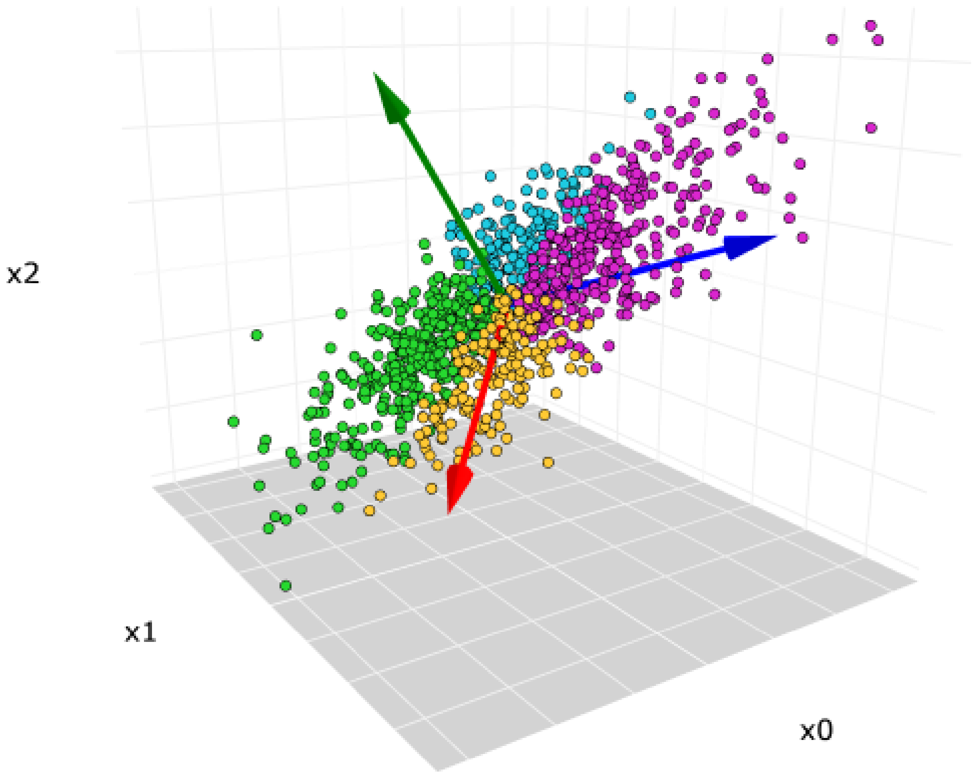
GEOMETRÍA COMPUTACIONAL

PRÁCTICA 4



DAVID SEIJAS PÉREZ

1. Introducción

En esta cuarta práctica utilizaremos el algoritmo de *Análisis de Componentes Principales: **PCA/EOF*** y aprenderemos a usarlo para reducir la dimensionalidad de un sistema y sacara sus componentes principales, así como sus pesos.

2. Material Usado

Para el desarrollo de esta práctica he utilizado diversas librerías dadas por Python. Primero, como siempre, hemos usado *matplotlib.pyplot* para la representación de gráfica y *np* para manejo de arrays. Además. hemos usado las librerías *math* y *dt* para poder hacer uso de operaciones matemáticas y de fechas, respectivamente. Por último, esta vez, hemos utilizado *Dataset* y *PCA* para representar el sistema a usar y poder utilizar el algoritmo de reducción de dimensionalidad y hallar las componentes principales de este.

Además, he cogido la plantilla *GCOM2022-practica4_plantilla* para implementar algunas funcionalidades y los archivos *air.2021.nc*, *air.2022.nc*, *hgt.2021.nc* y *hgt.2022.nc* para obtener los datos del sistema.

3. Metodología

Para el primer apartado, he hecho la función *apartado1*, reutilizando código de la plantilla 1, en la que calculo el sistema a estudiar en hgt21b juntando las variables longitud y latitud y calculo las componentes principales con el algoritmo **PCA/EOF**. Aunque el enunciado nos indica que las variables de estado son longitud y latitud (por lo que tendríamos que usar la matriz $m \times n$ con $m = 10512$ y $n = 365$), reduzco la dimensionalidad según la variable $tiempo$ (sistema Y) y según la variable $lat \times long$ (sistema X) y muestro los pesos y varianzas explicadas para ver qué variable es mejor reducir para conseguir un sistema transformado más similar al original. Una vez veo cuál es mejor utilizar, represento las componentes principales con la opción 1 dada en la plantilla.

En el segundo apartado, obtengo el subsistema en hgt21c y extraigo el elemento a_0 a estudiar con su índice según el día del archivo de datos hgt22, por lo que guardo en a_0 la matriz $presion \times lat \times long$ del día a estudiar. Posteriormente, gracias a la función implementada *dist_euclidea* con los pesos indicados, soy capaz de hallar la distancia de todos los elementos del subsistema al elemento a_0. Calculadas estas distancias, me quedo con las 4 menores, que corresponden a los elementos más análogos al que queremos estudiar. Además de las distancias, me guardo los índices en la dimensión del tiempo en el sistema de estos elementos para poder acceder a qué días son exactamente con el vector de tiempo dt_time21. Por último, calculo la media de los análogos de la variable temperatura y aplico la fórmula del error medio con la temperatura del año 2022 del elemento a_0 para ver cómo de bien hemos aproximado el elemento.

