Memoria Practica 4 – PCA

Jorge del Valle Vázquez

1 Introducción

El **PCA** es un algoritmo que proporciona una forma de compresión de datos, o desde otro punto de vista, un algoritmo no supervisado que aprende una proyección lineal dando lugar a mayor varianza en el sentido de los ejes. La emplearemos para maximizar la información minimizando la cantidad de variables.

2 Material usado

La práctica gira en torno al método **PCA** de la librería **sklearn. decomposition**. Los datos proceden de la fuente primaria de reanálisis proporcionados por el docente, y se trabajan con la librería **netCDF4**. La graficas se realizan por medio de **pyplot**.**contour** de **matplotlib**.**pyplot**.

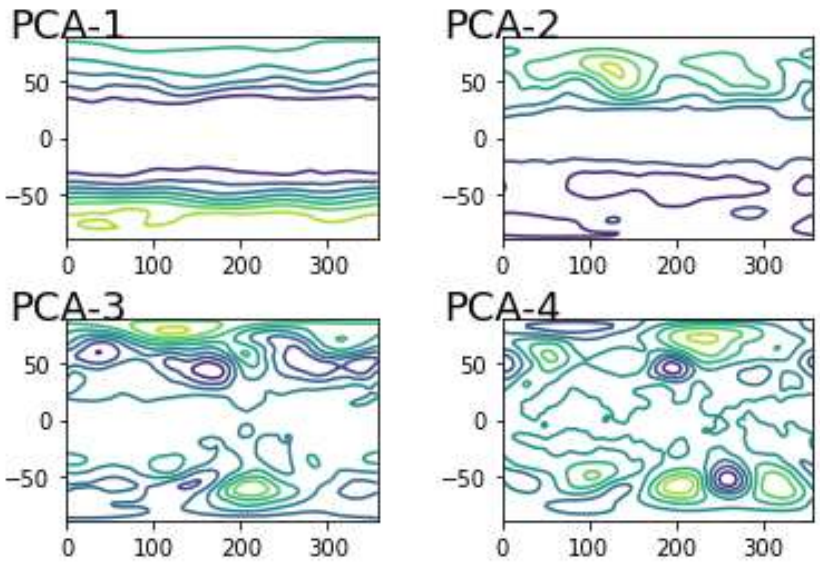
En cuanto a los ejercicios, se hace uso de la plantilla proporcionada en la obtención de datos de los ficheros air21, air22 y hgt21, y de entre las dos opciones plateadas para el primer ejercicio se hace uso de la primera, que trabaja con la traspuesta, pues proporciona los mejores resultados. De esta forma, en el primero de los ejercicios se obtienen las cuatro componentes principales deseadas por medio del PCA y en el segundo obtenemos los cuatro días de 2021 más semejantes al 11 de enero de 2022 atendiendo a la altura geopotencial Z para luego observar la temperatura media sobre estos cuatro y observar el error respecto a lo contemplado en los datos para el día 11-1-2022.

3 Resultados

i. Como se ha comentado, obtenemos las 4 componentes principales como combinación lineal de los datos correspondientes a cada día. Dando lugar a un "pca.explained_variance_ratio_" o porcentaje de varianza [0.8877314 0.05177598 0.00543984 0.00357636] que supone una perdida de información pequeña, no obteniendo el mismo resultado de haber empleado la opción de trabajar con X y no con su traspuesta, pues se tendría [0.4724878 0.0607269 0.03592641 0.02815221], perdiendo en torno a un 40 % de la información, mientras que con la opción elegida ronda el 5%, estando recogida la información del sistema en mayor medida en la primera componente y en menor en la cuarta.

