

Introducción a la Inteligencia Artificial
Facultad de Ingeniería
Universidad de Buenos Aires



Índice

1. Fin del repaso de Numpy
2. kMeans
3. Teoría - Principal Component Analysis
 - a. Concepto
 - b. Demostración Matemática
 - i. Enfoque de máxima varianza
 - ii. Enfoque de error de reconstrucción mínimo
 - iii. Enfoque de variables latentes
 - c. Otros métodos
4. Práctica - Principal Component Analysis
 - a. PCA en Numpy
 - b. Ejercicios de Aplicación



Fin temas Clase 1



Algoritmos no supervisados

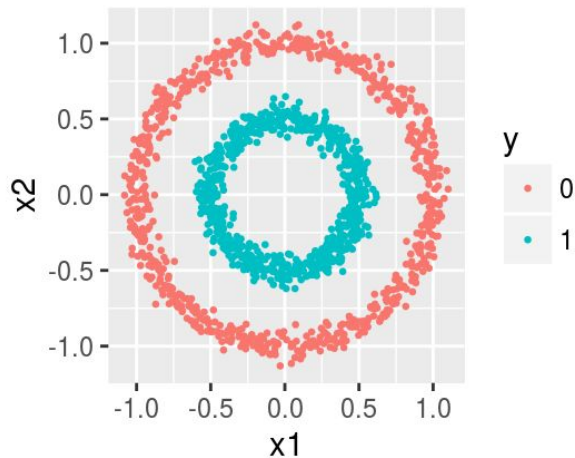


Clustering

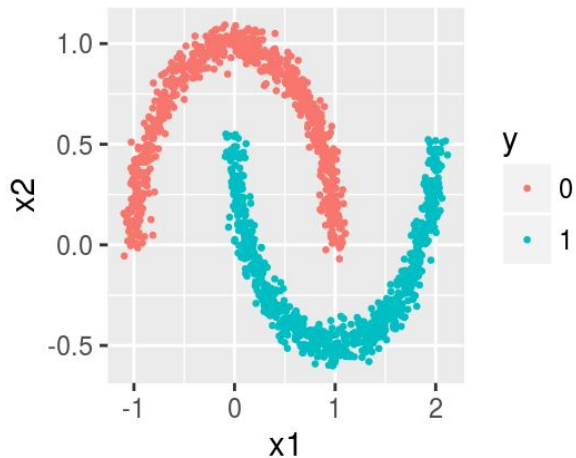
La clusterización o clustering, es el proceso de agrupar objetos en grupos de manera que sean más similares entre sí que con los objetos de otros clusters.

Para generar estos grupos existen diferentes técnicas y diferentes medidas de similaridad.

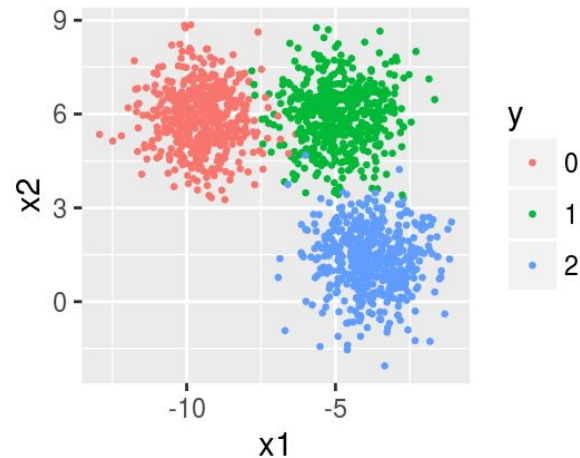
Circles



Moons



Blobs



kMeans

K-means es uno de los algoritmos más básicos en Machine Learning no supervisado. Es un algoritmo de **clusterización**, que agrupa los datos que comparten características similares. Recordemos que entendemos datos como n realizaciones del vector aleatorio X .

