Statistics 201B Project: DSTL Satellite Imagery Feature Detection

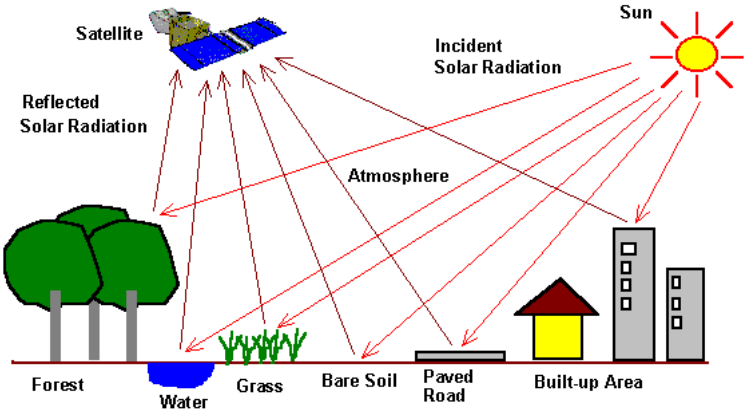
Jiatong Chen
Kaiwen Huang
Winter 2017

1. Introduction

This is a Kaggle project just finished recently. We are required to classify the types of objects found in 1km x 1km satellite images. We first processed and extracted features from WKT input format so that they are more readily accessible by our models. Then we apply Xgboost and SGD with logistic regression to train the model and use test set to evaluate the performance.

2. Background

The motivation for this challenge comes from how proliferation of satellite imagery has helped human understand our planet. Satellite imagery not only achieve monitoring through mobilizing resources which is especially useful during disasters or long-term global warming, but has given human the ability to automate detection of objects and patterns. The advancement in the functionality of satellite imagery largely relies on the labeling on features of significance, just as human identifies the features through eyes and cognition. In this sense, this project is analogous to "training an eye in the sky" and we want to ultimately build a model that automate the process of feature labeling on images.



The images in this competition are provided by Defence Science and Technology Laboratory (DSTL). We are challenged to accurately classify 10 object classes in the complex and large dataset of satellite images. The models will help DSTL to seek novel solution in image analytics and also make smart and decisions more quickly as well as yield innovations to computer vision technologies for satellite imagery.

3. Feature Processing

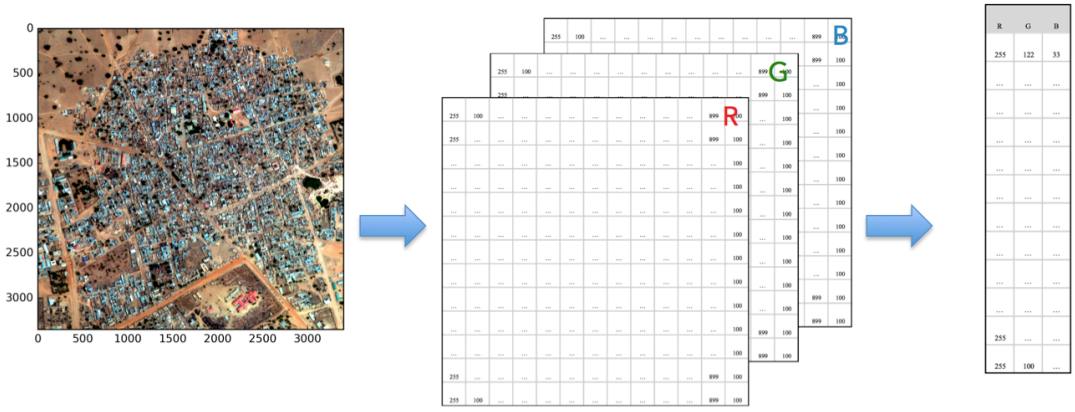
DStl provides 2 sets of data, 3-band and 16-band formatted 1km x 1km GeoTiff satellite images. 16-band images contain 8 Multispectra from 400 nm to

1040 nm, 8 SWIR (short wavelength infrared) from 1195 nm - 2365 nm, 1 Panchromatic energy from 450 to 800 nm. 3-band images are traditional RGB natural color images, but with higher resolution.

