Introducción a la Inteligencia Artificial

Clase 2

Índice

1. 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
2. kMeans
3. Práctica

Algoritmos no supervisados

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:

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

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

χ

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

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

Definimos transformaciones lineales:

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

$$B = [b_1, \cdots, b_m] \in \mathbb{R}^{DxM}, b_i^T b_j = 0 \ \forall \ i \neq j$$

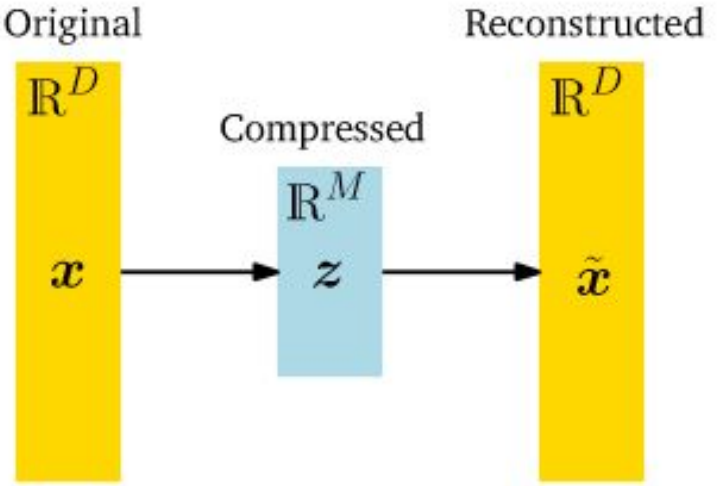
PCA

Buscamos un subespacio

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

donde proyectar los datos. Es decir encontrar para:

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



$$z_n = B^T x_n$$
$$\tilde{x}_n = B z_n$$

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

Desarrollo matemático de PCA:

buscamos una proyección $\tilde{x}_n$ de mis datos originales x_n / $\dim(\tilde{x}_n) \leq \dim(x_n)$

Que tenemos:

- Dataset $\mathcal{X}$ iid $= \{x_1, \dots, x_N\}$ / $x_i \in \mathbb{R}^D$, $\mu_0 = \emptyset$
- matriz de cov : $S = \frac{1}{N} \sum_{n=1}^{N} x_n x_n^t$

→ buscamos $z_n = B^t x_n$

↳ $\in \mathbb{R}^M$

Métodos para encontrar B (y z_n) → Maxima varianza (1)

↳ error de reconstrucción (2)

Máxima varianza:

formulación: Maximizar la varianza en una dimensión inferior → reteniendo la mayor cantidad de información

- Partimos con una columna de $B\ (\mathbb{R}^{M \times D})$, $b_1 \in \mathbb{R}^D$
  ↳ maximizar la varianza de z_1 de $z \in \mathbb{R}^M$:

$$Var[z] = Var[B^t(x-u)] = Var[B^t x - B^t u] = Var[B^t x]$$

$$Var_1 = Var[z_{1n}] = \frac{1}{N}\sum_{n=1}^{N} z_{1n}^2$$

$$z_{1n} = b_1^t x_n$$ → proyección ortogonal de x_n en el subespacio unidimensional formado por b_1

↳ $B = [b_1, b_2, \dots, b_n]$

$$Var_1 = \frac{1}{N}\sum_{n=1}^{N}(b_1^t x_n)^2 = \frac{1}{N}\sum_{n=1}^{N}(b_1^t x_n)^t(b_1^t x_n)$$

$$= \frac{1}{N}\sum_{n=1}^{N} b_1^t x_n x_n^t b_1 = b_1^t \underbrace{\left(\frac{1}{N}\sum_{n=1}^{N} x_n x_n^t\right)}_{S} b_1 \rightarrow \boxed{Var_1 = b_1^t S b_1}$$

Si aumento b_1 ⇒ incremento Var_1

⇒ buscamos maximizar todo pero restringo b_1 (unitario)

