TALLER DE PROCESAMIENTO DE SEÑALES

2do Cuatrimestre 2026 - Trabajo Práctico Nº 6 - PCA, K-Means

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

ECG5000 es una base de datos de señales electrocardiográficas (ECG) compuesta por 4998 registros, cada uno correspondiente a un latido cardíaco. Cada latido está representado mediante 140 valores que corresponden a muestras de la señal a lo largo del latido, obtenidas luego de normalizar su duración mediante interpolación. De esta manera, cada registro puede considerarse una serie temporal univariada de longitud 140. La base incluye además una etiqueta que identifica el tipo de latido (normal o anómalo).

(**a**) *Base de datos:*

1. Cargar el dataset desde `http://storage.googleapis.com/download.tensorflow.org/data/ecg.csv` (descarga autocontenida en el código) e indicar la cantidad de latidos, muestras por latido y latidos de cada clase.
2. Separar la base de datos de forma aleatoria entre entrenamiento y testeo con las proporciones 80 % y 20 %.
3. Graficar 20 latidos de cada clase superpuestos y el latido medio de la clase. ¿En qué tramos del latido difieren las clases?

(**b**) *PCA:*

1. Implementar un algoritmo PCA. 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. Verificar que con las 140 componentes la reconstrucción es exacta.
3. Graficar el porcentaje de energía por cada componente y la energía acumulada. ¿Cuántas componentes son necesarias para conservar el 90 % y el 95 % de la energía?
4. Elegir un latido normal y uno anómalo del conjunto de testeo y reconstruirlos con 1, 5 y 20 componentes principales. Graficar el original junto a cada reconstrucción e indicar el error cuadrático medio.
5. Proyectar los latidos de entrenamiento sobre las dos primeras componentes y graficar un scatter indicando con un color distinto cada clase. Analizar resultados.

(**c**) *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. Entrenar el algoritmo K-Means utilizando las dos componentes principales halladas anteriormente. Graficar un scatter de los datos, indicando con un color distinto cada cluster y superponer la frontera de decisión para todo $\mathbb{R}^2$.

3. Asignar a cada cluster la clase mayoritaria entre los latidos de entrenamiento y calcular el accuracy en testeo. ¿Es suficiente este algoritmo para separar los latidos normales de los anómalos? ¿Por qué? Justificar.