Práctica 4: PCA y Analogía

Guillermo Martín Sánchez

28/03/20

1 Introducción

El objetivo de esta práctica es el estudio del método *Principal Component Analysis* (PCA) y la búsqueda de los n elementos más análogos, aplicados a las condiciones atmosféricas del planeta en los años 2019 y 2020. Para ello usamos los archivos NetCDF; que registran la temperatura T y la altura geopotencial Z de cada día del año en 144 longitudes, 73 latitudes y 17 niveles de presión.

2 Material usado

2.1 Ejercicio 1

Considera el sistema $S = \{a_d, X_d\}_{d=1}^{365}$ *formado por los elementos 'días del año 2019' y las variables de estado* $X_d := \{Z_{i,j,k}\}_{i=1,j=1,k=1}^{i=144,j=73,k=17}$ *y estima las 4 componentes principales fijando* $p_k = 500hPa$*. Represéntalas espacialmente en* (x, y)*. ¿Qué porcentaje de varianza se explica?.*

Hemos leído los datos de altura geopotencial del archivo NetCDF y nos hemos quedado con el subsistema de todos los días con 144×73 alturas geopotenciales (las variables de estado), donde la presión es $p_k = 500hPa$. Luego, hemos usado la clase *PCA* de la librería *sklearn.decomposition* para calcular las 4 componentes principales. Como estas componentes son vectores en un espacio 144×73 dimensional, vemos el valor en cada componente como la altura geopotencial en ese punto (x, y), es decir, en esa longitud y latitud, y podemos representarlas como mapas de contorno de la altura geopotencial en cada punto del planeta. Finalmente, usamos el método *explained_variance_ratio_* de la clase *PCA* para ver que varianza explican cada una de estas componentes.

2.2 Ejercicio 2

Considera un subsistema $\sigma \subset S$ *delimitado por* $x \in (-20^o, 20^o)$*,* $y \in (30^o, 50^o)$*. Considerando sólo Z, encuentra los 4 días de 2019 más análogos al elemento* $a_0 :=$ "2020/01/20" *y calcula el error absoluto medio de la temperatura* $\{T_{i,j,1000hPa}\}_{i=1,j=1}^{i=144,j=73}$ *prevista para el elemento* a_0 *según la media de dichos análogos. Para la analogía,*

considera la distancia euclídea de elementos de σ con los pesos $w_{i,j,k} = w_{i,j}w_k$, donde $w_{i,j} = 1$ para las coordenadas (x, y) y $w_{500hPa} = w_{1000hPa} = 0,5$ y $w_k = 0$ para el resto de p_k.

Realizamos los siguientes pasos:

- 1. Cargamos los datos de altura geopotencial del año 2020 y extraemos el día "2020/01/20". Nos quedamos con el subsistema a_0 de la altura geopotencial en 17 presiones, 17 longitudes y 9 latitudes que representan los valores que satisfacen $x \in [-20^o, 20^o]$ y $y \in [30^o, 50^o]$ (incluimos los extremos).
- 2. Cargamos los datos de altura geopotencial del año 2019 y de nuevo extraemos de cada día el subsistema de $17 \times 17 \times 9$ valores.
- 3. Calculamos la distancia de cada uno de estos días a a_0 mediante la fórmula:

$$d(a,b) = \sqrt{0.5 \cdot \sum_{i=1,j=1}^{i=17,j=9} (a_{i,j,1000hPa} - b_{i,j,1000hPa})^2 + 0.5 \cdot \sum_{i=1,j=1}^{i=17,j=9} (a_{i,j,500hPa} - b_{i,j,500hPa})^2} \tag{1}$$

- 4. Seleccionamos los 4 días más cercanos a a_0: $\{a_d\}_{d=1}^4$
- 5. Cargamos los datos de temperatura del año 2019 y los del año 2020 y extraemos los días $\{a_d\}_{d=1}^4$ y a_0 respectivamente. Luego, nos quedamos con los subsistemas con presión $p_k = 1000hPa$
- 6. Calculamos el error absoluto medio.
  - 6.a. Calculamos la media de temperatura de los cuatro días $\{a_d\}_{d=1}^4$ en cada punto de la malla:

$$\widetilde{T}_{i,j} = \frac{\sum_{d=1}^4 T_{i,j}^{(a_d)}}{4} \quad \forall i = 1, .., 144, j = 1, .., 73. \tag{2}$$

