Supplementary Materials

A hybrid data balancing method for classification of imbalanced training data within Google Earth Engine: Case studies from mountainous regions

Amin Naboureh, Ainong Li*, Jinhu Bian, Guangbian Lei, and Meisam Amani

* Corresponding author.

E-mail addresses: amin.nabore@mails.ucas.ac.cn (A. Naboureh), ainongli@imde.ac.cn (A. Li), bianjinhu@imde.ac.cn (J. Bian), leiguangbin@imde.ac.cn (G, Lei), meisam.amani@woodplc.com (M. Amani).

Content:

1. Scripts for investigating the role of different complementary information on the accuracies of MLC classes.

2. Scripts for implementing PROSRUS for LC mapping using time-series Landsat images in the GEE platform.

Scripts for investigating the role of different complementary information on the accuracies of MLC classes:

```
var bandsAll=  ['B2', 'B3', 'B4', 'B5', 'B6', 'B7', 'slope','elevation','aspect',
            'NDVI','NDBI','NDWI','SAVI']


var bands=  ee.List([['B2', 'B3', 'B4', 'B5', 'B6', 'B7'],
                    ['B2', 'B3', 'B4', 'B5', 'B6', 'B7', 'slope','elevation','aspect'],
                    ['B2', 'B3', 'B4', 'B5', 'B6', 'B7', 'NDVI','NDBI','NDWI','SAVI'],
                    ['B2', 'B3', 'B4', 'B5', 'B6', 'B7', 'slope','elevation','aspect', 'NDVI','NDBI','NDWI','SAVI']])
```

Importing Landsat-8 images as an ImageCollection

```
var LL8 = ee.ImageCollection('LANDSAT/LC08/C01/T1_SR')
    .filterBounds(geometry)
    .filterDate('2019-05-01', '2019-10-01')
    .filterMetadata('CLOUD_COVER', 'less_than', 10)
    .map(function(image){return image.clip(geometry)});
```

Calculating complementary data

```
var addIndices = function(image) {
  var NDVI = image.normalizedDifference(['B5', 'B4']).rename('NDVI');
  var NDBI = image.normalizedDifference(['B6', 'B5']).rename('NDBI');
  var NDWI = image.normalizedDifference(['B3', 'B5']).rename('NDWI');
  var SAVI = image.expression(
    '(1 + L) * (NIR - RED) / (NIR + RED + L)', {
      'NIR':image.select('B5'),
      'RED':image.select('B4'),
      'L': 0.428}).rename ('SAVI');
  return image.addBands(NDVI).addBands(NDBI).addBands(NDWI).addBands(SAVI);
};
var srtm30 = ee.Image('USGS/SRTMGL1_003').clip(geometry);
var slope30 = ee.Terrain.slope(srtm30);
var aspect30 = ee.Terrain.aspect(srtm30);
```

Adding complementary information to image collection

```
var LL82 = LL8.map(addIndices)
var LL821= LL8.median().addBands(srtm30.select('elevation')).addBands(slope30).addBands(aspect30)
var LL83 = LL82.median().addBands(srtm30.select('elevation')).addBands(slope30).addBands(aspect30);

var DataOrig = table.randomColumn("random",12345);
var DataTrain = DataOrig.filter(ee.Filter.lte('random',0.5))
var DataValid = DataOrig.filter(ee.Filter.gt('random',0.5))

var DataSamp = LL83.select(bandsAll).sampleRegions({
  collection: DataTrain,
  properties: ['landcover','classTY'],
  scale: 30
});

var accuracyTable = bands.map(function(SBands){
        var classifier = ee.Classifier.smileRandomForest({numberOfTrees: 500,variablesPerSplit: 4})
        .train({
        features: DataSamp,
        classProperty: 'landcover',
        inputProperties: SBands,
        });

        var classified = LL83.select(SBands).classify(classifier);

        var validData = classified.sampleRegions({
        collection: DataValid,
        properties: ['landcover'],
        scale: 30
        });

        var errorMatrix = validData.errorMatrix('landcover', 'classification');

        return  ee.Feature(null, {
   "SCC": SBands,
    "OA": errorMatrix.accuracy(),
    "UA": errorMatrix.consumersAccuracy().project([1]),
    "PA": errorMatrix.producersAccuracy().project([0]),
    "KAP": errorMatrix.kappa()
  })
});
Export.table.toDrive({
  collection: ee.FeatureCollection(accuracyTable),
  description: 'LL8',
  folder: "AminLandsat8",
  fileNamePrefix: "LL8",
  fileFormat: 'CSV'
});
```

