Práctica 1. Análisis de componentes principales

Reconocimiento de patrones

---

1. Objetivo

Que el alumno comprenda, implemente e interprete el algoritmo de análisis de componentes principales (PCA), siendo capaz de aplicarlo sobre datos reales y analizar críticamente los resultados obtenidos. De igual forma será capaz de:

- entender geométricamente PCA
- implementarlo desde cero
- interpretar autovalores y varianza explicada
- evaluar perdida de información

2. Desarrollo

Para el desarrollo de esta práctica se utilizaran algunos conjuntos de datos de acceso publico, siendo estos Iris, Wine y Olivetti, los cuales pueden importarlos mediante la librería scikit-learn con el modulo sklearn.datasets, el cual tiene las funciones: load_iris, load_wine, y fetch_olivetti_faces.

2.1. Genera un dataset sintético de dos variables con el codigo:

```
1 np.random.seed(0)
2 n = 200
3 x1 = np.random.normal(0,1,n)
4 x2= 2*x1 + np.random.normal(0, 1.25, n)
5 X = np.column_stack((x1, x2))
```

1. Centra los datos usando la media.
2. Calcula la matriz de covarianza, los valores y vectores propios. Despliega estos tres.
3. Grafica los datos originales, así como los vectores propios como direcciones principales ponderados por sus respectivos valores propios.
4. Verifica si estos vectores son ortogonales entre sí

2.2. Usando el dateset Iris

1. Genera graficas de dispersión usando la función pairplot. ¿Observas algún par de característica que te permita discernir adecuadamente entre clases?

2. Centra los datos usando su media.

3. Crea una función pca_cov() que reciba la matriz de datos y el número de componentes k que se requieran, y entregue los datos proyectados Z, los k vectores propios que contengan la mayor varianza y valores propios asociados a los correspondientes vectores. Esto haciendo el calculo mediante la matriz de covarianza.

4. Aplica PCA sobre este conjunto de datos

5. Grafica la varianza explicada (scree plot) y la varianza explicada acumulada

6. Si se toma entre el 90 % y 95 % de la varianza explicada, ¿Cuántos vectores propios tomarías? Muestra cuales son los vectores elegidos

7. Proyecta los datos a las 2 componentes principales y grafica las características en el nuevo espacio, coloreando cada instancia según su clase. ¿Qué observas sobre la separación natural de los datos en el espacio PCA? ¿PCA elimina características o crea nuevas?

2.3. Usando el dateset Wine

1. Crea una función pca_svd() que reciba y entregue la misma información que la función pca_cov() del punto anterior. En este caso haciendo los cálculos mediante la técnica SVD.

2. Estandariza los datos usando su media y desviación estándar.

3. Aplica PCA usando la función pca_svd().

4. Grafica la varianza explicada (scree plot) y la varianza explicada acumulada

5. Con base en los criterios de decisión vistos en clase (regla del codo, umbral de varianza y criterio de kaiser) especifique cuantos vectores propios escogerías para cada caso.

6. Aplica PCA a este mismo dataset, pero sin estandarizar. Genera el mismo scree plot y varianza explicada acumulada. ¿Cómo cambia la distribución de varianza? ¿Qué efecto tiene el no estandarizar los datos?

7. Reconstrucción de los datos. Calcula el error cuadrático medio (MSE) para cuando se reconstruyen los datos con k=1,2,..13 y genere una gráfica que muestre como cambia el MSE conforme se aumenta el número de vectores propios (MSE vs número de componentes).

2.4. Comparación. Para el dataset Iris:

1. Aplica PCA usando las funciones pca_cov, pca_svd y PCA() implementada por la librería sklearn.
2. Despliega los 4 valores propios que entrega cada función.
3. Despliega los 4 vectores propios que entrega cada función
4. ¿Hay diferencia entre los valores y vectores propios que entrega cada función? ¿A qué se debe?

2.5. EigenFaces. Usando el dataset de imágenes de rostros Olivetti Faces

1. Asegúrate que las imágenes esten almacenados en una matriz X de (nxp) donde n es el número de imágenes y p es el número total de pixeles.
2. Centra la matriz X.
3. Aplica PCA sobre esta matriz utilizando la función pca_svd.
4. Despliega como se ve el rostro promedio (la media de la matriz X)
5. Despliega las primeras 10 eigenfaces y las 10 ultimas eigenfaces.
6. Reconstruye cualquier imagen utilizando 10,50,100,200 y 300 eigenvectores y muestra en una sola figura los resultados incluyendo la imagen original.
7. Genera la gráfica MSE vs número de componentes en donde se visualice como cambia el error conforme aumenta el numero de k vectores utilizados en la reconstrucción. Para disminuir el tiempo de procesamiento puedes utilizar intervalos de 20 vectores para generar la curva.
8. ¿Por qué el error disminuye al aumentar los componentes? ¿Cómo se relaciona el error con la varianza descartada?