OBIC_SVM

March 28, 2024

1 Project title:-

Advancing Earth Observation Data and ResUNet-Deep Learning Model for Irrigated Area Mapping: The Case of Along the Awash Valley, Ethiopia

2 Object Based Image Classification (OBIC) using Support Vector Machine (SVM) classifier

This Jupyter notebook demonstrates how to apply OBIC using RF classifier with the ESA EO-Africa inovation lab cloud computing environment.

Prerequisites for running this notebook

Several packages need to be installed and/or imported for running this script:

The `rasterio, skimage, sklearn,numpy and geopandas` modules should be installed first to apply OBIC based RF classifier ;

2.1 Importing the relevant modules

[15]:
```
import rasterio
import numpy as np
from rasterio.plot import show
from skimage import io, segmentation, color
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import geopandas as gpd
```

2.1.1 Load RGB image

[2]:
```
sentinel_image_path = "/home/eoafrica/Sentinel2_AWbasin/sentinel2_layerstack/
 ↪stacked_image.tif"
with rasterio.open(sentinel_image_path) as src:
   rgb_image = src.read()
```

2.1.2 Load the ground control points (GCP) data from Shapefile

[3]:
```
gcp_shapefile_path = "/home/eoafrica/Sentinel2_AWbasin/GCP_LULCawash/lulcgcp.
↪shp"
gcp_data = gpd.read_file(gcp_shapefile_path)
```

2.1.3 Extract coordinates and class labels from the GCP data

[4]:
```
gcp_coords = np.column_stack((gcp_data.geometry.x, gcp_data.geometry.y))
gcp_labels = gcp_data['class_labe'].values
```

2.1.4 Check shapes

[4]:
```
print("gcp_coords shape:", gcp_coords.shape)
print("gcp_labels shape:", gcp_labels.shape)
```

2.1.5 Convert GCP coordinates to image indices

[4]:
```
gcp_indices = np.floor(src.index(gcp_coords[:, 0], gcp_coords[:, 1]))
```

```
gcp_coords shape: (88, 2)
gcp_labels shape: (88,)
```

2.1.6 Extract RGB values at GCP locations

[5]:
```
gcp_rgb_values = rgb_image[:, gcp_indices[1].astype(int), gcp_indices[0].
↪astype(int)].T
```

2.1.7 Split data into training and testing sets

[6]:
```
X_train, X_test, y_train, y_test = train_test_split(gcp_rgb_values, gcp_labels,␣
↪test_size=0.2, random_state=42)
```

2.1.8 Remove rows with NaN values from the GCP data

[7]:
```
valid_indices = ~np.isnan(y_train)
X_train = X_train[valid_indices]
y_train = y_train[valid_indices]
```

2.1.9 Train a Random Forest classifier

[8]:
```
classifier = SVC(kernel='linear', C=1.0)
classifier.fit(X_train, y_train)
```

[8]:
```
SVC(kernel='linear')
```

2.1.10 Predict labels for the entire image

[9]:
```
rgb_image_reshaped = np.moveaxis(rgb_image, 0, -1)
segments = segmentation.slic(rgb_image_reshaped, compactness=20, n_segments=500)
segmented_image = color.label2rgb(segments, rgb_image_reshaped, kind='avg')
```

2.1.11 Extract features for each segment

[10]:
```
segment_features = []
for segment_id in np.unique(segments):
    segment_mask = segments == segment_id
    segment_rgb_values = rgb_image_reshaped[segment_mask]
    segment_features.append(np.mean(segment_rgb_values, axis=0))
```

2.1.12 Classify segments

[11]:
```
predicted_labels = classifier.predict(segment_features)
```

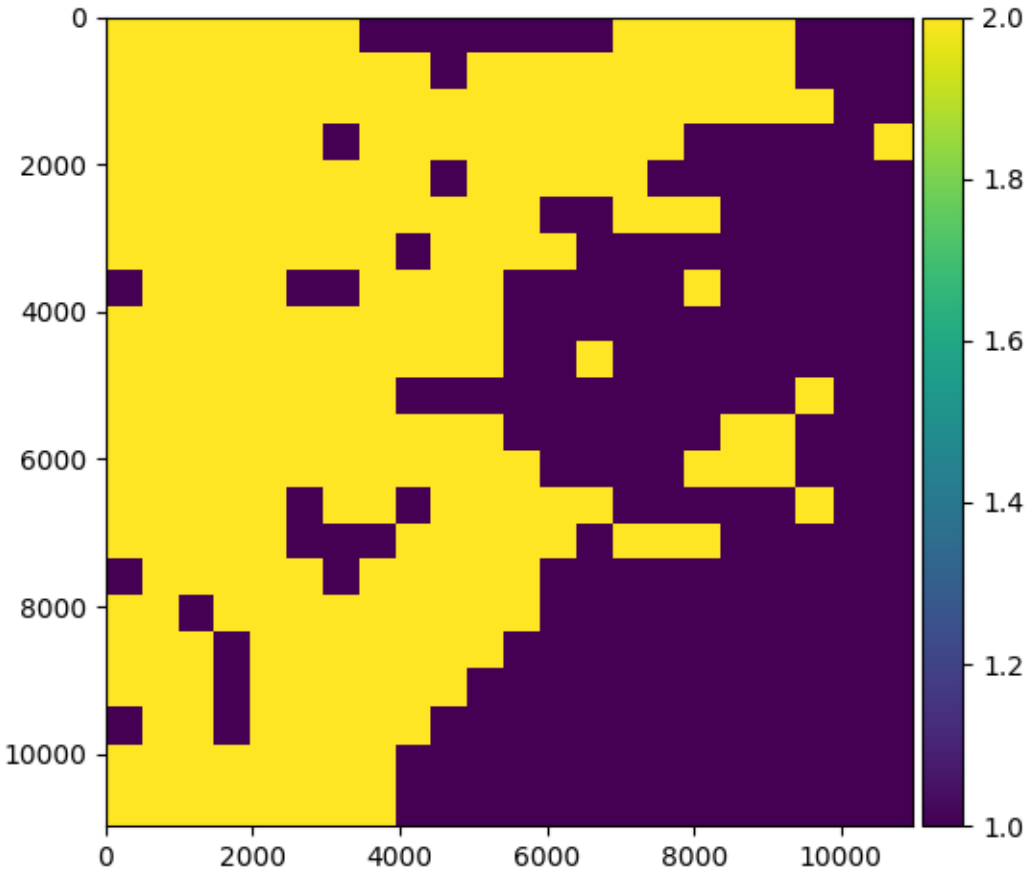
2.1.13 Visualize the classified image

[12]:
```
classified_image = np.zeros_like(segments)
for segment_id, label in zip(np.unique(segments), predicted_labels):
    classified_image[segments == segment_id] = label

io.imshow(classified_image)
```

```
/usr/local/lib/python3.8/dist-
packages/skimage/io/_plugins/matplotlib_plugin.py:149: UserWarning: Low image
data range; displaying image with stretched contrast.
  lo, hi, cmap = _get_display_range(image)
```

[12]:
```
<matplotlib.image.AxesImage at 0x7f2751ded850>
```


0
2000
4000
6000
8000
10000
0
2000
4000
6000
8000
10000
2.0
1.8
1.6
1.4
1.2
1.0

[ ]: