TALLER DE PROCESAMIENTO DE SEÑALES

1er Cuatrimestre 2026 - Trabajo Práctico Nº 7 - PCA, KMeans y EM

IMPORTANTE: En este ejercicio solamente se podrá importar numpy y matplotlib.pyplot(el resto deberá ser implementación propia).

Dado un conjunto de datos 2D no supervisado, se desea aprender una curva representativa del *manifold* utilizando un algortimo de *PCA condicional*.

(**a**) *Creación del dataset:*

1. Crear un conjunto de datos $\{X_i\}_{i=1}^{200}$ no supervisado de tamaño 200 y dimensión 2. El mismo debe ser generado como $X_i = g(U_i) + 0.05 \cdot Z_i$, donde las Z_i son vectores aleatorios i.i.d. normal estándar bivariados, y $U_i \sim \mathcal{U}(0,1)$ i.i.d. e independientes de las normales. La función $g : \mathbb{R} \to \mathbb{R}^2$ se define como:
$$g(u) = \begin{cases} (1.5u\ ;\ 3u) & 0 \leq u < \frac{1}{3} \\ (1.5u\ ;\ 2-3u) & \frac{1}{3} \leq u < \frac{2}{3} \\ (3-3u\ ;\ 0) & \frac{2}{3} \leq u \leq 1 \end{cases}$$
2. Graficar un *scatter* de los datos.
3. Graficar la curva representativa del *manifold* $M = \{g(u) \in \mathbb{R}^2 :\ 0 < u < 1\}$.

(**b**) *K-Means*: Dado que PCA proyecta sobre subespacios (rectas en este caso), se necesitarán más de un PCA para describir este *manifold*. Se denomina PCA condicional al proceso de combinar un algoritmo de clustering seguido de un PCA. Con el algoritmo de clustering se selecciona sobre que recta se proyectará y con el PCA se lo proyectará efectivamente sobre dicha recta (es decir, habrá un PCA distinto para cada recta). Como primer algoritmo de clustering se propone utilizar K-Means.

1. Entrenar un algoritmos K-Means para clusterizar los datos. El código debe estar estructurado de la siguiente manera:

```
class Kmeans:
    # Inicializar atributos y declarar hiperparámetros
    def __init__(self,...

    # Etapa de entrenamiento
    def fit(self,X):

    # Etapa de testeo:
    def predict(self,X):
```

2. Graficar un *scatter* de los datos, indicando con un color distinto cada cluster.
3. ¿Es suficiente este algoritmo para esta tarea? ¿Por qué? Justificar.

(**c**) *Algoritmo EM:*

1. Entrenar un algoritmo EM para clusterizar los datos. El algoritmo debe ser inicializado eligiendo muestras al azar para las medias, pesos equiprobables y covarianzas identidad. El código debe estar estructurado de la siguiente manera:

```
class EM:
    # Inicializar atributos y declarar hiperparámetros.
    def __init__(self,...):

    # Etapa de entrenamiento.
    def fit(self,X):

    # Etapa de testeo soft
    def predict_proba(self,X):
```

```
# Etapa de testeo hard
def predict(self,X):
```

2. Graficar un *scatter* de los datos, indicando con un color distinto cada cluster.

3. Superponer al `scatter` las fronteras de decisión para todo $\mathbb{R}^2$. : Funciones como `meshgrid` (numpy) y `contour` (matplotlib) pueden ser útiles para graficar las fronteras.

(**d**) *PCA:*

1. Implementar un algoritmo PCA que proyecte puntos del plano sobre una recta. El código debe estar estructurdo de la siguiente manera (implementar todos los métodos mencionados):

```
class PCA:

    # Inicializar atributos y declarar hiperparámetros. No aclarar aún la dimensión
    # del espacio latente
    def __init__(self,...

    # Etapa de entrenamiento. Se entrenan todos los autovectores, sin definir la
    # dimensión latente
    def fit(self,X):

    # Transformar del espacio original al espacio latente de dimensión k.
    def transform(self,X,k):

    # Transformar del espacio latente al espacio original
    def inverse_transform(self,U):

    # fit + transform
    def fit_transform(self,X,k):
```

2. Utilizar el algoritmo de clustering EM para separar el conjunto de datos en varios conjuntos (uno por cluster). Entrenar un PCA por cluster.

3. Cada algoritmo PCA proyecta sobre un espacio de una dimensión (una recta). Indicar los valores mínimos y máximos de la proyección para cada cluster.

4. Utilizar el método `inverse_transform` de cada PCA para graficar el manifold a partir de los valores extremos calculados previamente. Indicar la recta correspondiente a cada cluster con un color diferente.

(**e**) *Detector de Anomalías:* Se desea evaluar el desempeño del algoritmo como detector de anomalías. Se espera que los valores del plano lejanos al *manifold* sean catalogados como anómalos.

1. Definir una grilla de puntos en $[0, 1] \times [0, 1]$. Utilizar los algoritmos entrenados previamente para proyectar y reconstruir cada punto de la grilla. Clasificar como anómalo los puntos de la grilla donde el error cuadrático entre el punto y su reconstrucción supere un determinado umbral.

2. Graficar los puntos *no anómalos*. Elegir un tamaño de grilla y un umbral para que el gráfico represente el manifold.