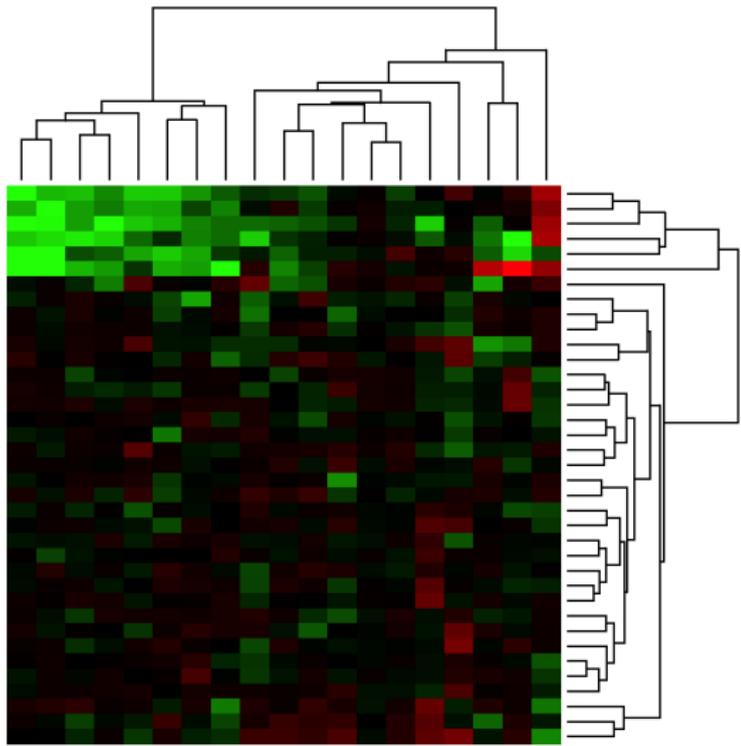
Scegliere il numero di cluster ( $k$ ) non è banale



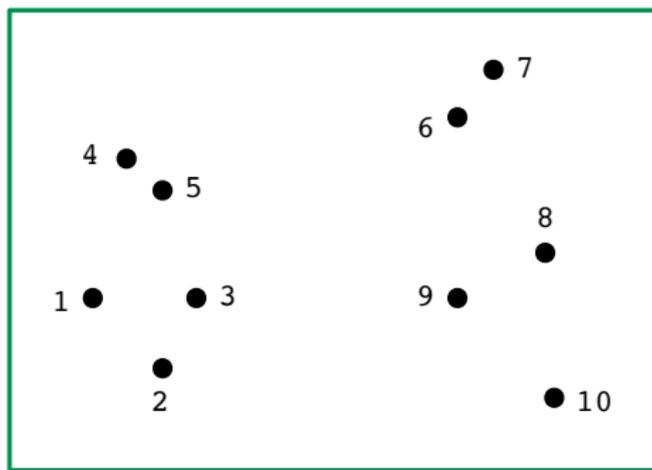
A causa della struttura **multiscala** dei dati, spesso non c'è un numero di cluster corretto in assoluto

Per questo può essere preferibile un approccio *gerarchico*





## L'algoritmo *single linkage*



- Inizia con ogni punto in un cluster a sé stante
- Ripeti fino ad avere un unico cluster:
  - Fondi i due cluster contenenti la coppia di punti più vicina

# Metodi di linkage

- Inizia con ogni punto in un cluster a sé stante
- Ripeti fino ad avere un unico cluster:
  - Fondi i due cluster “più vicini”

Come definiamo la **distanza** tra due cluster  $C, C'$ ?

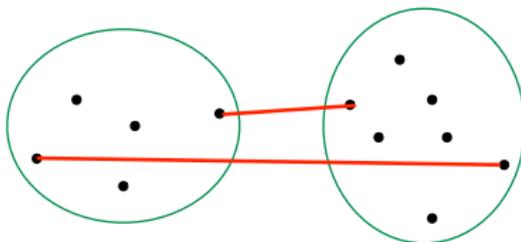
- *Single linkage*

$$\text{dist}(C, C') = \min_{x \in C, x' \in C'} \|x - x'\|$$

- *Complete linkage*

$$\text{dist}(C, C') = \max_{x \in C, x' \in C'} \|x - x'\|$$

# Metodi di linkage



## Average linkage: vari criteri

- 1** Distanza media tra coppie di punti nei due cluster:

$$\text{dist}(C, C') = \frac{1}{|C| \cdot |C'|} \sum_{x \in C} \sum_{x' \in C'} \|x - x'\| = \text{avg}_{x \in C, x' \in C'} \|x - x'\|$$

