Lab 4 – Image Classification and Time-based Analysis

Introduction

This lab covers two important topics in digital image analysis using GEE: image classification and time-series analysis. The overall objective of image classification is to automatically categorize all pixels in an image into land cover classes or themes. This lab covers how to conduct supervised and unsupervised classifications, apply spectral mixture analysis, and perform classification accuracy assessment. This lab also covers explores time series analysis.

Pre-lab requirement: i.e. you must complete this before lab starts

Read the following pages in Lillesand et al. (2015):

P537–538 Image Classification
P538–540 Supervised Classification
P546–556 The Training Stage
P557–560 Unsupervised Classification
P562–567 Classification of Mixed Pixels and Spectral Mixture Analysis
P575–582 Classification Accuracy Assessment
P587–591 Image Time Series Analysis

Read the following paper.

Pal, M. (2005). Random forest classifier for remote sensing classification. *International Journal of Remote Sensing*, *26*(1), 217–222. ttps://doi.org/10.1080/01431160412331269698.

Objectives

This lab focuses on guiding you through several image processing techniques. At the end of this lab you will be able to use GEE to perform the following tasks:

- Supervised Classification
- Classification Accuracy Assessment
- Unsupervised Classification
- Spectral Mixture Analysis
- Image Time Series Analysis

Supervised image classification and classification accuracy assessment

Supervised Classification

The overall objective of image classification procedures is to automatically categorize all pixels in an image into land cover classes or themes. Supervised classification procedure involves three basic components: (1) training, (2) classification, and (3) assessment. The analyst first identifies a set of training data representing the spectral attributes of each land cover type of interest. The analyst then selects the desired mathematical approach, i.e. the classifier, which will be used for spectral pattern recognition. The selected classifier is trained on a limited set of pixels (training data) and then used to categorize each pixel in the input image into the land cover classes.

In GEE, classification is performed using five steps:

1. Collect training data. Identify features and assign a property that stores the known class label and numeric values for the predictors.
2. Instantiate a classifier. Set parameters if necessary.
3. Train the classifier using the training data.
4. Classify an image or feature collection.
5. Estimate classification error with independent validation data.

The training data is a FeatureCollection with properties used to store the class label and predictor variables. Class labels in GEE should be small, consecutive, non-negative integers representing classes. If necessary, remap() is used to convert class values to consecutive integers starting from 0. The predictors should be numeric.

Training and validation data can come from a variety of sources. You can use the geometry drawing tools to collect training data interactively in Earth Engine. Alternatively, you can import predefined training data from an Earth Engine table asset or a fusion table. The example below aims to classify an image into vegetation and non-vegetation classes using a supervised classification approach.

```
// Determine a study area for further analysis
var studysite = ee.Geometry.Polygon(
        [[[-76.1015, 43.0793],
          [-76.0981, 43.0190],
          [-76.0203, 43.0225],
          [-76.0258, 43.0808]]]);

// Import all the available Landsat 5 images then filter by location and sort by cloud cover.
var L5filtered = ee.ImageCollection('LANDSAT/LT05/C01/T1_TOA')
            .filterBounds(studysite) //filter to limit images to those over the study site
            .sort('CLOUD_COVER'); //sort by cloud cover to obtain an ideal image for classification
```

```
// Select the first image (i.e. the scene with the least cloud cover) in the sorted collection above.
var input = ee.Image(L5filtered.first()) //first in sort on cloud cover is least cloudy
            .clip(studysite); //clip the selected image scene to the study site

// Set map display to center on the study area at a zoom level of 13
Map.centerObject(studysite, 13);

// Display the input image as a color infrared composite.
Map.addLayer(input, {bands: ['B4', 'B3', 'B2']}, 'input image');

/* Manually label training points as vegetation or non-vegetation.
To do this you need to first create two new layers (under "Geometry Imports").  Click on the gear
icon next to each layer name, renaming the layers as Veg_train and Nonveg_train, and change both
 to FeatureCollections. Once the layers are created, select one of the layers and then click on the
point icon and each identify reference points that belong to the class. After all of the reference
points are labeled click on the gear icon again and add a property for both Veg_ref and
Nonveg_ref layers, which has name = class and value = 1 for veg or 0 for non-veg.
*/

// Combine the veg and non-veg layers into a reference dataset
var training = Veg_train.merge(Nonveg_train);

/* Use the training data to sample the classified image and create a new band called class that
contains the corresponding value for each reference point based on the classified image.
*/
var training = input.sampleRegions({
   collection: training,
   properties: ['class'],
   scale: 30
});

/* Create and train a random forest classifier. A random forest classifier uses a combination of
decision tree classifiers where each is generated using a random vector sampled independently
from the input vector. Each tree casts a unit vote for the most popular class to classify an input
vector (Pal 2005). */
var classifier = ee.Classifier.randomForest(100) //set the number of trees in random forest to 100
   .train(training, 'nd'); //'nd' is 0 for non-vegetation and 1 for vegetation cover types.

// Classify the input imagery.
var classified = input.classify(classifier);

// Display classified image in the interactive map. Non-vegetation land cover has a pixel value of
0 and vegetation cover has a value of 1.
Map.addLayer(classified,{min:0,max:1},'classified image');
```

