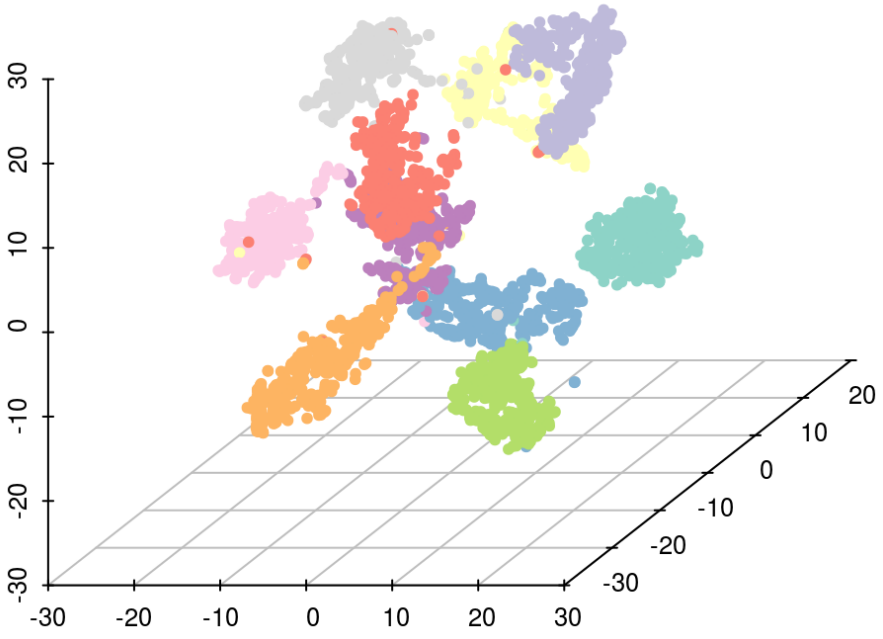
GEOMETRÍA COMPUTACIONAL

PCA y Analogía



Lucas de Torre

Índice

1. Introducción

Consideramos la atmósfera como un sistema de 6 variables: tiempo (t), longitud (x, discretizada en 144 valores), latitud (y, discretizada en 73 valores), presión (p, discretizada en17 valores), temperatura (T) y altura geopotencial (Z), siendo las dos últimas las únicas variables dinámicas respecto a la variable tiempo. A partir de ahí, considerando el sistema $S = \{a_d, X_d\}_{d=1}^{365}$, de los días a_d del año y las variables de estado $X_d = \{Z_{i,j,k}\}_{i=1,j=1,k=1}^{i=144,j=73,k=17}$, estimaremos las 4 componentes principales, las representaremos espacialmente en (x,y) y veremos el porcentaje de varianza explicado.

Además, consideraremos el subsistema $\sigma \in S$ tal que x $\in$ (340º,20º) e y $\in$ (30º,50º). En este subsistema σ, teniendo en cuenta solo la altura geopotencial, hallaremos los 4 días más análogos al día $a_0 = 2020/01/20$ y calcularemos el error absoluto de la temperatura prevista para a_0 por la media de esos 4 análogos.

2. Material

Para la estimación de las 4 componentes principales, primero cargamos los datos de altura geopotencial de 2019. De esos datos, nos quedamos con aquellos cuyo nivel de presión sea de 500hPa.

Después, haciendo uso de la biblioteca *sklearn*, utilizamos el método PCA (con 4 componentes principales como argumento) para estimar las 4 componentes principales. Por último, utilizamos otro método de la librería sklearn para mostrar el ratio de varianza explicado y dibujamos las componentes principales en el plano.

