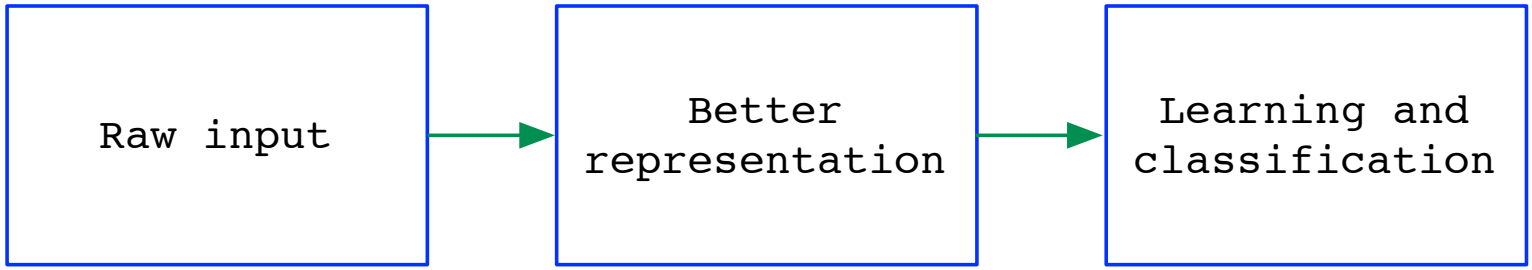
IN550 Machine Learning

Apprendimento non supervisionato: Clustering e Principal Component Analysis

Vincenzo Bonifaci

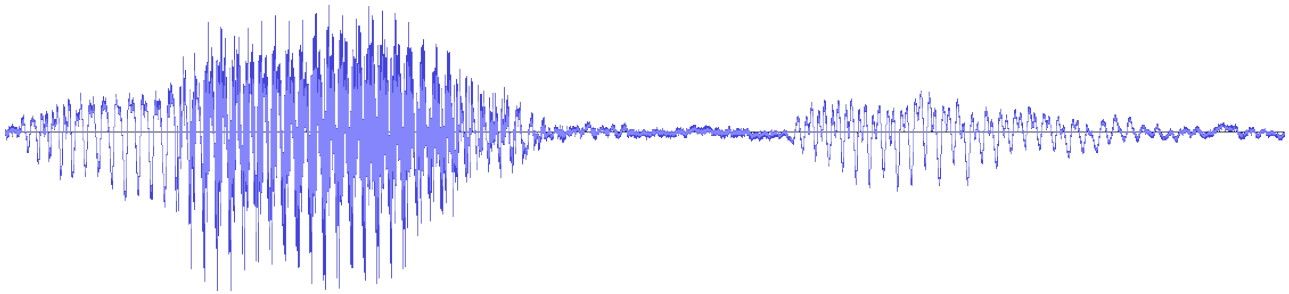
Apprendimento della rappresentazione



Una buona rappresentazione semplifica l'apprendimento:

- Catturando correttamente i gradi di libertà presenti nei dati
- Catturando strutture rilevanti su varia scala
- Mascherando 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

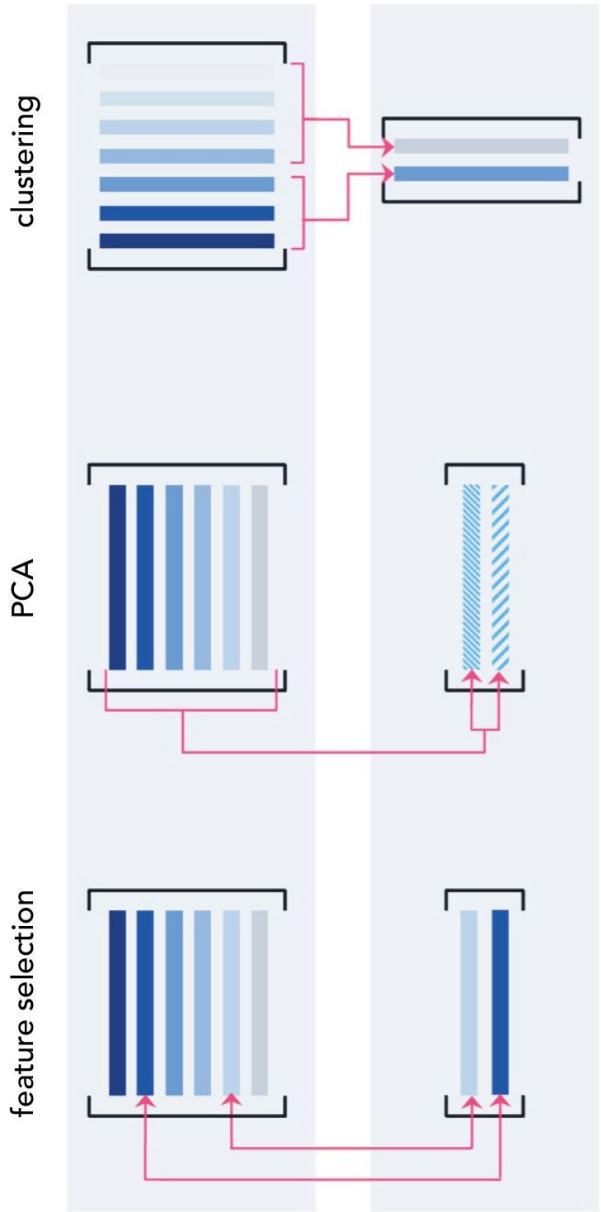
Esploreremo i seguenti approcci:

- Clustering
- Proiezioni lineari

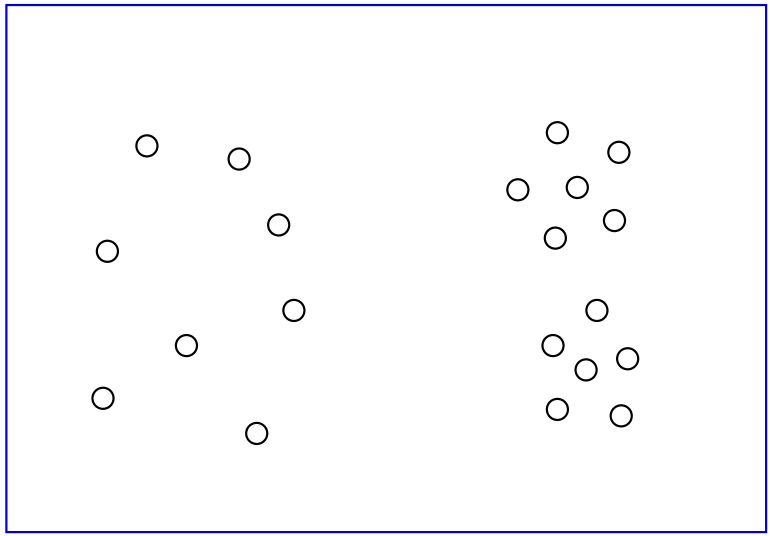
È una tipologia di apprendimento non supervisionato perché non ci sono variabili di uscita (etichette), né predizioni