- 2** Distanza tra i centri dei cluster:

$$\text{dist}(C, C') = \|\text{baricentro}(C) - \text{baricentro}(C')\|$$

- 3** Criterio di Ward

$$\text{dist}(C, C') = \frac{|C| \cdot |C'|}{|C| + |C'|} \|\text{baricentro}(C) - \text{baricentro}(C')\|^2$$

Coincide con l'incremento nel costo  $k$ -means che si avrebbe fondendo i cluster

# Clustering: riepilogo

- Metodo non supervisionato (nessuna variabile da predire)
- Ricerca sottoinsiemi “significativi” di esempi
- Difficile da formalizzare: non esiste una misura di clustering “ideale”
- Può essere utile per ridurre la mole di esempi in un metodo supervisionato
- O semplicemente per “esplorare” i dati

Metodi di clustering:

- *k*-means (`sklearn.cluster.KMeans`)
- metodi di linkage (`scipy.cluster.hierarchy.linkage`)
- altri metodi (clustering spettrale, information bottleneck, ...)

# Riduzione della dimensionalità dei dati

L'array dei dati  $X \in \mathbb{R}^{m \times d}$  ha due assi: gli  $m$  esempi e le  $d$  variabili

Il clustering  $k$ -means (o in genere, la quantizzazione vettoriale) può essere visto come un metodo per ridurre il numero di esempi ( $m$ )

I metodi di riduzione della dimensionalità hanno invece come obiettivo la riduzione del numero di variabili ( $d$ )

Esempi:

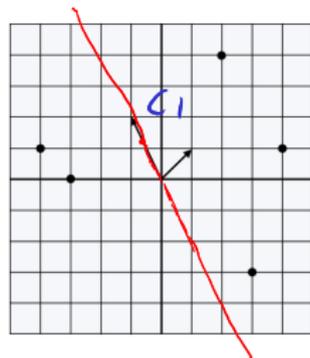
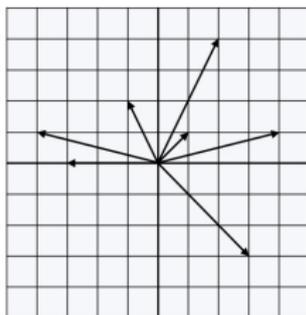
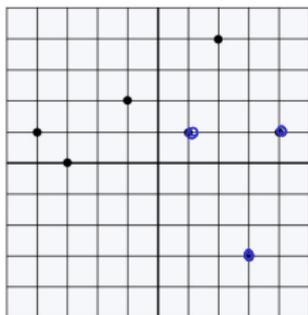
 *Principal Component Analysis* (PCA)

GES30

- Proiezioni casuali
- Compressed sensing

Analoghi ai metodi di *riduzione delle feature* discussi nell'apprendimento supervisionato, ma in un contesto **non supervisionato**

## Punti, vettori, spanning set, basi

 $\mathbb{R}^2$  $\mathbb{R}^2$ 

Punti di input:  $x^{(1)}, x^{(2)}, \dots, x^{(m)} \in \mathbb{R}^d$

Vettori  $c_1, c_2, \dots, c_K$ : definiscono il sottospazio lineare

$$\{x \in \mathbb{R}^d : x = \sum_{k=1}^K c_k w_k \text{ per qualche } w \in \mathbb{R}^K\}$$

L'insieme  $\{c_1, \dots, c_K\}$  è detto uno *spanning set*

Se linearmente indipendenti e  $K = \underline{d}$ , formano una base di  $\mathbb{R}^d$

# Coordinate nella base $C$

Se i vettori  $c_1, c_2, \dots, c_d$  formano una base di  $\mathbb{R}^d$ , allora per ogni  $x^{(i)} \in \mathbb{R}^d$  esiste  $w^{(i)} \in \mathbb{R}^d$  tale che

$$Cw^{(i)} = x^{(i)}$$

dove  $C \in \mathbb{R}^{d \times d}$  è la matrice formata dai vettori colonna  $c_1, \dots, c_d$ :

$$C = \begin{pmatrix} c_1 & c_2 & \dots & c_d \end{pmatrix} \in \mathbb{R}^{d \times d}$$