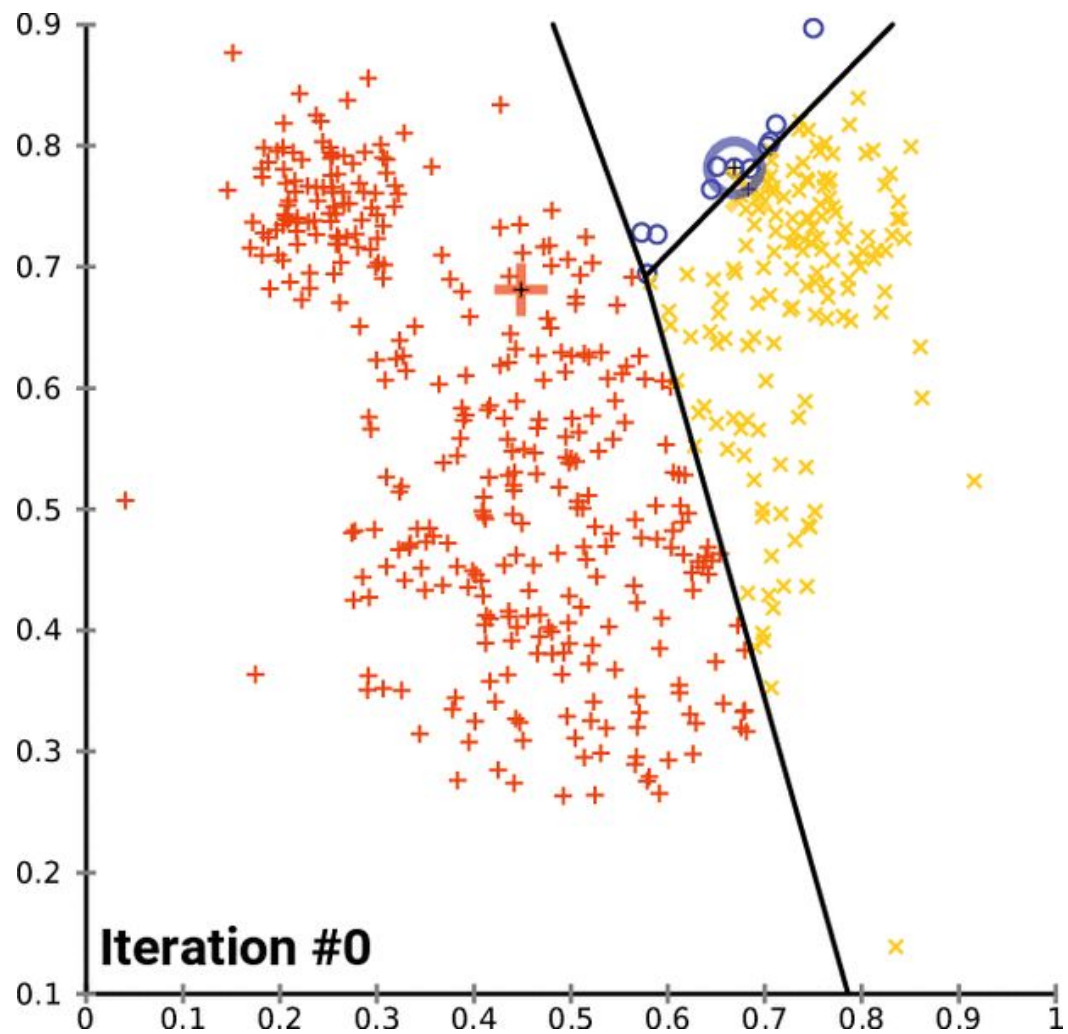
El algoritmo K-means funciona de la siguiente manera:

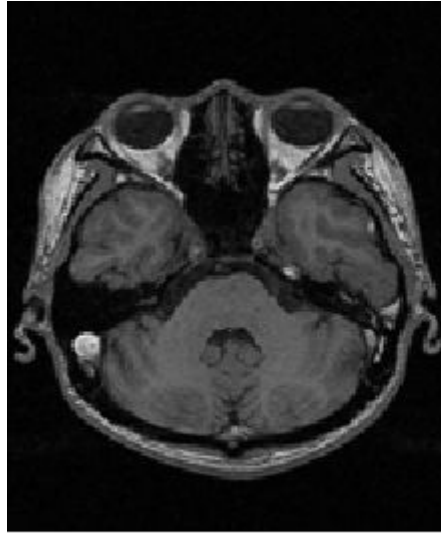
1. El usuario selecciona la cantidad de clusters a crear (n).
2. Se seleccionan n elementos aleatorios de X como posiciones iniciales de los centroides C .
3. Se calcula la distancia entre todos los puntos en X y todos los puntos en C .
4. Para cada punto en X se selecciona el centroide más cercano de C .
5. Se recalculan los centroides C a partir de usar las filas de X que pertenecen a cada centroide.
6. Se itera entre 3 y 5 una cantidad fija de veces o hasta que la posición de los centroides no cambie.