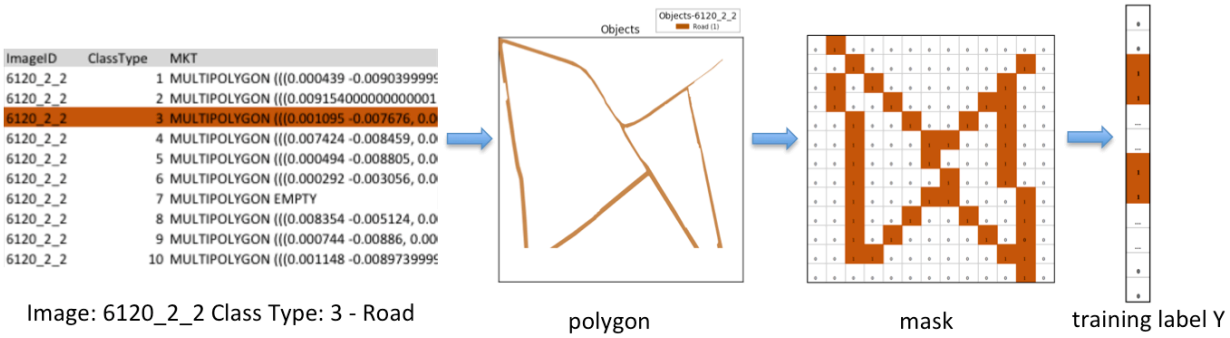
There are 10 types of objects we are expected to classify. They are stored as multipolygon coordinates in WKT files.

1. *Buildings* - large building, residential, non-residential, fuel storage facility, fortified building
2. *Misc. Manmade structures*
3. *Road*
4. *Track* - poor/dirt/cart track, footpath/trail
5. *Trees* - woodland, hedgerows, groups of trees, standalone trees
6. *Crops* - contour ploughing/cropland, grain (wheat) crops, row (potatoes, turnips) crops
7. *Waterway*
8. *Standing water*
9. *Vehicle Large* - large vehicle (e.g. lorry, truck,bus), logistics vehicle
10. *Vehicle Small* - small vehicle (car, van), motorbike

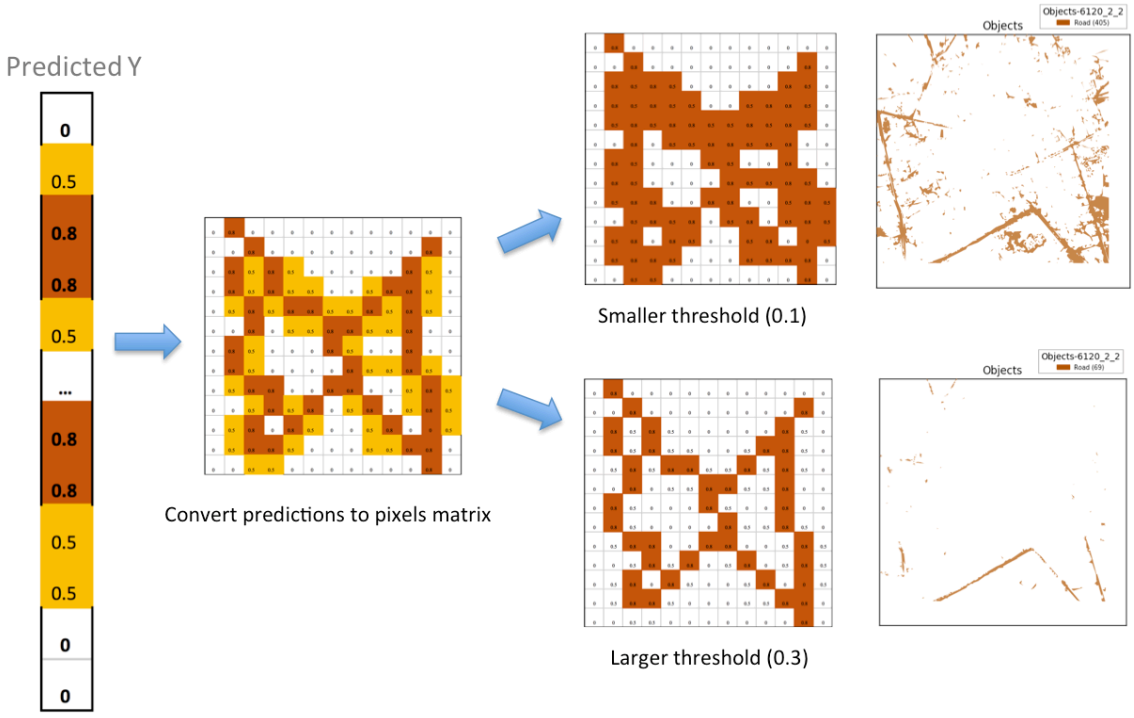
Step 1: GeoTiff images (3-band, 16-band) to pixels. The dataset also provides geo-coordinates for all images. With the help of these geo-coordinates, we project GeoTiff images into pixel matrices. Then we transform each matrix of a particular feature into one column. Take one 3-band image as an example, a pixel matrix with size 3391(h) x 3349(w) x 3(features) is transformed into an 11356459 (3391x3349) x 3 matrix. Each row represents a pixel. Each column is a feature. This matrix will serve as our X. Same for 16-band images. There X has 17 columns, corresponding to 17 features, but has fewer rows due to the fact that 16-band images have lower resolution.



