Geoinformatics | Course Remote Sensing (1)
Schmitt | Ulloa
Summer Semester 2020

Practice 4: Unsupervised classification

Overview

Objectives: Perform an unsupervised classification of your AOI, on a multiband image data using Python you need GDAL, Numpy and Sklearn.

Data: For this practice, use the following files:

- Raster file: GeoTIFF (S2A_L2A_T32UPU_rstck_id1.tif, S2A_L2A_T32UPU_rstck_id2.tif, S2A_L2A_T32UPU_rstck_id3.tif)

Tasks: load your rasterfile in Python and perform an unsupervised classification. Export the classification file and visualize it.

Procedure

To do our unsupervised classification we are going to use the Scikit-Learn library. It is a very commonly used python library for machine learning which includes all the classification algorithms presented in the lecture. For further information see this site. (https://en.wikipedia.org/wiki/Scikit-learn https://scikit-learn.org/stable/)

Load and visualize raster

In [1]:

```
from sklearn import cluster
```

If you wish to see the data you will also need Matplotlib

In [2]:

```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [4]:

```
from matplotlib.patches import Patch
from matplotlib.colors import ListedColormap
import matplotlib.colors as colors
```

These are the libraries we are going to use to read and write our imagery and prediction rasters

In [5]:

```
#from osgeo import gdal, gdal_array
import os
import rasterio as rio
import numpy as np
from rasterio.plot import show
from rasterio.plot import plotting_extent
```

1. Read your rasterstack using the function `open` from the package rio(rasterio). Assign your raster to an object called "s2_stack". Remeber to specify first the path where your data is located. Store your path in an object called "folder_src".

In [ ]:

```
# Specify the path

folder_src =

# Read in raster image

s2_stack =
```

To check that you loaded your raster, you can print some of the properties of that raster. I'll check here the number of bands.

In [7]:

```
# number of bands
s2_stack.count
```

Out[7]:

```
4
```

2. Apply the necessary functions (see Practice 1) to extract the name, dimensions and resolution of your raster.

In [ ]:

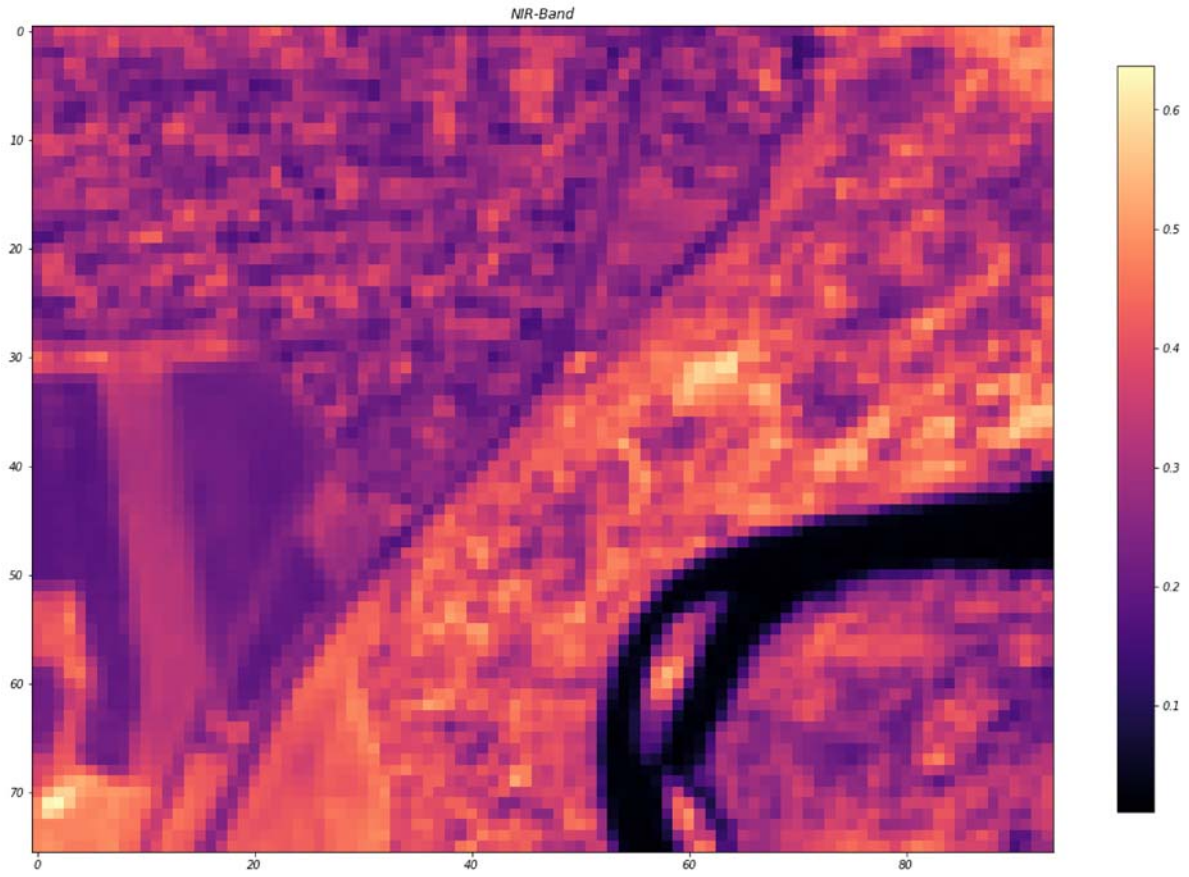
```

```

Plot a single band: NIR

Let's make a visualization of one band from your RasterStack, in this case the NIR band

In [12]:

```
#Plot Band NIR
fig,ax = plt.subplots(figsize=(20,20))
nirNpArray = s2_stack.read(4)
ax.set_title("NIR-Band")
# Be careful Numpy Array indexes start at zero!
#This command equals the one we used before to visualize band 4 using the show() function f
plot=ax.imshow(nirNpArray, cmap='magma')
plt.colorbar(plot,shrink=0.6)
plt.show()
```



3. Change the settings of the previous function with the following values:

- figsize=(5,5)
- plt.colorbar(plot,shrink=1)

In [ ]:

```

```

4. Explain the purpose of the parameters `figsize` and `shrink`

In [ ]:

```

```

5. `cmap` is a parameter that defines some defined color scales for maps in the matplotlib package. Using the

information on the next image, choose another color scale and change the color of your map.

In [ ]:

```
```

6. Using the previous line code, plot the bands BLUE and GREEN. Remember to update the title with the command `ax.set_title("your title")`

In [ ]:

```
```

7. Now we want to take a look at the structure of the RasterStack with a function you will use afterwards. Use the function `shape` to extract the information of the number of columns and rows of your rasterstack (hint: name_of_your_raster.shape)

In [ ]:

```
```

8. Please explain the meaning of the values from the result of the function `shape` . Which other functions so far in this practice have returned the same values?

In [ ]:

```
```

Now we will use the previous function and store the NIR band in an object called "band". This object will be turned into a vector for the classification purposes in the next step

In [15]:

```
# Create an object called "band" that stores the 4th value of your rasterstack = NIR band
band = s2_stack.read(4)
band.shape
```

Out[15]:

```
(76, 94)
```

Perform unsupervised classification with k-means algorithm on NIR band

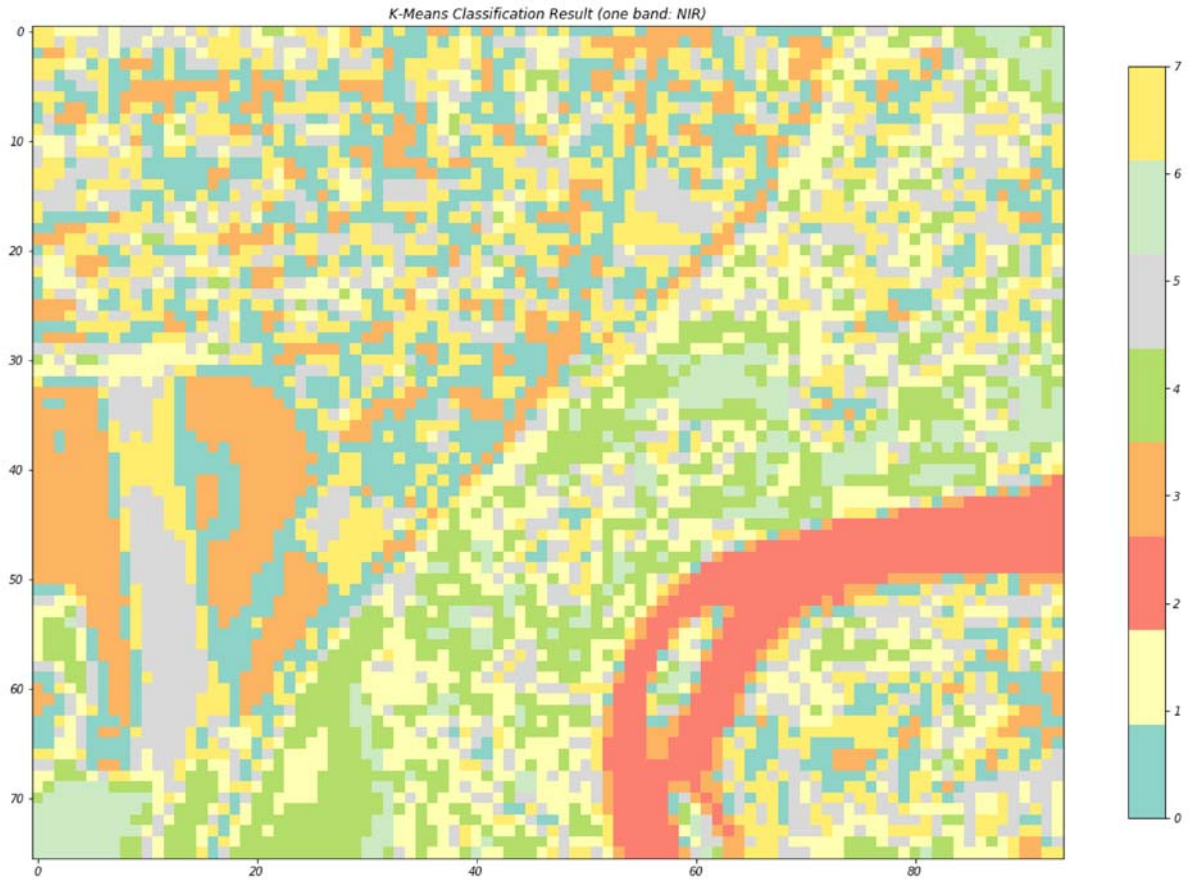
In [57]:

```
#Make linear vector for Sci-Kit Learn
X = band.reshape((-1,1))
print ('linear Vector:',X.shape)

# Lets use our classification Algorithm with 8 clusters.
# The kmeans classification of the NIR band will be stored in the object "kmeans_nir"
k_means = cluster.KMeans(n_clusters=8)
k_means.fit(X)
# Lets write and reshape our results into the original image dimensions
X_NIR = k_means.labels_
X_NIR = X_NIR.reshape(band.shape)
print ('reshaped vector:',X_NIR.shape)
```

```
linear Vector: (7144, 1)
reshaped vector: (76, 94)
```

In [58]:

```
#Lets plot the Results
fig,ax = plt.subplots(figsize=(20,20))
ax.set_title("K-Means Classification Result (one band: NIR)")
plot=ax.imshow(X_NIR, cmap=plt.cm.get_cmap('Set3', 8))
#plot.clim(-0.5, 2.5)
plt.colorbar(plot, ticks=[0, 1, 2, 3, 4, 5, 6, 7],shrink=0.6)

plt.show()
```



9. Run again the unsupervised classification. This time use 5 clusters instead of 8. Plot your results (hint: modify and adapt `k_means = cluster.KMeans(n_clusters=8);`

`plot=ax.imshow(X_NIR, cmap=plt.cm.get_cmap('Set3', 8));`

```
    plt.colorbar(plot, ticks=[0, 1, 2, 3, 4, 5, 6, 7],shrink=0.6) )
```

In [ ]:

```

```

10. Which differences can you see among the unsupervised classification with 8 clusters vs 5 clusters? What does the clusters mean?

In [ ]:

```

```

11. Repeat the previous procedure with 2 clusters. How accurate is the classification?

In [ ]:

```

```

12. Based on the previous plot with 2 clusters, can you properly classify the raster band? In other words, are all landclasses represented in the classification output?

In [ ]:

```

```

13. Based on the previous plot with 2 clusters, which is the minimum of clusters to choose in order to address all landclasses from your raster band?

In [ ]:

```

```

Perform unsupervised classification with k-means algorithm on the whole Rasterstack (4 bands)

Now you will apply everything you learned about kmeans classification on a RasterStack. Previously you did this for only one band: the NIR band. The procedure is similar, instead of 1 band, you will use now 4.

In [42]:

```
# First we have to write all 4 bands
stack = s2_stack.read()
print('Rasterio read multiband image shape equals',stack.shape)
s2_stack.meta["dtype"]

# Be careful Numpy Array indexes start at zero!
stack= np.zeros([stack.shape[1],stack.shape[2],stack.shape[0]],s2_stack.meta["dtype"])
print("Our desired format is",stack.shape)
```