Objetivo: $\max\ b_1^t S b_1$, $\|b_1\|^2 = 1$ ⇝ maximización condicionada (optimización de Lagrange)

$$L(b_1, \lambda_1) = b_1^t S b_1 + \lambda_1 (1 - \underbrace{b_1^t b_1}_{\|b_1\|^2})$$

$$\begin{cases} \partial_{\lambda_1} L = 0 \rightarrow 1 - b_1^t b_1 = 0 \rightarrow b_1^t b_1 = 1 \\ \partial_{b_1} L = 0 \rightarrow 2 b_1^t S - 2 \lambda_1 b_1^t = 0 \end{cases}$$

$$(b_1^t S)^t = (\lambda_1 b_1^t)^t$$

$$S^t b_1 = (b_1^t)^t \lambda_1^t$$

$$S b_1 = b_1 \lambda_1 \rightarrow S b_1 = \lambda_1 b_1$$

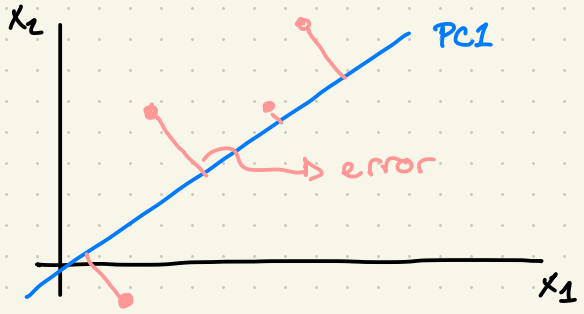
(b_1: vector propio de S; λ_1: autovalor correspondiente)

⇒ Seleccionamos los autovectores asociados a los m autovalores más grandes de la matriz de covarianza.

con esto:

- Varianza explicada: $\sum_{i=1}^{M} \lambda_i$
- Varianza perdida: ⟶ $\sum_{j=M+1}^{D} \lambda_j$

Minimización del error de reducción:



Si proyectamos un punto sobre una dirección la reconstrucción después va a ser sobre el mismo punto.

Vamos a minimizar $\|x - \tilde{x}\|$.

Si $\exists$ B base ortonormal de $\mathbb{R}^D$, cualquier $x \in \mathbb{R}^D$ se puede escribir como cl de la base:

$$x = \sum_{d=1}^{D} \alpha_d b_d = \underbrace{\sum_{i=1}^{M} \alpha_i b_i}_{\tilde{x}} + \sum_{j=M+1}^{D} \alpha_j b_j$$

queremos encontrar $\tilde{x} = \sum_{i=1}^{M} z_i b_i \in U \subseteq \mathbb{R}^D$ lo más similar posible a x:

$$\tilde{x} = \sum_{i=1}^{M} z_i b_i = B z_i$$

buscamos minimizar el MSE $\|x-\tilde{x}\|$:

① - optimizar z_n para una base

② - Encontrar esa base óptima.

$$J_M = \frac{1}{N} \sum_{i=1}^{N} \|x_n - \tilde{x}_n\|^2$$

$$\partial_{z_{in}} J_M = \frac{\partial J_M}{\partial \tilde{x}_n} \cdot \frac{\partial \tilde{x}_n}{\partial z_{in}}$$

$$= \left(-\frac{2}{N} (x_m - \bar{x}_m)^t \right) \cdot \underbrace{\left(\frac{\partial}{\partial z_{in}} \left(\sum_{i=1}^{M} z_{in} b \right) \right)}_{b_i}$$

$$= -\frac{2}{N} (x_m - \bar{x}_m)^t \cdot b_i$$

$$= -\frac{2}{N} \left(x_m - \sum_{i=1}^{M} z_{in} \cdot b_i \right)^t \cdot b_i = -\frac{2}{N} \left(x_m^t b_i - z_{im} \underbrace{b_i^t b_i}_{=1} \right)$$

$$= -\frac{2}{N} \left(x_m^t b_i - z_{im} \right)$$

igualamos a 0:

$$-\frac{2}{N}\left(x_m^t \cdot b_i - z_{in}\right) = 0 \longrightarrow x_m^t b_i = \boxed{z_{im} = b_i x_m^t}$$