Implementar la función `def k_means(X, n)` de manera tal que al finalizar devuelva la posición de los centroides y a qué cluster pertenece cada fila de X .

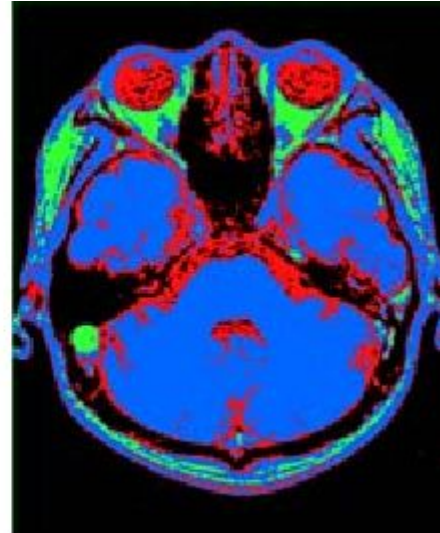
Hint: para (2) utilizar funciones de `np.random`, para (3) y (4) usar los ejercicios anteriores, para (5) es válido utilizar un `for`. Iterar 10 veces entre (3) y (5).





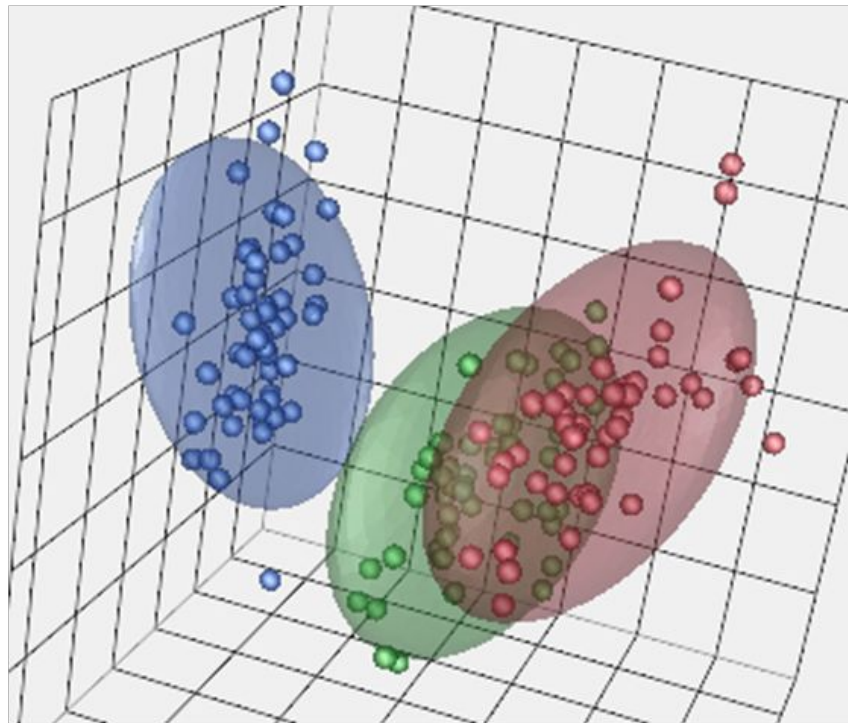
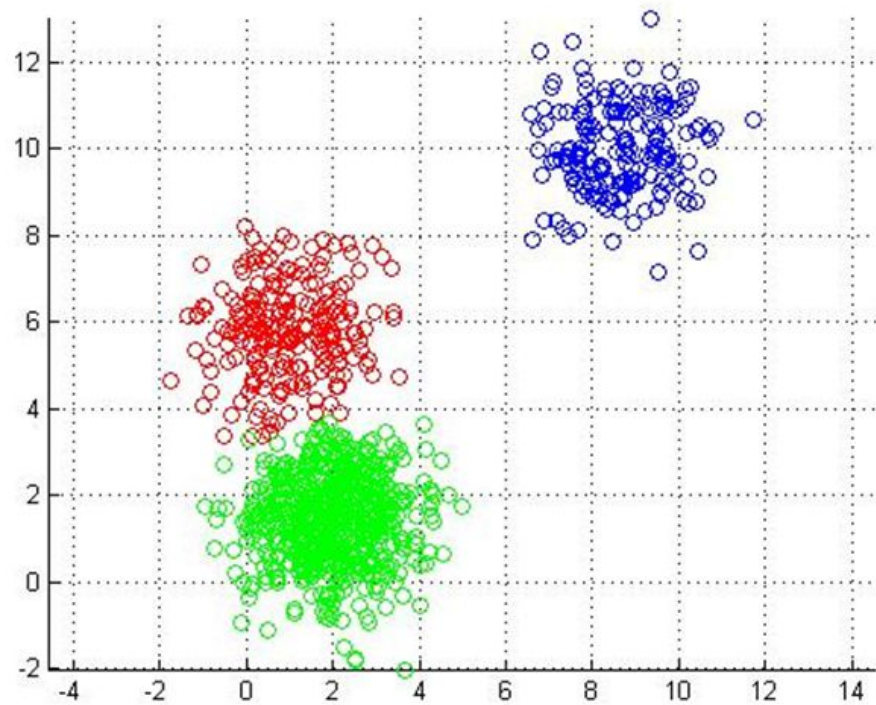


(a)



(b)

kMeans - Image segmentation



kMeans en R3

Reducción de dimensionalidad

El objetivo de los modelos de reducción de dimensionalidad es encontrar una “mejor” representación de los datos.

Con “mejor” nos referimos a una representación que preserve la mayor cantidad de información posible de los datos, bajo una determinada penalidad o restricción, que haga que la representación sea más accesible o simple.

Ejemplos de representaciones más simples:

- Representación de menor dimensionalidad
- Representación sparsa
- Representación independiente

Ingeniería de Features - PCA

En ocasiones los datos de entrada tienen muchas features y se torna costoso en tiempo y recursos entrenar modelos de ML con todo el dataset. En la práctica se pueden utilizar técnicas de reducción de la dimensión no supervisadas como PCA (Principal Component Analysis).

Casos de Uso

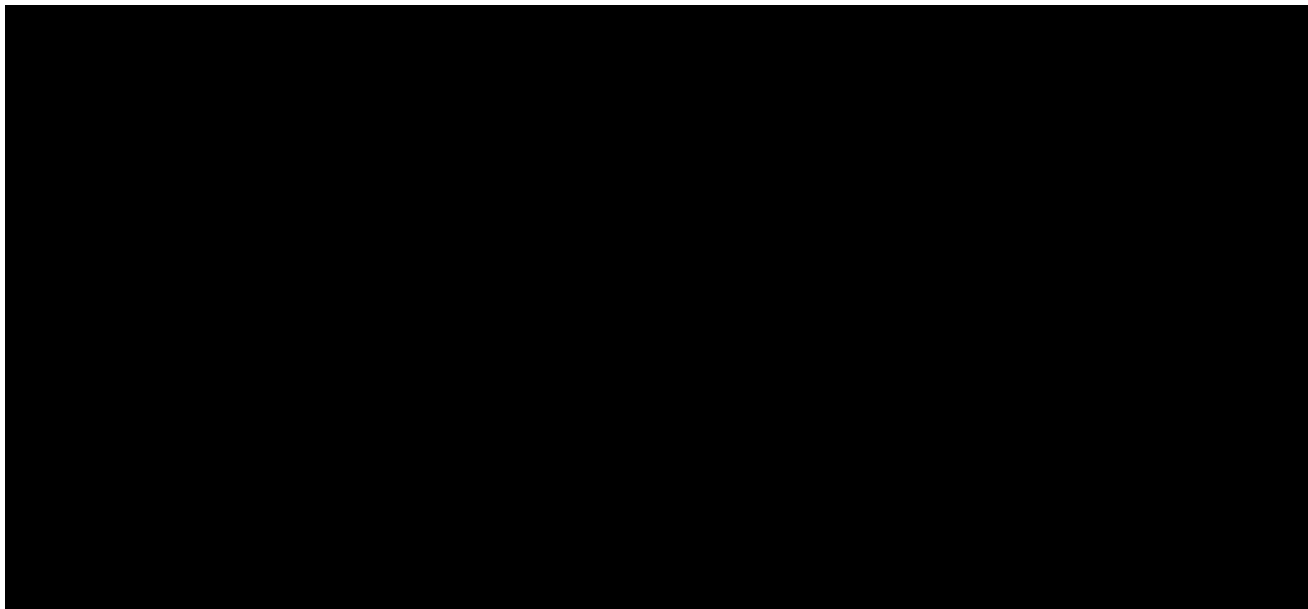
- Compresión de datos
- Identificación de patrones
- Factores latentes
- Visualización