Il vettore  $w^{(i)}$  fornisce le **coordinate** di  $x^{(i)}$  nella base  $C$

# Determinazione delle coordinate $w$ nella base $C$

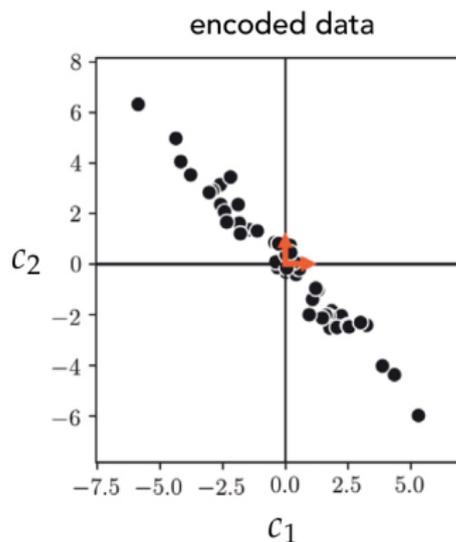
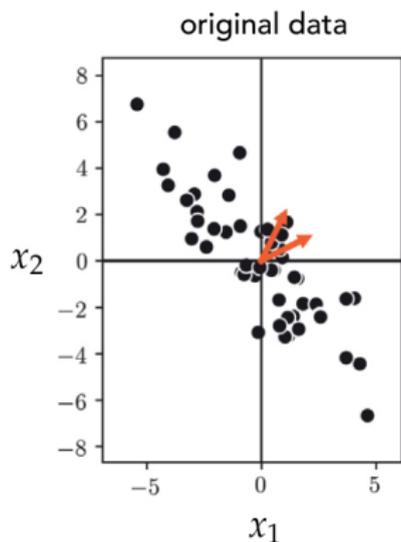
I vettori  $w^{(i)}$  possono essere determinati minimizzando la funzione

$$g(w^{(1)}, w^{(2)}, \dots, w^{(m)}) = \frac{1}{m} \sum_{i=1}^m \left\| \underbrace{Cw^{(i)}} - \underbrace{x^{(i)}} \right\|_2^2$$

In particolare, poiché ogni  $w^{(i)}$  può essere scelto indipendentemente dagli altri, annullando il gradiente di  $g$  si ottiene la condizione di ottimalità

$$C^T Cw^{(i)} = C^T x^{(i)}$$

## Codifica perfetta dei dati in una base



Esempio con  $C = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}$

## Caso di una base ortonormale

Un caso particolare si ha quando i vettori  $c_1, \dots, c_d$  formano una base **ortonormale**:

$$C = \begin{bmatrix} | & | & & | \\ c_1 & c_2 & \dots & c_d \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{d \times d} \quad c_i^\top c_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

$$\|c_i\|^2 = 1 \rightarrow \|c_i\| = 1 \\ c_i \perp c_j \quad \forall i \neq j$$

In questo caso abbiamo  $C^\top C = CC^\top = I$  dove  $I$  è la matrice identità  $d \times d$  (cioè  $C$  è una matrice **ortonormale**)

Quindi la condizione  $C^\top C w^{(i)} = C^\top x^{(i)}$  diventa

$$\underline{w}^{(i)} = C^\top x^{(i)}$$

$$C w^{(i)} = \underbrace{C C^\top}_{I} x^{(i)}$$

In altre parole, per **codificare** usiamo  $w^{(i)} = C^\top x^{(i)}$  e per **decodificare**  $x^{(i)} = C w^{(i)}$