```
Rasterio read multiband image shape equals (4, 76, 94)
Our desired format is (76, 94, 4)
```

In [43]:

```
for b in range(stack.shape[2]):
    # Let's print the band indexes
    # Be careful Numpy Array indexes start at zero!
    print(b)
    stack[:, :, b] = s2_stack.read(b+1)
```

```
0
1
2
3
```

14. What does (4, 76, 94) mean? Hint: see question 8

In [ ]:

```

```

To help you visualize all the bands that will be classified, you can use the command
`plt.imshow(stack[:,:,0], cmap='')`

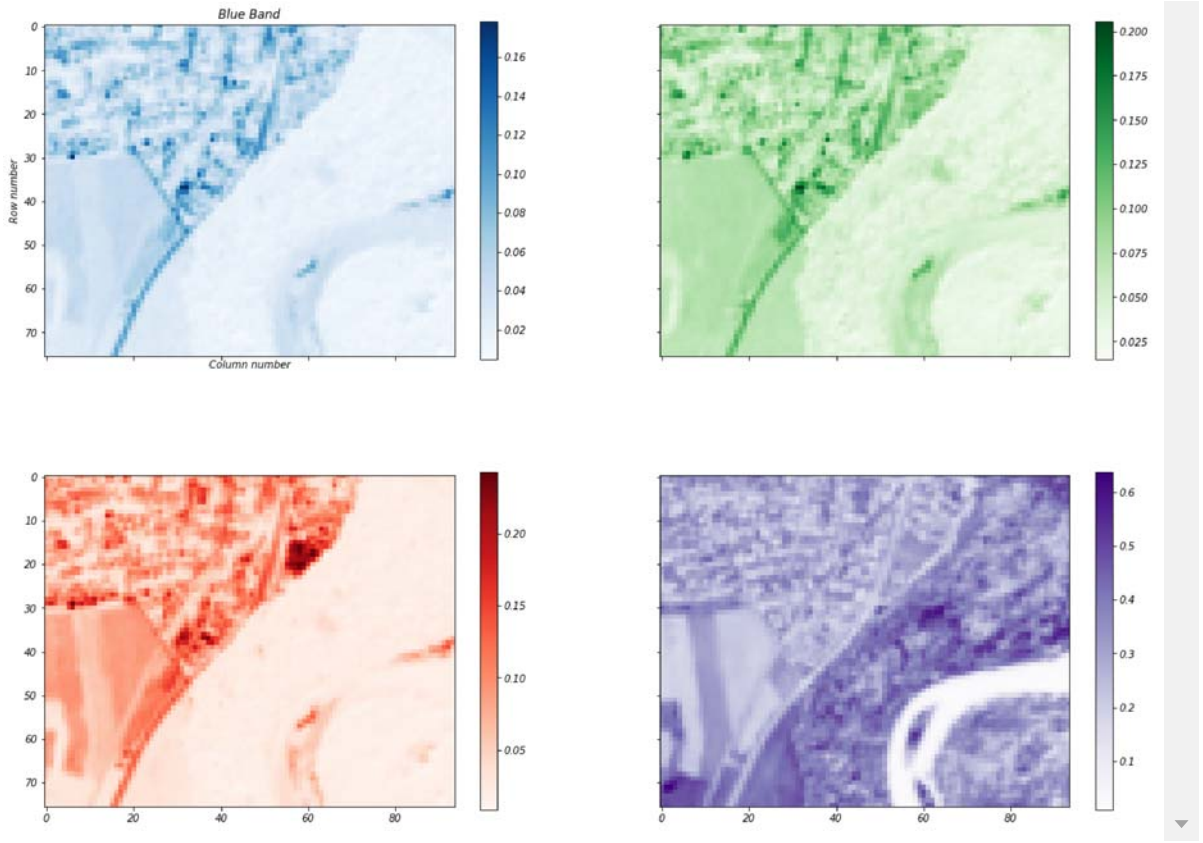
15. Adapt the following code to write the correct labels in the x,y axis and the title depending on the case.

In [46]:

```
fig, axes = plt.subplots(2,2, figsize=(20,15), sharex=True, sharey=True)
# Be careful Numpy Array indexes start at zero!

