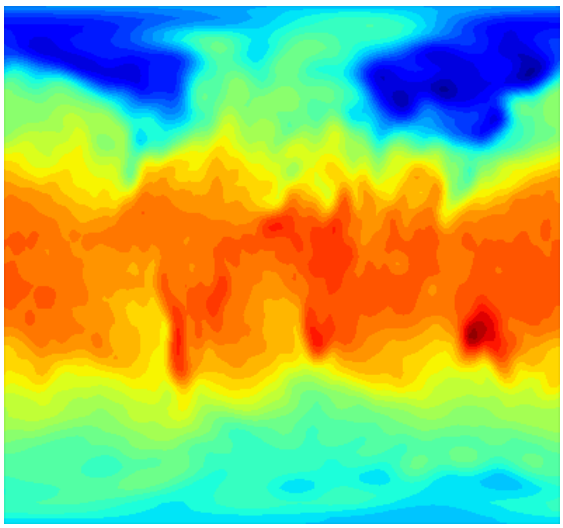
Práctica 4 - GCOMP

Celia Rubio Madrigal

29 de marzo de 2022

Índice



1. Introducción

En esta práctica trabajaremos con un sistema cuyos datos están tomado de la atmósfera terrestre.

Tendremos fijada una malla discreta de longitudes (variable x), latitudes (y) y niveles de presión (p), que no se verá modificada a lo largo del tiempo (t). En cada punto de la malla se habrá medido la temperatura (T) y la altura geopotencial (Z) cada día de año 2021 y parte del 2022. Se ha de observar que, por convenio, la malla no está definida a partir de alturas fijas sino de valores de presión fijos.

Con estos datos, primero reduciremos el sistema en su eje temporal mediante un análisis de componentes principales, y después trataremos de predecir la temperatura de un día a partir de sus días análogos en altura geopotencial.

2. Material usado y metodología

Hemos tomado los datos del reanálisis climatológico NCEP, cuyo formato es NetCDF4. Nos proporciona una malla fija de dimensión (144,73,17) por cada día, y los valores de T y Z por cada uno de sus puntos.

Además, hemos cambiado el sistema de referencia de las longitudes para que el 0 ($\equiv 360° \equiv 2\pi$ rad) —que se corresponde geográficamente con España— se encuentre en medio del intervalo y no al comienzo y final. Es decir, transformamos el intervalo $[0, 360) \mapsto [-180, 180)$ (en grados, como se encuentran los datos) o $[0, 2\pi) \mapsto [-\pi, \pi)$ (en radianes).

2.1. Apartado *i*)

En el primer apartado, hemos fijado la variable Z en el valor $500hPa$, y la variable t en los 365 días de 2021. Así, tenemos el sistema $S_1 = \{\ a_d,\ \{Z_{x,y,p}\}_{x=1,\ y=1,\ p=1}^{x=144,y=73,p=17}\ \}_{d=1}^{d=365}$.

Sobre el sistema hemos aplicado el algoritmo PCA. Consiste en encontrar la matriz de covarianzas entre sus elementos, que se descompone espectralmente para quedarnos solamente con los $N = 4$ vectores propios con mayor autovalor asociado. Al recomponer la matriz original, nos habremos quedado con una reducción del espacio de elementos.

Lo hemos aplicado a través de la clase `sklearn.decomposition.PCA`. La plantilla facilitada por el profesor proporciona dos posibles opciones para calcularlas, y debemos escoger la opción correcta.

2.2. Apartado *ii*)

En el segundo apartado, hemos fijado el subsistema S_2 donde la malla de puntos está delimitada en sus latitudes y longitudes: $x \in (20°, 20°)$, $y \in (30°, 50°)$.

Para el elemento del sistema a_d con $d =$ "11/01/2022", hemos calculado los 4 elementos más análogos a él respecto a su variable Z. Hallamos la distancia mínima entre él y cada elemento del año 2021 usando la métrica euclídea ponderada. Los pesos distintos de 1 corresponden a la variable presión: si $p_k = 500hPa$ o $p_k = 1000hPa$ entonces $w_k = 0.5$, y, para el resto, $w_k = 0$.

Hemos creado un nuevo elemento a'_d cuyos estados son la media de los estados de esos 4 elementos, tanto para su variable Z como para su variable T, y los hemos comparado con los de a_d. Habremos predicho la temperatura del elemento original solo a partir de su altura; así podremos calcular el error absoluto medio cometido de, por ejemplo, los puntos de la malla con $p = 1000hPa$.

