IN550 Machine Learning

Apprendimento non supervisionato: PCA e riduzione della dimensionalità

Vincenzo Bonifaci

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)

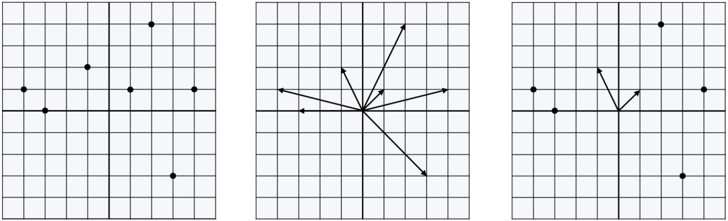
I metodi di *riduzione della dimensionalità* hanno invece come obiettivo la riduzione del numero di variabili (d)

Esempi:

- Analisi delle componenti principali (*Principal Component Analysis* o *PCA*)
- Proiezioni casuali
- Compressed sensing

Simili ai metodi di *riduzione delle feature* discussi nell'apprendimento supervisionato, ma nel 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 un sottospazio lineare

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

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

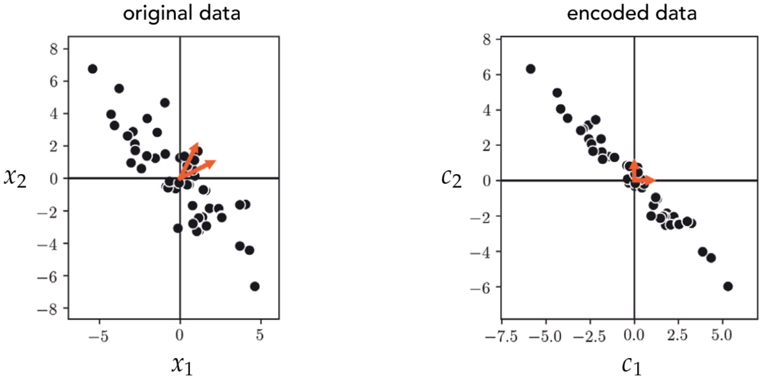
I vettori $w^{(i)}$ possono essere determinati minimizzando la funzione

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

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:

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

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

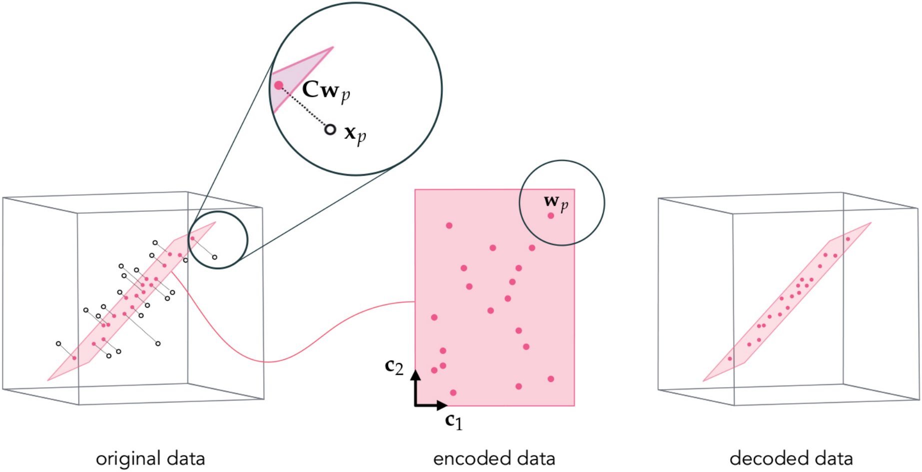
$$w^{(i)} = C^\top x^{(i)}$$

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

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 approssimata

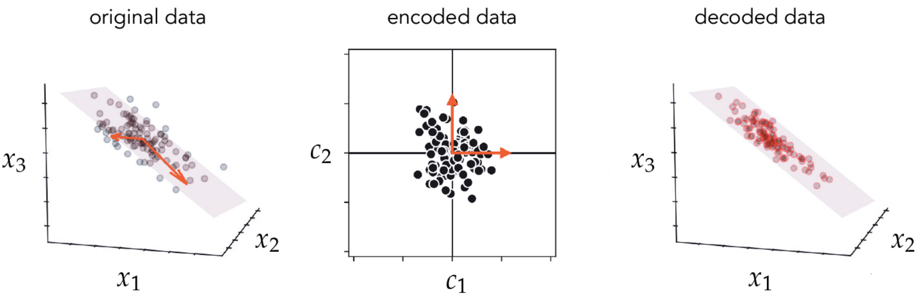
$$x^{(i)} \approx CC^\top x^{(i)}$$

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

La funzione g misura la distorsione media della proiezione

Apprendimento di uno spanning set

Come scegliamo le colonne di C?



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

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

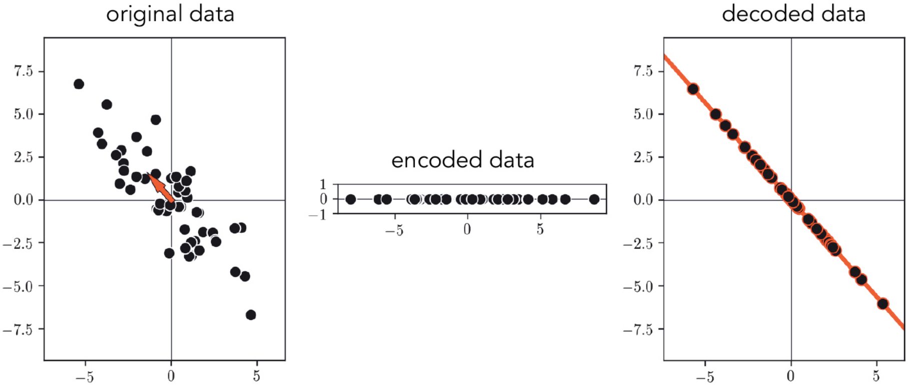
$$g(C) = \frac{1}{m} \sum_{i=1}^{m} \left\| CC^\top x^{(i)} - x^{(i)} \right\|_2^2$$

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.

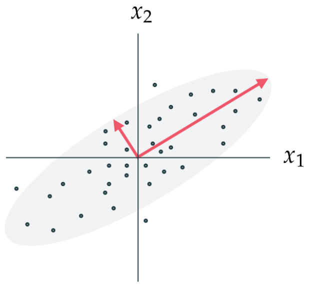
Autoencoder lineare



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

Essendo una matrice simmetrica, essa ammette una diagonalizzazione

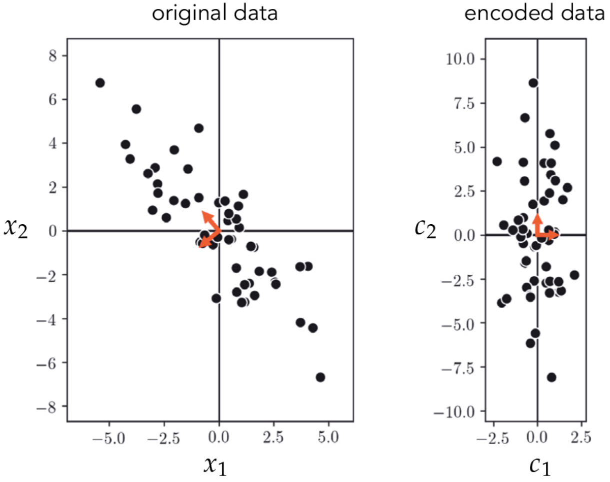
$$\Sigma = V\, D\, V^\top$$

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 $x^{(1)}, \dots, x^{(m)} \in \mathbb{R}^d$ 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: $\Sigma \leftarrow \frac{1}{m} X^\top X$
4. Diagonalizza la matrice di covarianza: $\Sigma = VDV^\top$
5. Riordina gli autovalori d_{kk} (e i corrispondenti autovettori v_k) in modo che
$$d_{11} \geq d_{22} \geq \dots \geq d_{kk} \geq \dots$$
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

