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

Ing. Lautaro Delgado
(lautarodc@unops.org)

Índice

1. Exposiciones ejercicios clase 1
2. Ejercicios k-means
    a. Datos sintéticos
    b. Clusterización
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

Repaso Ejercicios
Clase 1

Ejercicio #1 - C2 | Datasets sintéticos

En problemas de Machine Learning es muy importante contar con el dataset correcto. Pero muchas veces, el dataset nos es fácil de conseguir o el equipo de data engineers aún lo está generado.

Una manera sencilla de generar datos para probar soluciones de Machine Learning es crear datasets sintéticos. Por ejemplo, es simple crear datasets con grados de clusterización variables y probar cómo se comportan nuestros algoritmos en diferentes escenarios.

Objetivo: Utilizar numpy para crear datos clusterizados A/B en 4 dimensiones.

Hint:

- Definir una matriz con centroides [1,0,0,0] y [0,1,0,0]
- Utilizar una constante para separar o alejar los centroides entre sí.
- Utilizar np.repeat para crear n/2 muestras de cada centroide.
- Sumar a cada centroide un vector aleatorio normal i.i.d. con media 0 y desvío (np.random.normal).
- Armar un arreglo que tenga n enteros indicando si la muestra pertenece a A o a B.

Ejercicio #6 - C1 | Distancia a centroides

Dada una nube de puntos X y centroides C, obtener la distancia entre cada vector X y los centroides utilizando operaciones vectorizadas y broadcasting en NumPy. Utilizar como referencia los siguientes valores:

- X = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
- C = [[1, 0, 0], [0, 1, 1]]

Ejercicio #7 - C1 | Enunciado

Obtener para cada fila en X, el índice de la fila en C con distancia euclídea más pequeña.
Es decir, decir para cada fila en X a qué cluster pertenece en C.
Por ejemplo, si el resultado anterior fue:

```
# [[ 3.60555128  8.36660027 13.45362405]
#  [ 2.44948974  7.54983444 12.72792206]]
```

El programa debería devolver [1, 1, 1]

Hint: utilizar np.argmin

Ejercicio #8 - C1 | Implementacion basica de K-means en Numpy

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).

Notebook Consolidado Ejercicios Dataset Sintético y K-means

Aprendizaje no supervisado

El objetivo de los modelos no supervisados es encontrar la “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

Principal Component Analysis - Teoría

PCA

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

Principal Component Analysis - Teoría

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 medio cero, la matriz de covarianza es:

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

Definimos transformaciones lineales:

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

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

Principal Component Analysis - Teoría

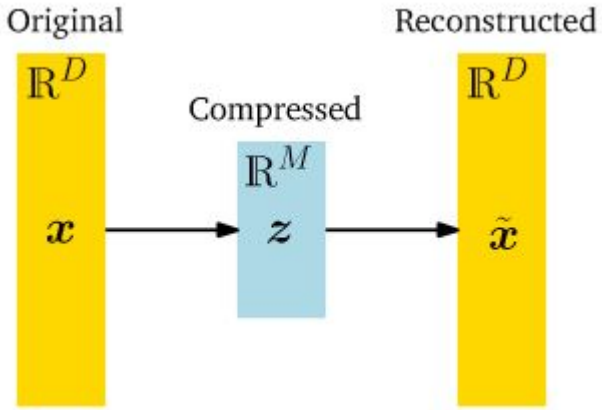
PCA

Buscamos un subespacio

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

donde proyectar los datos. Es decir encontrar para:

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



$$z_n = B^T x_n$$

$$\tilde{x}_n = B z_n$$

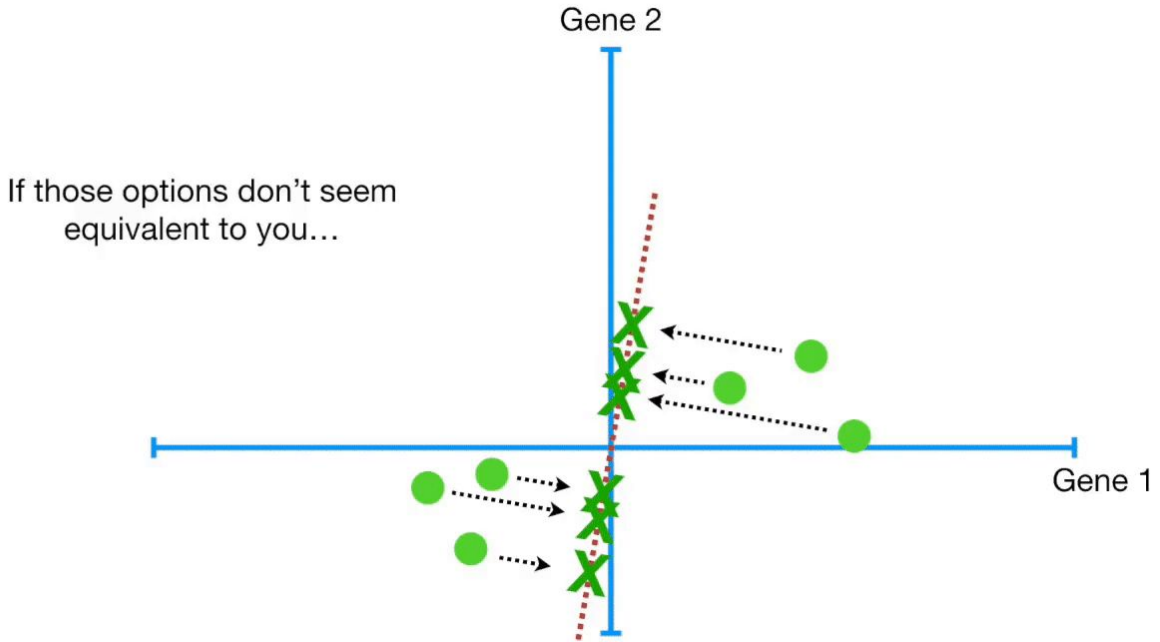
i. Enfoque de máxima varianza
ii. Enfoque de error de reconstrucción mínimo
iii. Enfoque de variables latentes

Jamboard - Desarrollo Matemático PCA

- Introducción
- Enfoque de maximización de varianza
- Enfoque de minimización de error de reconstrucción
- Enfoque por variables latentes

PCA

Comparación métodos 1 y 2.



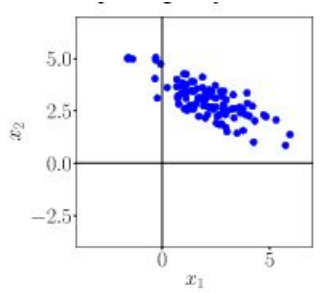
Principal Component Analysis - Teoría

PCA

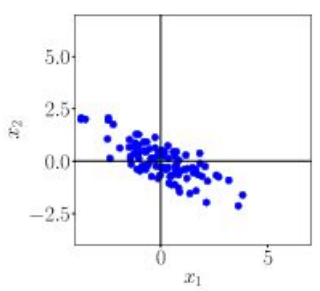
Pasos principales:

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

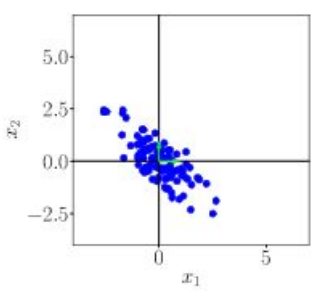
$$z_n = B^T x_n$$



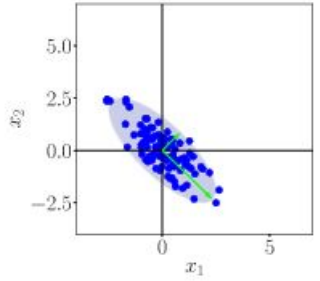
(a) Original dataset.