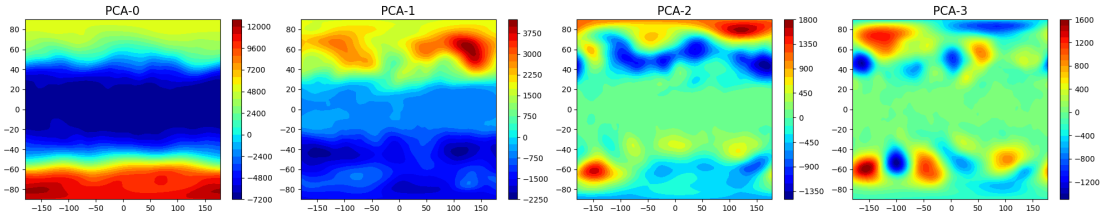
3. Resultados y conclusiones

3.1. Apartado *i*)

La opción correcta para calcular los 4 componentes principales del sistema es la primera; es decir, la opción "0", que primero transpone la matriz que queremos reducir. Esto es porque el algoritmo PCA se suele aplicar para la reducción de variables de estado, pero nosotros lo usamos para la reducción del número de elementos. Podemos comprobar que la segunda opción (la "1", la errónea) transforma nuestra matriz de tamaño (365, 10512) en una de (365, 4).

Los ratios de varianza explicada obtenidos son [0.888, 0.052, 0.005, 0.004] ±0.001, cuya suma es 0.949 ±0.001. Así, el porcentaje de varianza explicada es 94.5 %, una cifra considerable. Quiere decir que hemos conseguido reducir un sistema con 365 elementos en uno con 4, de forma que somos capaces de explicar el 94.5 % de la varianza de los estados originales mediante los reducidos.

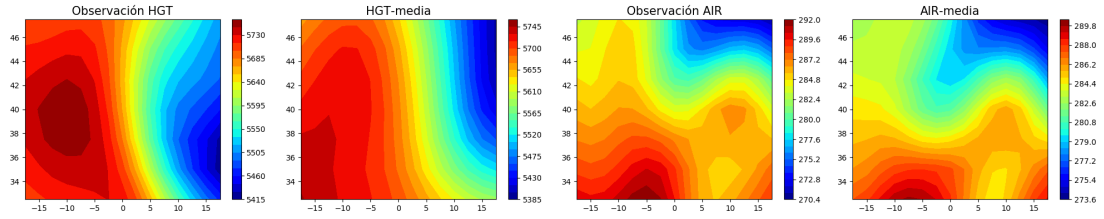
A continuación se muestran las gráficas de esta nueva descomposición de los elementos.



3.2. Apartado *ii*)

Para los valores de altura geopotencial Z de la malla del subsistema S_2 en el día d, y usando la distancia definida anteriormente, se han obtenido las 4 fechas más parecidas a ésta en 2021: 23 de marzo, 16 de enero, 12 de enero y 16 de marzo.

Con los valores de los estados de estos 4 elementos hemos calculado su media aritmética. A continuación se muestran, respectivamente, las gráficas de los valores reales de Z ($p = 500hPa$), la media de los valores de Z más análogos, los valores reales de temperatura T ($p = 1000hPa$) y sus valores predichos a partir del elemento calculado.



Se puede observar que los valores reales de temperatura y los predichos son muy cercanos, por lo que el algoritmo produce resultados satisfactorios. De hecho, para $p = 1000hPa$ se obtiene un error absoluto medio de 1.419K ±0.001. Este algoritmo se podría utilizar para predecir temperaturas que no supiéramos de antemano, habiéndolo ya probado en algunas que sí.

En conclusión, hemos aplicado algunos algoritmos para entender los datos de un sistema altamente complejo y de persistente aplicación en el mundo real, que es la atmósfera. Hemos conseguido reducir el número de elementos del sistema, y hemos conseguido predecir los estados de un elemento a partir de otros.