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



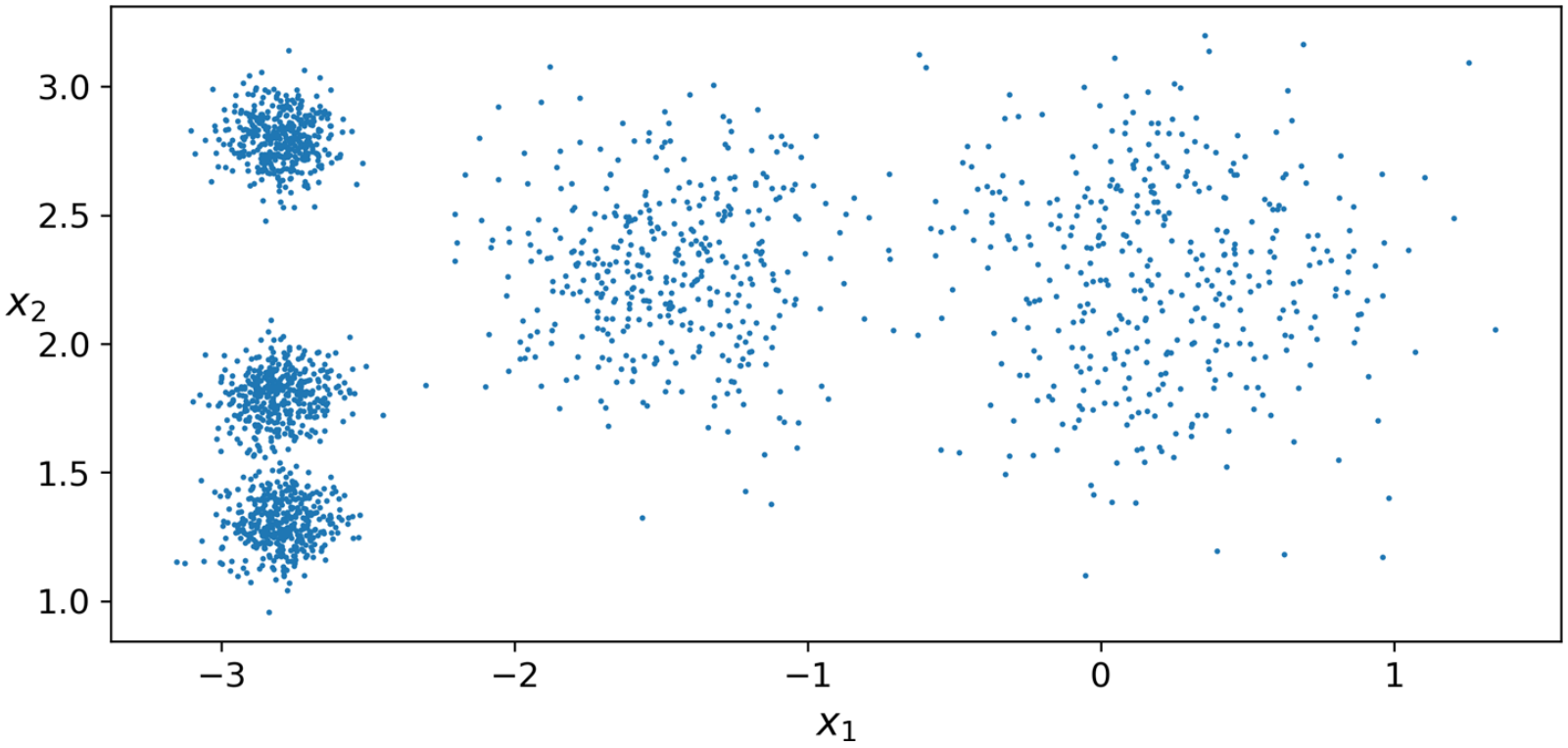
Clustering in $\mathbb{R}^d$



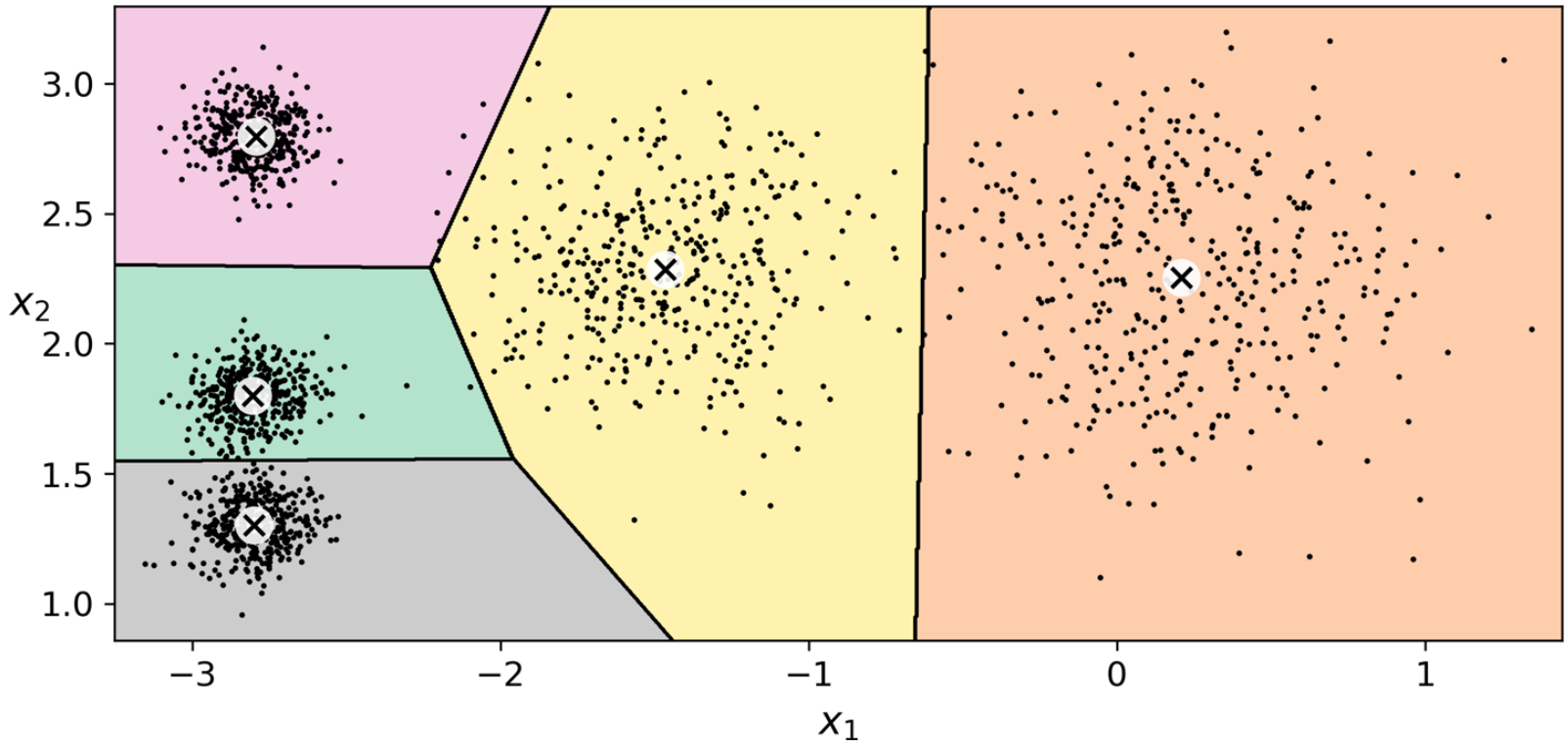
Due comuni utilizzi del clustering:

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

Esempio



Esempio



Due approcci al clustering

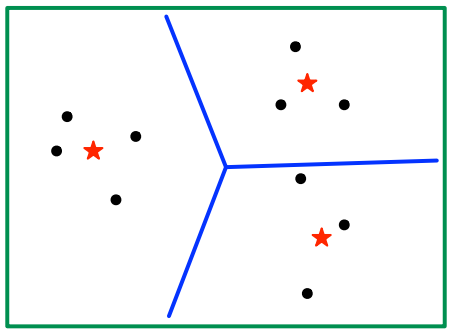
Qui discuteremo due approcci al clustering:

- Clustering k-means
- Clustering gerarchico

Il problema di ottimizzazione k-means

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

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



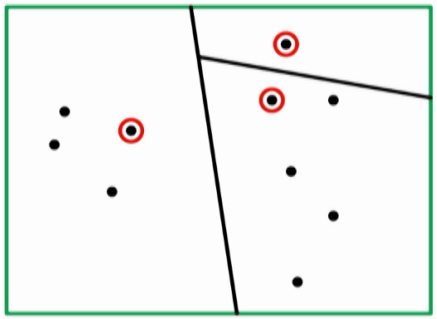
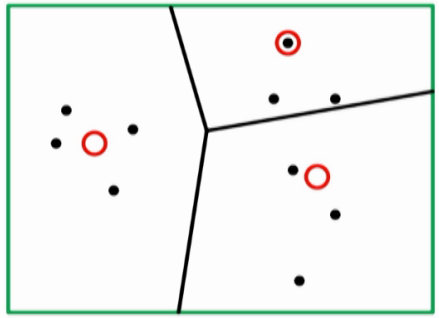
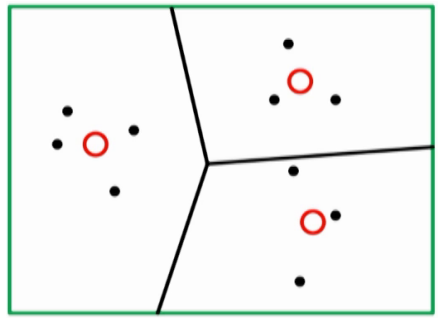
I centri partizionano $\mathbb{R}^d$ in k regioni convesse
La regione j consiste di tutti i punti il cui centro più vicino è $\mu^{(j)}$

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

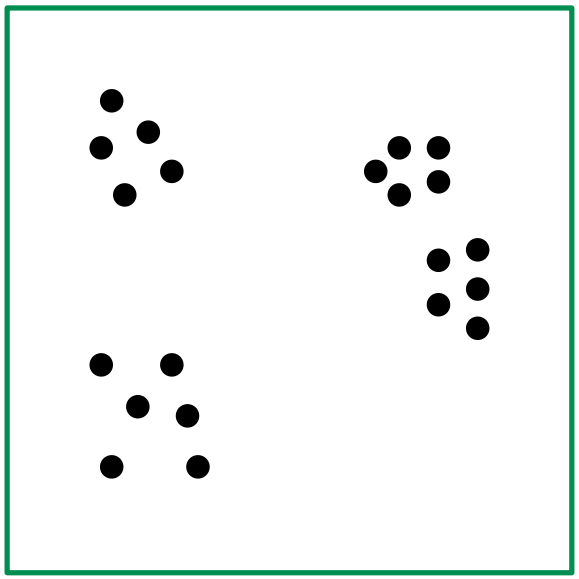
- Inizializza i centri $\mu^{(1)}, \ldots, \mu^{(k)}$ (in qualche modo)
- Ripeti fino ad avere convergenza:
  - Assegna ogni punto al suo centro più vicino
  - Aggiorna ciascun $\mu^{(j)}$ al baricentro dei punti assegnati a $\mu^{(j)}$