Classification accuracy assessment

Classification accuracy assessment verifies the accuracy of the classified image with reference data that is typically generated either through interpretation of imagery with higher spatial resolution or ground data collection. It is essential to document the categorical accuracy of the

classification results before moving onto other endeavors and a classification is not complete until its accuracy is assessed. This lab utilizes the most common means of expressing classification accuracy preparation using a classification error matrix, also known as a confusion matrix. This matrix compares, on a category-by-category basis, the relationship between reference data and the corresponding classification results.

This exercise continues the code example from the supervised classification example above. The accuracy assessment verifies the accuracy of the supervised classification using reference points generated from a NAIP image that was collected around the same time as the Landsat 5 image to.

```
// Find and import a NAIP image taken around the same time when the input Landsat was taken.
var referenceimage = ee.ImageCollection('USDA/NAIP/DOQQ')
                .filterBounds(studysite) //filter images by the previously defined study area
                .filterDate('2011-01-01','2011-12-31'); //filter to match Landsat image date

// Display the filtered reference NAIP image.
Map.addLayer(referenceimage,"",'reference NAIP image');

// Produce 100 random sampling points for producing the reference points.
var referencep = ee.FeatureCollection.randomPoints(studysite,100,0);

// Display the reference points in red.
Map.addLayer(referencep,{color:'red'},"reference points");

/* Manually label the reference points as vegetation or non-vegetation.
To do this you need to first create two new layers (under "Geometry Imports").  Click on the gear
icon next to each layer name, renaming the layers as Veg_ref and Nonveg_ref, and changing both
to FeatureCollections. Once the layers are created, select one of the layers and then click on the
point icon and each identify reference points that below to the class. After all of the reference
points are labeled click on the gear icon again and add a property for both Veg_ref and
Nonveg_ref layers, which has name = class and value = 1 for veg or 0 for non-veg.
*/

// Combine the veg and non-veg layers into a reference dataset
var reference = Veg_ref.merge(Nonveg_ref);

/* Use the reference data to sample the classified image and obtain a new band called
classification that contains the corresponding value for each reference points based on the
classified image.
*/
var validated = classified.sampleRegions({
   collection: reference,
   properties: ['class'],
   scale: 30
});

// Generate and display a confusion matrix representing expected accuracy.
var testAccuracy = validated.errorMatrix('class', 'classification');
```

```
print('Validation error matrix: ', testAccuracy);
print('Validation overall accuracy: ', testAccuracy.accuracy()); //Display overall accuracy

//Note the accuracy is very low. What do you think contributes to this problem? How can you
improve the classification accuracy?
```

Unsupervised image classification

Unsupervised classification does not utilize training data as the basis for classification. Unsupervised classifiers examine the pixels in an image and aggregate them into classes based on natural groupings or clusters present in the image values. This approach is based on the premise that a particular cover type is likely to similar in the measurement space.

The ee.Clusterer package handles unsupervised classification (or clustering) in Earth Engine. These algorithms are currently based on the algorithms with the same name in Weka, an open source collection of machine learning algorithms. Clusterers are used the same manner as classifiers in Earth Engine. The general workflow for clustering is:

1. Assemble features with numeric properties in which to find clusters.
2. Instantiate a clusterer. Set its parameters if necessary.
3. Train the clusterer using the training data.
4. Apply the clusterer to an image or feature collection.
5. Label the clusters.

