

Riduzione Dimensionalità

- Introduzione
 - Definizioni
 - Le Principali Tecniche
 - PCA vs LDA
- Principal Component Analysis (PCA)
- Linear Discriminant Analysis (LDA)
- t-SNE

Definizioni

Obiettivo dei metodi per la riduzione di dimensionalità (*dimensionality reduction*) è quello di eseguire un **mapping** dallo spazio iniziale \mathcal{R}^d a uno spazio di dimensione inferiore \mathcal{R}^k , $k < d$.

Può essere vista come una forma di compressione (con perdita di informazione). Obiettivo è **scartare le informazioni** non rilevanti o **meno rilevanti** per il problema di interesse:

- allevia i problemi collegati alla **curse of dimensionality**: operare in spazi ad elevata dimensionalità, a causa del fatto che i pattern sono molto sparsi, richiede ingenti moli di dati per l'addestramento.
- operare in spazi a dimensionalità inferiore rende più **semplice** addestrare algoritmi di machine learning. Scartando dati **ridondanti** (informazioni correlate) e **rumorosi** talvolta si migliorano anche le prestazioni.

Attenzione: riduzione di dimensionalità non significa mantenere alcune «dimensioni» e cancellarne altre, ma «**combinare**» le dimensioni in modo opportuno.

Le principali tecniche

Le più note tecniche di riduzione di dimensionalità (che vedremo) sono:

- **Principal Component Analysis (PCA)**: trasformazione **non-supervisionata** nota anche come Karhunen Loeve (KL) transform. Esegue un mapping **lineare** con l'obiettivo di **preservare** al massimo l'**informazione** dei pattern.
- **Linear Discriminant Analysis (LDA)**: il mapping è ancora **lineare**, ma in questo caso è **supervisionato**. Mentre **PCA** privilegia le dimensioni che **rappresentano** al meglio i pattern, **LDA** privilegia le dimensioni che **discriminano** al meglio i pattern del TS.
- **t-distributed Stochastic Neighbor Embedding (t-SNE)**: trasformazione **non lineare** e **non supervisionata**, specificatamente ideata per ridurre dimensionalità a 2 o 3 dimensioni onde poter **visualizzare** dati multidimensionali.

Altre tecniche di interesse:

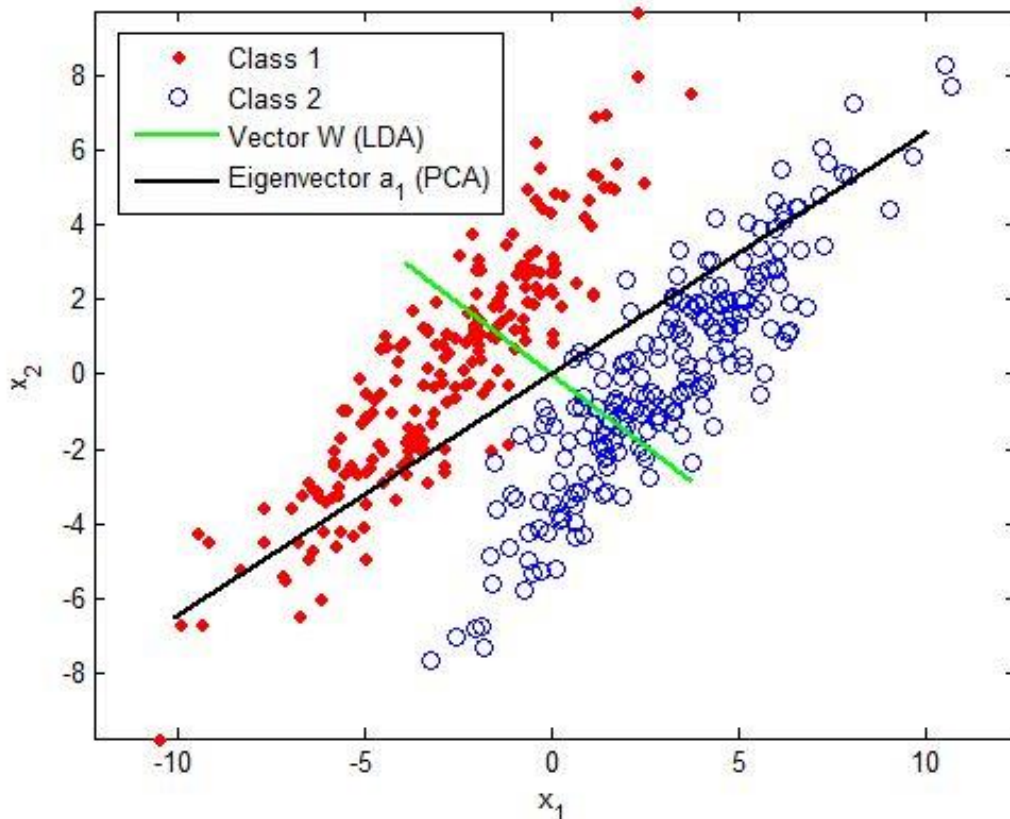
- **Independent Component Analysis (ICA)**: trasformazione lineare orientata a proiettare i pattern su una base di componenti (statisticamente **indipendenti**).
- **Kernel PCA**: simile a PCA ma più potente perché il mapping è **non-lineare**. Utilizza un «trucco» simile a quello che permette di passare da SVM lineare a SVM non lineare.
- **Local Linear Embedding (LLE)**: trasformazione **non-lineare** che invece di calcolare un mapping «globale», considera relazioni tra gruppi di pattern vicini.