← las z_{im} son las coord. que vamos a encontrar para cada vector $(b_i x_m^t)$

buscamos ahora la base óptima:

$$\tilde{x}_n = \sum_{n=1}^{M} z_{mn} b_n = \sum_{n=1}^{M} (x_n^t b_n) b_n$$

$$= \sum_{n=1}^{M} (b_n b_n^t) x_n$$

$$x_n = \left(\sum_{n=1}^{M} b_n b_n^t\right) x_n + \left(\sum_{j=M+1}^{D} b_j b_j^t\right) x_n$$

distancia (error) $x - \tilde{x}_m = \sum_{j=M+1}^{D} b_j b_j^t x_j = \sum_{j=M+1}^{D} (x_m^t b_j) b_j$

$$J_M = \frac{1}{N}\sum_{n=1}^{N}\|x_n - \tilde{x}_n\|^2 = \frac{1}{N}\sum_{n=1}^{N}\left\|\sum_{j=M+1}^{D}(x_n^T b_j)\,b_j\right\|^2$$

$$= \sum_{j=M+1}^{D} b_j^T \left(\frac{1}{N}\sum_{n=1}^{N} x_n x_n^T\right) b_j$$

$$= \sum_{j=M+1}^{D} b_j^T S\, b_j = \sum_{j=M+1}^{D} \mathrm{tr}\left(b_j^T S\, b_j\right) = \sum_{j=M+1}^{D} \mathrm{tr}\left(S\, b_j b_j^T\right)$$

$$= \mathrm{tr}\left(\underbrace{\sum_{j=M+1}^{D}(b_j b_j^T)}_{\text{Matriz de proyección.}} \cdot S\right)$$

El error de reconstrucción se puede pensar como la matriz S proyectada sobre el complemento ortogonal de U (subespacio).

↳ Equivalente a minimizar la var de los datos en el subespacio I (ignorado)

Al reducir dimensiones:

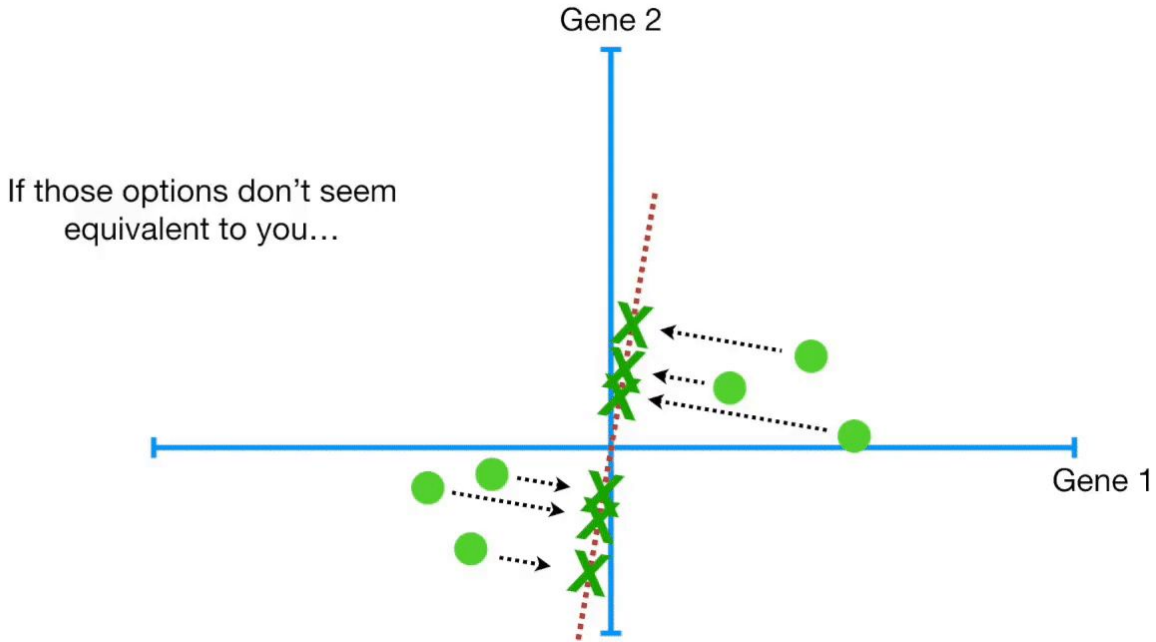
$\mathbb{R}^M \to U + I$

U: subespacio 'explicado'

I: subespacio 'ignorado'

PCA

Comparación métodos 1 y 2.