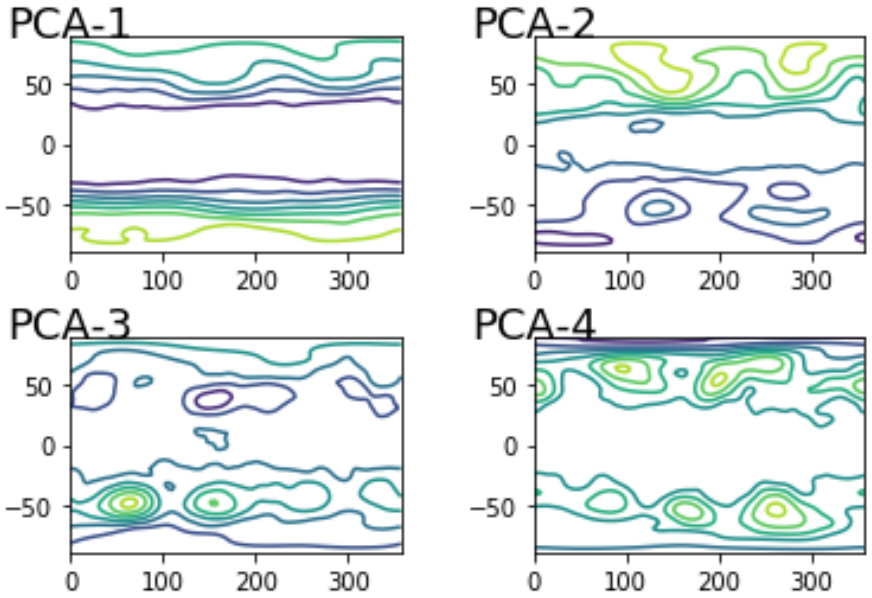
Procedemos ahora a buscar los 4 días más análogos al día a_0. Para ello, cargamos los datos de altura geopotencial de dicho día (para lo que hallamos el índice que corresponde a dicho día en la matriz). Después, cargamos los datos de altura geopotencial de 2019 y calculamos la distancia euclídea de cada uno de los días de 2019 al día a_0. Nos quedamos con los 4 días más análogos al día a_0 (aquellos cuya distancia a a_0 fuera menor), digamos $d1$, $d2$, $d3$, $d4$.

Después, cargamos los datos de temperatura del día a_0 y los datos de temperatura de 2019 (estos datos y los del día a_0 los transformamos a grados kelvin). De estos últimos, hacemos la media ($\overline{d}$) únicamente de los 4 días ($d1$, $d2$, $d3$, $d4$) calculados antes (eran los más parecidos respecto a altura geopotencial, y ahora calculamos su media respecto a temperatura).

Por último, calculamos la media del valor absoluto de las diferencias entre $\overline{d}$ y a_0. De esta manera, hemos calculado el error absoluto de la temperatura prevista para a_0 por la media $\overline{d}$ de los 4 días más análogos a a_0 según la altura geopotencial.

3. Resultados

Los valores explicados en función del número de componentes principales son: 0.86579346, 0.93789939, 0.94300602 y 0.94717256, en función de si el número de componentes principales es 1, 2, 3 ó 4, respectivamente. Por tanto, los porcentajes son 86.579346, 93.789939, 94.300602 y 94.717256, respectivamente. Su representación espacial en (x,y) es:



Por otro lado, los 4 días de 2019 más análogos al día a_0 según la altura geopotencial son: 78, 79, 109 y 335. El error absoluto de la temperatura prevista para a_0 por la media de esos 4 análogos es: 4.2725863.

4. Conclusiones

Llama la atención que los días más análogos al día a_0 se reparten bastante a lo largo del año, habiendo 2 en invierno, 1 en primavera y otro en otoño. Sin embargo, los 4 días son muy próximos al comienzo o al final del invierno (lo que es lógico, porque tendrán unas condiciones parecidas).