Esempio PCA vs LDA

In figura due esempi di riduzione di dimensionalità da $d = 2$ a $k = 1$ dimensione:

- Il segmento nero che identifica la **soluzione PCA** è l'iperpiano sul quale proiettando i pattern (indipendentemente dalla loro classe) conserviamo al massimo l'informazione.
- Il segmento verde che identifica la **soluzione LDA** è l'iperpiano sul quale proiettando i pattern siamo in grado di distinguere al meglio le due classi (pattern rossi contro blu).

Entrambi sono mapping lineari $\mathcal{R}^2 \rightarrow \mathcal{R}^1$ ma la soluzione (retta) è profondamente diversa.



Principal Component Analysis (PCA)

Dato un training set $TS = \{\mathbf{x}_i \in \mathbb{R}^d, i = 1 \dots n\}$, siano:

$$\bar{\mathbf{x}} = \frac{1}{n} \sum_{i=1 \dots n} \mathbf{x}_i$$

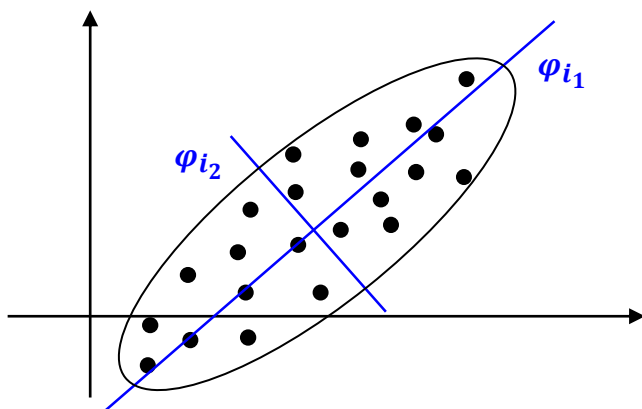
$$\Sigma = \frac{1}{n-1} \sum_{i=1 \dots n} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^t$$

il **vettore medio** $\in \mathbb{R}^d$ e la **matrice di covarianza** $\in \mathbb{R}^{d \times d}$ (definizioni simili a quelle usate per il classificatore di Bayes parametrico con multinormali. La divisione per $n-1$ invece di n dovuta a correzione di Bessel per caso unbiased).

allora per un dato k ($k < d, k < n, k > 0$), lo spazio k dimensionale ($S_{\bar{\mathbf{x}}, \Phi_k}$) è univocamente definito dal **vettore medio** e dalla **matrice di proiezione** $\Phi_k \in \mathbb{R}^{d \times k}$ le cui **colonne** sono costituite dagli **autovettori** di Σ corrispondenti ai k più grandi **autovalori**:

$$\Phi_k = [\boldsymbol{\varphi}_{i_1}, \boldsymbol{\varphi}_{i_2} \dots \boldsymbol{\varphi}_{i_k}] \text{ con } \lambda_{i_1} \geq \lambda_{i_2} \geq \dots \lambda_{i_k} \geq \dots \lambda_{i_d}$$

$\boldsymbol{\varphi}_{i_r}$ **autovettore** di Σ corrispondente all'**autovalore** λ_{i_r} , $r = 1 \dots d$