Conocimientos Previos

- Bases y cambio de bases
- Proyecciones
- Valores y vectores propios
- Distribución gaussiana
- Optimización con restricciones

PCA

Queremos encontrar proyecciones ... de observaciones de datos ..., que sean lo más similares posibles a los originales, pero con significativamente menos dimensiones.



PCA

Dado un dataset i.i.d:

$$\chi = \{x_1, \dots, x_N\}, x_N \in \mathbb{R}^D$$

con **media cero**, la matriz de covarianza es:

$$S = \frac{1}{N} \sum_{n=1}^N x_n x_n^T$$

Definimos transformaciones lineales:

$$z_n = B^T x_n \in \mathbb{R}^M$$

$$B = [b_1, \dots, b_m] \in \mathbb{R}^{D \times M}, b_i^T b_j = 0 \quad \forall i \neq j$$

x_{11}	$\cdot \cdot \cdot$	$\cdot \cdot \cdot$	x_{1n}
$\cdot \cdot \cdot$	$\cdot \cdot \cdot$	$\cdot \cdot \cdot$	$\cdot \cdot \cdot$
x_{d1}	$\cdot \cdot \cdot$	$\cdot \cdot \cdot$	x_{dn}

 χ

PCA

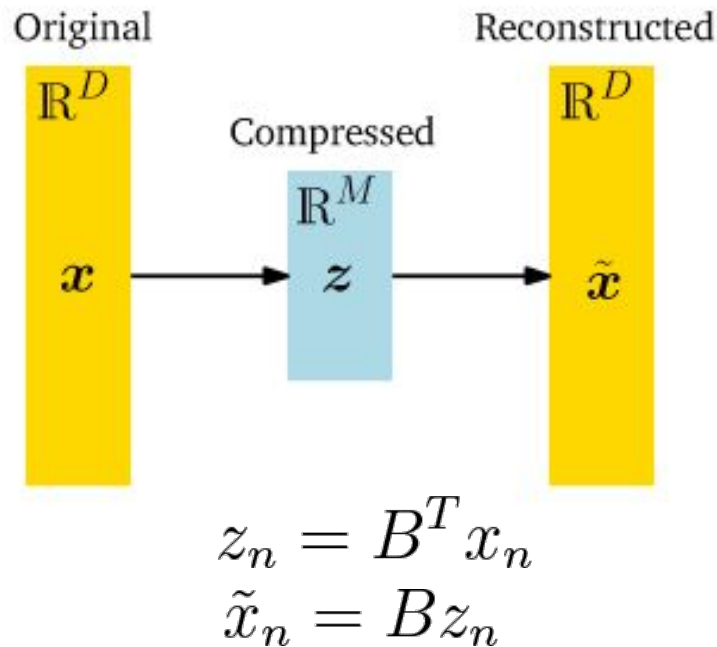
Buscamos un subespacio

$$U \subseteq \mathbb{R}^D / \dim(U) = M < D$$

donde proyectar los datos. Es decir encontrar para:

$$\tilde{x}_n \in \mathbb{R}^D \begin{cases} \rightarrow z_n \\ \rightarrow [b_1, \dots, b_m] \end{cases}$$

- i. Enfoque de máxima varianza
- ii. Enfoque de error de reconstrucción mínimo
- iii. Enfoque de variables latentes