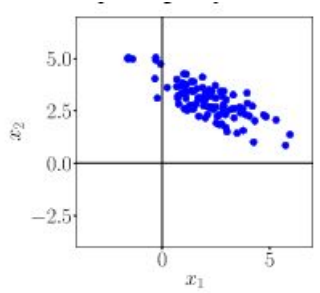
Principal Component Analysis - Teoría

PCA

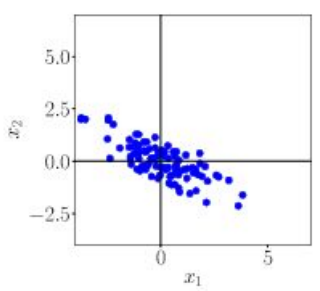
Pasos principales:

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

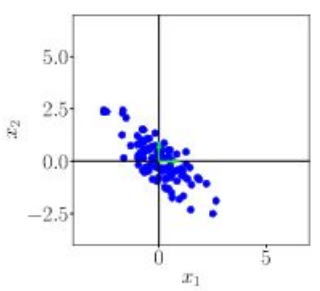
$$z_n = B^T x_n$$



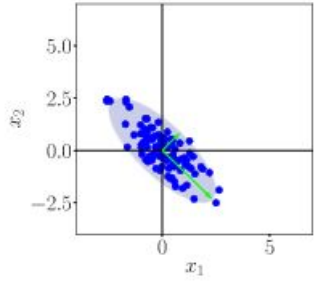
(a) Original dataset.



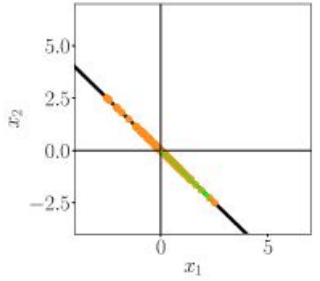
(b) Step 1: Centering by subtracting the mean from each data point.



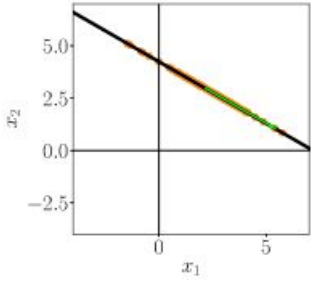
(c) Step 2: Dividing by the standard deviation to make the data unit free. Data has variance 1 along each axis.



(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).