$\boldsymbol{\varphi}_{i_1}$ indica la direzione di maggior varianza nel training set TS

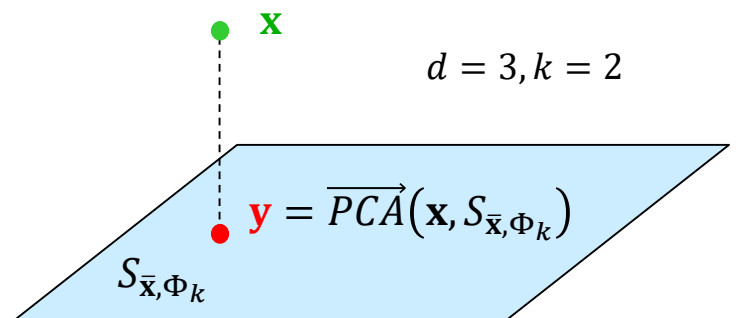
I primi k autovettori sono detti **componenti principali**

PCA: proiezione e retroproiezione

- **Proiezione** (\rightarrow): una volta determinato lo spazio PCA, la proiezione di un pattern su tale spazio è semplicemente la **proiezione geometrica** di un vettore sull'iperpiano che definisce lo spazio. **In realtà** la vera proiezione geometrica è un vettore che ha la stessa dimensionalità del vettore originale mentre **in questo contesto** indichiamo con **proiezione** il vettore (ridotto) nello **spazio PCA**. Matematicamente questa operazione è eseguita come prodotto della **matrice di proiezione trasposta** per il **pattern** al quale è preventivamente **sottratta la media**.

$$\overrightarrow{PCA}: \mathbb{R}^d \rightarrow \mathbb{R}^k$$

$$\overrightarrow{PCA}(\mathbf{x}, S_{\bar{\mathbf{x}}, \Phi_k}) = \Phi_k^t (\mathbf{x} - \bar{\mathbf{x}})$$

