PRÁCTICA 4: PCA y Analogía

Geometría Computacional

Belén Sánchez Centeno
belsan05@ucm.es

29 de marzo de 2022

Date Performed:	9 y 16 de marzo de 2022
Code Partner:	Martín Fernández de Diego

1. Introducción

Considerar que la atmósfera es un sistema de 6 variables de estado $(t, x, y, p, Z, T) \in \mathbb{R}^6$ pero sólo dos de ellas (T y Z) son dinámicas respecto a la variable t que llamaremos tiempo. Para poder trabajar con *elementos numerables*, discretizaremos las x en 144 valores posibles, y en 73, p en 17 y el parámetro tiempo t en lapsos de 1 día. Es decir, podemos considerar que tenemos 144 × 73 × 17 elementos de aire que no modifican sus coordenadas (x_i, y_j, p_k), pero sí su temperatura T y altura geopotencial Z en función de la coordenada temporal t. Desde un punto de vista geométrico, podemos redefinir el sistema como un conjunto numerable de elementos (días) con 144 × 73 × 17 × 2 variables de estado diferentes (pares de temperatura-altura) representados como $\{(T_{i,j,k}, Z_{i,j,k})\}_{i=1,j=1,k=1}^{i=144,j=73,k=17} \subset \mathbb{R}^3 \times \mathbb{R}^3$.

Primer Objetivo

Considerar el sistema $S = \{a_d, X_d\}_{d=1}^{365}$ formado por los elementos 'días' a_d del año 2021 y las variables de estado $X_d := \{Z_{i,j,k}\}_{i=1,j=1,k=1}^{i=144,j=73,k=17}$ y estimar las 4 componentes principales fijando $p_k = 500hPa$. Representarlas espacialmente en (x, y). ¿Qué porcentaje de varianza se explica?

Segundo Objetivo

Considerar un subsistema $\lambda \subset S$ delimitado* por $x \in (-20°, 20°)$, $y \in (30°, 50°)$. Considerando sólo $\mathbb{Z}$, encontrar los 4 días de 2019 más análogos al elemento a_0 := \2022/01/11" y calcular 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, considerar la distancia euclídea de elementos de λ 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.

2. Material utilizado

Se resuelve el problema con un script de Python, en el que se usan funciones de implementación propia, de las plantillas proporcionadas y de las siguientes bibliotecas: `matplotlib.pyplot`, `numpy`, `math`, `netCDF4`, `sklearn.decomposition`, `datetime` y `copy`. Las de implementación propia quedan detalladas a continuación:

- `lons_normal_ref`, que, dado un dataset con una matriz $(latitud, longitud)$ con valores entre $[0, 2\pi]$ en el dominio de longitud, devuelve una copia de esa matriz con los valores desplazados longitudinalmente, quedando en el domino $[-\pi, \pi]$. Se utiliza para cambiar la perspectiva de los mapas y que España quede en el medio.
- `dist_euclidea`, que, dados datos de dos días, devuelve la distancia euclídea que los separa en función de los pesos indicados en el enunciado.

Se estudian datos reales de un re-análisis climatológico conocido como NCEP, que vienen dados en dos archivos de temperatura T (air*) y dos de altura geopotencial Z (hgt*), de los años 2021 y 2022. Cada archivo se ordena en 144 longitudes (x), 73 latitudes (y), 17 niveles de presión (p). Los 4 archivos vienen en formato NETCDF4, por lo que en primer lugar se leen con `Dataset` (de su librería) como se hace en la plantilla proporcionada y, posteriormente, se normalizan con `lons_normal_ref`.

Para el **primer apartado** se toman los datos de altura geopotencial de 2021 con presión igual a $500hPa$ como una matriz X. Se toma además Y, la transpuesta de X, para quedarnos con la que más varianza explicada nos permita obtener al estimar las componentes principales. Para realizar dicha estimación se utiliza el algoritmo `PCA`, de `sklearn.decomposition`, indicando que se quieren las 4 componentes principales. Finalmente, se dan los ratios de varianza explicada y se muestran de manera gráfica las componentes, en este caso con la matriz Y que da más varianza explicada.