**Formula di autocodifica** [autoencoder formula]

$$x^{(i)} = C \underbrace{C^\top x^{(i)}}_{w^{(i)}}$$

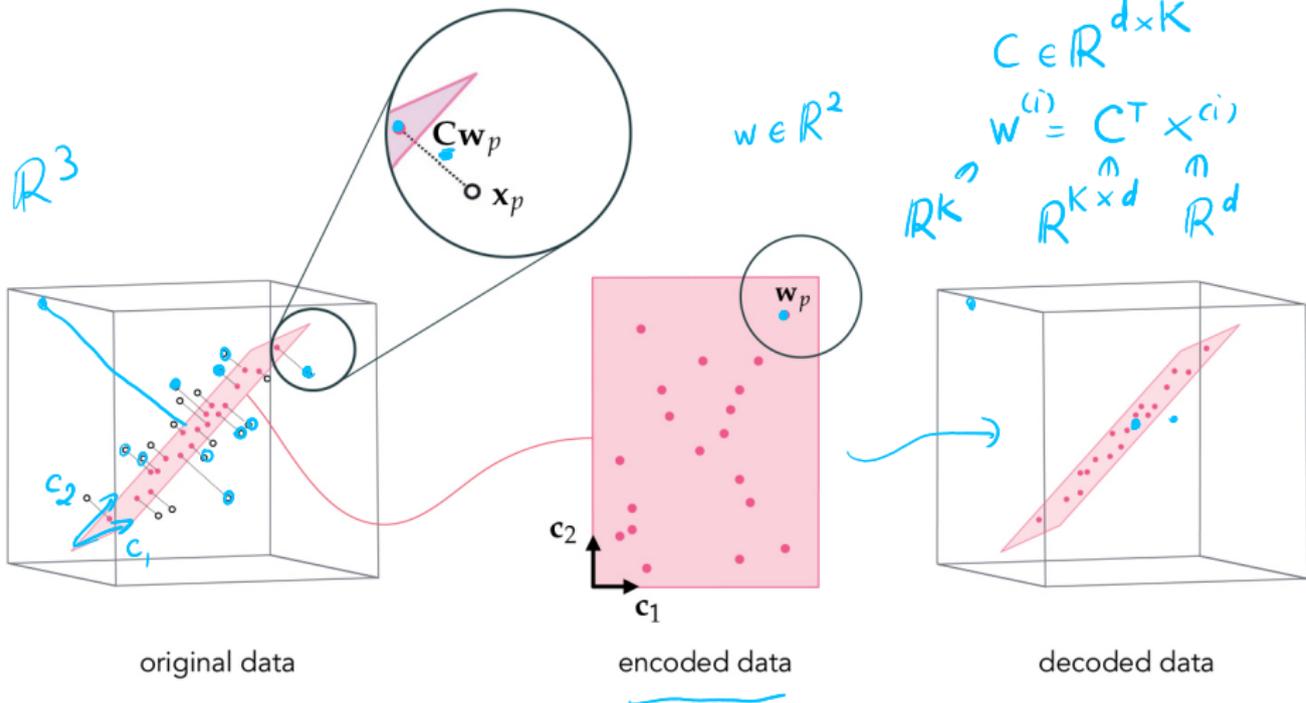
$$C^\top = C^{-1}$$

# Codifica imperfetta dei dati con uno spanning set

$$K \leq d$$

$$c_1 \dots c_K$$

$$(K < d)$$



## Codifica imperfetta dei dati con uno spanning set

Se il numero di colonne di  $C$  (numero di vettori nello spanning set) è  $K < d$  allora la codifica diventa **imperfetta**

Se scegliamo sempre i  $w^{(i)}$  in modo da minimizzare la funzione  $g(w)$ ,  $Cw^{(i)}$  coincide con la **proiezione** di  $x^{(i)}$  sul **sottospazio** generato dalle colonne di  $C$

Se  $x^{(i)}$  è **vicino** a questo sottospazio avremo  $Cw^{(i)} \approx x^{(i)}$

La relazione di autocodifica diventa approssimata

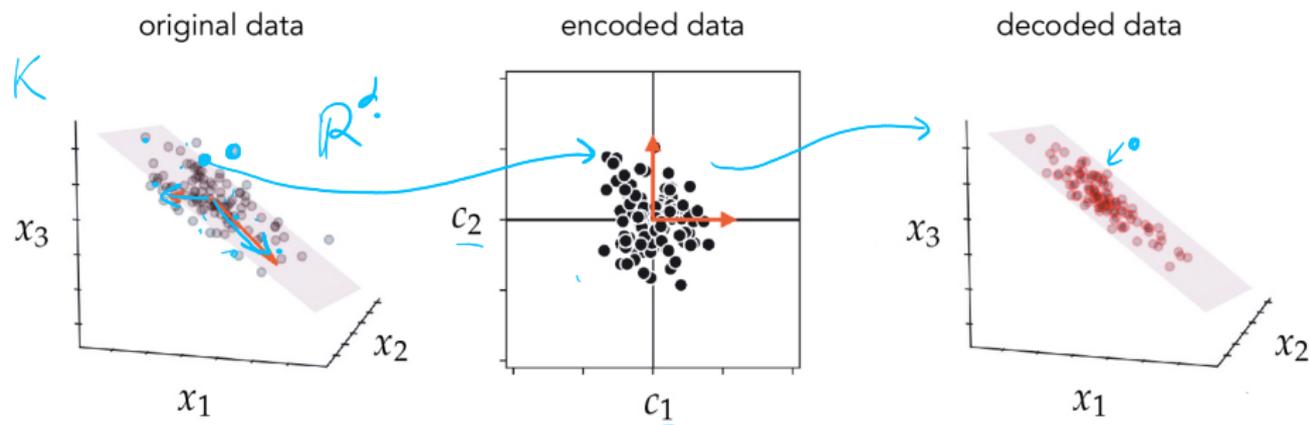
*Formula di autocodifica* [autoencoder formula]

$$x^{(i)} \approx CC^T x^{(i)}$$

L'approssimazione è buona nella misura in cui ogni  $Cw^{(i)}$  è vicino a  $x^{(i)}$

# Apprendimento di uno spanning set

Come **scegliamo** le colonne di  $C$ ?