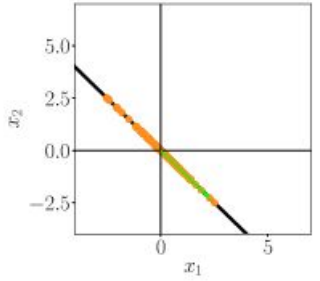
(b) Step 1: Centering by subtracting the mean from each data point.



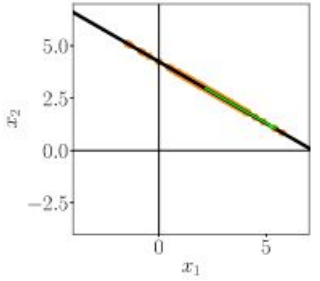
(c) Step 2: Dividing by the standard deviation to make the data unit free. Data has variance 1 along each axis.



(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).

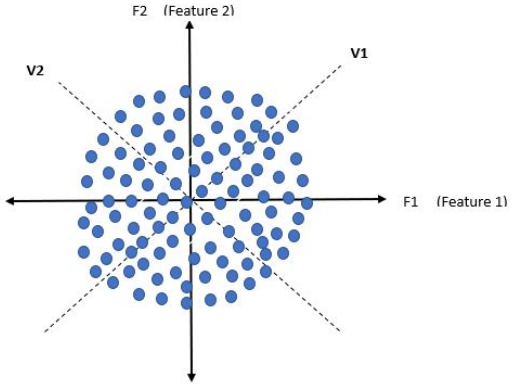
PCA

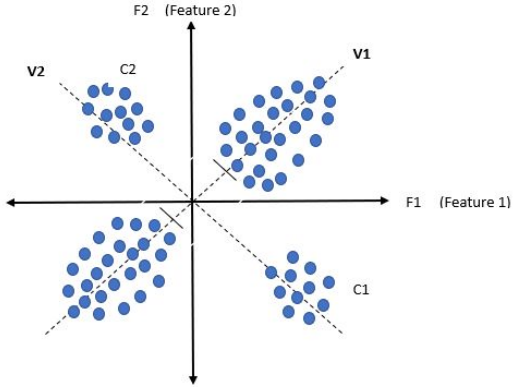
Derivaciones

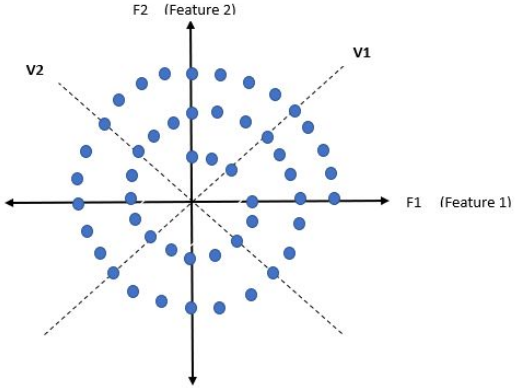
- Si en PCA cambiamos el mapeo lineal por uno no-lineal, obtenemos un auto-encoder. Si el mapeo no-lineal es una red neuronal, tenemos un deep auto-encoder.
- Cuando la varianza del ruido gaussiano es cero, PPCA → PCA.
- Si para cada dimensión, el ruido tiene una varianza distinta → Factor Analysis.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