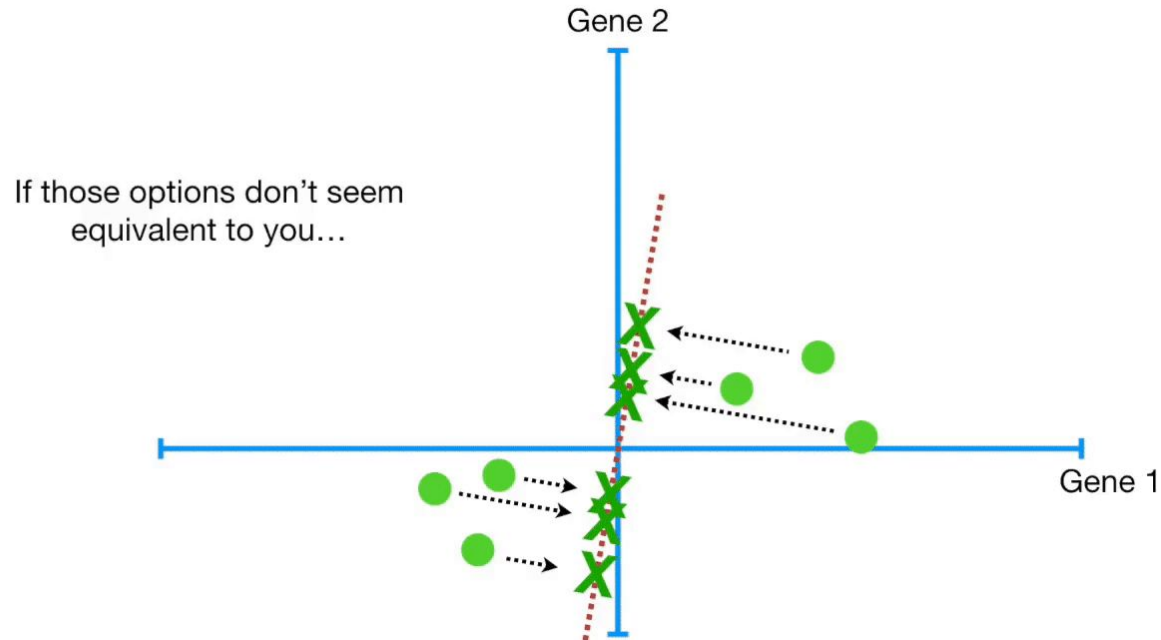
Jamboard - Desarrollo Matemático PCA

- [Introducción](#)
- [Enfoque de maximización de varianza](#)
- [Enfoque de minimización de error de reconstrucción](#)
- [Enfoque por variables latentes](#)



PCA

Comparación métodos 1 y 2.

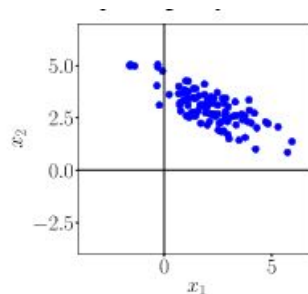


PCA

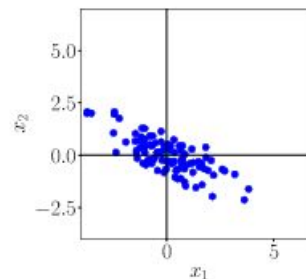
Pasos principales:

1. Centramos los datos
2. Estandarización
3. Autovalores de la matriz de covarianza
4. Proyección

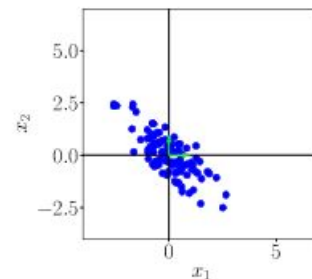
$$z_n = B^T x_n$$



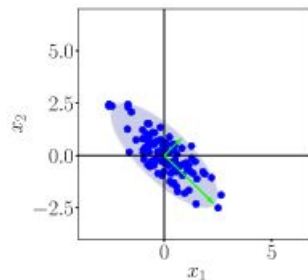
(a) Original dataset.