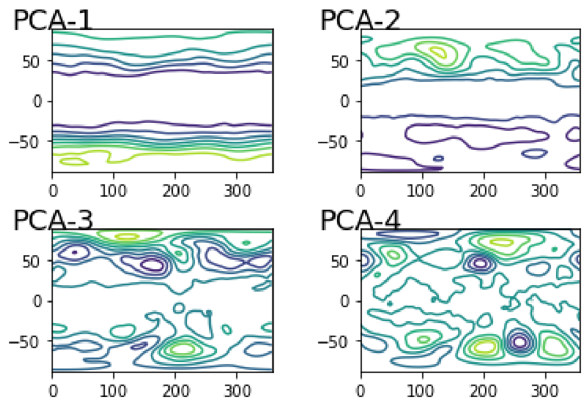
$$EAM = \frac{\sum_{i=1}^{n} \mid y_i - x_i \mid}{n}$$

4. Resultados

En el primer apartado he obtenido que las varianzas para el sistema Y (en el que reducimos los 365 elementos de tiempo) son:
$[0.8877314,\ 0.05177594,\ 0.00543983,\ 0.00357636]$
Para el sistema X (en el que reducimos los 10512 elems $lat \times long$) las varianzas obtenidas son:
$[0.4724878,\ 0.06072694,\ 0.0359264,\ 0.02815221]$

De esta forma, vemos que el sistema Y recoge mucha más información del sistema original. Vemos que la 1ª componente principal recoge un 88 % de info del sistema original, mientras que el resto recogen mucha menos. Conseguimos, aproximadamente y en total, un 94 % de información del sistema original, por lo que hemos sintetizado el sistema de una manera muy fiel.

La representación de las 4 componentes principales del sistema Y es:



En el segundo apartado, los 4 días más análogos al elemento $a0 = 2022/01/11$ (con sus distancias eulídeas al elemento) son:
$[(81,\ 396.4154622993407),\ (15,\ 467.1849874514377),\ (11,\ 524.7436874322549),$
$74,\ 525.9499738568298)]$
que corresponden, respetivamente, a los días: $2021/03/23,\ 2021/01/16,\ 2021/01/12,\ 2021/03/16$

Por último, el error medio absoluto de la temperatura previsto para el elemento a_0 con la media de la temperatura de los análogos es: $EAM = 3.072786049574296$

5. Conclusión

Nunca había utilizado antes un algoritmo de reducción de dimensionalidad y me parece increíble la utilidad de estos. He aprendido muchos conceptos nuevos y a saber cómo hallar sistemas transformados con menos dimensiones que podamos representar y entender mejor, así como aprender cómo poder reducir según distintas variables y el estudio de las varianzas para saber cuál sería mejor de reducir.

En concreto, en esta práctica, me ha sorprendido la diferencia en las varianzas que obtenemos al reducir el sistema según el tiempo, mucho más altas, que según latitud y longitud (lo cual tiene sentido al pasar de muchos más elementos a solo 4), viendo que reduciendo el tiempo sintetizamos casi a la perfección el sistema con el algoritmo PCA. Además, hemos visto como el elemento estudiado en el apartado está muy bien aproximado por los análogos pues el EAM es muy pequeño.