PCA

Limitaciones





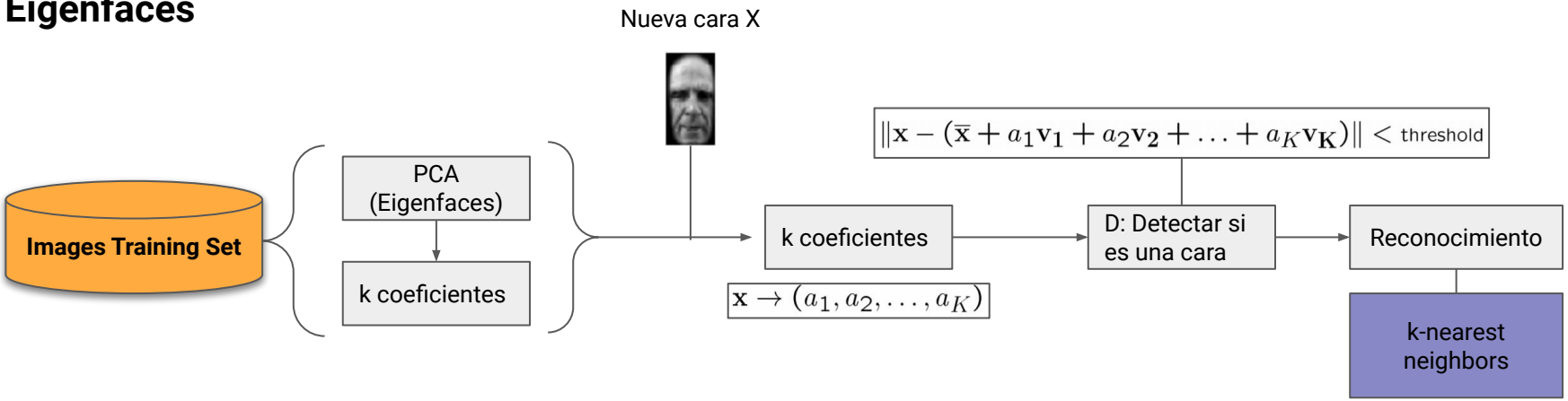


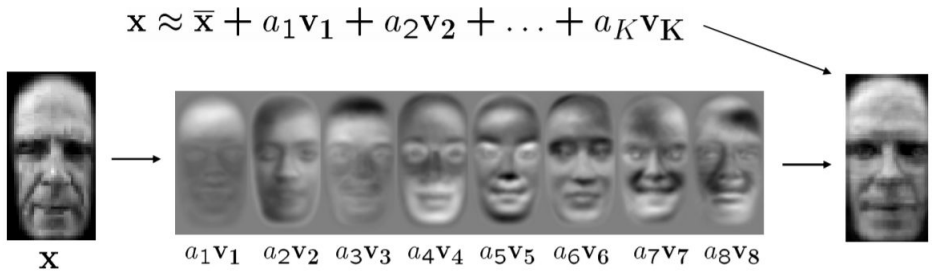
PCA- Casos Prácticos

Principal Component Analysis - Teoría

PCA

Eigenfaces





Ejercicios

Principal Component Analysis - Práctica

Ejercicio #1 | Dado un dataset X, calcular PCA para reducir dimensión.

Siguiendo los pasos vistos en la teoría, se requiere utilizar numpy para calcular PCA del dataset de entrada X, utilizando la componente más importantes.

```
X = np.array( [ [0.8, 0.7] , [0.1, -0.1] ] )
```

Al finalizar la implementación en numpy, corroborar obtener los mismos resultados que utilizando el código de la librería scikit-learn. Escribir un test para comparar las matrices.

Ejercicio #2 | Aplicar PCA y K-means al dataset de dígitos

Siguiendo los ejemplos vistos en clase sobre los datasets de Human Activity Recognition y Fashion MNIST, realizar las siguientes consignas:

1. Aplicar PCA sobre el dataset para poder explicar el 90% de la varianza. ¿Cuántos componentes se requieren?
2. Graficar un scree plot (varianza contemplada en función del número de componentes considerados)
3. Visualizar gráficamente los primeros 5 componentes ¿Qué conclusiones se puede sacar de cada componente? [OPCIONAL].
4. Visualizar la imagen original vs. la reconstruida con los m componentes del punto 1.
5. Graficar una matriz de correlación del dataset reducido.
6. Graficar los clusters de dígitos en 2 y 3 dimensiones usando los componentes obtenidos en PCA.
7. Aplicar K-means para clusterizar los dígitos ¿Cómo son los resultados?
8. Realizar un gráfico de inercia para obtener el número óptimo de clusters k.
9. Analizar visualmente los límites del cluster de algún dígito y "generar" artificialmente el dígito dándole valores a los primeros dos componentes de PCA.

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