Codigo Metodo de La Distancia Inversa Ponderada (IDW)

Miguel Angel Cañon Ramos
Tutor: Efraín Dominguez

Facultad de Estudios Ambientales y Rurales
Departamento de Ecología y Territorio
miguelca27@gmail.com

Contenido

Algoritmo para Python I

1. Crear el script 10_Metodo_IDW.py en Pycharm;
2. Import numpy as np;
3. Import pandas as pd;
4. from osgeo import gdal;
5. Import matplotlib.pyplot as plt;
6. Crear una variable con la ruta al archivo de Excel de lectura;
7. Crear funcion Tamaño de pixel
8. crear funcion Salida del Raster con informacion Interpolada
9. Crear funcion con el metodo de La Distancia Inversa Ponderada (IDW)
10. Crear Funcion con la Metodologia de validacion JackKnife

Código en Python

Para el desarrollo del Metodo de interpolacion se requiere la lectura de archivos de Excel por lo que importamos el modulo pandas, para soporte de graficas importamos el modulo pyplot desde matplotlib y el modulo osgeo desde la lireria gdal

Requerimientos de Módulos

```
1 import pandas as pd
2 from osgeo import gdal
3 import osr
4 from pylab import *
5 import matplotlib.pyplot as plt
```

Código en Python I

El archivo Excel de entrada contiene los Totales anuales de precipitacion y las respectivas cordenadas correspodientes a cada uno de los puntos. Es necesario separar la informacion del Data frame los valores correspondientes a x,y,z y Generar la Ruta de Salida para el Archivo .tif

Lectura y Separacion de datos

```
1 fn='Estaciones_Betania.xlsx'
2 df = pd.read_excel(fn,'Datos',index_col=0)
3 outRuta ='Interpolacion.tif'
4
5 x=df['Este']
6 y=df['Norte']
7 z=df['z']
```

Código en Python II

Como datos de entrada es necesario dar los valores del numero de Celdas en x - y, y agregar el Dominio de Nuestra Interpolacion, como un primer ejercicio se tendran en cuenta las coordenadas de nuestros Datos, luego se agregara Un Dominio Diferente en nuestra interpolacion.

Datos de entrada

```
1 nx=100 # numero de Celdas en X
2 ny=100 # numero de Celdas en Y
3
4 xmax=np.max(x)#873346.050446
5 xmin=np.min(x)#708428.281178
6
7 ymax=np.max(y)#838345.286362
8 ymin=np.min(y)#652604.566429
```

Código en Python III

se presenta de forma grafica la ubicacion de los puntos conocidos.

Presentacion grafica de los Puntos Conocidos

```
1 fig , ax = subplots()
2 ax.scatter(df.Este, df.Norte, c=df.z, cmap='gray')
3 ax.set_aspect(1)
4 xlabel('Este [m]')
5 ylabel('Norte [m]')
6 title('Precipitacion [mm]')
7 plt.show()
```

Código en Python IV

se construira una funcion para el tamaño del pixel Y la salida raster

Construccion de Funciones Salidas graficas

```
1
2
3 def  pixel(xmax,xmin,ymax,ymin,nx,ny):
4
5    pixelWidth=abs(xmin-xmax)/nx
6    pixelHeight=abs(ymin-ymax)/ny
7
8    return  pixelWidth,pixelHeight
```

Código en Python V

Generamos una funcion para la salida raster

```
1  def array2raster(newRasterfn, array, pixelWidth,
       pixelHeight, cols, rows, xorg, yorg):
2    originX = xorg
3    originY = yorg
4    driver = gdal.GetDriverByName('GTiff')
5    outRaster = driver.Create(newRasterfn, cols, rows, 1,
       gdal.GDT_Float64)
6    outRaster.SetGeoTransform((originX, pixelWidth, 0,
       originY, 0, pixelHeight))
7    outband = outRaster.GetRasterBand(1)
8    outband.WriteArray(array)
9    outRasterSRS = osr.SpatialReference()
10   outRasterSRS.ImportFromEPSG(3116)
11   outRaster.SetProjection(outRasterSRS.ExportToWkt())
12   outband.FlushCache()
```

Código en Python VI

se construira la funcion para el metodo de interpolacion de Distancia Ponderada (IDW)