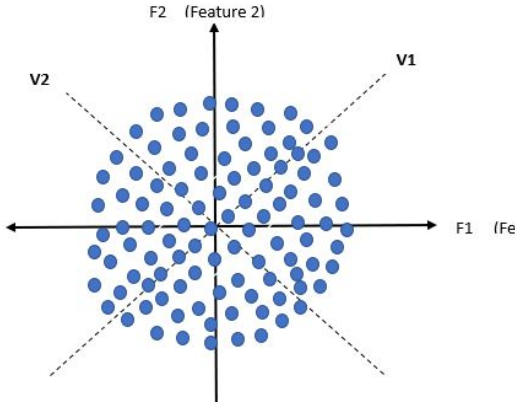
PCA

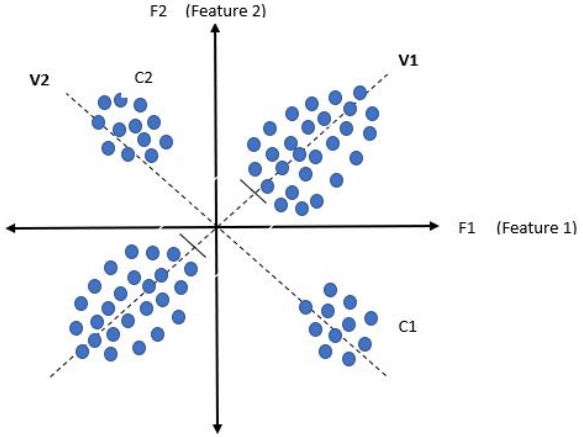
Derivaciones

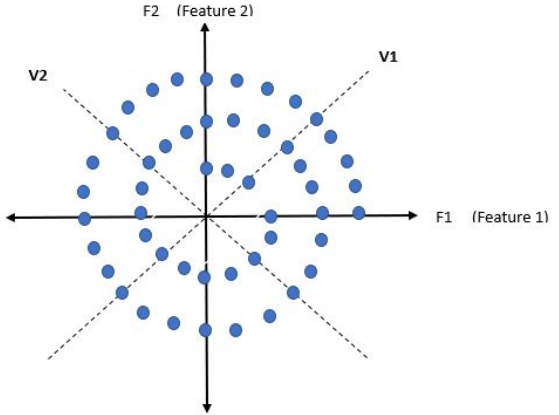
- Si en PCA cambiamos el mapeo lineal por uno no-lineal, obtenemos un auto-encoder. Si el mapeo no-lineal es una red neuronal, tenemos un deep auto-encoder.
- Cuando la varianza del ruido gaussiano es cero, PPCA → PCA.
- Si para cada dimensión, el ruido tiene una varianza distinta → Factor Analysis.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