Para el **segundo apartado** se restringen los datos como se pide con `logical_and`, de `numpy`, y se almacenan los días de 2021 y 2022 en dos arrays de objetos `datetime`. Para obtener los días más análogos a a_0 se calculan las distancias euclídeas entre sus datos en el subsistema geopotencial de 2022 y los de todos los días del subsistema geopotencial de 2021, se ordenan, se toman las 4 menores y se dan los días a los que corresponden. Haciendo la media de las temperaturas de esos días se puede obtener una predicción de la temperatura de a_0, y devolver el error absoluto medio.

3. Resultados

3.1. Primer objetivo

Ratio de varianza explicada de las 4 componentes principales para X y para Y (X^t), respectivamente:

[**0.47248763 0.06072691 0.0359264 0.02815223**]
[**0.8877314 0.05177598 0.00543983 0.00357636**]

Es decir que, con X, se explica un 59,7 % de varianza y, con Y, un 94,9 %.

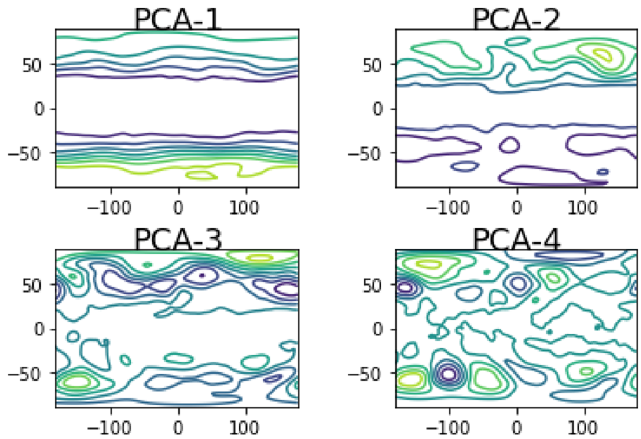

Figura 1: Componentes principales

3.2. Segundo objetivo

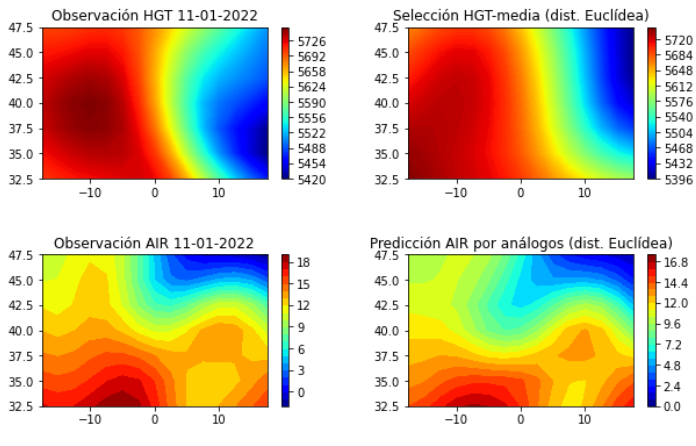

Figura 2: Resultado obtenido:
Los 4 días de 2021 localmente más análogos al 2022-01-11 son:
[**'2021-03-23', '2021-01-16', '2021-01-12', '2021-03-16'**]
El error absoluto medio local de la temperatura prevista para el 2022-01-11 es:
1.4193446568080357

4. Conclusión

En el primer apartado se ha observado que la primera componente principal es con la que más varianza se explica, con mucha diferencia respecto de las siguientes. En el segundo apartado han salido días invernales o del principio de la primavera, por lo que sus temperaturas son similares a las de a_0, como se puede comprobar en las gráficas inferiores de la Figura 2.