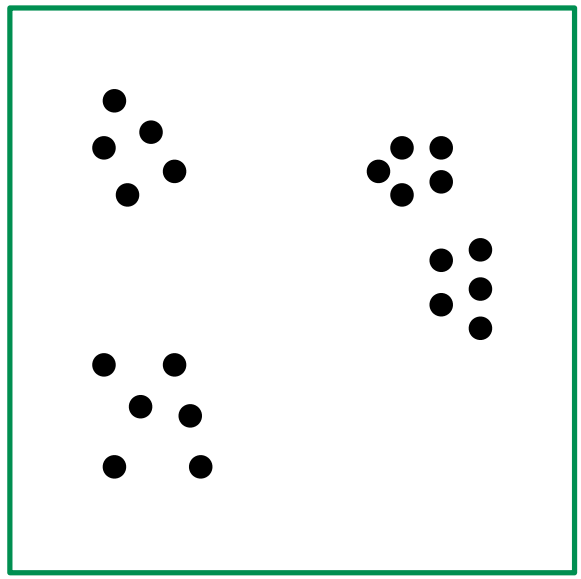




Si può dimostrare che ogni iterazione riduce il costo

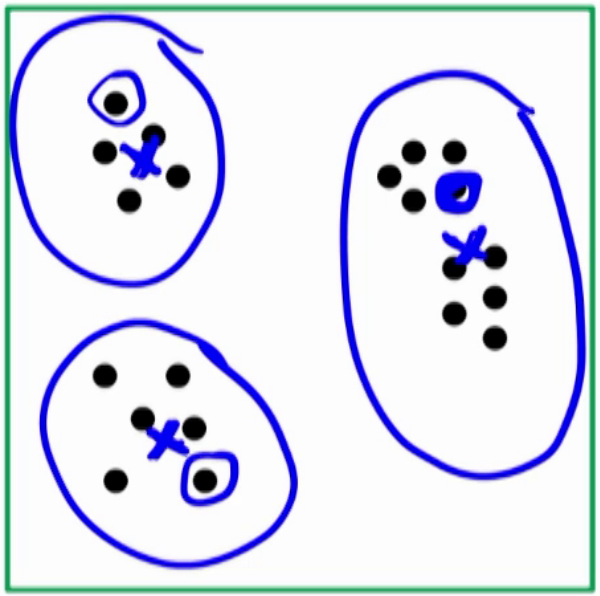
Quindi si ha convergenza ad un ottimo locale della funzione costo

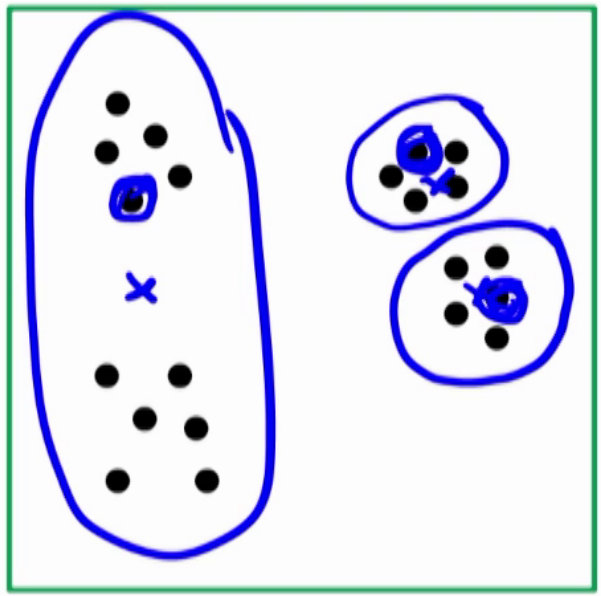
L'inizializzazione può avere un grosso impatto





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:

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

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

Due esempi di utilizzo del clustering k-means

- *Quantizzazione vettoriale*:
  Trovare un insieme finito di rappresentanti che "coprano 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 $\{1, 2, \ldots, k\}$

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

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

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

Clustering k-means: pregi e difetti

Pregi:

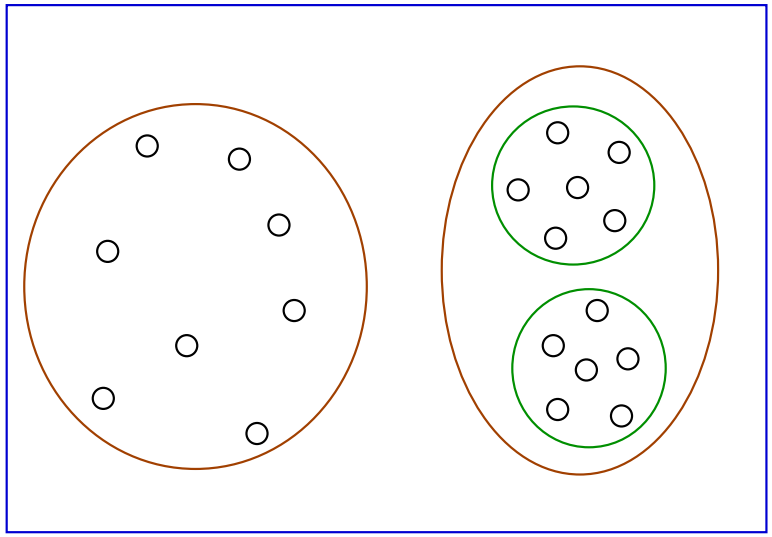
- Rapido e semplice
- Approccio efficace alla quantizzazione vettoriale

Difetti:

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

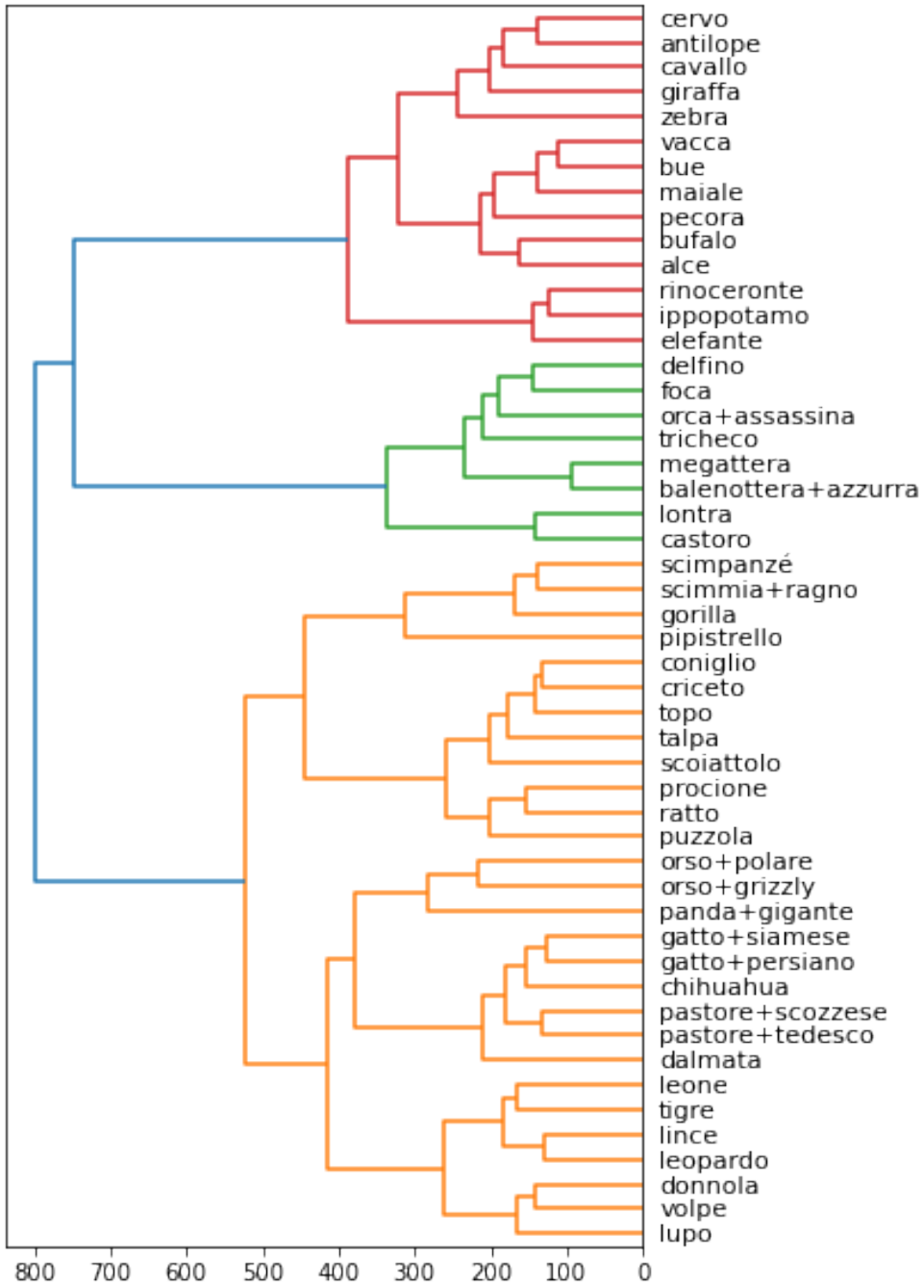
Clustering gerarchico

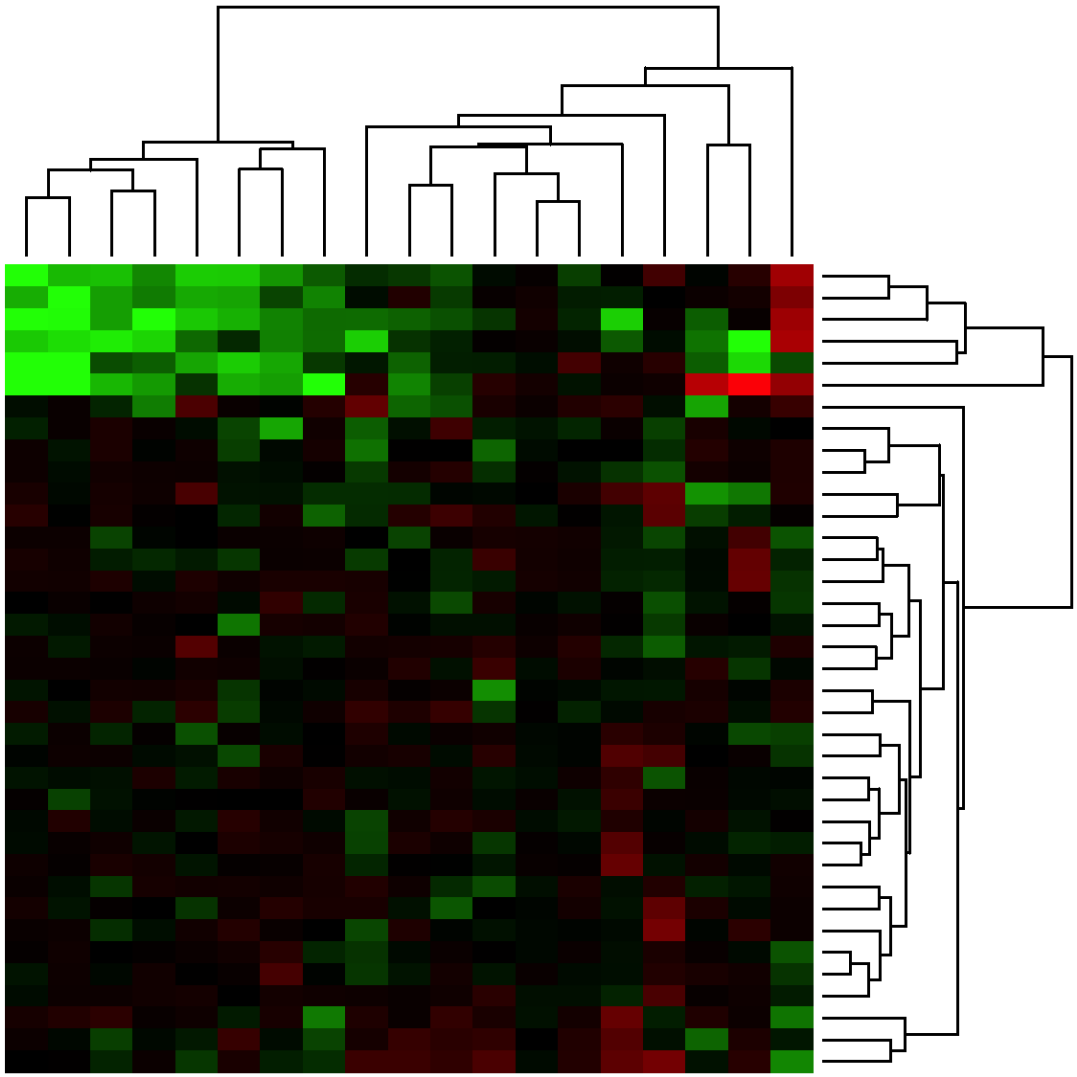
Scegliere il numero di cluster (k) non è banale



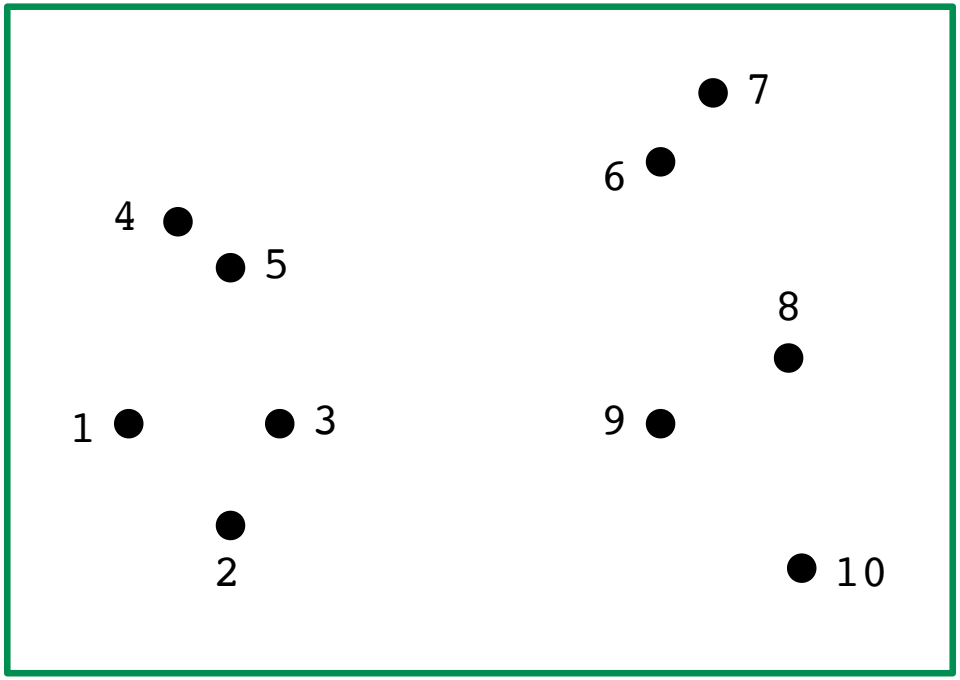
A causa della struttura multiscala dei dati, spesso non c'è un numero di cluster corretto in assoluto

Per questo può essere preferibile un approccio *gerarchico*


cervo
antilope
cavallo
giraffa
zebra
vacca
bue
maiale
pecora
bufalo
alce
rinoceronte
ippopotamo
elefante
delfino
foca
orca+assassina
tricheco
megattera
balenottera+azzurra
lontra
castoro
scimpanzé
scimmia+ragno
gorilla
pipistrello
coniglio
criceto
topo
talpa
scoiattolo
procione
ratto
puzzola
orso+polare
orso+grizzly
panda+gigante
gatto+siamese
gatto+persiano
chihuahua
pastore+scozzese
pastore+tedesco
dalmata
leone
tigre
lince
leopardo
donnola
volpe
lupo
800
700
600
500
400
300
200
100
0



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'?

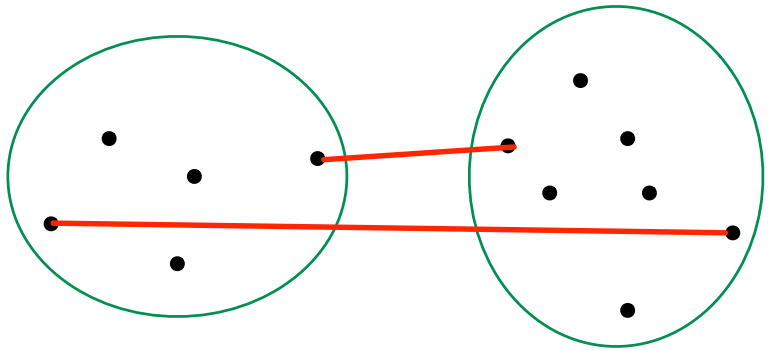
- *Single linkage*

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

- *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:

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

2. Distanza tra i centri dei cluster:

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

3. Criterio di Ward

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

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

Clustering in `scikit-learn`

Approccio	Iperparametri	Interfaccia `scikit-learn`
k-Means	k	`sklearn.cluster.KMeans`
Linkage	`linkage`	`sklearn.cluster.AgglomerativeClustering`

Opzioni per `linkage`: `'ward'`, `'complete'`, `'average'`, `'single'` (`default='ward'`)

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

Metodologie per il clustering:

- k-means
- metodi di 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)

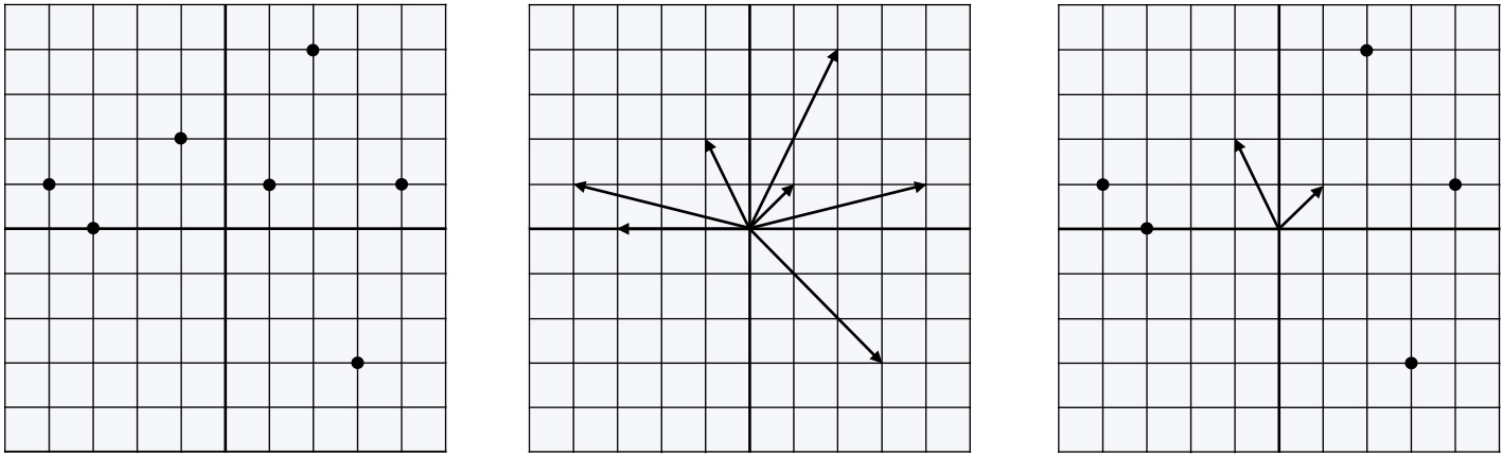
I metodi di *riduzione della dimensionalità* hanno invece come obiettivo la riduzione del numero di variabili (d)

Esempi:

- *Principal Component Analysis* (PCA)
- 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



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

Vettori $c_1, c_2, \ldots, c_K$: 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 $\{c_1, \ldots, c_K\}$ è detto uno *spanning set*
Se linearmente indipendenti e $K = d$, formano una *base* di $\mathbb{R}^d$

Coordinate nella base C

Se i vettori $c_1, c_2, \ldots, c_d$ 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 $c_1, \ldots, c_d$:

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

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

Determinazione delle coordinate w nella base C

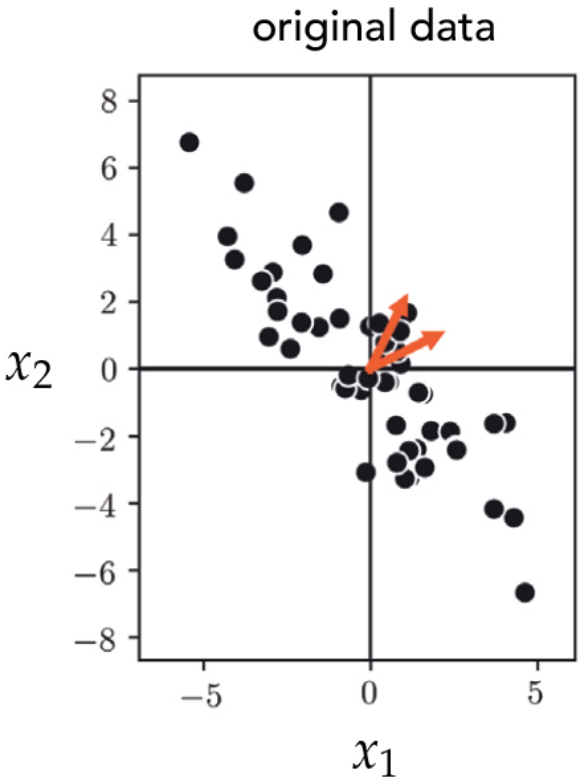
I vettori $w^{(i)}$ possono essere determinati minimizzando la funzione

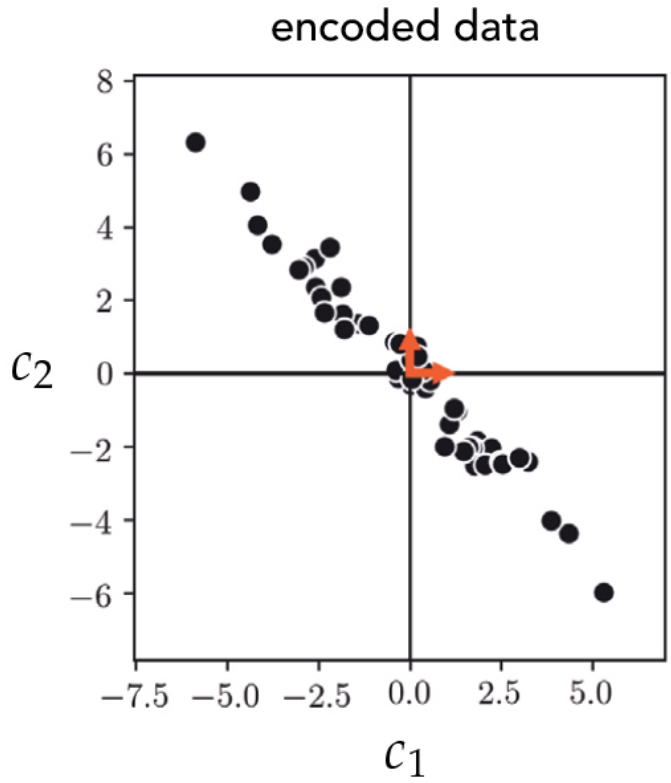
$$g\left(w^{(1)}, w^{(2)}, \ldots, w^{(m)}\right) = \frac{1}{m} \sum_{i=1}^{m} \left\| Cw^{(i)} - x^{(i)} \right\|_2^2$$

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

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

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 $c_1, \ldots, c_d$ formano una base ortonormale:

$$c_i^\top c_j = \begin{cases} 1 \text{ if } i = j \\ 0 \text{ if } i \neq j \end{cases}$$