Step 2: Extract WKT file to labels. We generate masks from WKT files containing (multi)polygon representation of all 10 object class. If the object appears in one pixel, we label it as 1, otherwise 0. Then we further arrange each mask into a single column. This serves as our training label Y.



Step 3: Pixels to predicted polygons. After we make a prediction from machine learning algorithms, we further tune the threshold parameter for each class in order to have the highest Jaccard index. Then we transform the matrix representation back to polygon with scaling and alignment. The polygons are output to WKT format using the python package “shapely”.



4. Methods

We use one image to train and another image for out-of-sample testing. This choice of number of images to train and test is reasonable given the number of classes we have to train, the image sizes and the limitation of computational resource.

We applied two methods in this project: XgBoost and logistic regression with stochastic gradient descent. We tested on 3-band and 16-band images respectively using *scikit-learn* packages and did parameter tunings based on our observations of the output. We also adjust the threshold values in rendering the prediction images.

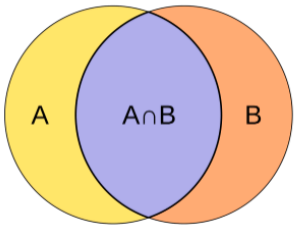
XGBoost. XGboost has been recently one of the best performed models for supervised learning. It uses gradient boosting decision tree algorithm that is known for its execution speed and performance. The most important advantage of using XGboost in this project is its speed in training the classifier. Considering the complexity of the features: there are 10 object classes and each is represented using a multi-polygons, the amount of work it takes to train a model on a specific feature is considerably large. So a classifier that trains fast is very desirable in our experiment. The good performance it gives is another reason that we choose it at first place.

Logistic Regression with Stochastic Gradient Descent (SGD). When the training set is enormous and no simple formulas exist, evaluating the sums of gradients becomes very expensive, because evaluating the gradient requires evaluating all the summand functions' gradients. To economize on the computational cost at every iteration, stochastic gradient descent samples a subset of summand functions at every step. Hence stochastic gradient descent is very effective when dealing with large scale minimization.

5. Model Evaluation

Instead of using AUC, we use Jaccard index as core metric to evaluate the model performance. The Jaccard Index for two regions A and B, also known as the "intersection over union", is defined as,

$$Jaccard = \frac{TP}{TP + FP + FN} = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$



where TP is the true positives area, FP is the false positives area, and FN is the false negatives area.

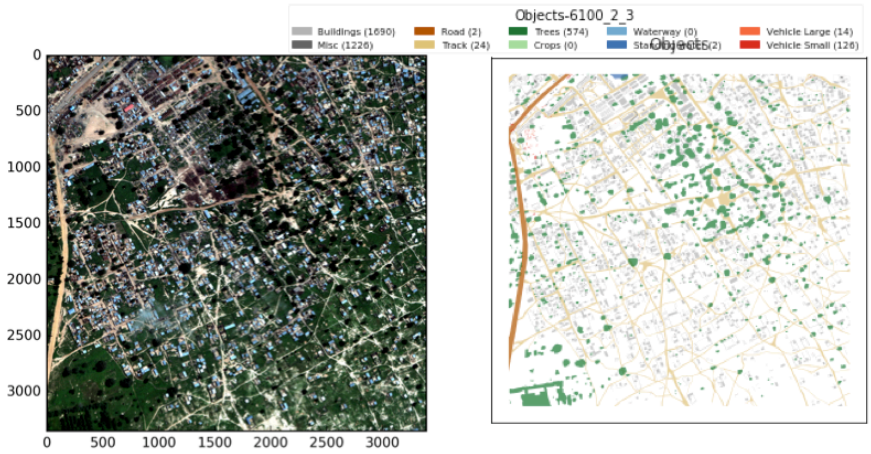
For each object class, we calculate the total TP, FP, and FN areas across all images. Then the Jaccard index is calculated for that class using total TP, FP, and FN. We can either evaluate our model performance with Jaccard index of every individual class or average Jaccard Indexes of all classed to evaluate the overall performance of the classifier.

6. Results

Note: Due to the limitation of our computational resource, we use one image to train and another image to test. One XGBoost training takes about 45 mins on Mac. One SGD training takes about 20 mins on Mac.

Training data:

Notice that in our testing data, crops and waterway are empty, so it is reasonable their Jaccard Index is 0.



Testing data:

The figures below visualize a comparison among original labels and two models' prediction.