The example below classifies a Landsat 5 image scene into 3 classes, which are then labeled through visual interpretation.

```
// Construct a study area for further analysis.
var studysite = ee.Geometry.Polygon(
      [[[-76.1015, 43.0793],
        [-76.0981, 43.0190],
        [-76.0203, 43.0225],
        [-76.0258, 43.0808]]]);

// Find and import a Landsat 5 image scene to classify.
var input = ee.ImageCollection('LANDSAT/LT05/C01/T1_TOA')
          .filterBounds(studysite) //filter based on the study site
          .filterDate('2011-06-01','2011-06-04') //to match the NAIP imagery used for validation
          .sort('CLOUD_COVER') //sort all images by cloud cover to obtain the best image to classify
          .mosaic() //this converts the output of this ee.ImageCollection into an image
          .clip(studysite); //clip the selected image scene with the study site

// Set map display to center on the study area at a zoom level of 13
Map.centerObject(studysite, 13);
```

```
// Display the input image as a CIR composite.
Map.addLayer(input, {bands: ['B4', 'B3', 'B2']}, 'input image');

/* Sample the image to construct a "training" sample of pixels to characterize the image.
Unsupervised classifiers do not need training data; however, processing time is decreased by
generating image statistics from a subset of pixels.  Rather than sampling pixels, you can increase
the numPixels value to include all pixels in the scene but this will also increase script run time.*/

var training = input.sample({
  region: studysite,
  scale: 30,
  numPixels: 5000
});

// Instantiate the unsupervised classifier and train it. Here we specify three classes.
var clusterer = ee.Clusterer.wekaKMeans(3).train(training);

// Classify the input image using the trained unsupervised classifier.
var result = input.cluster(clusterer);

// Display the classified image with random colors for each of the three classes
Map.addLayer(result.randomVisualizer(), {}, 'classified image');

//Visually interpret the classified image. What do you think the three classes should be?
```

Spectral mixture analysis

Spectral mixture analysis is also called spectral unmixing in GEE. This type of analysis uses a range of techniques wherein mixed spectral signatures are compared to a set of "pure" reference spectra. The result is an estimate of the approximate proportions of the ground area of each pixel that is occupied by each of the reference classes.

```
// Load a Landsat 5 image and select the bands we want to unmix.
var bands = ['B1', 'B2', 'B3', 'B4', 'B5', 'B6', 'B7'];
var image = ee.Image('LANDSAT/LT05/C01/T1/LT05_015030_20100531')
  .select(bands);
Map.setCenter(-76.1467, 43.0458,12);
Map.addLayer(image, {bands: ['B4', 'B3', 'B2'], min: 0, max: 127}, 'image');

/* Define spectral endmembers. Each number corresponds to the DN values (from 0–127) of the
seven Landsat image bands. You can obtain values of endmembers by going to the Inspector tab
and clicking on "pure" pixels of a selected land cover type. More endmembers can also be found
in the USGS or NASA JPL spectral libraries. */
var urban = [88, 42, 48, 38, 86, 115, 59];
```

```
var veg = [64, 30, 20, 118, 74, 138, 22];
var water = [63, 23, 16, 14, 6, 131, 4];

// Unmix the input image based on the provided spectral endmembers.
var fractions = image.unmix([urban, veg, water]);

/*Display unmixed image. In the unmixed image, the red band shows the proportion of urban
cover, green shows vegetation and blue shows water based on the spectral endmember you
provided earlier.  When viewed as an RGB image, the color shows the proportion of each cover
types e.g. cyan pixels are a mixture of water and vegetation, magenta pixels are a mixture of
urban and vegetative cover. */
Map.addLayer(fractions, {}, 'unmixed');

/*With only three endmembers, then is clearly some limitation to this unmixing e.g. based on the
color shown in the RGB image, what type of land cover are the bare fields west of Syracuse?
```

Image time series analysis

Time series analysis utilizes long-term, systematically collected data to explore seasonal to decadal variations in agricultural, ecology, hydrologic and other regional-scale systems (Lillesand et al. 2015).

The example below shows an image time series analysis on the surface temperature of three locations near Syracuse.

```
// Construct a FeatureCollection for three different locations near Syracuse
var points = ee.FeatureCollection([
  ee.Feature(    // Syracuse
    ee.Geometry.Point(-76.1505,43.0498), {label: 'City of Syracuse downtown'}),
  ee.Feature(  // Clark Reservation
    ee.Geometry.Point(-76.0899,42.9983), {label: 'Forest in Clark Reservation'}),
  ee.Feature(  // Oneida Lake
    ee.Geometry.Point(-75.9045,43.2007), {label: 'Water in Oneida Lake'})
]);

// Import Landsat 8 brightness temperature data for 3 years.
var temps = ee.ImageCollection('LANDSAT/LC8_L1T_TOA')
    .filterBounds(points) // Filter image scenes based on the three defined locations above
    .filterDate('2013-01-01', '2016-12-31') // Filter image scenes from 2013 to 2016
    .select('B11') // Select band 11 for the thermal infrared response
    .filterMetadata('CLOUD_COVER', 'less_than', 40); //filter based on image cloud cover

/* Create a time series chart showing surface temperature change for the three defined locations
based on three years of Landsat 8 images. */
var tempTimeSeries = ui.Chart.image.seriesByRegion(
    temps, points, ee.Reducer.mean(), 'B11', 100, 'system:time_start', 'label')
```

```
        .setChartType('ScatterChart') // Set the chart to be a scatter plot
        .setOptions({ //Set options of the plot
          title: 'Temperature over time for locations near City of Syracuse', //Set title
          vAxis: {title: 'Temperature (Kelvin)'}, //Set x axis label
          lineWidth: 1, // Set the width of the line
          pointSize: 4, // Set the point size for each data point
          series: {
            0: {color: 'FF0000'}, // color for City of Syracuse downtown
            1: {color: '00FF00'}, // color for Forest in Clark Reservation
            2: {color: '0000FF'}  // color for Water in Oneida Lake
}});

// Display time series chart.
print(tempTimeSeries);
```