PCA come fattorizzazione di matrici

PCA può essere visto come la ricerca di una fattorizzazione $X^\top \approx CW$

Obiettivo:

$$\underset{C,W}{\text{minimize}} \frac{1}{m} \sum_{i=1}^{m} \left\| Cw^{(i)} - x^{(i)} \right\|_2^2 = \frac{1}{m} \left\| CW - X^\top \right\|_F^2$$

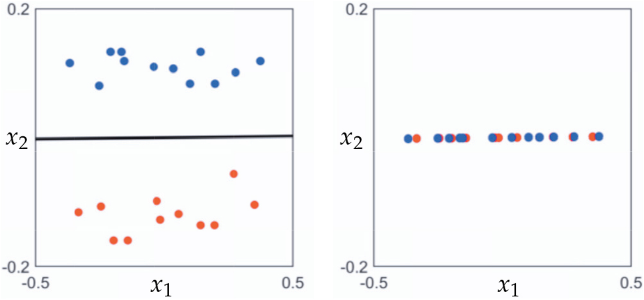
Vincoli:

$$C^\top C = I$$

$$C \in \mathbb{R}^{d \times K}, W \in \mathbb{R}^{K \times m}$$

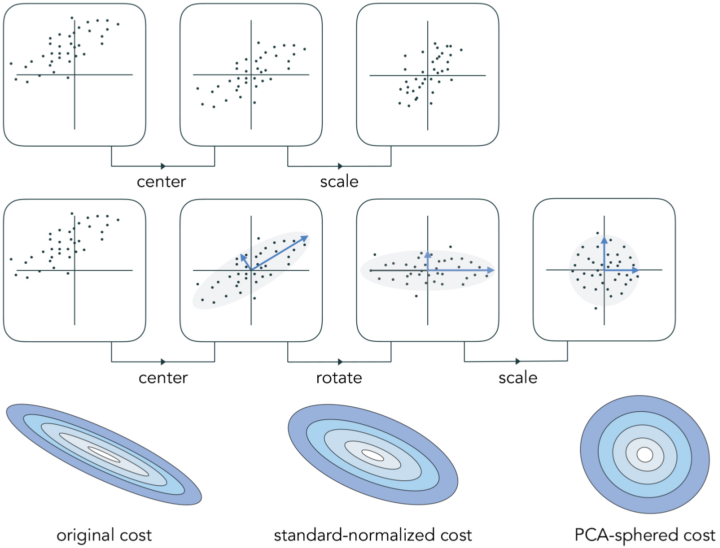
Un esempio disastroso

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)

Variante: Sparse PCA

Obiettivo:

$$\underset{C,W}{\text{minimize}} \frac{1}{m} \left\| CW - X^\top \right\|_F^2 + \lambda \left\| C \right\|_1$$

Vincoli:

$$\left\| W_j \right\|_2 \leq 1 \qquad \text{per } j = 1, 2, \ldots, K$$

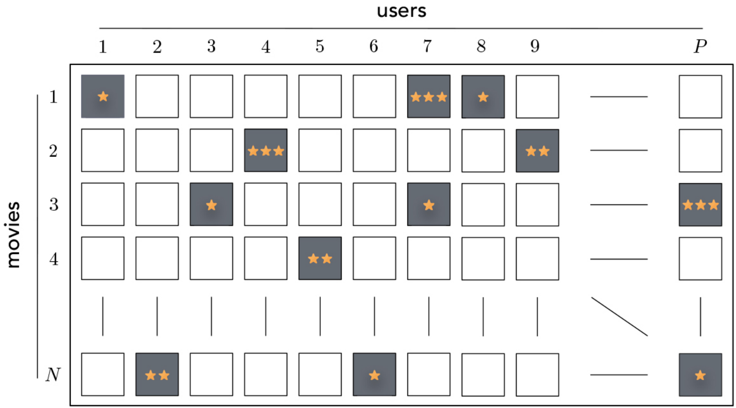
$$C \in \mathbb{R}^{d \times K}, W \in \mathbb{R}^{K \times m}$$

Il termine di regolarizzazione $\lambda \left\| C \right\|_1$ incentiva combinazione lineari sparse

Rispetto a PCA, migliora l'interpretabilità in termini delle variabili di input originali

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.