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

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

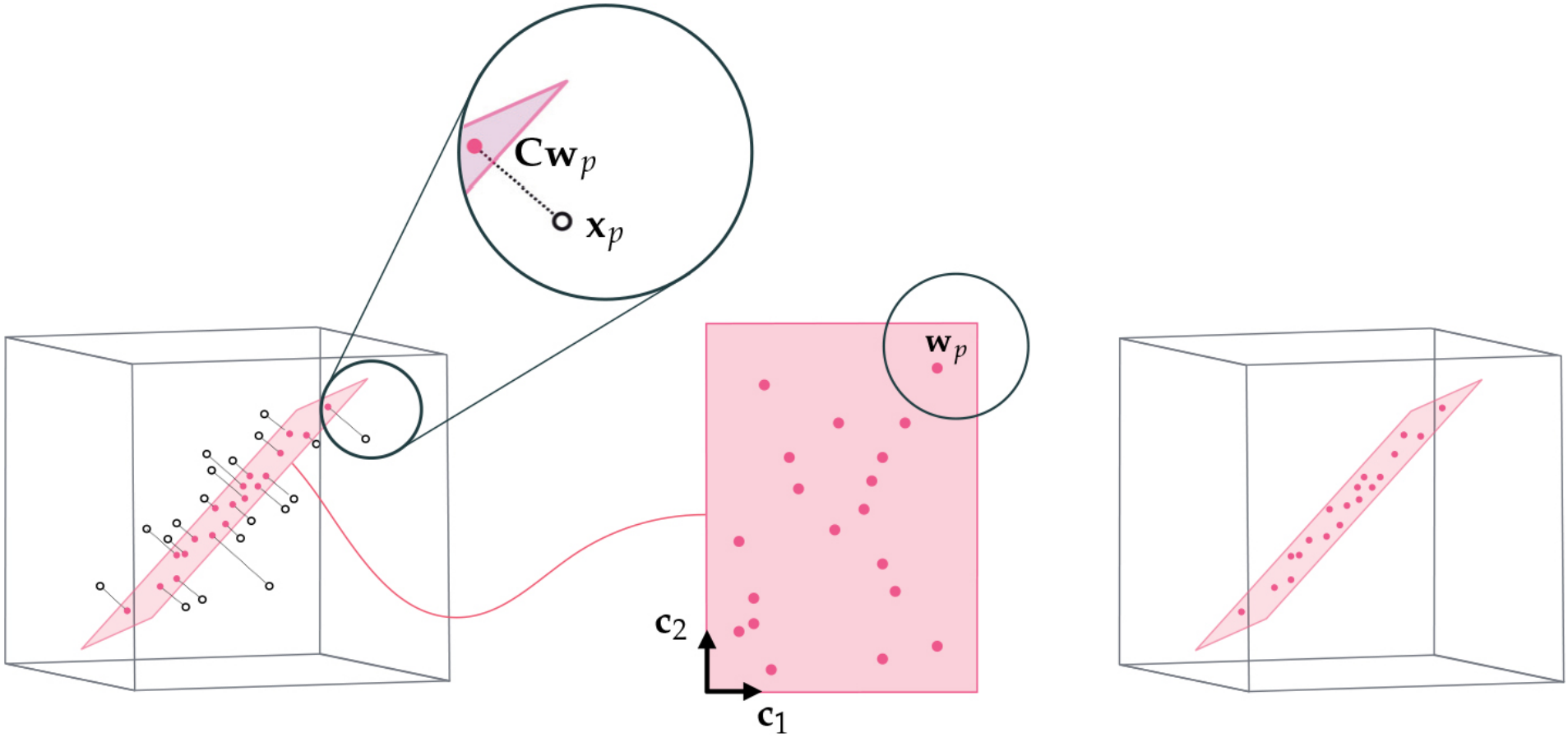
$$w^{(i)} = C^\top x^{(i)}$$

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

Formula di autocodifica [autoencoder formula]

$$x^{(i)} = CC^\top x^{(i)}$$

Codifica imperfetta dei dati con uno spanning set



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

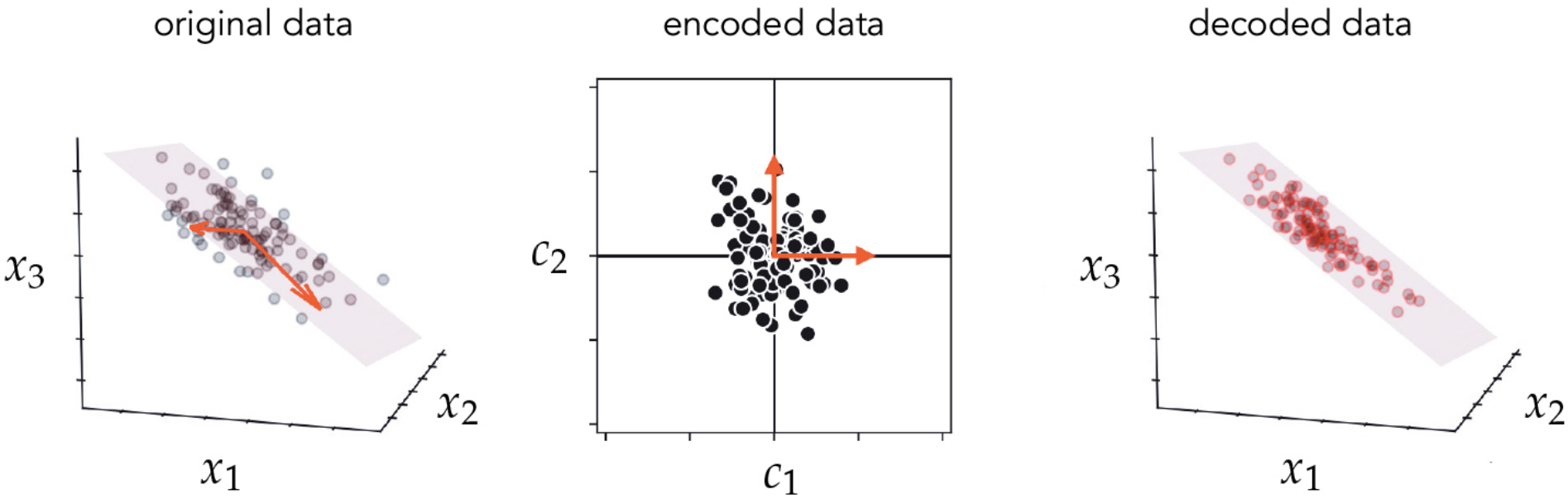
Formula di autocodifica [autoencoder formula]

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

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?



$$\text{Criterio:} \qquad \text{minimize}\, g(W, C) = \frac{1}{m} \sum_{i=1}^{m} \left\| Cw^{(i)} - x^{(i)} \right\|_2^2$$

g ora è una funzione (non convessa ma *biconvessa*) sia di $W = (w^{(1)} \dots w^{(m)})$ che di C

Autoencoder lineare

Aggiungendo l'assunzione che i vettori c siano ortonormali tra loro abbiamo, come prima, $w^{(i)} = C^\top x^{(i)}$ e quindi

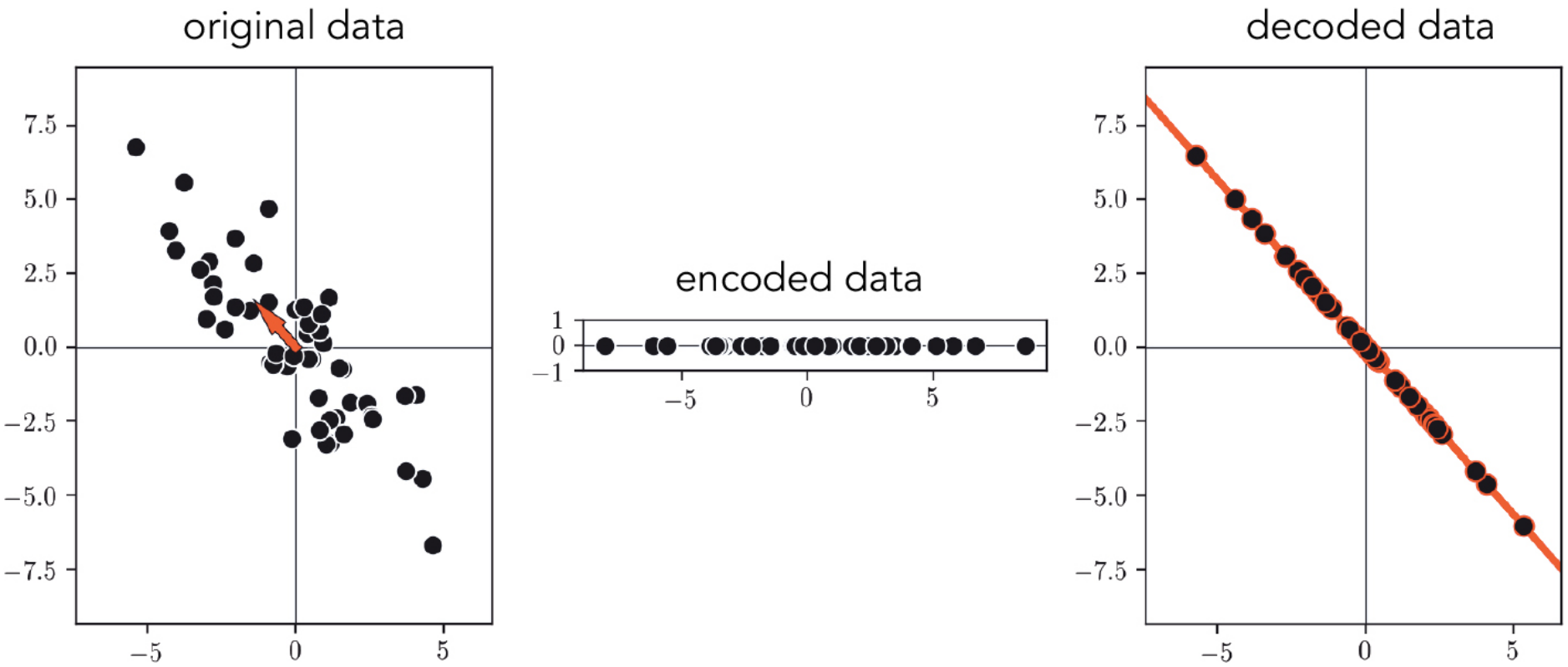
$$g(C) = \frac{1}{m} \sum_{i=1}^{m} \left\| CC^\top x^{(i)} - x^{(i)} \right\|_2^2$$

è ora esprimibile come funzione della sola matrice C

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

NB. Sebbene la matrice cercata sia ortonormale, in effetti non è necessario forzare questo vincolo perché si può mostrare che *tutti* i minimizzanti di $g(C)$ sono ortonormali.