5. Anexo: Código utilizado

```
"""
PRÁCTICA 4: PCA Y ANALOGÍA
Belén Sánchez Centeno
Martín Fernández de Diego
"""

"""
Referencias:

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

    Altura geopotencial en niveles de presión
    https://psl.noaa.gov/cgi-bin/db_search/DBListFiles.pl?did=59&tid
        =97457&vid=1498

    Temperatura en niveles de presión:
    https://psl.noaa.gov/cgi-bin/db_search/DBListFiles.pl?did=59&tid
        =97457&vid=4237

    Temperatura en niveles de superficie:
    https://psl.noaa.gov/cgi-bin/db_search/DBListFiles.pl?did=59&tid
        =97457&vid=1497
"""

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

# FORMATO
class Formato:
    BOLD = "\033[1m"
    RESET = "\033[0m"

f = Dataset("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
#air_scale = f.variables['air'].scale_factor
#air_offset = f.variables['air'].add_offset
f.close()

f = Dataset("air.2022.nc", "r", format="NETCDF4")
time = f.variables['time'][:].copy()
time_bnds = f.variables['time_bnds'][:].copy()
```

```python
time_units = f.variables['time'].units
air22 = f.variables['air'][:].copy()
f.close()

f = Dataset("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
#hgt_scale = f.variables['hgt'].scale_factor
#hgt_offset = f.variables['hgt'].add_offset
f.close()

#f = nc.netcdf_file(workpath + "/" + files[0], 'r')
f = Dataset("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()



"""
Dada una matriz (latitud,longitud) con valores en el dominio de
    longitud [0,2pi]
devuelve la lista con valores en el dominio de longitud [-pi,pi]
"""
def lons_normal_ref(dataset):
    return_dataset = copy(dataset) # Necesario para que no copie por
        referencia
    for i in range(72):
        return_dataset[:,:,:,i] = dataset[:,:,:,i+72]
    for i in range(72):
        return_dataset[:,:,:,i+72] = dataset[:,:,:,i]
    return return_dataset

# Normalizamos los mapas para que España esté en medio
hgt21a = lons_normal_ref(hgt21)
hgt22a = lons_normal_ref(hgt22)
air21a = lons_normal_ref(air21)
air22a = lons_normal_ref(air22)



# APARTADO i)
print("\n" + Formato.BOLD + "Apartado i)" + Formato.RESET)

hgt21b = hgt21a[:,level==500.,:,:].reshape(len(time21),len(lats)*len(
    lons))

n_components = 4

X = hgt21b
Y = hgt21b.transpose()
pca = PCA(n_components=n_components)
```

```
# Interpretar el siguiente resultado
pca.fit(X)
print(pca.explained_variance_ratio_)
out = pca.singular_values_
"""
Salida: [0.4724878  0.06072688 0.03592642 0.02815213]
La primera componente principal explica el 47% de la varianza del
    dataset.
En total, las cuatro primeras componentes principales explican entorno
    al 60% del dataset al entrenar X.
"""

# Interpretar el siguiente resultado
pca.fit(Y)
print(pca.explained_variance_ratio_)
out = pca.singular_values_
"""
Salida: [0.8877314  0.05177603 0.00543984 0.00357636]
La primera componente principal explica más del 88% de la varianza del
    dataset.
En total, las cuatro primeras componentes principales explican más del
    94% del dataset al entrenar Y = tr(X).
"""

"""
Dado que es capaz de explicar más varianza con menos componentes,
se entrena el análisis de componentes principales PCA con Y = tr(X)
"""
Element_pca0 = pca.fit_transform(Y)
Element_pca0 = Element_pca0.transpose(1,0).reshape(n_components,len(
    lats),len(lons))

# 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-180, lats, Element_pca0[i-1,:,:])
plt.show()



# APARTADO ii)
print("\n" + Formato.BOLD + "Apartado ii)" + Formato.RESET)

