Branch: master

Find file | Copy path

pgser-ser347 / 2019 / aula-19-gdal-parte-3.ipynb


tkorting Add files via upload

a426b3e on 13 May 2019

1 contributor

Raw | Blame | History

359 lines (359 sloc) 13.3 KB

GDAL - Parte III

Passos necessários para manipular imagens:

- importar bibliotecas (GDAL, NumPy, etc)
- definir o caminho correto dos arquivos raster
- abrir um dataset para cada arquivo
- verificar compatibilidade dos metadados
- obter as bandas de cada raster
- converter as bandas para matrizes no formato NumPy
- manipular números digitais presentes nas matrizes
- (se necessário) salvar as informações em um novo arquivo raster

In [1]:
```
# importar bibliotecas
from osgeo import gdal
import numpy as np
from matplotlib import pyplot as plt

# importar constantes
from gdalconst import *

# informar o uso de exceções
gdal.UseExceptions()

# biblioteca de funções relacionadas ao sistema
# sys: System-specific parameters and functions
import sys
```

Exercício - Abrir duas imagens (crop-1-band-5.tif e crop-1-band-7.tif), realizar uma aritmética de bandas (usar NDVI) e aplicar um fatiamento (pixels > 0.5)

In [2]:
```
# abrir 2 imagens com 1 banda cada
filename_crop_1_band_5 = "./raster/crop-1-band-5.tif" # red
filename_crop_1_band_7 = "./raster/crop-1-band-7.tif" # nir

try:
    dataset_crop_1_band_5 = gdal.Open(filename_crop_1_band_5, GA_Rea
dOnly)
    dataset_crop_1_band_7 = gdal.Open(filename_crop_1_band_7, GA_Rea
dOnly)
except:
    print ("Erro na abertura de algum arquivo!")

# todas as imagens possuem uma banda cada
crop_1_band_5 = dataset_crop_1_band_5.GetRasterBand(1)
crop_1_band_7 = dataset_crop_1_band_7.GetRasterBand(1)

# para se realizar cálculos com as bandas, usamos a conversão para m
atriz numpy
numpy_crop_1_band_5 = crop_1_band_5.ReadAsArray()
numpy_crop_1_band_7 = crop_1_band_7.ReadAsArray()

# criar banda de índice de vegetação e aplicar fatiamento
```

```
numpy_crop_1_ndvi = (numpy_crop_1_band_7 - numpy_crop_1_band_5) / \
                    (numpy_crop_1_band_7 + numpy_crop_1_band_5)
numpy_output = numpy_crop_1_ndvi > 0.5
```

In [3]:
```
print(numpy_crop_1_ndvi)
print(numpy_output)
```

```
[[1.85628743e-01 3.76623377e-01 6.61909091e+02 ... 1.17852975e-01
  5.85585586e-02 8.41750842e-02]
 [4.74869565e+02 6.17283951e-03 1.68831169e-01 ... 1.62123386e-01
  1.55555556e-01 8.75000000e-02]
 [1.73524150e-01 1.51428571e-01 1.36294028e-01 ... 1.93875740e+02
  1.01214575e-01 1.68975069e-01]
 ...
 [1.75000000e-01 1.03448276e-01 1.37313433e-01 ... 3.16770186e-01
  3.17073171e-01 2.28187919e-01]
 [1.62790698e-01 1.66007905e-01 1.31313131e-01 ... 3.97849462e-01
  2.42236025e-01 3.63344051e-01]
 [1.49466192e-01 1.43369176e-01 1.81969950e-01 ... 1.69491525e-02
  1.63398693e-01 1.15942029e-01]]
[[False False  True ... False False False]
 [ True False False ... False False False]
 [False False False ...  True False False]
 ...
 [False False False ... False False False]
 [False False False ... False False False]
 [False False False ... False False False]]
```

Salvando um raster

Neste exemplo, vamos abrir duas bandas, realizar uma aritmética (NDVI) seguida de um fatiamento, e salvar o resultado num arquivo GeoTIFF. Para salvar um arquivo com imagens, é preciso criar um novo dataset, informar todos os metadados relacionados ao contexto geográfico (sistema de projeção, limite geográfico, etc.) além do número de bandas, número de linhas e colunas.

In [4]:
```
# obter metadados
linhas = dataset_crop_1_band_5.RasterYSize
colunas = dataset_crop_1_band_5.RasterXSize
bandas = 1

# salvar banda em arquivo GeoTIFF
# definir nome do arquivo
filename_output = "./raster/crop-1-ndvi-threshold.tif"
# definir driver
driver = gdal.GetDriverByName('GTiff')
# copiar tipo de dados da banda já existente
data_type = crop_1_band_5.DataType
# criar novo dataset
dataset_output = driver.Create(filename_output, colunas, linhas, ban
das, data_type)
# copiar informações espaciais da banda já existente
dataset_output.SetGeoTransform(dataset_crop_1_band_5.GetGeoTransform
())
# copiar informações de projeção
dataset_output.SetProjection(dataset_crop_1_band_5.GetProjectionRef
())
# escrever dados da matriz NumPy na banda
dataset_output.GetRasterBand(1).WriteArray(numpy_output)
```

```
# salvar valores
dataset_output.FlushCache()
# fechar dataset
dataset_output = None
```

Exercício - criar uma função `salvar_banda` para salvar um raster de *uma banda*, tendo como parâmetros uma matriz NumPy com os pixels, o nome do arquivo GeoTIFF, e um dataset de referência (para copiar os metadados)

In [ ]:
```
def salvar_banda(matriz_de_pixels, nome_do_arquivo, dataset_de_refer
encia):
    # obter metadados
    linhas = dataset_de_referencia.RasterYSize
    colunas = dataset_de_referencia.RasterXSize
    bandas = 1
    # definir driver
    driver = gdal.GetDriverByName('GTiff')
    # copiar tipo de dados da banda já existente
    data_type = dataset_de_referencia.GetRasterBand(1).DataType
    # criar novo dataset
    dataset_output = driver.Create(nome_do_arquivo, colunas, linhas,
bandas, data_type)
    # copiar informações espaciais da banda já existente
    dataset_output.SetGeoTransform(dataset_de_referencia.GetGeoTrans
form())
    # copiar informações de projeção
    dataset_output.SetProjection(dataset_de_referencia.GetProjection
Ref())
    # escrever dados da matriz NumPy na banda
    dataset_output.GetRasterBand(1).WriteArray(matriz_de_pixels)
    # salvar valores
    dataset_output.FlushCache()
    # fechar dataset
    dataset_output = None
```

Exercício - dados dois rasters de 1 banda cada (1 mapa temático, 1 mapa de referência), calcule a taxa de acerto da classificação do mapa temático e salve um arquivo GeoTIFF contendo um mapa de concordância entre as imagens

In [ ]:
```
# abrir as imagens
filename_reference = "./raster/referencia_area_urbana.tif"
filename_classification = "./raster/classificacao_area_urbana.tif"

try:
    dataset_reference = gdal.Open(filename_reference, GA_ReadOnly)
    dataset_classification = gdal.Open(filename_classification, GA_R
eadOnly)
except:
    print ("Erro na abertura de algum arquivo!")
```

```
# verificar compatibilidade de metadados
if (dataset_reference.GetProjectionRef() != dataset_classification.G
etProjectionRef()):
    print("Sistemas de referência diferentes")
elif (dataset_reference.GetGeoTransform() != dataset_classification.
GetGeoTransform()):
    print("Metadados espaciais diferentes")
else:
    # obter metadados
    linhas = dataset_reference.RasterYSize
    colunas = dataset_reference.RasterXSize

    # obter as bandas
    band_reference = dataset_reference.GetRasterBand(1)
    band_classification = dataset_classification.GetRasterBand(1)

    # gerar matrizes de pixels
    numpy_reference = band_reference.ReadAsArray()
    numpy_classification = band_classification.ReadAsArray()

    # gerar matriz de comparação
    numpy_comparison = (numpy_reference == numpy_classification)
    accuracy = 100 * numpy_comparison.sum() / (linhas * colunas)

    # plotar resultados
    plt.figure(figsize=(15, 4))

    plt.subplot(131)
    plt.imshow(numpy_reference)
    plt.title('Imagem de referência')

    plt.subplot(132)
    plt.imshow(numpy_classification)
    plt.title('Imagem classificada')

    plt.subplot(133)
    plt.imshow(numpy_comparison)
    plt.title('Imagem de comparação, ' + str(accuracy) + '% de acurá
cia')

# salvar imagem de concordância
nome_do_arquivo = "./raster/comparacao.tif"
salvar_banda(numpy_comparison, nome_do_arquivo, dataset_reference)

# fechar imagens
dataset_reference = None
dataset_classification = None
```

Exercício - Dadas 2 imagens (`raster_alvo` e `raster_entrada`), encontrar a localização da imagem `raster_alvo` dentro da imagem `raster_entrada` e plotar o resultado (`raster_entrada` com um x sobreposto na localização)

In [ ]:
```
# definir onde estao as imagens
raster_entrada = './raster/american-anticipation-audience-163368.jp
g'
raster_alvo = './raster/target_2.jpg'
```

```
# gerar datasets gdal
dataset_entrada = gdal.Open(raster_entrada, GA_ReadOnly)
dataset_alvo = gdal.Open(raster_alvo, GA_ReadOnly)

# obter as bandas
banda_entrada = dataset_entrada.GetRasterBand(1).ReadAsArray()
banda_alvo = dataset_alvo.GetRasterBand(1).ReadAsArray()

# obter metadados da imagem de entrada
Linhas_entrada = dataset_entrada.RasterYSize
Colunas_entrada = dataset_entrada.RasterXSize

# obter metadados da imagem de alvo
Linhas_alvo = dataset_alvo.RasterYSize
Colunas_alvo = dataset_alvo.RasterXSize

# encontrar linha/coluna do alvo na imagem de entrada
linha = 0
coluna = 0
# criar variável para armazenar a região de maior semelhança
maior_semelhanca = 0
for r in range(Linhas_entrada - Linhas_alvo):
    for c in range(Colunas_entrada - Colunas_alvo):
        # criar janela com recorte do mesmo tamanho da banda_alvo
        janela = banda_entrada[r:r+Linhas_alvo, c:c+Colunas_alvo]
        # comparar quantos pixels são iguais entre as duas janelas
        pixels_iguais = (banda_alvo == janela)
        somatorio = pixels_iguais.sum()
        if (somatorio > maior_semelhanca):
            linha = r
```