3-band dataset:

In-sample predictions

Actual | SGD (logistic regression) | XGBoost

Out-of-sample predictions

Actual | SGD (logistic regression) | XGBoost

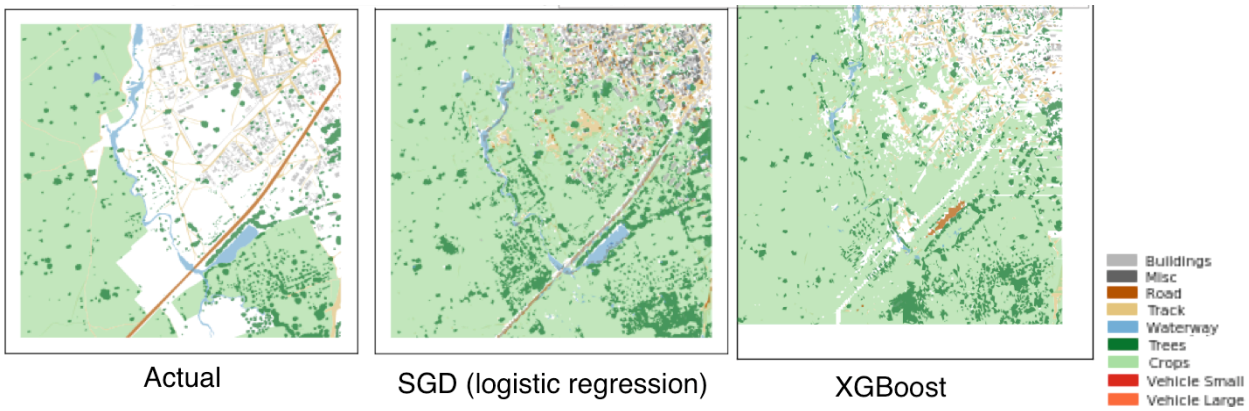
Jaccard Index (3-band)	SGD training	SGD testing	XGB training	XGB testing
1.Buildings	0.410	0.519	0.476	0.539
2.Misc. Manmade	0.0	4.924e-05	0.024	0.033
3.Road	0.0	0.0	0.506	0.00359
4.Track	0.156	0.185	0.199	0.232
5.Trees	0.378	0.453	0.393	0.511
6.Crops	0.525	0.0	0.539	0.0
7.Waterway	0.00186	0.0	0.580	0.0
8.Standing water	0.0	0.0	0.0	0.0
9.Vehicle Large	0.0	0.0	0.0	0.0
10.Vehicle Small	0.0	0.0	0.0	0.0
Average	0.14718	0.1446	0.271673	0.164823

As we can see, the Jaccard indexes for standing water, large and small vehicles are zero. This is because the testing image only has a few pixels labeling standing water. The sizes of large and small Vehicles are only comparable with one pixel, and there are not many of them. Hence the data sizes with respect to these classes are too small to fit a correct model.

Apart from this, SGD and XGB are both pretty good at classifying buildings, track, trees and crops. XGB outperforms SGD significantly in predicting waterway and roads. In general, XGB is a boosting machine, which can fit models with high nonlinearity. It also takes advantage of tree structure, so that we can estimate the weight of each weak classifier in close form. However, SGD does not give us close form solution of estimates and even not use the gradients of all individual loss functions. Therefore, XGB performs better than SGD.

16-band dataset:

In-sample predictions



Actual SGD (logistic regression) XGBoost

Out-of-sample predictions

Actual SGD (logistic regression) XGBoost

Jaccard Index (16-band)	SGD training	SGD testing	XGB training	XGB testing
1.Buildings	0.357	0.478	0.0934	0.0465
2.Misc. Manmade	0.0755	0.0139	0.0209	0.00704
3.Road	0.150	0.0113	0.140	7.718e-05
4.Track	0.219	0.169	0.206	0.165
5.Trees	0.340	0.455	0.290	0.317
6.Crops	0.532	0.0	0.584	0.0
7.Waterway	0.620	0.0	0.230	0.0
8.Standing water	0.0	0.0	0.483	0.0
9.Vehicle Large	0.0	0.0	0.0	0.0
10.Vehicle Small	0.0	0.0	0.0	0.0
Average	0.22933	0.1409	0.204807	0.066943

The 16-band dataset provides us more information on geo-objects because it captures energies that span wider wavelengths. Therefore, as we can see, standing water can be characterized with XGB. Predictions on Misc. Manmade Structures and crops also have good improvement.

7. Future Improvement

In this project, we proposed a pixel-based approach that does not consider interaction between pixels. It is important to note that the depth of our attempts in this project was limited by the time and computing resource that are accessible. But we did obtain promising results from the two models that we selected and get valuable insights from the difference in performance subject to the features and images. This tells us the trade-off that is seen in every model. In terms of data-wise improvement, in the future, given faster processor and larger capacity in memory, we can train using classifiers that are slower than XgBoost and SGD and more insights from the relative performance among different models can be achieved.

There are also some method-wise improvement that we can make in the future. We can merge the 3-band and 16-band datasets together by model ensemble. A more precise treatment will process images by segmentation first, then extract features like mean, variance, geometry shapes and texture from each segment. Convolutional Neural network or U-network can also be used to train and improve prediction. This object-based approach may save computational effort and enhance the accuracy.