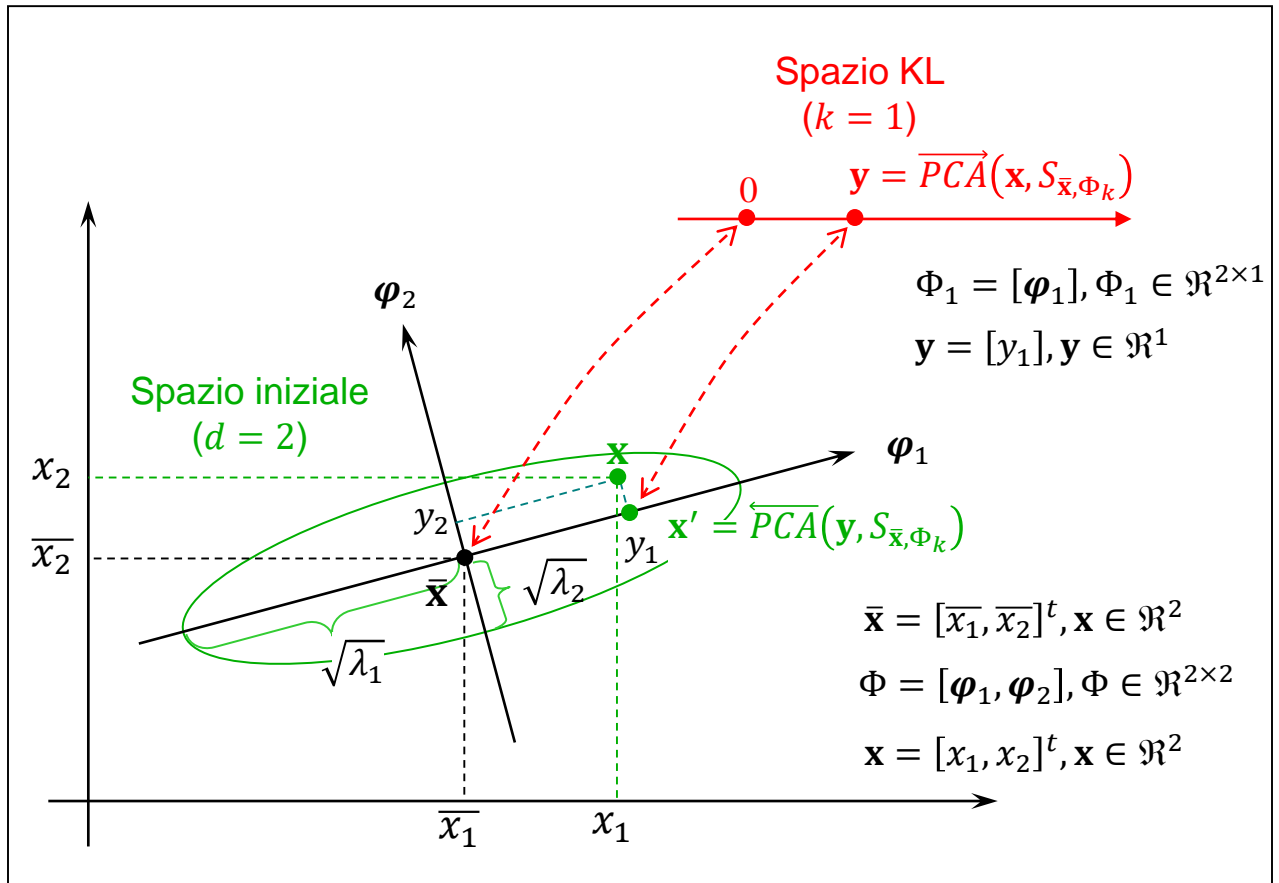


- **Retro-proiezione** (\leftarrow): Dato un vettore nello spazio PCA, la sua retro-proiezione verso lo spazio originale si ottiene moltiplicando il vettore per la **matrice di proiezione** e sommando il **vettore medio**. Questa trasformazione **non sposta** spazialmente il vettore, **che giace ancora sullo spazio PCA**, ma opera un cambiamento di coordinate che ne permette la codifica in termini delle d componenti dello spazio originale.

$$\overleftarrow{PCA}: \mathbb{R}^k \rightarrow \mathbb{R}^d$$

$$\overleftarrow{PCA}(\mathbf{y}, S_{\bar{\mathbf{x}}, \Phi_k}) = \Phi_k \mathbf{y} + \bar{\mathbf{x}}$$

PCA: esempio riduzione 2→1



- L'**ellisse** rappresenta la **distribuzione dei pattern** nel training set.
- φ_1 e φ_2 sono gli **autovettori** della matrice di covarianza.
- Gli **autovalori** λ_1 e λ_2 sono le varianze della distribuzione lungo gli assi φ_1 e φ_2 .
- y_1 e y_2 sono le proiezioni di x sugli assi φ_1 e φ_2 .
- Se λ_2 è piccolo, x può essere approssimato con x' (retroproiezione di y) senza perdite significative di informazione.
- Si **può dimostrare** che tra tutte le riduzioni di dimensionalità lineari **PCA** è quella che **preserva al massimo l'informazione** dei vettori originali.

PCA: scelta di k

- Talvolta la scelta di k è obbligata: ad esempio per la visualizzazione 2D o 3D dei dati.
- Quando invece l'obiettivo è quello di scartare informazione inutile e dati correlati **mantenendo gran parte del contenuto informativo** si può scegliere k nel modo seguente:
 - Fissata una percentuale t del contenuto informativo che si vuole preservare (es. $t = 95\%$) si sceglie il minimo valore di k per cui la somma dei più grandi k autovalori, rispetto alla somma di tutti gli autovalori, è maggiore o uguale a t .
 - Considerando gli autovalori ordinati in ordine decrescente:

$$k = \arg \min_z \left\{ \frac{\sum_{i=1 \dots z} \lambda_i}{\sum_{i=1 \dots d} \lambda_i} \geq t \right\}$$

Infatti, ricordando che gli autovalori denotano la varianza lungo i diversi assi, il rapporto nella formula indica la varianza «conservata» rispetto alla varianza totale.

PCA: codifica di un'immagine

i primi 8 autovettori o componenti principali
(denominati *eigenfaces* nell'applicazione al riconoscimento volto)



Immagine originale



Ricostruzione (retroproiezione)

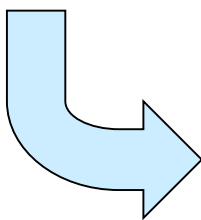
$$\mathbf{x} \in \mathcal{R}^{16500}$$

$$\mathbf{x}' \in \mathcal{R}^{16500}$$

$$\mathbf{y} = \overrightarrow{PCA}(\mathbf{x}, S_{\bar{\mathbf{x}}}, \Phi_{15})$$