Criterio: minimize  $g(w, C) = \frac{1}{m} \sum_{i=1}^m \left\| \underbrace{Cw}_{\substack{\uparrow \\ \uparrow}}^{(i)} - x^{(i)} \right\|_2^2$

$g$  ora è una funzione (non convessa) sia di  $w$  che di  $C$

# Autoencoder lineare

Aggiungendo l'assunzione che i vettori  $\underline{c}$  siano ortonormali tra loro abbiamo, come prima,  $\underline{w}^{(i)} = C^T x^{(i)}$  e quindi

$$\rightarrow g(C) = \frac{1}{m} \sum_{i=1}^m \left\| \underbrace{CC^T}_{\underline{w}^{(i)}} x^{(i)} - x^{(i)} \right\|_2^2$$

è ora esprimibile come funzione della sola matrice  $C$

In altre parole: la scelta fondamentale è il sottospazio su cui proiettare

# Autoencoder lineare

$\mathbb{R}^2$

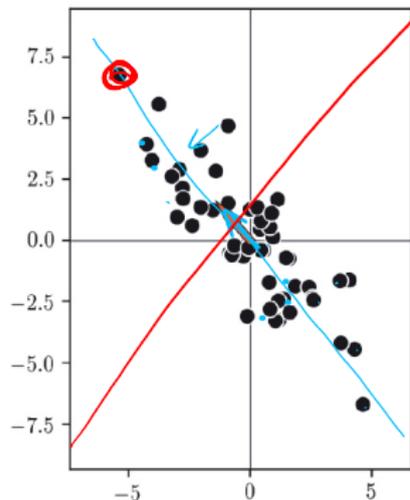


$\mathbb{R}^1$

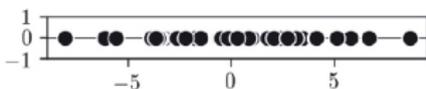


$\mathbb{R}^2$

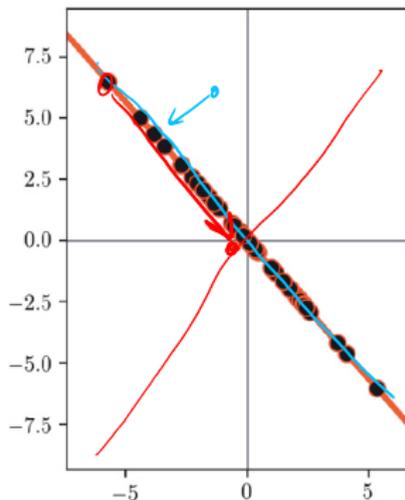
original data



encoded data



decoded data

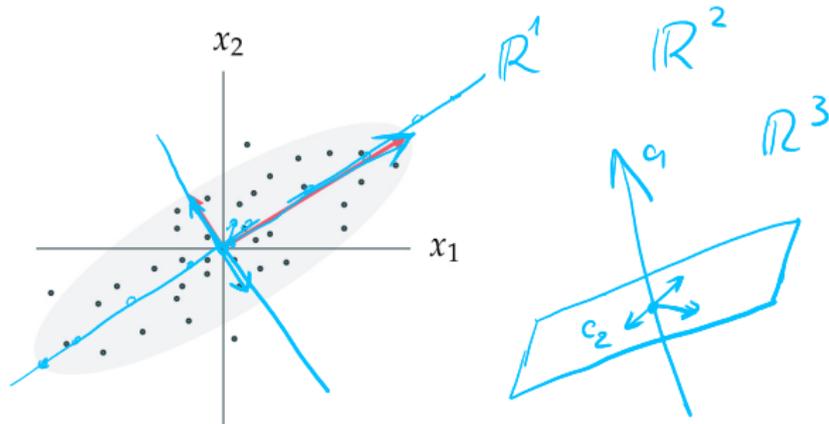


# Componenti principali di un dataset

Possono esistere **molte** matrici  $C$  che minimizzano la funzione  $g$

Le **componenti principali** forniscono uno di questi minimi

$c_1, c_2, \dots, c_k$   
ortogonali



**Intuizione.** La prima componente principale  $c_1$  è la direzione lungo la quale la varianza dei dati è massima

La  $k$ -esima componente principale  $c_k$  è la direzione, **tra quelle ortogonali a  $c_1, \dots, c_{k-1}$** , lungo la quale la varianza dei dati è massima

