Crop Classification Report

Bhanu Kappala

February 3, 2019

After first examining the .tif images in Windows Photo Viewer, I proceeded to start by installing the GDAL (Geospatial Data Abstraction Library) package and associated python bindings. I did a quick info dump (gdalinfo) on the .tif files to just take a quick look at what kind of data I was working with before I applied any algorithms (this data can be see in north.txt for 20130824_RE3_3A_Analytic_Champaign_north.tif and cdal.txt for CDAL_2013_Champaign_north.tif). The resulting info gave me a good idea of what I was working with: the RapidEye satellite imagery consists of 5 bands (blue, red, green, near infrared, and red edge). Using the GDAL Python library (all this code is done in all the python files at the very beginning), if I read the image as Numpy array I would get a 3D array which had shape of (5, 5959, 9425). The 5 represents a value for each of the 5 spectral bands, while 5959x9425 is the image size so each individual element in that array represents a numerical representation of one of the 5 spectral bands in each pixel of the .tif image. I decided to proceed by first cleaning the CDL file; I grouped all of the values that were not corn or soybean into an "Other" value (our classification problem really only has 3 categories - corn, soybean, and other).

I decided to first implement a k-nearest neighbor algorithm as a simple approach to see how accurate it could be (note that all computations were done on a t3.large AWS EC2 instance). In order to decrease computation time, I trained using only 20% of the data (still around 10 million observations). I also applied other common algorithms (all in their respective *.py files) and wrote all their results to results.txt. Based on that, I decided that I would use a random forest classifier to train on the whole northern half image due its slightly higher accuracy. The full model is implemented and trained in fullrandomforest.py. Afterwards, the prediction is run by passing the 20130824_RE3_3A_Analytic_Champaign_south.tif as an array and predicting using the model. The resulting test1.tif file has no image, so another python script is used to convert "test1.tif" into a PNG file, then a web tool was used to convert the resulting PNG file and reshape into a 9425x5959 .tif file.