1. Introducción

En esta práctica estudiaremos la atmósfera considerada como un sistema de 6 variables de estado (t, x, y, p, Z, T) de las cuales dos de ellas (Z y T) dependen del tiempo t.

En el primer apartado estimaremos las 4 componentes principales del sistema $S = \{a_d, X_d\}$ formado por los días a_d del año 2019 y las variables de estado $X_d = \{Z_{i,j,k}\}$ con $p_k = 500hPa$.

En el segundo apartado buscaremos los 4 elementos de un subsistema de S más análogos a uno a_0 del año 2020. Después calcularemos el error absoluto medio de la temperatura prevista para dicho elemento a_0 según la media de los análogos.

2. Material empleado

2.1. Análisis de componentes principales

En este primer apartado, empezamos cargando los datos de la altura geopotencial del año 2019 sobre los que vamos a trabajar. A continuación buscamos el índice correspondiente al nivel de presión $500hPa$ y extraemos la submatriz correspondiente a la que aplicaremos el algoritmo de análisis de componentes principales (PCA).

El algoritmo de análisis de componentes principales toma la matriz anterior y la interpreta como una evolución temporal del día d_0 correspondiente al 1 de enero de 2019. A partir de la matriz de covarianza, extrae los autovectores y los ordena de mayor a menor según los autovalores. De estos autovectores selecciona los p mejores (en nuestro caso 4 que es el número de componentes que buscamos) y esos son las componentes principales. Estas componentes principales no tienen porque ser días de nuestro conjunto si no que pueden ser combinaciones lineales de estos.

2.2. Analogía

En este apartado primero cargamos los datos de la altura geopotencial del año 2020 para poder extraer los del día 20 de enero. Después extraemos los índices que nos van a permitir delimitar σ que es el subsistema de S.

Cargamos también los datos de temperaturas del día 20 de enero de 2020 a $1000hPa$ d_0a que es sobre el que calcularemos el error.

A continuación, guardamos los datos de la altura geopotencial del año 2019 y calculamos su distancia al día 20 de enero de 2020 con la siguiente expresión:

$$d = \sqrt{\sum_{x\in(-20°,20°)} \sum_{y\in(30°,50°)} w_{500hPa}(Z_{x,y,500} - Z^0_{x,y,500})^2 + w_{1000hPa}(Z_{x,y,1000} - Z^0_{x,y,1000})^2}$$

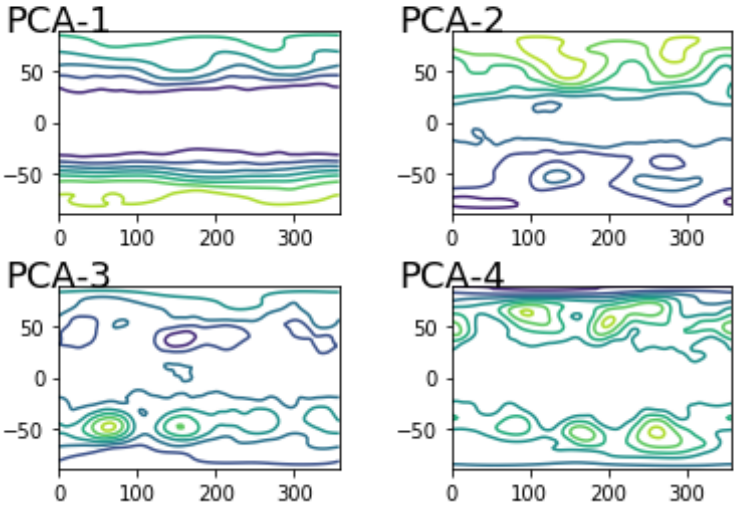
Donde $w_{500hPa} = w_{1000hPa} = 0,5$ son los pesos de dichas presiones y $Z_{x,y,p}$ y $Z^0_{x,y,p}$ son los valores de la altura geopotencial, en esas coordenadas y presión, del día a comparar

y el 20 de enero de 2020, respectivamente.

Extraemos los índices de los cuatro días más análogos (menor distancia) y cargamos los datos de las temperaturas de esos días del año 2019. Calculamos la media de estos cuatro días y así obtendríamos nuestra predicción de d_0a. Para calcular el error absoluto medio hacemos el valor absoluto de la diferencia entre los dos días y calculamos la media de la matriz obtenida.

3. Resultados

```
Valores explicados en función del número de componentes:
[0.86579346 0.93789939 0.94300602 0.94717256]
```




```
Apartado 2:

Los 4 días más análogos son: [78, 79, 109, 335]

El error absoluto medio de la temperatura prevista
para el día a0 es 4.2725863
```

4. Conclusión

En el primer apartado vemos que ya una única componente principal nos permite explicar el 86 % del ratio de varianza y que la tercera y la cuarta componente, a penas aportan información en este aspecto.

En el segundo apartado vemos que los días del año 2019 más análogos al 20 de enero de 2020 no son días próximos al 20 de enero de 2019 si no días de mediados de marzo, mediados de abril y principios de diciembre.