## Definizione delle componenti principali

Siano  $x^{(1)}, \dots, x^{(m)}$  degli esempi **già centrati sulla loro media** ( $\sum_i x^{(i)} = 0$ )

Se  $X \in \mathbb{R}^{m \times d}$  è la matrice dati (esempi-feature), la sua **matrice di covarianza** è la matrice  $\Sigma = \frac{1}{m} X^T X \in \mathbb{R}^{d \times d}$   $\frac{1}{m} X^T X = \Sigma^T = \Sigma$

Essendo una matrice simmetrica, essa ammette una **diagonalizzazione**

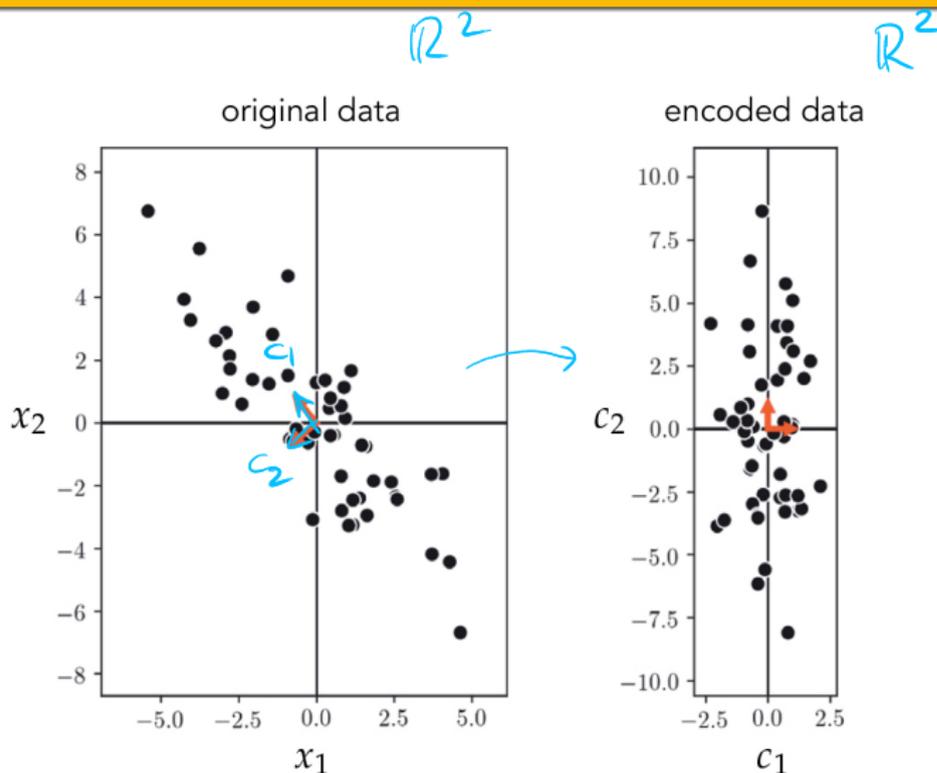
$$V = \begin{bmatrix} v_1 & v_2 & \dots & v_d \end{bmatrix} \quad \Sigma = V D V^T \quad D = \begin{bmatrix} d_{11} & & & \\ & d_{22} & & \\ & & \dots & \\ & & & d_{dd} \end{bmatrix}$$

con  $V \in \mathbb{R}^{d \times d}$  matrice **ortogonale** e  $D \times \mathbb{R}^{d \times d}$  matrice **diagonale**

I valori sulla diagonale di  $D$  (**autovalori** di  $\Sigma$ ) quantificano **la varianza dei dati lungo ciascuna componente** (colonna di  $V$ )

Le  $K$  **componenti principali** sono le colonne di  $V$  (**autovettori** di  $\Sigma$ ) corrispondenti ai  $K$  autovalori più grandi