PCA

Limitaciones



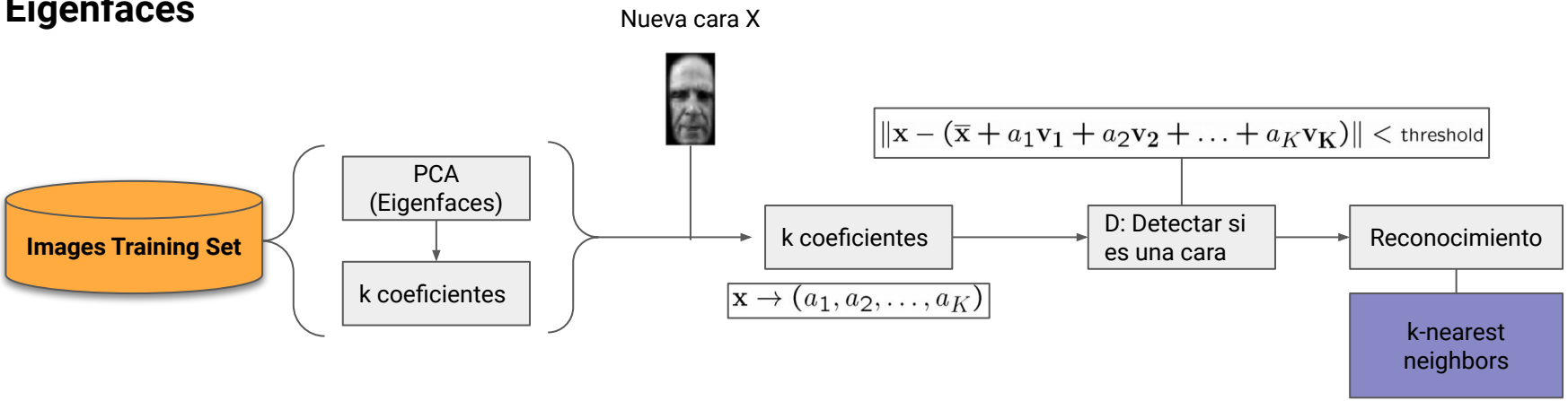


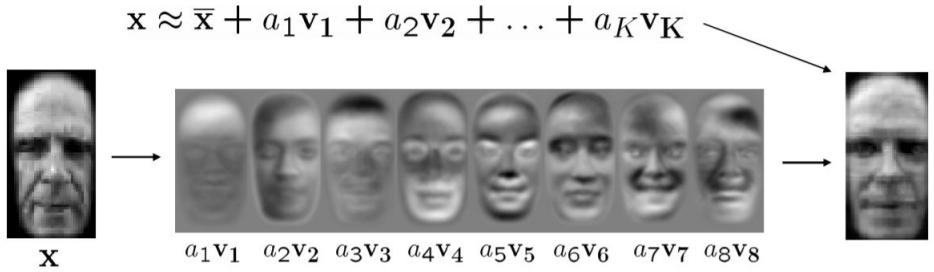


PCA - Ejemplo

PCA

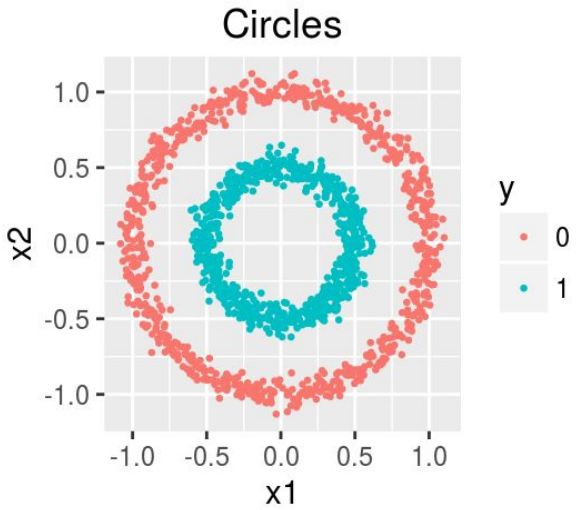
Eigenfaces

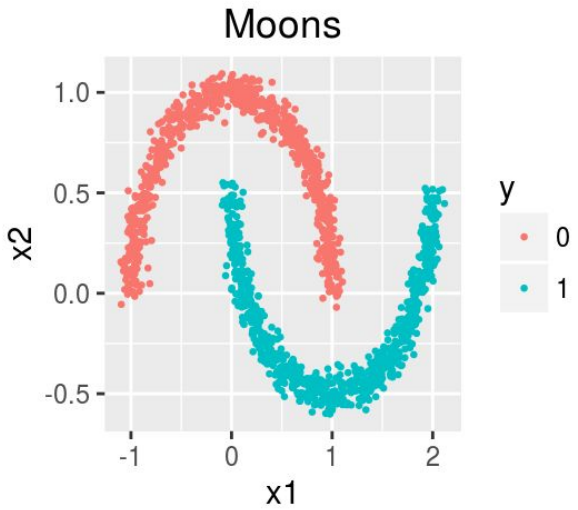


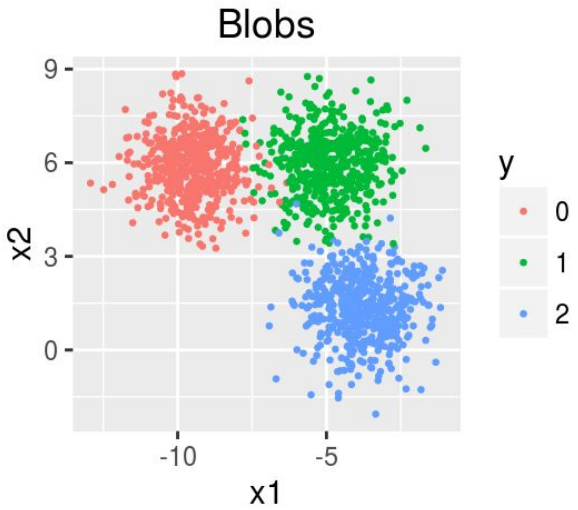


Clustering

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.