Come minimizzare

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

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 $g(W, \cdot)$

Anche il problema dei sistemi di raccomandazione può essere interpretato come una fattorizzazione di matrici, con l'obiettivo

$$g(W, C) = \frac{1}{m} \left\| \{CW - X^\top\}_\Omega \right\|_F^2$$

PCA in `scikit-learn`

Moduli: `sklearn.decomposition`

	Iperparametri	Interfaccia `scikit-learn`
PCA	K	`PCA(n_components)`
Kernel PCA	K, kernel	`KernelPCA(n_components, kernel)`
Sparse PCA	K, α	`SparsePCA(n_components, alpha)`

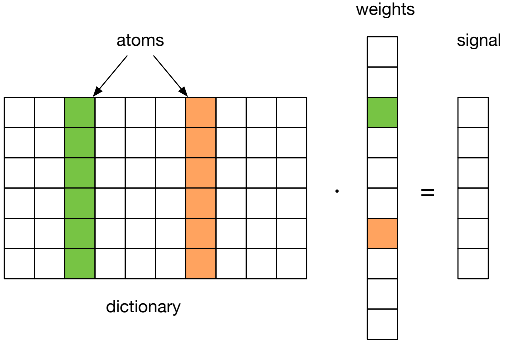
α è un iperparametro che controlla la sparsità delle componenti ricostruite: una maggiore sparsità favorisce una maggior quantità di coordinate nulle

Sistemi di raccomandazione nella libreria `surprise`

Moduli: `surprise.prediction_algorithms.matrix_factorization`

	Iperparametri	Interfaccia `surprise`
Sistema di raccomandazione (algoritmo SVD)	K	`SVD(n_factors,biased=False)`

Dati $C \in \mathbb{R}^{d \times K}$, $x^{(i)} \in \mathbb{R}^d$: risolvere $Cw^{(i)} = x^{(i)}$ per $w^{(i)} \in \mathbb{R}^K$



Se $K < d$ il sistema è sovradeterminato e non ha soluzione esatta
In tal caso si cerca di soddisfare $Cw^{(i)} \approx x^{(i)}$

Sparse dictionary learning (sparse coding)

Obiettivo:

$$\underset{C,W}{\text{minimize}}\, \frac{1}{m} \left\| CW - X^\top \right\|_F^2 + \lambda \left\| W \right\|_1$$

Vincoli:

$$\left\| C_j \right\|_2 \leq 1 \qquad \text{per } j = 1, 2, \ldots, K$$

$$C \in \mathbb{R}^{d \times K}, W \in \mathbb{R}^{K \times m}$$

Il termine di regolarizzazione $\lambda \left\| W \right\|_1$ incentiva codifiche sparse

Cerca simultaneamente il dizionario (C) e i pesi (codifiche dei dati) (W)

Metodi di fattorizzazione di matrici

Tutti i seguenti metodi cercano fattorizzazioni $X^\top \approx CW$, ma con vincoli diversi:

Problema	**Vincoli su C e W**
PCA	C ortonormale (implica $W = C^\top$)
Sparse PCA	Ogni colonna di C è sparsa
Sistemi di raccomandazione	Nessun vincolo su C o W
	X è solo parzialmente nota
Clustering k-means	Ogni colonna di W è un vettore canonico
Sparse dictionary learning	Ogni colonna di W è sparsa
Fattorizzazione nonnegativa	C e W sono nonnegative

Altri metodi di fattorizzazione in scikit-learn

Moduli: `sklearn.decomposition`

	Iperparametri	Interfaccia `scikit-learn`
Dictionary Learning (batch)	K, α	`DictionaryLearning(n_components, alpha)`
Dictionary Learning (mini-batch)	K, α	`MiniBatchDictionaryLearning(n_components, alpha)`
Fattorizzazione nonnegativa (batch)	K	`NMF(n_components)`
Fattorizzazione nonnegativa (mini-batch)	K	`MiniBatchNMF(n_components)`

α è un iperparametro che controlla la sparsità delle codifiche