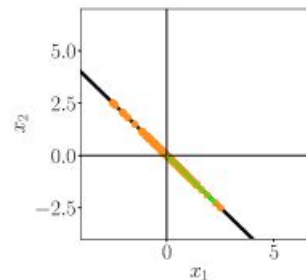
(b) Step 1: Centering by subtracting the mean from each data point.



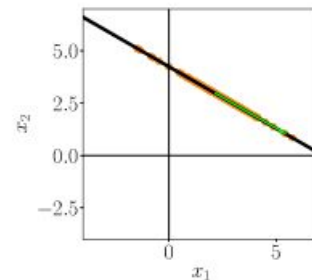
(c) Step 2: Dividing by the standard deviation to make the data unit free. Data has variance 1 along each axis.



(d) Step 3: Compute eigenvalues and eigenvectors (arrows) of the data covariance matrix (ellipse).



(e) Step 4: Project data onto the principal subspace.



(f) Undo the standardization and move projected data back into the original data space from (a).

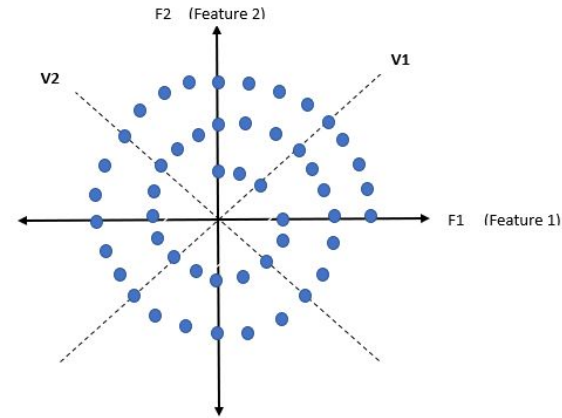
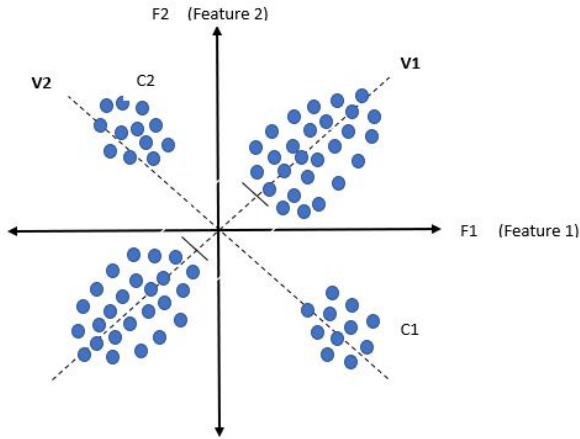
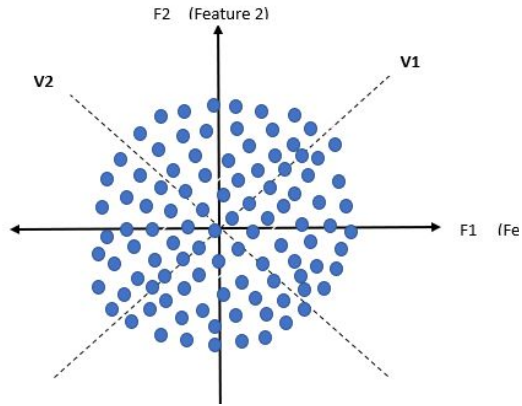
PCA

Derivaciones

- Si en PCA cambiamos el mapeo lineal por uno no-lineal, obtenemos un auto-encoder. Si el mapeo no-lineal es una red neuronal, tenemos un deep auto-encoder.
- Cuando la varianza del ruido gaussiano es cero, PPCA \rightarrow PCA.
- Si para cada dimensión, el ruido tiene una varianza distinta \rightarrow Factor Analysis.
- Si cambiamos la distribución a priori de z por una no gaussiana \rightarrow ICA