def dist_euclidea(dia0, dia):
    dist = 0
    for lat in range(len(dia0[0])):
        for lon in range(len(dia0[0][0])):
            dia0_500 = 0.5*dia0[level==500.,lat,lon]
            dia0_1000 = 0.5*dia0[level==1000.,lat,lon]
            dia_500 = 0.5*dia[level==500.,lat,lon]
            dia_1000 = 0.5*dia[level==1000.,lat,lon]
            dist += (dia0_500 - dia_500)**2 + (dia0_1000 - dia_1000)**2
    dist = math.sqrt(dist)
    return dist
```

```
"""
Sistemas base
"""
# Restringimos el espacio de búsqueda a las longitudes (-20,20) y
    latitudes (30,50)
hgt21c = hgt21a[:, :, :, np.logical_and(160 < lons, lons < 200)]
hgt21c = hgt21c[:, :, np.logical_and(30 < lats, lats < 50),:]

hgt22c = hgt22a[:, :, :, np.logical_and(160 < lons, lons < 200)]
hgt22c = hgt22c[:, :, np.logical_and(30 < lats, lats < 50),:]

air21c = air21a[:, :, :, np.logical_and(160 < lons, lons < 200)]
air21c = air21c[:, :, np.logical_and(30 < lats, lats < 50),:]

air22c = air22a[:, :, :, np.logical_and(160 < lons, lons < 200)]
air22c = air22c[:, :, np.logical_and(30 < lats, lats < 50),:]

lons = lons[np.logical_and(0 < lons, lons < 40)]-20
lats = lats[np.logical_and(30 < lats, lats < 50)]

# Almacenamos los días de 2021 y 2022 en las siguientes estructuras
dt_time21 = [dt.date(1800, 1, 1) + dt.timedelta(hours=t) for t in
    time21]
dt_time22 = [dt.date(1800, 1, 1) + dt.timedelta(hours=t) for t in
    time22]

"""
Obtención de los días más análogos
"""
# Tomamos el índice correspondiente al día 11 de enero de 2022
dia0 = dt.date(2022, 1, 11)
idx0 = dt_time22.index(dia0)
# Obtenemos los datos del día 11 de enero de 2022
a0 = hgt22c[idx0, :, :, :]

# Calculamos la distancia euclidia entre el día 11 de enero de 2022 y
    los días de 2021
distancia_idx = [[dist_euclidea(a0,hgt21c[i, :, :, :]), i] for i in
    range(hgt21c.shape[0])]

# Mostramos los 4 días más análogos considerando solo Z
num_dias = 4
distancia_idx.sort()
idx_analogos = [idx for _, idx in distancia_idx[0:num_dias]]

# Buscamos el día correspondiente al índice almacenados en el vector de
     pares distancia-índice
dias_analogos = [dt_time21[idx_analogos[i]].isoformat() for i in range(
    num_dias)]
print("Los", num_dias, "días de 2021 localmente más análogos al", dia0,
     "son:")
print(dias_analogos)

"""
Error absoluto medio de T según la media de los análogos
"""
# Media de los análogos
media_analogos = np.mean(air21c[idx_analogos,level==1000.,:,:],axis=0)
```

```
# Error absoluto medio
error_abs_medio = np.mean(abs(media_analogos - air22c[idx0,level
    ==1000.,:,:]))
print("El error absoluto medio local de la temperatura prevista para el
    ", dia0, "es:")
print(error_abs_medio)

"""
Opcional: Comprobación gráfica de los resultados
"""
fig = plt.figure(figsize=(10,6))
fig.subplots_adjust(hspace=0.5, wspace=0.3)

ax = fig.add_subplot(2, 2, 1)
ax.set_title('Observación HGT 11-01-2022')
p = plt.contourf(lons, lats, hgt22c[idx0,5,:,:], 200, cmap='jet')
fig.colorbar(p)

ax = fig.add_subplot(2, 2, 2)
ax.set_title('Selección HGT media (dist. Euclídea)')
p = plt.contourf(lons, lats, np.mean(hgt21c[idx_analogos,5,:,:],axis=0)
    , 200, cmap='jet')
fig.colorbar(p)

ax = fig.add_subplot(2, 2, 3)
ax.set_title('Observación AIR 11-01-2022')
p = plt.contourf(lons, lats, air22c[idx0,0,:,:]-273, 20, cmap='jet')
fig.colorbar(p)

ax = fig.add_subplot(2, 2, 4)
ax.set_title('Predicción AIR por análogos (dist. Euclídea)')
p = plt.contourf(lons, lats, media_analogos-273, 20, cmap='jet')
fig.colorbar(p)

plt.show()
```