5. Anexo A: Código

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

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

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

    Temperatura en niveles de presi n:
    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 d a
    f.close()

    """
    Ejemplo de evoluci n temporal de un elemento de hgte

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

```
45
46      PRESION = 500
47      for i in range(len(level)):
48          if level[i]==PRESION:
49              p500=i
50              break
51
52
53      plt.title("Distribuci n espacial de la altura geopotencial en
        el nivel de 500hPa, para el primer d a")
54      plt.contour(lons, lats, hgt[0,p500,:,:])
55      plt.show()
56
57
58      #Seleccionamos los datos con p = 500hPa y redimensionamos la
        matriz
59      hgt2 = hgt[:,p500,:,:].reshape(len(time),len(lats)*len(lons))
60
61      n_components=4
62
63      Y = hgt2.transpose()
64      pca = PCA(n_components=n_components)
65
66      #Aplicamos el algortimo PCA
67      pca.fit(Y)
68      print("Valores explicados en funci n del n mero de
        componentes:",pca.explained_variance_ratio_.cumsum())
69
70
71      #Ejercicio de la pr ctica
72      #Representaci n espacial de las componentes principales
73      Element_pca = pca.fit_transform(Y)
74      Element_pca = Element_pca.transpose(1,0).reshape(n_components,
        len(lats),len(lons))
75      fig = plt.figure()
76      fig.subplots_adjust(hspace=0.4, wspace=0.4)
77      for i in range(1, n_components+1):
78          ax = fig.add_subplot(2, 2, i)
79          ax.text(0.5, 90, 'PCA-'+str(i),
80                  fontsize=18, ha='center')
81          plt.contour(lons, lats, Element_pca[i-1,:,:])
82      plt.show()
83      print("\n\n\n\n")
84
85
86
87
88  """
89  apartado 2
```

```
"""
IND_500=5
IND_1000=0

#Calcula la distancia eucl dea entre dos d as 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 d a
    f.close()

    #Calculamos el  ndice  correspondiente al d a 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 d a 2020/01/20
    d0h = hgt[t,:,:,:]
```

```
136
137
138     #cargamos los datos de temperatura del 2020
139     f = nc.netcdf_file("air.2020.nc", 'r')
140     air = f.variables['air'][:].copy() #los valores de cada d a
141     tem_scale_factor=f.variables['air'].scale_factor.copy()
142     tem_add_offset=f.variables['air'].add_offset.copy()
143     f.close()
144
145     #cargamos los datos de temperatura del d a 2020/01/20
146     d0a = air[t,0,:,:]*tem_scale_factor+tem_add_offset
147
148
149     #cargamos los datos de altura geopotencial del 2019
150     f = nc.netcdf_file("hgt.2019.nc", 'r')
151     lons = f.variables['lon'][:].copy() #los valores de x
152     hgt = f.variables['hgt'][:].copy() #los valores de cada d a
153     f.close()
154
155
156
157     #Calculamos la distancia de cada d a de nuestro subsistema al
        d a 2020/01/20
158     distancias=[[dist_euc(d0h,hgt[i,:,:,:],min_lat,max_lat,min_lon,
        max_lon,len(lons)),i]for i in range(hgt.shape[0])]
159     distancias.sort()
160
161     take=[j for i,j in distancias[0:4]]
162     print("Los 4 d as m s an logos son:",take,"\n")
163
164     #cargamos los datos de temperatura de 2019
165     f = nc.netcdf_file("air.2019.nc", 'r')
166     air = f.variables['air'][:].copy() #los valores de cada d a
167     tem_scale_factor=f.variables['air'].scale_factor.copy()
168     tem_add_offset=f.variables['air'].add_offset.copy()
169     f.close()
170
171     #Transformamos los datos para que est n en grados Kelvin
172     air=air*tem_scale_factor+tem_add_offset
173
174     #Calculamos la media de los 4 d as m s an logos a 2020/01/20
175     mediaDias=np.mean(air[take,0,:,:],axis=0)
176
177     print("El error absoluto medio de la temperatura prevista para
        el d a a0 es",np.mean(abs(mediaDias-d0a)))
178
179
180 apartado1()
181 apartado2()
```