Lab 07 – Unsupervised Classification

In Lab 06, we looked at a supervised classification approach using the "ensemble" decision tree classifier called RandomForest (RF). RF tends to be the "Go To" classifier these days, but it by no means is the only one available, nor is it the best one in all occasions.

The main sticking point with supervised classification techniques like RF is that we need a large number of a-priori training data in order to generate the output. Sometimes we do not have that luxury.

The alternative to the Supervised approach is the **Unsupervised** approach. In an unsupervised classification, the user employs various clustering algorithms to generate an output where pixels that have similar spectral properties are grouped together into a single "**spectral cluster**". Once the individual pixels in an image are grouped into similar spectral clusters, effectively reducing the image into a set of categories, the user must then identify what each spectral cluster represents on the ground. So…., no matter what you do, you still need to have field data to generate a land cover classification. The biggest difference, therefore, is that the user guides a supervised classification, but in an unsupervised classification the image "guides" (in a way) the user to identify the land cover type that each spectral cluster represents.

You could say that using one method or the other is a matter of "six of one, and a half dozen of the other" since you arrive at the same end point, but in some ways, the unsupervised approach allows the user to explore the data and gain a better understanding of how land cover types can be confused with one another.

In this exercise, I will provide the script and I will then ask you to modify that script to create and evaluate the differences you get when we add different data sets to our classification.

This is the link to the script

The script tends to push the capacity of GEE – probably because I'm not experienced enough to optimize it. If you get a capacity exceeded error, run it again.

This is what it does:

1. Identify the input data
   a. Sentinel 2, Level-1C imagery filtered to mid-summer 2020, clouds, and a boundary (BND)
   b. USGS National Elevation Dataset 1/3 arc-second (10m resolution)
2. Sets up the Sentinel-2 cloud mask function and sets parameters to rename bands and to establish the maximum number of clusters to generate. This is a guess, but 15 seems like the right number for our purposes. Not too few, but not so many that it takes too much time to evaluate for this exercise. As a rule of thumb, I set the max number of clusters to 3x the number of land cover categories that I expect to map. So, if I want to map 10 different types, I usually start with 30 clusters.
3. Filters the data.
4. Sets up the training, cluster generation, and image clustering (3 steps)

R. Douglas Ramsey
Dept. of Wildland Resources
Quinney College of Natural Resources
Utah State University
Logan, Utah 84322-5230
doug.ramsey@usu.edu

a. The training consists of sampling 5000 pixels (not sure of the .sample() command uses a random or systematic sample)

b. Training the clustering algorithm (wekaXMeans) with the 5000 samples we collected. wekaXMeans is a K-means clustering protocol with the ability to determine the optimum number of clusters that the image will support.

c. The .cluster() command applies the result of the wekaXMean algorithm to the image.

5. Displays the image, classification (visibility off, turn on if you want to see it), and a selected cluster from the classification.
6. The remainder of the script generates some diagnostic plots that can help us identify some of the clusters relative to land cover type. I will explain these in more detail in lecture.

You need to:

1. From the **'combined'** image, don't add the **Terrain** bands. Comment out everything past the .addBands.

```
var combined = input.addBands(Terrain,['DEM','slope'])
```

2. Rerun the classification and evaluate the output. Can you see any confusion between land cover types?

   a. To evaluate the classification, I have added a **Map.addLayer** function that only shows one cluster at a time:

   ```
   Map.addLayer(result.mask(result.eq(2)), {palette: ['red'], opacity: 0.75}, 'clusters');
   ```

   Change the .eq value to a cluster number (0-14 since we have 15 clusters) and re-run the script to show that particular cluster

3. Extract from the Sentinel-2 image collection a fall image (say mid Sept – mid Oct)
4. Add the fall, cloud-free, composite image to the stack of bands in the initial Sentinel-2 mid-Summer mosaic that I generated. You should have a final image with 20 spectral bands

   ['B02','B03','B04','B05','B06','B07','B08','B08A','B11','B12'] 10 from summer and

   ['B02','B03','B04','B05','B06','B07','B08','B08A','B11','B12'] 10 from fall

   You will also have the DEM and slope layers for a total of 22 bands.

5. Regenerate the unsupervised classification with 15 clusters - you will have to change parameters in the script, so read through it carefully to see where you have to change things to account for the extra bands.
6. Compare the clusters from the first (single image) classification to the second (two image) classification. Are there any significant differences, improvements, etc.?

   a. Use the technique in step 2 above to show the individual clusters.

7. If you do see confusion in the clusters with the final (22-band) classification, what is that confusion and could there be a way to correct it?

R. Douglas Ramsey
Dept. of Wildland Resources
Quinney College of Natural Resources
Utah State University
Logan, Utah 84322-5230
doug.ramsey@usu.edu