Homework 5 : Spatial autocorrelation

Alexandre Boucher

February 5, 2010

Due date: February 11th 2010

Compute the experimental variogram and covariance of a 2D raster

Experimental spatial structure

Write the function `ComputeSpatialStructure` to compute the experimental variogram or covariance in N-S and E-W directions from a 2D scipy array.

```
ComputeSpatialStructure(raster,dimPixel=[1,1],maxLag=None,type="variogram")
```

- `raster`: is a 2d scipy array of continuous attribute (it may have missing data : `scipy.nan`)
- `dimPixel`: size of one pixel (e.g `[10,10]` indicates that each pixel is of size 10×10 units).
- `maxLag`: The maximum *distance* in each direction (EW and NS) to compute the variogram/covariance values. For example `[200,300]` will compute the values up to 200 units in EW direction and 300 units in the NS direction. If set to `None` then `maxLag` is to be reset to half the dimension of the raster in each direction.
- `type`: A string that can take two values `"variogram"` or `"covariance"`

the function returns an object of type `SpatialStructure`.

SpatialStructure class

The resulting experimential variogram or covariance values are stored in:

```
class SpatialStructure :
  def __init__(self, ... )
  def __call__(self,dx,dy)
  def nPairs(self,dx,dy)
```

where `dx` and `dy` is the lag in EW-and NS in real coordinates, i.e. not the number of pixels. If no value exists for a pair (`dx`,`dy`) then it must return `scipy.nan`.

Example of usage

```
vario = ComputeSpatialStructure(myData,[30,30],[500,500],"variogram")
print  vario(0,90), vario.nPairs(0,90)
print  vario(120,0), vario.nPairs(120,0)
```

Data

Two data sets are available from coursework; `landsatBand1.dat` is complete while `landsatBand1Nan.dat` contains a significant number of missing values.

```
fid = open("landsatBand1.dat",'rb')
data = scipy.fromfile(file=fid, dtype=scipy.float32).reshape((500,500))
fid.close()

fid = open("landsatBand1Nan.dat",'rb')
datanan = scipy.fromfile(file=fid, dtype=scipy.float32).reshape((500,500))
fid.close()

pylab.figure()
pylab.subplot(121)
pylab.imshow(data,interpolation='nearest')
pylab.subplot(122)
pylab.imshow(datanan,interpolation='nearest')
pylab.show()
```