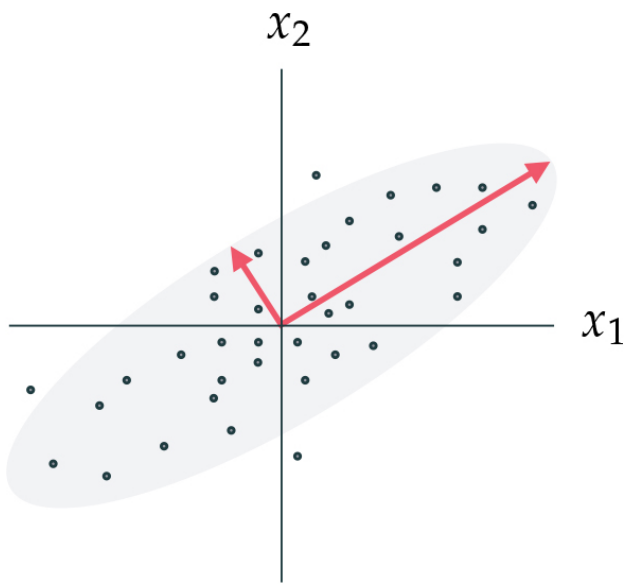
Autoencoder lineare



Componenti principali di un dataset

Possono esistere molte matrici C che minimizzano la funzione g

Le *componenti principali* forniscono uno di questi minimi



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 $c_1, \ldots, c_{k-1}$, lungo la quale la varianza dei dati massima

Definizione delle componenti principali

Siano $x^{(1)}, \ldots, x^{(m)}$ 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 $\Sigma = \frac{1}{m} X^\top X \in \mathbb{R}^{d \times d}$

Essendo una matrice simmetrica, essa ammette una diagonalizzazione

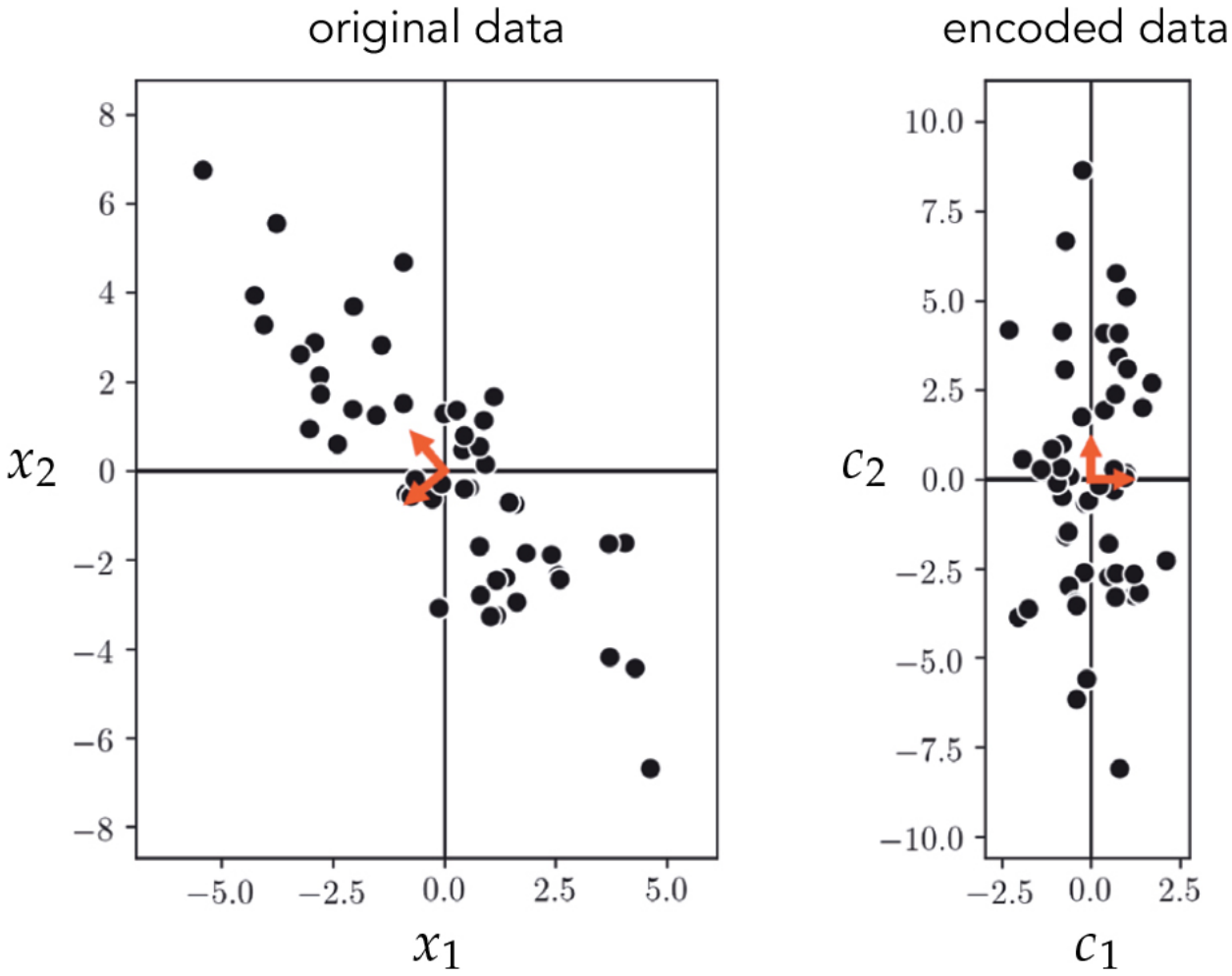
$$\Sigma = V D V^\top$$

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 Σ) quantificano la varianza dei dati lungo ciascuna componente (colonna di V)

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

Codifica tramite componenti principali



Algoritmo PCA

Principal Components Analysis (PCA)

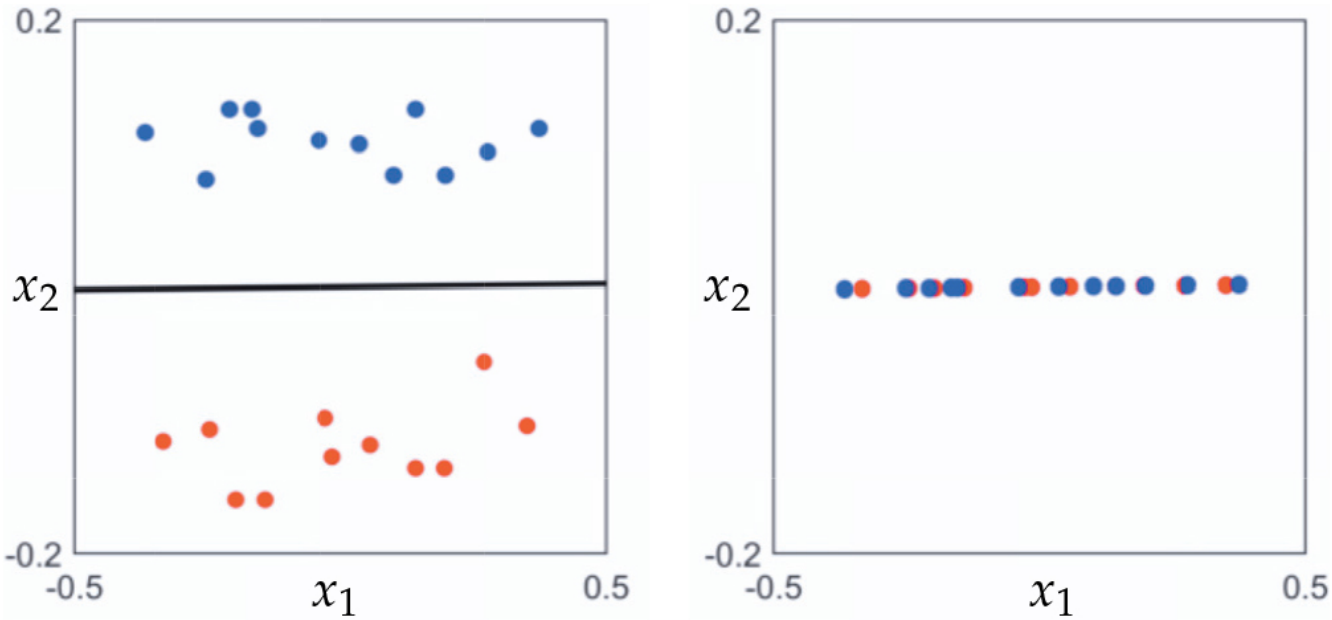
Dati: m vettori di input $x^{(1)}, \ldots, x^{(m)} \in \mathbb{R}^d$ ed un intero $K \leq d$
Trova: le K componenti principali del dataset

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 $i = 1, \ldots, m$
3. Calcola la matrice di covarianza: $\Sigma \leftarrow \frac{1}{m} X^\top X$
4. Diagonalizza la matrice di covarianza: $\Sigma = VDV^\top$
5. Riordina gli autovalori d_{kk} (e i corrispondenti autovettori v_k) in modo che
$$d_{11} \geq d_{22} \geq \ldots \geq d_{kk} \geq \ldots$$
6. Restituisci i vettori $v_1, v_2, \ldots, v_K$ (e i corrispondenti autovalori)