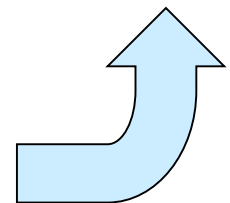
$$\mathbf{x}' = \overleftarrow{PCA}(\mathbf{y}, S_{\bar{\mathbf{x}}}, \Phi_{15})$$

$$\mathbf{y} \in \mathcal{R}^{15}$$



proiezione

-2532	2193	-2179	2099	491
427	-324	961	35	-40
-149	-624	317	-158	-142



retro-proiezione

Calcolo PCA in pratica

Per d elevato (tipico nel caso di immagini, audio, ecc.) la matrice di covarianza può essere molto grande.

- per $d = 16500$, $\Sigma \in \mathfrak{R}^{16500 \times 16500}$, oltre **272 milioni di valori** !
- se $n \ll d$, è conveniente calcolare la matrice di proiezione (primi k autovettori) attraverso **decomposizione SVD** della matrice **rettangolare** dei pattern centralizzati $\mathbf{X} \in \mathfrak{R}^{d \times n}$ senza passare per la matrice di covarianza (vedi [1]).

$$\mathbf{X} = \left[\begin{array}{c|c|c|c} \mathbf{x}_1 - \bar{\mathbf{x}} & \mathbf{x}_2 - \bar{\mathbf{x}} & \cdots & \mathbf{x}_n - \bar{\mathbf{x}} \end{array} \right]$$

Attenzione

*formato trasposto rispetto a \mathbf{X} usata in regressione.
Ogni pattern una colonna.*

decomposizione SVD per $d > n$: $\mathbf{X} = \mathbf{U}\mathbf{\Gamma}\mathbf{V}^t$, con $\mathbf{U} \in \mathfrak{R}^{d \times n}$ ortonormale, con $\mathbf{V} \in \mathfrak{R}^{n \times n}$ ortonormale, $\mathbf{\Gamma} \in \mathfrak{R}^{n \times n}$ diagonale.

$$\Sigma = \frac{1}{n} \mathbf{X}\mathbf{X}^t = \frac{1}{n} \mathbf{U}\mathbf{\Gamma}\mathbf{V}^t \mathbf{V}\mathbf{\Gamma}\mathbf{U}^t = \frac{1}{n} \mathbf{U}\mathbf{\Gamma}^2 \mathbf{U}^t$$

*decomposizione
spettrale della
matrice quadrata Σ*

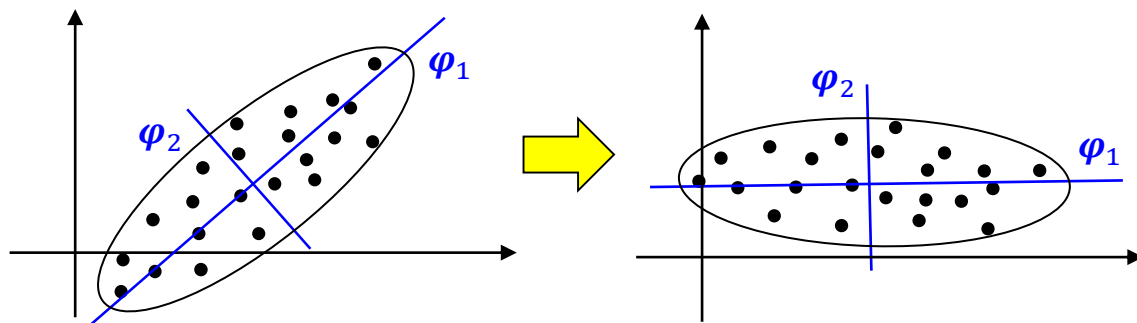
Autovettori e **autovalori** di Σ possono dunque essere ottenuti dalle colonne di \mathbf{U} (*vettori singolari sinistri di \mathbf{X}*) e corrispondenti elementi diagonali di $\mathbf{\Gamma}$ al quadrato (*valori singolari al quadrato di \mathbf{X}*).