5. Código

```
# -*- coding: utf-8 -*-
"""
Referencias:

Fuente primaria del reanálisis
https://www.esrl.noaa.gov/psd/data/gridded/
    data.ncep.reanalysis2.pressure.html

Altura geopotencial en niveles de presión
https://www.esrl.noaa.gov/psd/cgi-bin/db_search/
    DBListFiles.pl?did=59&tid=81620&vid=1498

Temperatura en niveles de presión:
https://www.esrl.noaa.gov/psd/cgi-bin/db_search/
    DBListFiles.pl?did=59&tid=81620&vid=1497

"""
#import os
import datetime as dt  # Python standard library datetime  module
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import netcdf as nc
from sklearn.decomposition import PCA
import math

"""
Apartado 1
"""

def apartado1():
    print("Apartado 1:\n")

    #Cargamos los datos de altura geopotencial del 2019
    f = nc.netcdf_file("hgt.2019.nc", 'r')
    time = f.variables['time'][:].copy()
    level = f.variables['level'][:].copy() #los valores de p
    lats = f.variables['lat'][:].copy() #los valores de y
    lons = f.variables['lon'][:].copy() #los valores de x
    hgt = f.variables['hgt'][:].copy() #los valores de cada día
    f.close()
```

```
"""
Ejemplo de evolución temporal de un elemento de hgte

plt.plot(time, hgt[:, 1, 1, 1], c='r')
plt.show()

"""

PRESION = 500
for i in range(len(level)):
    if level[i]==PRESION:
        p500=i
        break

plt.title("Distribución espacial de la temperatura en
    el nivel de 500hPa, para el primer día")
plt.contour(lons, lats, hgt[0,p500,:,:])
plt.show()


#Seleccionamos los datos con p = 500hPa y redimensionamos la matriz
hgt2 = hgt[:,p500,:,:].reshape(len(time),len(lats)*len(lons))

n_components=4

Y = hgt2.transpose()
pca = PCA(n_components=n_components)

#Aplicamos el algoritmo PCA
pca.fit(Y)
print("Valores explicados en función del número de componentes:",
    pca.explained_variance_ratio_.cumsum())

#Ejercicio de la práctica
#Representación espacial de las componentes principales
Element_pca = pca.fit_transform(Y)
Element_pca = Element_pca.transpose(1,0)
    .reshape(n_components,len(lats),len(lons))
fig = plt.figure()
fig.subplots_adjust(hspace=0.4, wspace=0.4)
for i in range(1, n_components+1):
    ax = fig.add_subplot(2, 2, i)
    ax.text(0.5, 90, 'PCA-'+str(i),
```

```
                fontsize=18, ha='center')
            plt.contour(lons, lats, Element_pca[i-1,:,:])
        plt.show()
        print("\n\n\n\n")



"""
apartado 2
"""
IND_500=5
IND_1000=0

#Calculala distancia euclídea entre dos días en un subconjunto del sistema
def dist_euc(d0,d,min_lat,max_lat,min_lon,max_lon,lons):
    dist=0
    for i in range(min_lat+1,max_lat):
        for j in range(min_lon+1,max_lon):
            dist+=
                0.5*(d0[IND_500,i,j%lons]-d[IND_500,i,j%lons])**2+
                0.5*(d0[IND_1000,i,j%lons]-d[IND_1000,i,j%lons])**2
    return math.sqrt(dist)


def apartado2():
    print("Apartado 2:\n")

    #Cargamos los datos de altura geopotencial del 2020
    f = nc.netcdf_file("hgt.2020.nc", 'r')
    time = f.variables['time'][:].copy()
    lats = f.variables['lat'][:].copy() #los valores de y
    lons = f.variables['lon'][:].copy() #los valores de x
    hgt = f.variables['hgt'][:].copy() #los valores de cada día
    f.close()

    #Calculamos el índice correspondiente al día 2020/01/20
    for t in range(len(time)):
        if dt.date(1800, 1, 1) + dt.timedelta(hours=time[t]) ==
            dt.date(2020,1,20):
            break

    double_lons=np.concatenate((lons, lons), axis=None)
```

```
min_lon=-1
for max_lon in range(len(double_lons)):
    if double_lons[max_lon]==340:
        min_lon=max_lon
    if double_lons[max_lon]==20 and not min_lon==-1:
        break

for max_lat in range(len(lats)):
    if lats[max_lat]==50:
        min_lat=max_lat
    if lats[max_lat]==30:
        break


#cargamos los datos de altura geopotencial del día 2020/01/20
d0h = hgt[t,:,:,:]


#cargamos los datos de temperatura del 2020
f = nc.netcdf_file("air.2020.nc", 'r')
air = f.variables['air'][:].copy() #los valores de cada día
tem_scale_factor=f.variables['air'].scale_factor.copy()
tem_add_offset=f.variables['air'].add_offset.copy()
f.close()

#cargamos los datos de altura geopotencial del día 2020/01/20
d0a = air[t,0,:,:]*tem_scale_factor+tem_add_offset


#cargamos los datos de altura geopotencial del 2019
f = nc.netcdf_file("hgt.2019.nc", 'r')
lons = f.variables['lon'][:].copy() #los valores de x
hgt = f.variables['hgt'][:].copy() #los valores de cada día
f.close()



#Calculamos la distancia de cada día de nuestro subsistema al
    día 2020/01/20
distancias=
    [[dist_euc(d0h,hgt[i,:,:,:]
        ,min_lat,max_lat,min_lon,max_lon,len(lons)),i]
            for i in range(hgt.shape[0])]
```

```
    distancias.sort()

    take=[j for i,j in distancias[0:4]]
    print("Los 4 días más análogos son:",take,"\n")

    #cargamos los datos de temperatura de 2019
    f = nc.netcdf_file("air.2019.nc", 'r')
    air = f.variables['air'][:].copy() #los valores de cada día
    tem_scale_factor=f.variables['air'].scale_factor.copy()
    tem_add_offset=f.variables['air'].add_offset.copy()
    f.close()

    #Transformamos los datos para que estén en grados Kelvin
    air=air*tem_scale_factor+tem_add_offset

    #Calculamos la media de los 4 días más análogos a 2020/01/20
    mediaDias=np.mean(air[take,0,:,:],axis=0)

    print("El error absoluto medio de la temperatura prevista para
        el día a0 es", np.mean(abs(mediaDias-d0a)))


apartado1()
apartado2()
```