## Codifica tramite componenti principali



# Algoritmo PCA

## Principal Components Analysis (PCA)

**Dati:**  $m$  vettori di input  $x^{(1)}, \dots, x^{(m)} \in \mathbb{R}^d$  ed un intero  $K \leq d$

**Trova:** le  $K$  componenti principali del dataset

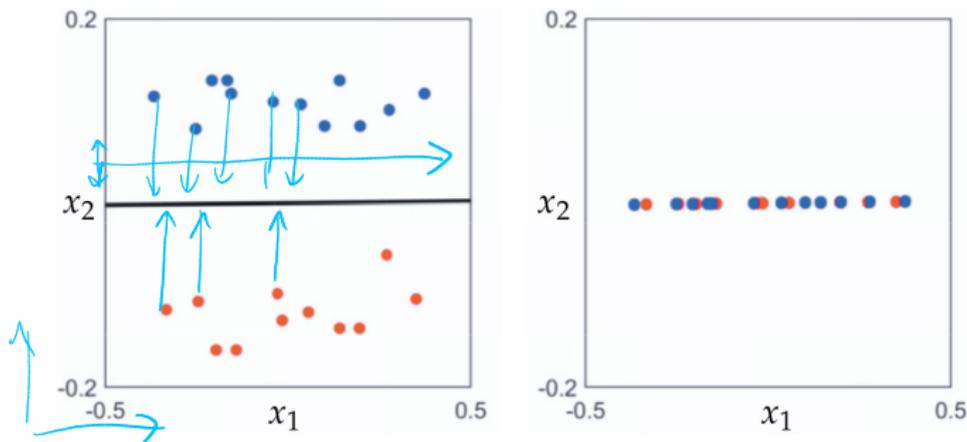
- 1 Calcola la media:  $\mu = \frac{1}{m} \sum_{i=1}^m x^{(i)}$
- 2 Centra i vettori:  $x^{(i)} \leftarrow x^{(i)} - \mu$ , per ogni  $i = 1, \dots, m$
- 3 Calcola la matrice di covarianza:  $\Sigma \leftarrow \frac{1}{m} X^T X$
- 4 Diagonalizza la matrice di covarianza:  $\Sigma = V D V^T$   $D, V$
- 5 Riordina gli autovalori  $d_{kk}$  (e i corrispondenti autovettori  $v_k$ ) in modo che
 

$\rightarrow d_{11} \geq d_{22} \geq \dots \geq d_{kk} \geq \dots$
- 6 Restituisci i vettori  $v_1, v_2, \dots, v_K$  (e i corrispondenti autovalori)

Per il passo (4) si può usare ad esempio `np.linalg.eigh` in NumPy

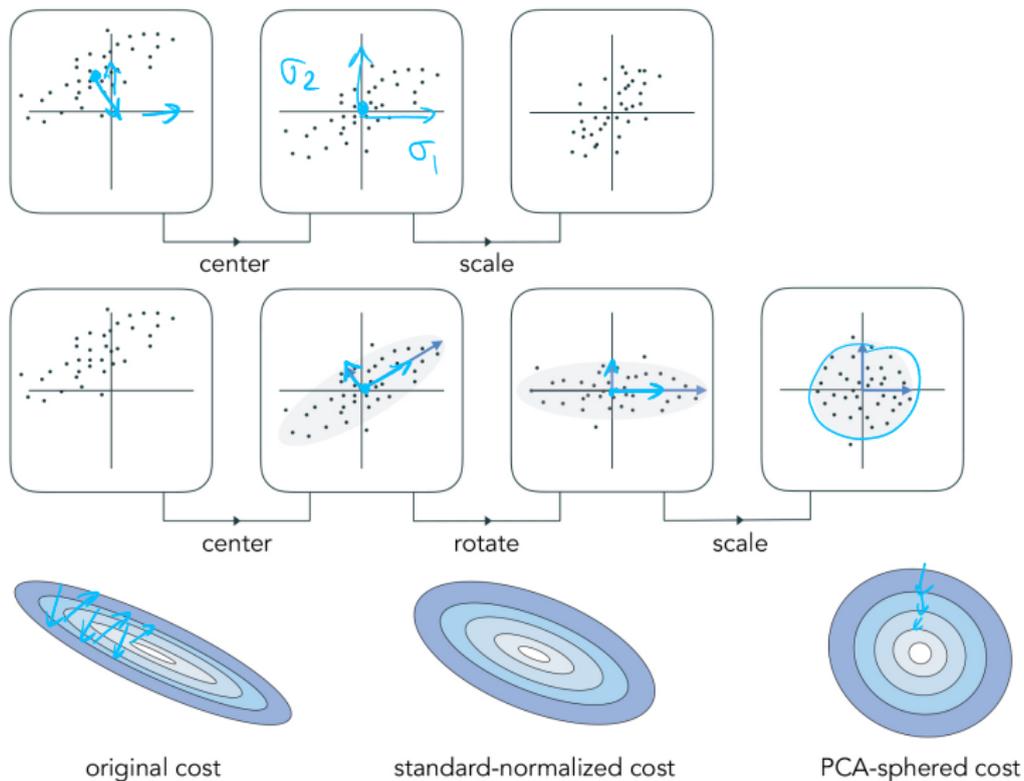
# Un esempio disastroso

**Attenzione:** in una problema di predizione, il metodo PCA con  $K < d$  rischia di tagliare via informazioni cruciali!