  - 6.b. Calculamos el error cometido en cada punto con el día a_0:

$$\delta T_{i,j} = |T_{i,j}^{(a_0)} - \widetilde{T}_{i,j}| \quad \forall i = 1, .., 144, j = 1, .., 73. \tag{3}$$

  - 6.c. Calculamos la media de los errores:

$$MAE = \frac{\sum_{i=1,j=1}^{i=144,j=73} \delta T_{i,j}}{144 \cdot 73} \tag{4}$$

3 Resultados y discusión

3.1 Ejercicio 1

En la Figura 1 vemos cada uno de las 4 componentes principales representadas espacialmente. Obtenemos que, en orden, cada una explica un porcentaje de varianza de, aproximadamente, 86.6%, 7.2%, 0.5% y 0.4%. En total, estos 4 componentes explican un total del 94.7%. Es sorprendente ver que tenemos un porcentaje de explicación tan alta usando sólo 4 vectores en vez de $144 \cdot 73 = 10512$. Además, vemos que la primera componente ya explica por sí sola mucho porcentaje de la varianza, y este decae rápidamente conforme cogemos más componentes.

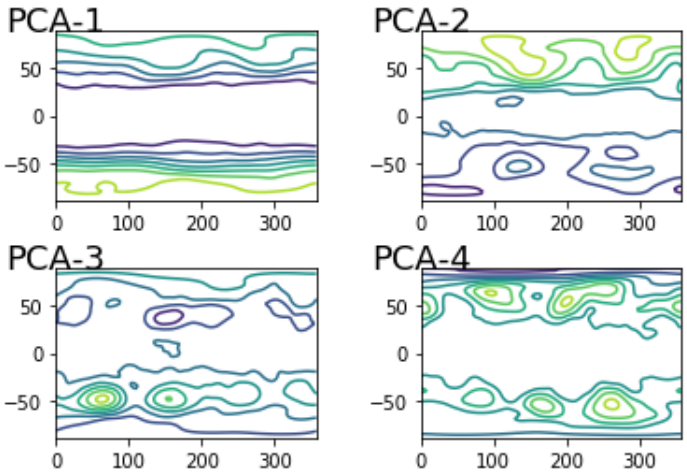

Figure 1: Cada uno de las 4 componentes principales representadas como un mapa de contorno de alturas geopotenciales en una malla que cubre el planeta

3.2 Ejercicio 2

Obtenemos que los cuatro días de 2019 más análogos con respecto a la altura geopotencial al "2020/01/20" en la región determinada (que incluye principalmente la Península Ibérica) son, en orden de menor a mayor distancia, "2019/03/20", "2019/04/20", "2019/03/21" y "2019/12/02". Además vemos que estos días son bastante similares en temperatura al "2020/01/20" puesto que el MAE calculado es de $3.6187907687869236K \approx 3.62K$ (equivalentemente 3.62ºC) lo cuál es bastante bajo.

4 Conclusión

Por un lado hemos visto la gran utilidad de métodos de reducción de dimensiones como es el PCA, que expresan datos en espacios con muchas dimensiones con un número mucho menor de componentes principales. Por otro lado hemos visto cómo calcular elementos análogos a un elemento dado definiendo una distancia y cómo el estudio de estos elementos análogos nos puede dar información desconocida de otros elementos parecidos. En concreto, hemos visto que elementos parecidos en cuanto a altura geopotencial también son parecidos respecto a temperatura y podemos usar para estudiar un elemento, los que sean parecidos a él.

5 Código

```
"""
Coursework 4: PCA and Analogous finding


References:

    https://www.esrl.noaa.gov/psd/data/gridded/data.ncep.reanalysis2.pressure.html

    https://www.esrl.noaa.gov/psd/cgi-bin/db_search/DBListFiles.pl?did=59&tid=81620&vid=149

    https://www.esrl.noaa.gov/psd/cgi-bin/db_search/DBListFiles.pl?did=59&tid=81620&vid=149

"""
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

# Compute the euclidean distance between two days
def distance(a,b):
    fact1 = 0.5*np.sum((a[5,:,:] - b[5,:,:]).astype('int64')**2)
    fact2 = 0.5*np.sum((a[0,:,:] - b[0,:,:]).astype('int64')**2)
    return np.sqrt(fact1 + fact2)

# Compute the n most analogous days to a given target day a0 from a set an
def analogues(a0,an,n):
    dis = [distance(a0,a) for a in an]
    ind = np.argsort(dis)[:n]
    return ind
```

```
#%%
"""
Exercise 1: PCA
"""

# Load data and attributes

workpath = "C:/Users/guill/Documents/Carrera/GEOComp/PCA"
os.getcwd()
files = os.listdir(workpath)

f = nc.netcdf_file(workpath + "/hgt.2019.nc", 'r')

print(f.history)
print(f.dimensions)
print(f.variables)
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()
hgt = f.variables['hgt'][:].copy()
hgt_units = f.variables['hgt'].units
hgt_scale = f.variables['hgt'].scale_factor
hgt_offset = f.variables['hgt'].add_offset
print(hgt.shape)

f.close()

"""
Example of the evolution of an air element
"""
plt.plot(time, hgt_offset + hgt[:, 1, 1, 1]*hgt_scale, c='r')
plt.show()


dt_time = [dt.date(1800, 1, 1) + dt.timedelta(hours=t)
           for t in time]
np.min(dt_time)
np.max(dt_time)

"""
Spatial distribution of the geopotential altitude at level 500hPa, for the first day
"""
```

```
plt.contour(lons, lats, hgt[0,5,:,:])
plt.show()

hgt2 = hgt[:,5,:,:].reshape(len(time),len(lats)*len(lons))

# Find with PCA the 4 principal components
n_components=4
Y = hgt2.transpose()
pca = PCA(n_components=n_components)
pca.fit(Y)
print(pca.explained_variance_ratio_)
out = pca.singular_values_
Element_pca = pca.fit_transform(Y)
Element_pca = Element_pca.transpose(1,0).reshape(n_components,len(lats),len(lons))

# Plot 4 principal components spacially
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_pca[i-1,:,:])
plt.show()

#%%
"""
Exercise 2: Analogous finding
"""

f = nc.netcdf_file(workpath + "/hgt.2019.nc", 'r')
hgt_19 = f.variables['hgt'][:].copy()
f.close()

f = nc.netcdf_file(workpath + "/hgt.2020.nc", 'r')
hgt_20 = f.variables['hgt'][:].copy()
time_bnds_20 = f.variables['time_bnds'][:].copy()
f.close()

# Indexes of x in (-20,20) and y in (30,50)
lats_index = np.arange(16,25)
lons_index = np.arange(-8,9)

# Get day 2020/01/20 in desired subset
hours = (dt.date(2020,1,20) - dt.date(1800,1,1)).days*24
idx = np.where(time_bnds_20[:,0] == hours)
```

```
a0 = hgt_20[idx[0][0],:,:,:]
aux = a0[:,lats_index,:]
a0_sub = aux[:,:,lons_index]

# Get 2019 days in desired subset
aux = hgt_19[:,:,lats_index,:]
an = aux[:,:,:,lons_index]

# Find the 4 days most analogous to 2020/01/20 in 2019
days = analogues(a0_sub,an,4)
dt_time = [dt.date(1800, 1, 1) + dt.timedelta(hours=t)
           for t in time_bnds[days][:,0]]
print("The 4 days more analogous to 2020-01-20 are", [str(date) for date in dt_time])

f = nc.netcdf_file(workpath + "/air.2020.nc", 'r')
air_20 = f.variables['air'][:].copy()
air_scale = f.variables['air'].scale_factor
f.close()

f = nc.netcdf_file(workpath + "/air.2019.nc", 'r')
air_19 = f.variables['air'][:].copy()
f.close()

# Get day 2020/01/20 in desired subset with p = 1000hPa
ta0 = air_20[idx[0][0],:,:,:]
aux = ta0[:,lats_index,:]
aux2 = aux[:,:,lons_index]
ta0_sub = aux2[0,:,:]

# Get 2019 analogous days in desired subset with p = 1000hPa
tdays = air_19[days,:,:,:]
aux = tdays[:,:,lats_index,:]
aux2 = aux[:,:,:,lons_index]
tdays_sub = aux2[:,0,:,:]

# Compute the mean temperature of the analogous days in each point
av = np.mean(tdays_sub,axis = 0)

# Compute the Mean Absolute Error with 2020/01/20
diff = np.abs(ta0_sub - av)
mae = np.sum(diff)/(9*17)*air_scale
print('Mean absolute error = ',mae, 'K')
```