kMeans

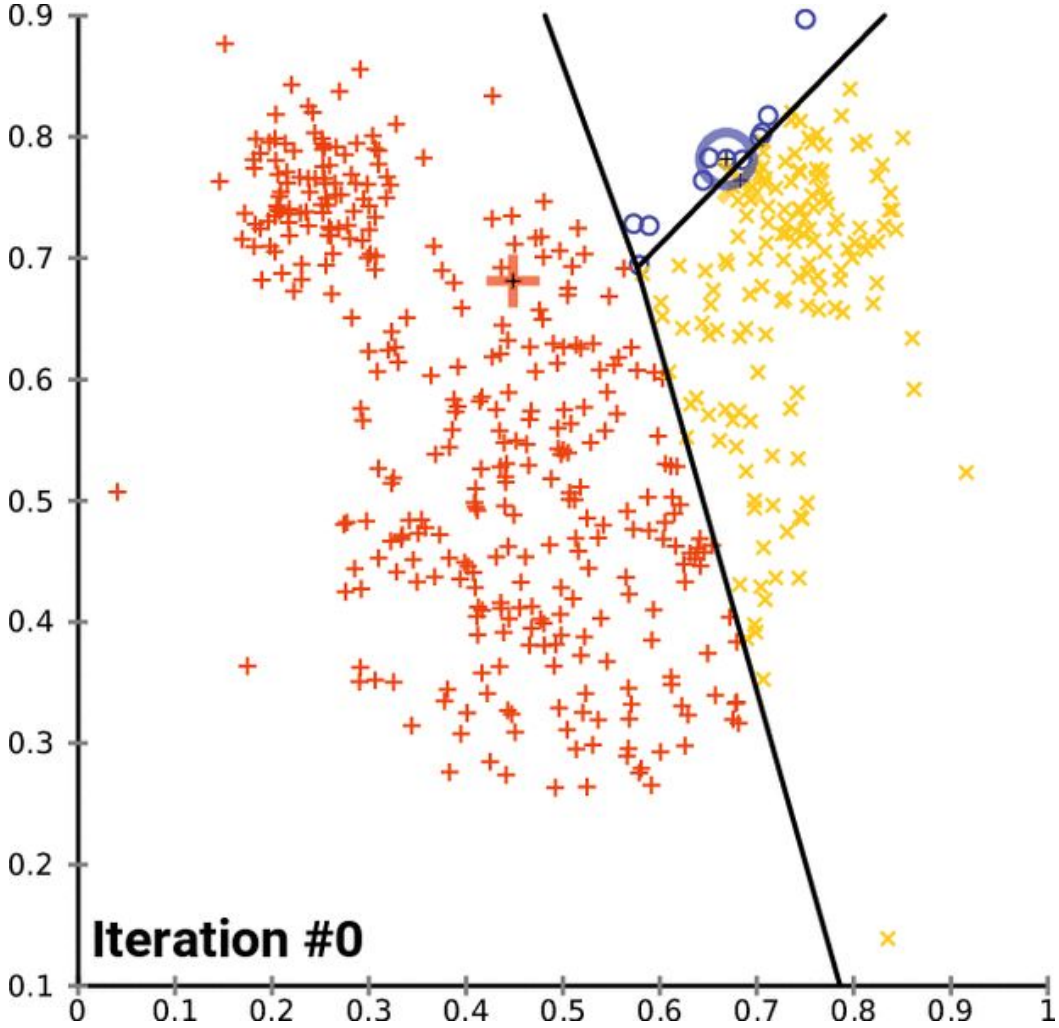
K-means es uno de los algoritmos más básicos en Machine Learning no supervisado. Es un algoritmo de **clusterización**, que agrupa los datos que comparten características similares. Recordemos que entendemos datos como n realizaciones del vector aleatorio X.

El algoritmo K-means funciona de la siguiente manera:

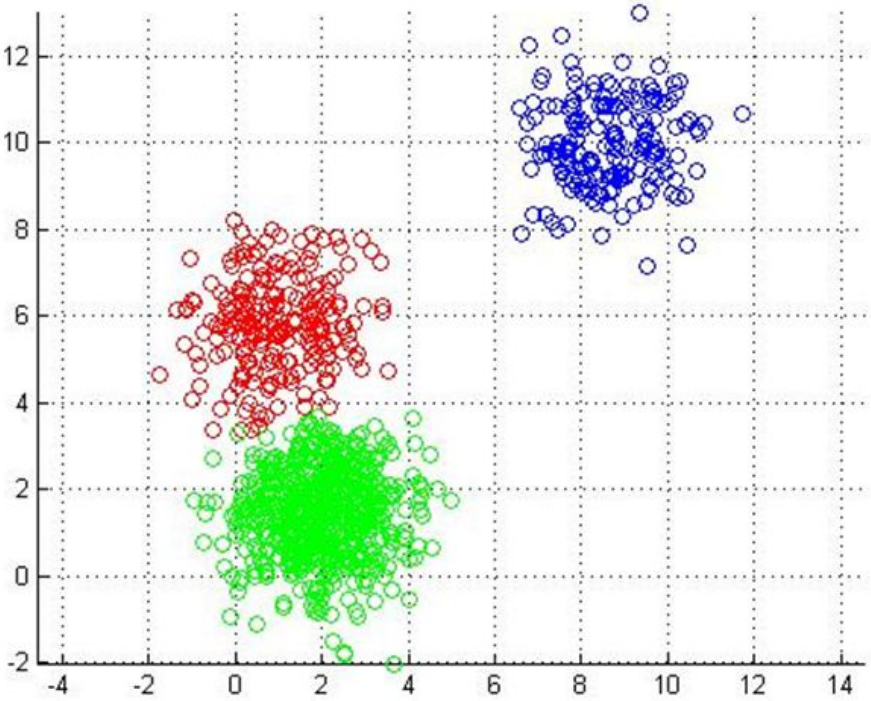
1. El usuario selecciona la cantidad de clusters a crear (n).
2. Se seleccionan n elementos aleatorios de X como posiciones iniciales del los centroides C.
3. Se calcula la distancia entre todos los puntos en X y todos los puntos en C.
4. Para cada punto en X se selecciona el centroide más cercano de 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.

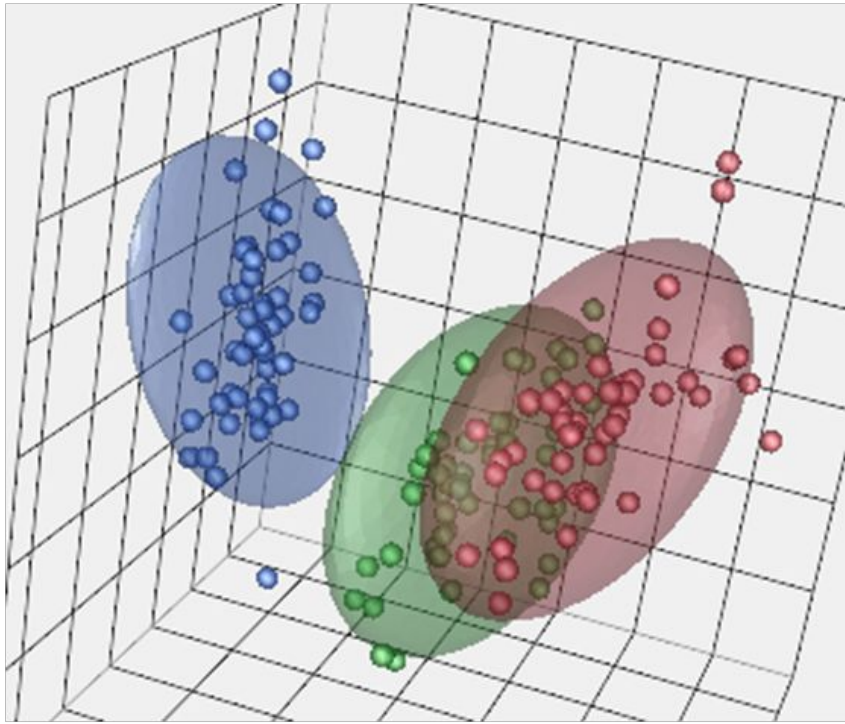
Implementar la función *def k_means(X, n)* de manera tal que al finalizar devuelva la posición de los centroides y a qué cluster pertenece cada fila de X.

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).


0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Iteration #0

kMeans - Image segmentation





kMeans en R3

Trabajo práctico 1

1) Tomar las primeras 63 componentes principales y calcular la varianza contemplada. Realizar las operaciones internas con numpy.linalg.
2) Utilizando KMeans (de scipy o Scikit Learn. Agrupar el dataset transformado (ejercicio de PCA) y agrupar en clusters de k=2 y 6. Graficar los casos de k=2 y k=6 con las primeras dos componentes principales.
3) Comparar los resultados anteriores con lo visto en clase.
4) Con las implementaciones de sklearn, tomar las componentes principales que capturen el 90% de la varianza y aplicar kmeans para agrupar los dígitos en 10 clusters. Analizar los resultados.

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