4. Código

```
# -*- coding: utf-8 -*-
"""
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 math
import datetime as dt # Python standard library datetime module
import numpy as np
import matplotlib.pyplot as plt
from netCDF4 import Dataset
from sklearn.decomposition import PCA

LEV = 30
CMAP = "jet"


f = Dataset("air.2021.nc", "r", format="NETCDF4")
level = f.variables["level"][:].copy()
lats = f.variables["lat"][:].copy()
lons = f.variables["lon"][:].copy()
air21 = f.variables["air"][:].copy()
f.close()
f = Dataset("hgt.2021.nc", "r", format="NETCDF4")
time21 = f.variables["time"][:].copy()
hgt21 = f.variables["hgt"][:].copy()
f.close()


f = Dataset("air.2022.nc", "r", format="NETCDF4")
air22 = f.variables["air"][:].copy()
f.close()
```

```
f = Dataset("hgt.2022.nc", "r", format="NETCDF4")
time22 = f.variables["time"][:].copy()
hgt22 = f.variables["hgt"][:].copy()
f.close()

# Cambio de coordenadas

air21[:,:,:,lons >= 180], air21[:,:,:,lons < 180] = air21[:,:,:,lons < 180],
    air21[:,:,:,lons >= 180]
air22[:,:,:,lons >= 180], air22[:,:,:,lons < 180] = air22[:,:,:,lons < 180],
    air22[:,:,:,lons >= 180]
hgt21[:,:,:,lons >= 180], hgt21[:,:,:,lons < 180] = hgt21[:,:,:,lons < 180],
    hgt21[:,:,:,lons >= 180]
hgt22[:,:,:,lons >= 180], hgt22[:,:,:,lons < 180] = hgt22[:,:,:,lons < 180],
    hgt22[:,:,:,lons >= 180]

lons = np.roll(lons, len(lons[lons < 180]))
lons[lons >= 180] -= 360


""" Apartado i) """


"""
Distribución espacial de la temperatura en el nivel de 500hPa, para el primer d
    ía
"""
cs = plt.contourf(lons, lats, air21[0, 0, :, :], cmap=CMAP, levels=LEV)
plt.colorbar(cs)
plt.savefig("1")

hgt21b = hgt21[:, level == 500.0, :, :].reshape(len(time21), len(lats) * len(
    lons))
n_components = 4


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.explained_variance_ratio_)

fig, axs = plt.subplots(nrows=1, ncols=4, figsize=(20, 4))
fig.tight_layout()
plt.subplots_adjust(top=0.90)
for i in range(4):
    ax = axs[i]
```

```
    ax.set_title("PCA-" + str(i), fontsize=15, ha="center")
    cs = ax.contourf(lons, lats, Element_pca0[i, :, :], cmap=CMAP, levels=LEV)
    fig.colorbar(cs, ax=ax)

plt.savefig("2")


""" Apartado ii) """


dt_time21 = np.array([dt.date(1800, 1, 1) + dt.timedelta(hours=t) for t in
    time21])
dt_time22 = np.array([dt.date(1800, 1, 1) + dt.timedelta(hours=t) for t in
    time22])

lats_logic = np.logical_and(lats > 30, lats < 50)
lons_logic = np.logical_and(lons > -20, lons < 20)

hgt_sub = hgt21[:, :, lats_logic, :][:, :, :, lons_logic]
air_sub = air21[:, :, lats_logic, :][:, :, :, lons_logic]
lats_sub = lats[lats_logic]
lons_sub = lons[lons_logic]

dia_0 = dt.date(2022, 1, 11)
hgt_0 = hgt22[dt_time22 == dia_0, :, :, :][0][:, lats_logic, :][:, :,
    lons_logic]
air_0 = air22[dt_time22 == dia_0, :, :, :][0][:, lats_logic, :][:, :,
    lons_logic]


def dist_analogia(u, v):
    w = u - v
    a, b, c = w.shape
    total = 0
    for i, p in enumerate(level):
        wk = 0.5 if p == 500.0 or p == 1000.0 else 0
        if wk == 0:
            continue
        for j in range(b):
            for k in range(c):
                total += (w[i, j, k] ** 2) * wk
    return math.sqrt(total)


distancias = [
    (dist_analogia(hgt_0, hgt_sub[dt_time21 == d, :, :, :][0]), d) for d in
        dt_time21
]
distancias.sort()
```

```
top4 = [d for _, d in distancias[:4]]
print(top4)
hgt_1 = sum([hgt_sub[dt_time21==d,:,:,:][0] for d in top4 ])/4
air_1 = sum([air_sub[dt_time21==d,:,:,:][0] for d in top4 ])/4


fig, axs = plt.subplots(nrows=1, ncols=4, figsize=(20, 4))
fig.tight_layout()
plt.subplots_adjust(top=0.90)

ax = axs[0]
ax.set_title("Observación HGT", fontsize=15, ha="center")
plotted = hgt_0[level==500.,:,:]
cs = ax.contourf(lons_sub, lats_sub, plotted[0], cmap=CMAP, levels=LEV)
fig.colorbar(cs, ax=ax)

ax = axs[1]
ax.set_title("HGT-media", fontsize=15, ha="center")
plotted = hgt_1[level==500.,:,:]
cs = ax.contourf(lons_sub, lats_sub, plotted[0], cmap=CMAP, levels=LEV)
fig.colorbar(cs, ax=ax)

ax = axs[2]
ax.set_title("Observación AIR", fontsize=15, ha="center")
plotted = air_0[level==1000.,:,:]
cs = ax.contourf(lons_sub, lats_sub, plotted[0], cmap=CMAP, levels=LEV)
fig.colorbar(cs, ax=ax)

ax = axs[3]
ax.set_title("AIR-media", fontsize=15, ha="center")
plotted = air_1[level==1000.,:,:]
cs = plt.contourf(lons_sub, lats_sub, plotted[0], cmap=CMAP, levels=LEV)
fig.colorbar(cs, ax=ax)

plt.savefig("3")

print(np.array([abs(a) for a in air_1[level==1000.,:,:] - air_0[level
    ==1000.,:,:]]).mean())
```