   De los resultados obtenidos podemos descartar que cada componente principal quede asociada a una estación del año, pues de los pesos asociados por cada componente a cada día no se aprecia preferncia o valores mayores en ninguna sección,

   [-0.05076396 -0.05139843 -0.05165739 -0.05144301 -0.05158139 -0.05144246
   -0.05094209 -0.04949634 -0.04805459 -0.04811923 -0.04835887 ……
   …….. -0.04661608 -0.0466941 -0.04641966 -0.04663102 -0.04886592
   -0.04981095 -0.05035869 -0.05003722 -0.0487005 -0.04851938]



ii. En el segundo apartado buscamos los días de 2021 con valores de Z más semejantes al día 11-01-2022, atendiendo a la distancia euclídea con pesos 0 salvo para presiones de 500hPa y 1000hPa, que tenían peso 0.5. Obteniendo así los días (distancia asociada):

Dia **81**: distancia 396.4154622993407, Dia **15**: distancia 467. 1849874514377

Dia **11**: distancia 524. 7436874322549 Dia **74**: distancia 525. 9499738568298

Ahora observamos los datos sobre temperaturas de estos días para obtener una media de ellos y obtener el error absoluto medio respecto al día 11 de enero de 2022, teniendo este un valor de **3.072786049574296**

4 Conclusión

De los resultados obtenidos podemos destacar la evidente mejora del PCA al permitir trabajar con solo 4 componentes sacrificando tan solo un 5% de la información, junto con la característica previamente comentada de la independencia de las componentes con las estaciones del año.

Del segundo ejercicio podemos apreciar que días parejos, en lo que altura geopotencial se refiere, tienen a su vez temperaturas similares

5 Anexo: Código

```
import datetime as dt
import numpy as np
import matplotlib.pyplot as plt
from netCDF4 import Dataset
from sklearn.decomposition import PCA
workpath = ""
import os
os.getcwd()
import math
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")
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()

#############################################################################
#
# dimension 365 * (73*144)
hgt21b = hgt21[:,level==500.,:,:].reshape(len(time21),len(lats)*len(lons))
air21b = air21[:,level==500.,:,:].reshape(len(time21),len(lats)*len(lons))
n_components=4
X = hgt21b
Y = hgt21b.transpose()
pca = PCA(n_components=n_components)
Element_pca0 = pca.fit_transform(Y)
Element_pca0 =
Element_pca0.transpose(1,0).reshape(n_components,len(lats),len(lons))
print(pca.components_[0],pca.explained_variance_ratio_)

#Ejercicio de la práctica - Opción 1
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()

###############################################################################
#

#hgt21b = hgt21[:,:,np.logical_and(30 < lats, lats < 50),np.logical_or(340 <
lons, lons < 20)]
hgt21b = hgt21[:,:,np.logical_and(30 < lats, lats < 50),:]
hgt21b = hgt21b[:,:,:,np.logical_or(340 < lons, lons < 20)]
dt_time22 = [dt.date(1800, 1, 1) + dt.timedelta(hours=t) for t in time22]
dia0_fecha = dt.date(2022, 1, 11)
dia0_ind = dt_time22.index(dia0_fecha)
#dia0 = hgt22[dia0_ind,:,np.logical_and(30 < lats, lats <
50),np.logical_or(340 < lons, lons < 20)]
dia0 = hgt22[dia0_ind,:,:,:]
dia0 = dia0[:,np.logical_and(30 < lats, lats < 50),:]
dia0 = dia0[:,:,np.logical_or(340 < lons, lons < 20)]
#dia0 = (latitud,longitud)
def dist_euclidea(dia):
    s = 0
    for i in range(len(dia0[0])):
        for j in range(len(dia0[0][0])):
            s += 0.5*((dia0[level == 500.][0][i][j] - dia[level ==
500.][0][i][j])**2)
            s += 0.5*((dia0[level == 1000.][0][i][j] - dia[level ==
1000.][0][i][j])**2)
    return math.sqrt(s)
dias = 4
distancias = [(i, dist_euclidea(hgt21b[i])) for i in range(len(hgt21b))]
distancias_ord = sorted(distancias, key=lambda e : e[1])
distancias_best = distancias_ord[0:dias]
print('(dia,distancias) mas analogas',distancias_best)
temp_sum = air21[distancias_best[0][0]][level == 1000]
#dim (1,73,144)
for i in range(1,dias):
    temp_sum = np.add(temp_sum, air21[distancias_best[i][0]][level == 1000])
temp_media = temp_sum*(1/dias)
dia0 = (air22[dia0_ind][level == 1000])*(-1)# -1 para restar a temp_media
error_abs_med = (np.sum(abs(np.add(temp_media,dia0))))/(73*144)
print('error absoluto medio',error_abs_med)
```