6. Anexo: Código

```python
"""
DAVID SEIJAS PEREZ
PRACTICA 4
"""

import datetime as dt  # Python standard library datetime  module
import numpy as np
import matplotlib.pyplot as plt
from netCDF4 import Dataset
from sklearn.decomposition import PCA
import math

workpath = "."

f = Dataset(workpath + "/air.2021.nc", "r", format="NETCDF4")
time = f.variables['time'][:].copy()
time_bnds = f.variables['time_bnds'][:].copy()
time_units = f.variables['time'].units
level = f.variables['level'][:].copy()
lats = f.variables['lat'][:].copy()
lons = f.variables['lon'][:].copy()
air21 = f.variables['air'][:].copy()
air_units = f.variables['air'].units
f.close()

f = Dataset(workpath + "/air.2022.nc", "r", format="NETCDF4")
time = f.variables['time'][:].copy()
time_bnds = f.variables['time_bnds'][:].copy()
time_units = f.variables['time'].units
air22 = f.variables['air'][:].copy()
f.close()

f = Dataset(workpath + "/hgt.2021.nc", "r", format="NETCDF4")
time21 = f.variables['time'][:].copy()
time_bnds = f.variables['time_bnds'][:].copy()
time_units = f.variables['time'].units
hgt21 = f.variables['hgt'][:].copy()
hgt_units = f.variables['hgt'].units
f.close()

f = Dataset(workpath + "/hgt.2022.nc", "r", format="NETCDF4")
```

```python
time22 = f.variables['time'][:].copy()
time_bnds = f.variables['time_bnds'][:].copy()
time_units = f.variables['time'].units
hgt22 = f.variables['hgt'][:].copy()
f.close()



def apartado1():
    hgt21b = hgt21[:,level==500.,:,:].reshape(len(time21),len(lats)*len(lons))  #365x10512
    n_components = 4
    X = hgt21b #365 elementos de 10512 variables
    Y = hgt21b.transpose() #10512 elementos de 365 variables

    pca = PCA(n_components=n_components) #crea el pca con 4 componentes principales
    Element_pca0 = pca.fit_transform(Y) #reducimos la dimensionalidad y nos quedamos con 4 comp princ (que son comb lineales de las 10512 variables)
    Element_pca0 = Element_pca0.transpose(1,0).reshape(n_components,len(lats),len(lons)) #proyecciones de hgt sobre los ejes (las 4 componentes principales) volviendo a separar lat y long

    pesos = pca.components_ #sacamos los alpha de cada variable a cada uno de las componentes principales (los coeficientes de la combinacion lineal)
    print(pesos)
    print("Varianzas de las 4 componentes principales en el sistema transformado Y:")
    print(pca.explained_variance_ratio_) #varianzas: nos dice cuanto y como se parece el sistema que transformado con las comp princ al original
    #la varianza de la 1a comp: 0.88 -> nos dice que la primera cmp es la que mas info recoge del sistema y las demas van a adiendo (5% de info cada una aprox.)
    #no tienen por qu  sumar 1 -> se puede perder info

    pca.fit(X)
    print("Varianzas de las 4 componentes principales en el sistema transformado X:")
    print(pca.explained_variance_ratio_)
    #las varianzas son peores y se pierde mucha info -> el sistema transformado no es tan bueno

    #Representacion espacial de las pca en (x, y)
    fig = plt.figure()
    fig.subplots_adjust(hspace=0.4, wspace=0.4)
    for i in range(1, 5):
        ax = fig.add_subplot(2, 2, i)
        ax.text(0.5, 90, 'PCA-'+str(i),
               fontsize=18, ha='center')
        plt.contour(lons, lats, Element_pca0[i-1,:,:])
    plt.show()
```

```python


def dist_euclidea(a0, dia_aux):
    d = 0
    for i in range(len(a0[0])): #latitud
        for j in range(len(a0[0][0])): #longitud
            #Para cada elemento aplico su peso w_k segun su p_k
            #Al hacer a0[level == 500.] me quedo con la fila de level = 500, pero sigo teniendo matriz de dim 3 aunque en la dim 1 solo tengo una "fila"
            d += 0.5*((a0[level == 500.][0][i][j] - dia_aux[level == 500.][0][i][j])**2)
            d += 0.5*((a0[level == 1000.][0][i][j] - dia_aux[level == 1000.][0][i][j])**2)
    return math.sqrt(d)


def apartado2():
    #Subsistema de S
    hgt21c = hgt21[:,:,:,np.logical_or(340 < lons, lons < 20)]
    hgt21c = hgt21c[:,:,np.logical_and(30 < lats, lats < 50),:]

    dt_time22 = [dt.date(1800, 1, 1) + dt.timedelta(hours=t) for t in time22]
    dia_a0 = dt.date(2022, 1, 11)
    index_a0 = dt_time22.index(dia_a0)

    #Dia a0 a estudiar
    a0 = hgt22[index_a0,:,:,:]
    a0 = a0[:,:,np.logical_or(340 < lons, lons < 20)]
    a0 = a0[:,np.logical_and(30 < lats, lats < 50),:]

    #Sacamos las distancias euclideas de cada dia con el a0 a estudiar y nos quedamos con los 4 m s peque as
    n_dias = 4
    distancias = [(i, dist_euclidea(a0, hgt21c[i])) for i in range(len(hgt21c))]
    distancias = sorted(distancias, key=lambda x : x[1])
    dist_analogos = distancias[0:n_dias]
    print("An logos a a0 seg n variable Z con su distancia:")
    print(dist_analogos)

    #Vemos cuales son los 4 d as an logos (menor distancia eucl dea) a a0
    dt_time21 = [dt.date(1800, 1, 1) + dt.timedelta(hours=t) for t in time21]
    analogos = [dt_time21[dist_analogos[i][0]] for i in range(n_dias)]
    print("D as an logos a a0 seg n variable Z:")
    print(analogos)

    #Hallamos la media de la variable de estado T (para p_k=1000) de los analogos
    media = 0
    for i in range(n_dias):
```

```
        media = np.add(media, air21[dist_analogos[i][0]][level == 1000])
    media = media*(1/n_dias)

    #Hallamos el error absoluto medio de esta variable
    a0 = (air22[index_a0][level == 1000])*(-1) #variable T para dia a0
    eam = (np.sum(abs(np.add(media, a0))))/(73*144) #formula del eam
    print("Error absoluto medio de la temperatura:")
    print(eam)

print("------ APARTADO 1 ------")
apartado1()
print("------ APARTADO 2 ------")
apartado2()
```