FUNCION IDW PARTE I

```
1  def IDW (x,y,z,nx,ny,xmax,xmin,ymax,ymin,power=2):
2
3    def distancia(x, y, xi, yi):
4    #genera la distancia euclidea entre dos puntos
5      obs = np.vstack((x, y)).T
6      interpolar = np.vstack((xi, yi)).T
7      d0 = np.subtract.outer(obs[:,0], interpolar[:,0])
8      d1 = np.subtract.outer(obs[:,1], interpolar[:,1])
9      Md=np.hypot(d0, d1)
10
11     return Md
```

Código en Python VII

FUNCION IDW PARTE II

```
1
2  def malla(xmax,xmin,ymax,ymin,nx,ny):
3      #genera la malla de puntos desconocidos los cuales
       se van a interpolar
4
5      xi=np.linspace(xmin,xmax,nx)
6      yi=np.linspace(ymin,ymax,ny)
7
8      xi, yi=np.meshgrid(xi,yi)
9      xi, yi = xi.flatten(), yi.flatten()
10
11     return xi,yi
```

Código en Python VIII

FUNCION IDW PARTE III

```
# divisiones de la malla
   xi , yi=malla(xmax , xmin , ymax , ymin , nx , ny)
# Generamos las disntacias entre los puntos
   dist = distancia(x , y , xi , yi)
# W pesos
   W = 1.0 / dist**power
   W /= W.sum(axis=0)
# multiplica los pesos por los valores observados z
   zi = np.dot(W.T, z)
# se genera una matrix con el tama o de las celdas
   zi=zi.reshape(nx , ny)
   return xi , yi , zi
```

Código en Python IX

FUNCION JACKKNIFE PARTE I

```
1
2 def jackknife(df):
3 # Funciones metricas
4     def ME(obs,sim):
5         A=np.mean(obs-sim)
6         return A
7     def MARE(obs,sim):
8         A=np.mean(abs(obs-sim)/obs)
9         return A
```

Código en Python X

FUNCION JACKKNIFE PARTE II

```
1   def distancia(x, y, xi, yi):
2       #genera la distancia euclidea entre dos puntos
3     obs = np.vstack((x, y)).T
4     interpolar = np.vstack((xi, yi)).T
5     d0 = np.subtract.outer(obs[:,0], interpolar[:,0])
6     d1 = np.subtract.outer(obs[:,1], interpolar[:,1])
7     Md=np.hypot(d0, d1)
8     return Md
9       #Matriz de almacenamiento
10      errores=np.zeros([np.size(df,axis=0),2])
```

Código en Python XI

FUNCION JACKKNIFE PARTE III

```
#Ciclo que recorra todos los datos
   for i in range (np.size(df,axis=0)):
   #Separa datos de data frame original
    observados=df[i:i+1]
    id= observados.index
    data = df.drop(id,axis=0)
    # matriz N-1
    x=data['Este']
    y=data['Norte']
    z=data['z']
    # datos Extraidos
    xi=observados['Este']
    yi=observados['Norte']
    obs=observados['z']
```

Código en Python XII

FUNCION JACKKNIFE PARTE IV

```
1
2   # Metodo idw
3
4    dist=distancia(x, y, xi, yi)
5   W = 1.0 / dist**2
6   W /= W.sum(axis=0)
7    sim = np.dot(W.T, z)
8
9   # Metricas
10
11   errores[i,0:]=ME(obs,sim)
12   errores[i,1:]=MARE(obs,sim)
```

Código en Python XIII

FUNCION JACKKNIFE PARTE V

```
#Guardamos los resultados en un data frame y generamos un histograma

E=pd.DataFrame(errores)
E.columns=['Mean error','Mean Absolute relative error']
E.to_excel('Errores_jacknife.xlsx','Errores')
E.hist()
plt.savefig('Histograma Errores.png', fmt='png', dpi=300)
plt.show()
return E
```

Código en Python XIV

Para Finalizar se llaman las funciones realizadas y se ejecuta el Codigo

Ejecucion del Codigo

```
# se llaman las funciones generadas
# jackKnife
s=jackknife(df)
# Metodo IDW
xi,yi,zi=IDW(x,y,z,nx,ny,xmax,xmin,ymax,ymin)
#salida en archivo raster
pixelW,pixelH=pixel(xmax,xmin,ymax,ymin,nx,ny)
array2raster(newRasterfn=outRuta, array=zi ,pixelWidth=
    pixelW, pixelHeight=pixelH,cols=nx,rows=ny,xorg=
    xmin,yorg=ymin)
```