[1] R. Madsen, L. Hansen, O. Winther, "Singular Value Decomposition and Principal Component Analysis", http://www2.imm.dtu.dk/pubdb/views/edoc_download.php/4000/pdf/imm4000.pdf

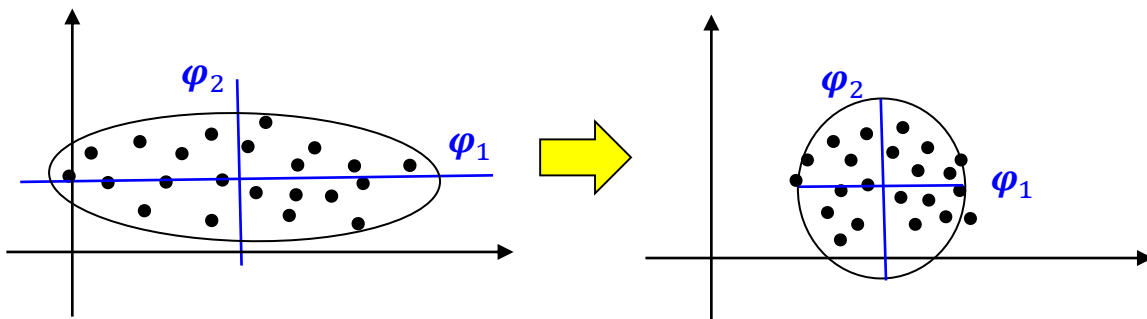
PCA Whitening

È una tecnica di **pre-normalizzazione** dei dati, che:

- **Rimuove** le correlazioni tra le dimensioni, ruotando la nuvola di punti per allineare gli assi di variazione principale dei dati (autovettori) agli assi cartesiani.



- **Sfericizza** l'ellissoide, uniformando le varianze (denotate dagli autovalori) a 1 lungo tutti gli assi

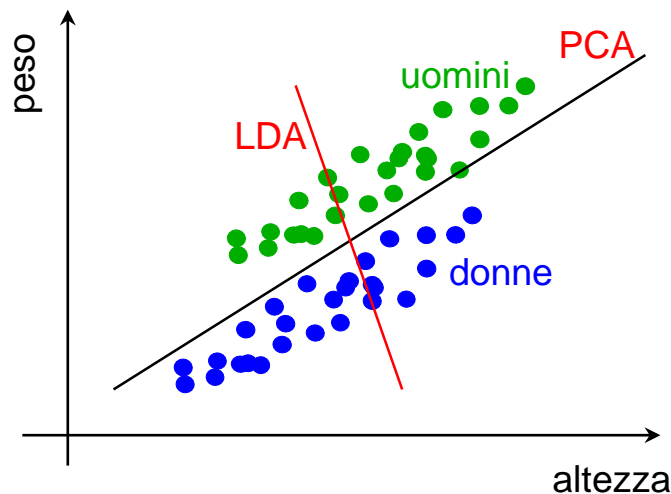


Dopo aver proiettato i pattern sullo spazio PCA (definito dai primi k autovettori) è sufficiente **dividere ogni dimensione** per la **radice quadrata dell'autovalore** corrispondente (deviazione standard).

La matrice di covarianza dei dati normalizzati è l'identità.

Linear Discriminant Analysis (LDA)

- **Riduzione** di dimensionalità lineare e **supervisionata** il cui obiettivo è **massimizzare** la **separazione tra le classi** (che nel TS sono etichettate). L'esempio seguente mostra che al fine della **discriminazione** la soluzione ottimale può essere anche **molto diversa** dalla soluzione PCA.



- Per formulare il **critero di ottimizzazione** di massima separazione tra le classi sono definite le seguenti **matrici di scattering** (in italiano “sparpagliamento”):
 - **within-class** S_w : indica come i **vettori sono scattered** rispetto al **centro delle classi** (ciascuno rispetto alla propria classe).
 - **between-class** S_b : indica come i **centri delle classi** sono scattered **rispetto al centro generale** della distribuzione (ovvero quanto le classi sono scattered).

Una matrice di scatter **si calcola** come una matrice di covarianza senza normalizzare per il numero di pattern

Calcolo LDA

Dato un training set **TS** contenente n pattern $(\mathbf{x}_1, y_1) \dots (\mathbf{x}_n, y_n)$, dove $\mathbf{x}_i \in \mathbb{R}^d$ sono i **pattern multidimensionali** e $y_i \in [1 \dots s]$ le **etichette** delle s classi. Siano n_i e $\bar{\mathbf{x}}_i$ il numero di pattern e il vettore medio della classe i -esima. Allora le **matrici di scattering** sono definite come:

■ **within-class:**

$$\mathbf{S}_w = \sum_{i=1 \dots s} \mathbf{S}_i, \quad \mathbf{S}_i = \sum_{\mathbf{x}_j | y_j=i} (\mathbf{x}_j - \bar{\mathbf{x}}_i)(\mathbf{x}_j - \bar{\mathbf{x}}_i)^t$$

← pattern della classe i -esima

■ **between-class:**

$$\mathbf{S}_b = \sum_{i=1 \dots s} n_i \cdot (\bar{\mathbf{x}}_i - \bar{\mathbf{x}}_0)(\bar{\mathbf{x}}_i - \bar{\mathbf{x}}_0)^t, \quad \bar{\mathbf{x}}_0 = \frac{1}{n} \sum_{i=1 \dots s} n_i \cdot \bar{\mathbf{x}}_i$$

← media globale

Tra i diversi **criteri di ottimizzazione** possibili quello più frequentemente utilizzato è la **massimizzazione** della quantità:

$$J_1 = \text{tr}(\mathbf{S}_w^{-1} \mathbf{S}_b) = \sum_{i=1 \dots d} \lambda_i$$

dove tr è la **traccia** (somma degli autovalori) della matrice. Il criterio è **intuitivo** in quanto cerca di **massimizzare** lo scattering tra le classi (\mathbf{S}_b) **minimizzando** al contempo (matrice inversa \mathbf{S}_w^{-1}) quello all'interno di ogni classe.

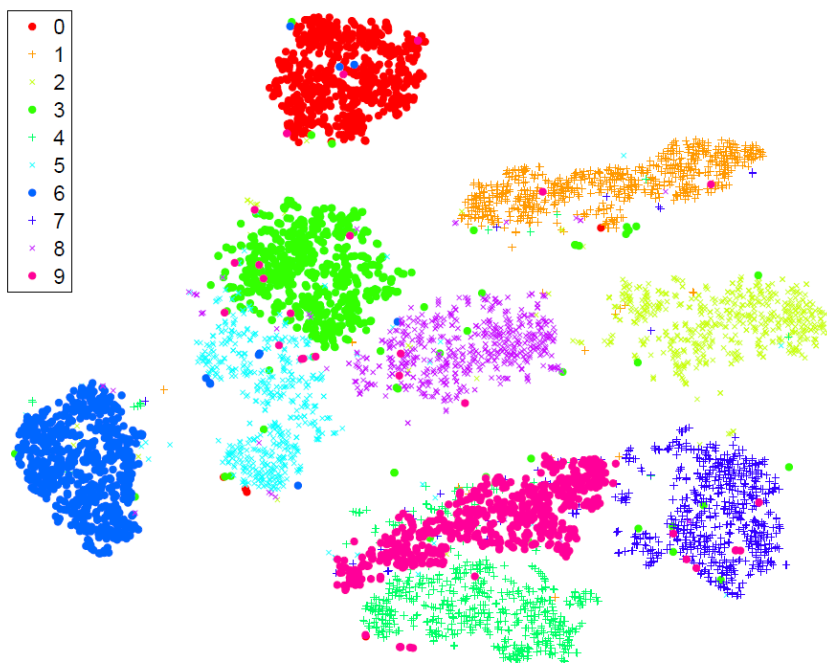
Si dimostra che per la **massimizzazione** di J_1 lo spazio LDA è definito (*analogia con PCA*) dagli **autovettori** relativi ai primi k ($k < n, k < s, k < d$) **autovalori** della matrice $\mathbf{S}_w^{-1} \mathbf{S}_b$.



valore massimo di $k = s - 1$

t-distributed Stochastic Neighbor Embedding (t-SNE)

- È una tecnica **non lineare** (**non supervisionata**) per la riduzione di dimensionalità introdotta nel 2008 da Van der Maaten e Hinton [1].
- Rappresenta lo stato dell'arte per la visualizzazione 2D o 3D di dati multidimensionali. Implementazione disponibile in molti linguaggi in [2].
- Anche PCA (con $k = 2$ o $k = 3$) può essere utilizzata a tale scopo, ma dati con distribuzioni spiccatamente **non multinormali** non possono essere efficacemente «ridotti» attraverso un mapping lineare.
- **Esempio**: visualizzazione 2D di MNIST (digit scritti a mano).



[1] L.J.P. van der Maaten and G.E. Hinton. Visualizing High-Dimensional Data Using t-SNE. *Journal of Machine Learning Research*, 2008.

[2] <https://lvdmaaten.github.io/tsne/>