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

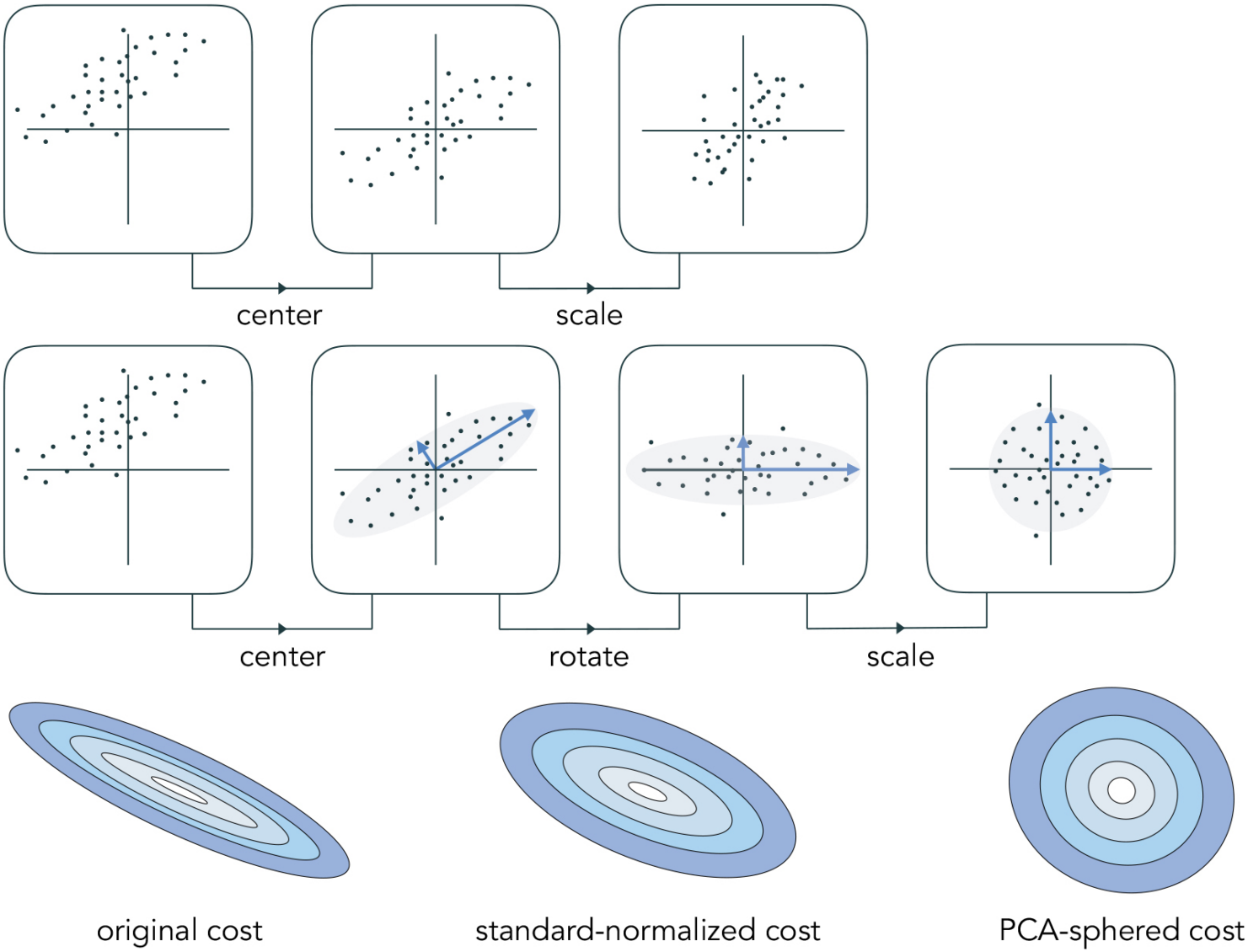
Un esempio disastroso

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

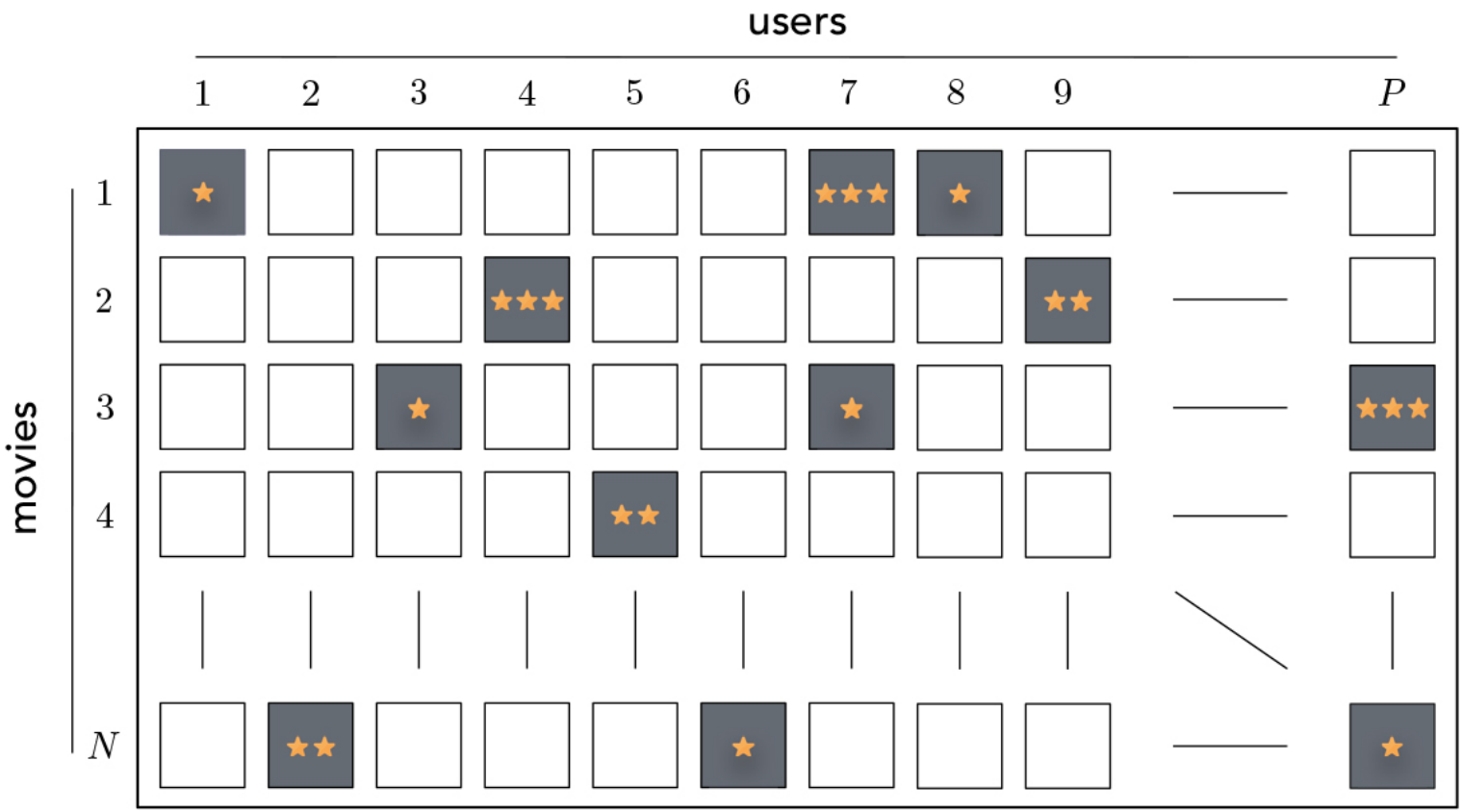
1. Centra: $x^{(i)} \leftarrow x^{(i)} - \mu$ dove μ è la media degli $x^{(i)}$
2. Scala: $x_j^{(i)} \leftarrow \frac{x_j^{(i)}}{\sqrt{\sigma_j^2}}$ dove $\sigma_j^2 = (1/m)\sum_i x_j^{(i)2}$ è la varianza della j-esima variabile

Sferificazione PCA [*PCA-sphering*]

1. Centra: $x^{(i)} \leftarrow x^{(i)} - \mu$ dove μ è la media degli $x^{(i)}$
2. Ruota: $x^{(i)} \leftarrow V^\top x^{(i)}$
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 Ω_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

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

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

Sistemi di raccomandazione [*Recommender systems*]

Come minimizzare

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

Non si può più forzare l'ortonormalità di C come in PCA. Ma $g(\cdot, C)$ è convessa per C fissata e $g(W, \cdot)$ è convessa per W fissata. Si può quindi usare uno schema di *minimizzazione alternata* del seguente tipo:

- Ripeti per $t = 1, 2, \ldots$:
  - Fissa C, trova W col metodo del gradiente per minimizzare $g(\cdot, C)$
  - Fissa W, trova C col metodo del gradiente per minimizzare $g(W, \cdot)$

Principal Component Analysis in `scikit-learn`

Approccio	Iperparametri	Interfaccia `scikit-learn`
PCA	K	`sklearn.decomposition.PCA`
Kernel PCA	K, kernel	`sklearn.decomposition.KernelPCA`
Sparse PCA	K, α	`sklearn.decomposition.SparsePCA`

α è un iperparametro che controlla la sparsità delle componenti ricostruite: una maggiore sparsità favorisce una maggior quantità di coordinate nulle