PCA

Limitaciones

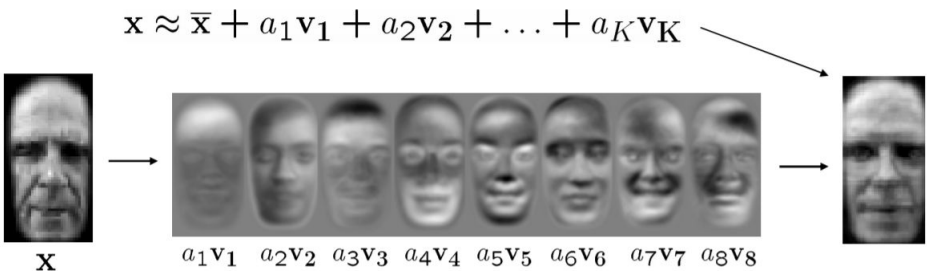
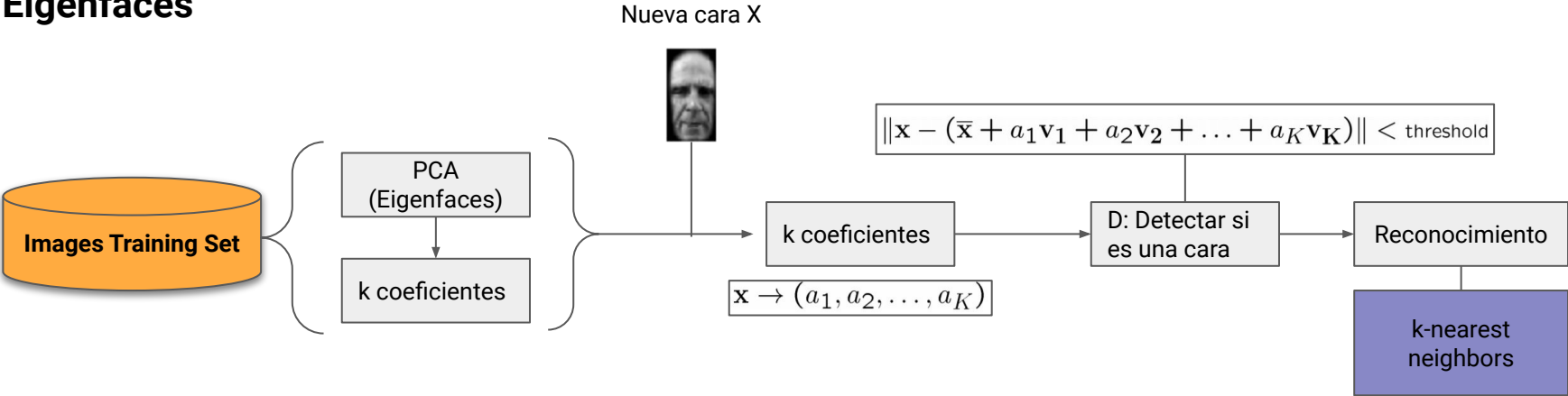


PCA- Casos Prácticos



PCA

Eigenfaces



Trabajo práctico 1

- 1) Implementar clase PCA con numpy. Tomar las primeras 63 componentes y calcular la varianza contemplada. Las operaciones internas pueden realizarse con linalg.
- 2) Implementar clase kMeans con numpy. Agrupar en clusters tomando k de 2 a 6, graficar con las primeras 2 componentes principales.
- 3) Comparar los resultados anteriores con lo visto en clase.
- 4) Utilizando las implementaciones de sklearn: Tomar las componentes principales que conserven el 90% de la varianza de MNIST y con ellas aplicar kmeans para agrupar los dígitos.

Deben maximizarse la cantidad de operaciones vectorizadas en las implementaciones. La notebook debe ser comentada, no únicamente el código.

Datasets: 1 y 2 usar Human Activity Recognition. 3 MNIST.

Entrega: Debe subirse 1 Jupyter Notebook a Github, repositorio público.

Deadline: En 2 clases.

Bibliografía

- The Elements of Statistical Learning | Trevor Hastie | Springer
- An Introduction to Statistical Learning | Gareth James | Springer
- Deep Learning | Ian Goodfellow | <https://www.deeplearningbook.org/>
- Stanford | CS229T/STATS231: Statistical Learning Theory | <http://web.stanford.edu/class/cs229t/>
- Mathematics for Machine Learning | Deisenroth, Faisal, Ong
- Artificial Intelligence, A Modern Approach | Stuart J. Russell, Peter Norvig