Assignment – Answer the Following Questions

Submit your code and text answers for this assignment by clicking on "Get Link" in the Code Editor and sending the link generated to the TA (your link will look something like https://code.earthengine.google.com/XXXXXXXX. Any written responses should be in comments with each question separated by a line of forward slashes. For example:

//Q1. Text answer to Q1
Code answer to Q1 //All code responses need suitable comments.

//
//Q2. Text answer to Q2
Code to Q2
...

1. Select one Landsat 8 image scene within Onondaga County. Using the random forest classification approach shown in this lab, classify the land cover into vegetation and non-vegetation classes.

2. Compute the accuracy of the classification in the above task. What is the error matrix and what is the overall accuracy?

3. Referring back to the unsupervised classification approach shown in this lab. What labels do you think should be assigned to the three classified land cover classes?

4. Select one Landsat 8 image scene within Onondaga County. Find your own spectral endmembers for urban, vegetation and water in this image, and perform a spectral mixture analysis using the endmembers you created. How would you interpret the unmixed image?

5. Perform a time series analysis of the three Syracuse locations—Syracuse, Clark Reservation, and Oneida Lake—using three years of Landsat 5 NDVI data. How does the NDVI change over time at these three locations?

Appendix A - Glossary of Terms

Page numbers refer to Lillesand et al. (2015).

- Accuracy Assessment - P575–582. Classification accuracy assessment verifies the accuracy of the classified image with reference data typically generated either through images with smaller spatial resolution or ground data.
- Change Detection - P582–587. Change detection refers to the identification and characterization of changes over time, through preserved records of remote sensing images at different points of time.
- Image Time Series Analysis - P587–591. Image time series analysis explores temporal trends using long-term, systematically collected data.
- Spectral unmixing - P 564–567. Also known as spectral mixture analysis. Spectral unmixing uses a range of techniques wherein mixed spectral signatures are compared to a set of "pure" reference spectra.
- Supervised Classification - P538–540. The overall objective of supervised image classification procedures is to automatically categorize all pixels in an image into *a priori* land cover classes that the user identifies through training data.
- Unsupervised Classification - P557–560. Unsupervised classification does not utilize training data as the basis for classification. This approach statistically analyzes the input dataset and develops classes based on the premise that pixels within a given cover type should be close together in measurement space.

Appendix B – Useful Resources

- Google Earth Engine Java Style Guide
  http://google.github.io/styleguide/javascriptguide.xml

- Google Earth Engine Guides/Cookbook/Dictionary
  https://developers.google.com/earth-engine/

- Google Earth Engine API Tutorials
  https://developers.google.com/earth-engine/tutorials

- Google Earth Engine Workshop (Beginning)
  https://docs.google.com/document/d/1ZxRKMie8dfTvBmUNOO0TFMkd7ELGWf3WjX0JvESZdOE

- Google Earth Engine Workshop (Intermediate)
  https://docs.google.com/document/d/1keJGLN-j5H5B-kQXdwy0ryx6E8j2D9KZVEUD-v9evys

- Google Earth Engine tutorials on AmericaView
  https://americaview.org/program-areas/education/google-earth-engine-tutorials/

General references

Lillesand, T. M., Kiefer, R. W., & Chipman, J. W. (2015). Remote Sensing and Image Interpretation. 7th edition. John Wiley & Sons.

Pal, M. (2005). Random forest classifier for remote sensing classification. International Journal of Remote Sensing, 26(1), 217–222. ttps://doi.org/10.1080/01431160412331269698.

This work was produced by Ge (Jeff) Pu and Dr. Lindi Quackenbush at State University of New York-College of Environmental Science and Forestry. Suggestions, questions and comments are all welcome and can be directed to Jeff at gpu100@syr.edu.