plt.sca(axes[0,0])
plt.imshow(stack[:,:,0], cmap='Blues')
plt.colorbar(shrink=0.9)
plt.title('Blue Band')
plt.xlabel('Column number')
plt.ylabel('Row number')

plt.sca(axes[0,1])
plt.imshow(stack[:,:,1], cmap='Greens')
plt.colorbar(shrink=0.9)
plt.title('')
plt.xlabel('')
plt.ylabel('')

plt.sca(axes[1,0])
plt.imshow(stack[:,:,2], cmap='Reds')
plt.colorbar(shrink=0.9)
plt.title('')
plt.xlabel('')
plt.ylabel('')

#This command equals the one we used before to visualize band 4 using the show() function f
plt.sca(axes[1,1])
plt.imshow(stack[:,:,3], cmap='Purples')
plt.colorbar(shrink=0.9)
plt.title('')
plt.xlabel('')
plt.ylabel('')
```

Out[46]:

```
Text(0, 0.5, '')
```



Now starts the classification code.

In [48]:

```
# Now lets reshape our stack into 4 linear vectors like the one we had before
# First we are going to calculate the dimnesions
new_shape = (stack.shape[0] * stack.shape[1], stack.shape[2])
print('The new dimensions are',new_shape)
```

```
The new dimensions are (7144, 4)
```

In [49]:

```
# Secondly we can use the calculated shape
X = stack[:, :, :4].reshape(new_shape)
```

In [50]:

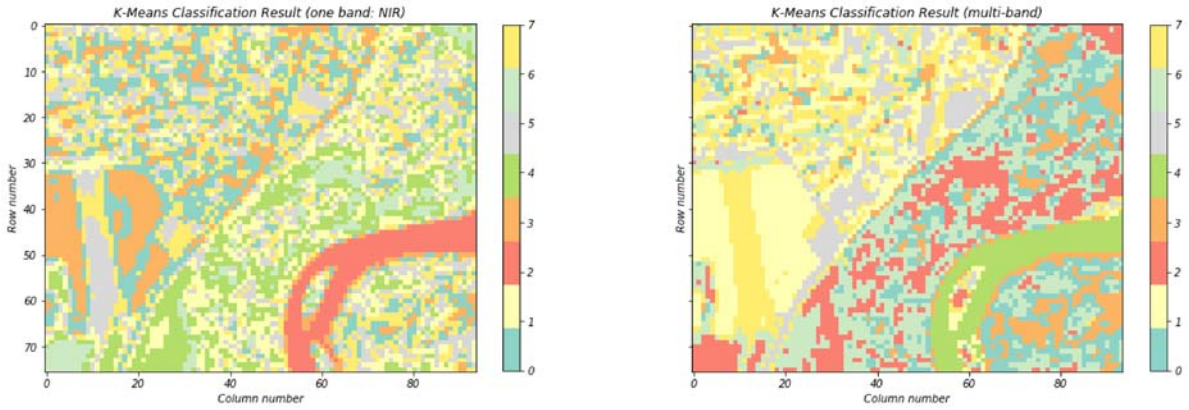
```
#Now we can use k_means like we did before
k_means = cluster.KMeans(n_clusters=8)
k_means.fit(X)

X_cluster = k_means.labels_
# and reshape it
X_cluster = X_cluster.reshape(stack[:, :, 0].shape)
```

In [60]:

```
#Now lets plot our multiband classification together with the classfication if the NIR-Band
fig, axes = plt.subplots(1,2, figsize=(20, 20), sharex=True, sharey=True)

plt.sca(axes[0])
plt.imshow(X_NIR, cmap=plt.cm.get_cmap('Set3', 8))
plt.colorbar(ticks=[0, 1, 2, 3, 4, 5, 6, 7],shrink=0.3)
plt.title("K-Means Classification Result (one band: NIR)")
plt.xlabel('Column number')
plt.ylabel('Row number')

plt.sca(axes[1])
plt.imshow(X_cluster, cmap=plt.cm.get_cmap('Set3', 8))
plt.colorbar(ticks=[0, 1, 2, 3, 4, 5, 6, 7],shrink=0.3)
plt.title("K-Means Classification Result (multi-band)")
plt.xlabel('Column number')
plt.ylabel('Row number')
```

Out[60]:

```
Text(0, 0.5, 'Row number')
```



16. Repeat the unsupervised (kmeans) classification of the Rasterstack with the right number of clusters based on the landclasses that you recognize (hint: see questions 9, 11).

In [ ]:

NOTE: for more information about the color scales, visit: https://matplotlib.org/tutorials/colors/colormaps.html (https://matplotlib.org/tutorials/colors/colormaps.html)

This tutorial was prepared with the support from Gabriel Cevallos. May 2020