Nei problemi di predizione, è più comune usare PCA con  $K = d$  come forma di preprocessamento (*sferificazione PCA*)

# Standardizzazione vs. "sferificazione" PCA



## Standardizzazione vs. "sferificazione" PCA

## Standardizzazione

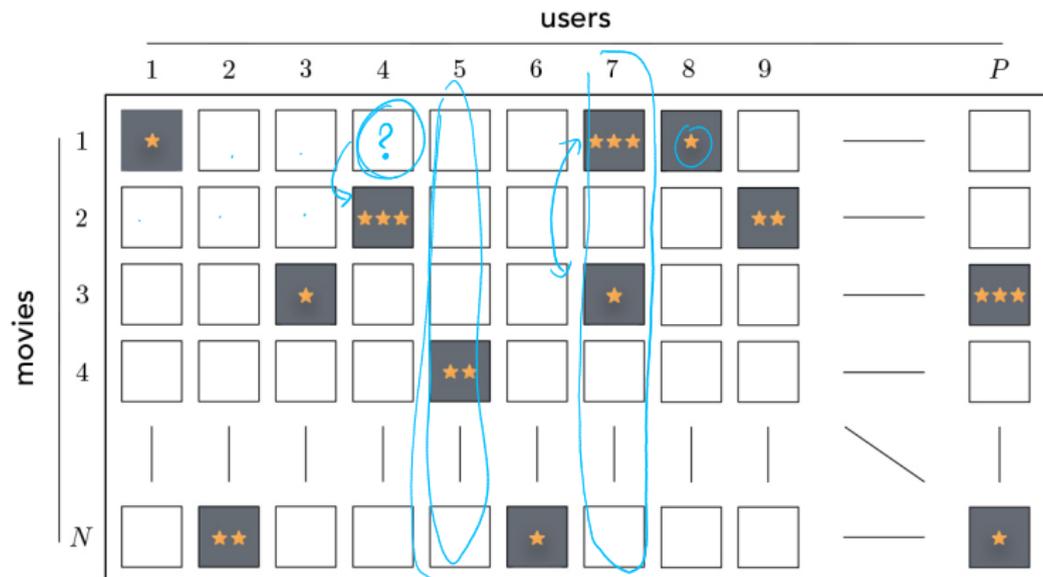
- 1 Centra:  $x^{(i)} \leftarrow x^{(i)} - \mu$  dove  $\mu$  è la **media** degli  $x^{(i)}$
- 2 Scala:  $x_j^{(i)} \leftarrow \frac{x_j^{(i)}}{\sqrt{\sigma_j^2}}$  dove  $\sigma_j^2 = (1/m) \sum_i x_j^{(i)2}$  è la **varianza** della  $j$ -esima variabile  
 deviazione standard delle  $j$ -esime variabile

Sferificazione PCA [*PCA-sphering*]

- 1 Centra:  $x^{(i)} \leftarrow x^{(i)} - \mu$  dove  $\mu$  è la **media** degli  $x^{(i)}$
- 2 Ruota:  $x^{(i)} \leftarrow V^T x^{(i)}$
- 3 Scala:  $x_j^{(i)} \leftarrow \frac{x_j^{(i)}}{\sqrt{d_{jj}}}$  dove  $d_{jj}$  è il  $j$ -esimo **valore sulla diagonale** della matrice  $D$  ( $\equiv$  varianza lungo la  $j$ -esima componente)

Sistemi di raccomandazione [*Recommender systems*]

Scenario applicativo: matrice di voti film–utenti  
 Come stimiamo i voti mancanti?



Si può utilizzare una generalizzazione di PCA a matrici “incomplete”

Sistemi di raccomandazione [*Recommender systems*]

Nella PCA minimizzavamo

$$g(w, C) = \frac{1}{m} \sum_{i=1}^m \left\| Cw^{(i)} - x^{(i)} \right\|_2^2$$

Ora però solo un **sottoinsieme**  $\Omega_i$  degli elementi di  $x^{(i)}$  è accessibile:

$$\Omega_i = \{(j, i) \mid \text{l'utente } i \text{ ha dato un voto al film } j\}$$

per cui minimizziamo

$$g(w, C) = \frac{1}{m} \sum_{i=1}^m \left\| \{Cw^{(i)} - x^{(i)}\}_{\Omega_i} \right\|_2^2$$

cioè teniamo conto solo delle componenti di  $Cw^{(i)} - x^{(i)}$  che ricadono nell'insieme  $\Omega_i$ . Il prodotto  $Cw^{(i)}$  stimerà anche i **voti mancanti** di  $i$ .