Scripts for implementing PROSRUS for LC mapping using time-series Landsat images in the GEE platform:

```
var bandsAll=  ['B2', 'B3', 'B4', 'B5', 'B6', 'B7', 'slope','elevation','aspect', 'NDVI','NDBI','NDWI','SAVI']
```

Importing Landsat-8 images as an ImageCollection

```
var LL8 = ee.ImageCollection('LANDSAT/LC08/C01/T1_SR')
    .filterBounds(geometry)
    .filterDate('2019-05-01', '2019-10-01')
    .filterMetadata('CLOUD_COVER', 'less_than', 10)
    .map(function(image){return image.clip(geometry)});
```

Calculating spectral indices

```
var addIndices = function(image) {
  var NDVI = image.normalizedDifference(['B5', 'B4']).rename('NDVI');
  var NDBI = image.normalizedDifference(['B6', 'B5']).rename('NDBI');
  var NDWI = image.normalizedDifference(['B3', 'B5']).rename('NDWI');
  var SAVI = image.expression(
    '(1 + L) * (NIR - RED) / (NIR + RED + L)', {
      'NIR':image.select('B5'),
      'RED':image.select('B4'),
      'L': 0.428}).rename ('SAVI');
  return image.addBands(NDVI).addBands(NDBI).addBands(NDWI).addBands(SAVI);
};
```

Importing SRTM data and Generating topographic data

```
var srtm30 = ee.Image('USGS/SRTMGL1_003').clip(geometry);
var slope30 = ee.Terrain.slope(srtm30);
var aspect30 = ee.Terrain.aspect(srtm30);
```

Adding optimal features to image collection

```
var LL82 = LL8.map(addIndices)
var LL83 = LL82.median().addBands(srtm30.select('elevation')).addBands(slope30).addBands(aspect30);
var DataOrig = table.randomColumn("random",12345);
var DataTrain = DataOrig.filter(ee.Filter.lte('random',0.5))
var DataValid = DataOrig.filter(ee.Filter.gt('random',0.5))

var DataSamp = LL83.select(bandsAll).sampleRegions({
  collection: DataTrain,
  properties: ['landcover','classTY'],
  scale: 30
});
```

Splitting data to minority, majority and middle groups, and introducing 200 different fractions for balancing majority and minority classes (middle classes stay unchanged)

```
var DataMinority = DataSamp.filter(ee.Filter.inList('classTY',
                    ee.List(['water','wetland','snow','grassland']))).randomColumn("random");

var DataMajority = DataSamp.filter(ee.Filter.inList('classTY',
                    ee.List(['cropland','bareland']))).randomColumn("random");

var DataMidle = DataSamp.filter(ee.Filter.inList('classTY',
                    ee.List(['artificial']))).randomColumn("random");

var percc = ee.List([[0.1,0.1],[0.2,0.1],[0.3,0.1],[0.4,0.1],[0.5,0.1],[0.6,0.1],[0.7,0.1],[0.8,0.1],[0.9,0.1],[1,0.1],
[0.1,0.2],[0.2,0.2],[0.3,0.2],[0.4,0.2],[0.5,0.2],[0.6,0.2],[0.7,0.2],[0.8,0.2],[0.9,0.2],[1,0.2],
[0.1,0.3],[0.2,0.3],[0.3,0.3],[0.4,0.3],[0.5,0.3],[0.6,0.3],[0.7,0.3],[0.8,0.3],[0.9,0.3],[1,0.3],
[0.1,0.4],[0.2,0.4],[0.3,0.4],[0.4,0.4],[0.5,0.4],[0.6,0.4],[0.7,0.4],[0.8,0.4],[0.9,0.4],[1,0.4],
[0.1,0.5],[0.2,0.5],[0.3,0.5],[0.4,0.5],[0.5,0.5],[0.6,0.5],[0.7,0.5],[0.8,0.5],[0.9,0.5],[1,0.5],
[0.1,0.6],[0.2,0.6],[0.3,0.6],[0.4,0.6],[0.5,0.6],[0.6,0.6],[0.7,0.6],[0.8,0.6],[0.9,0.6],[1,0.6],
[0.1,0.7],[0.2,0.7],[0.3,0.7],[0.4,0.7],[0.5,0.7],[0.6,0.7],[0.7,0.7],[0.8,0.7],[0.9,0.7],[1,0.7],
[0.1,0.8],[0.2,0.8],[0.3,0.8],[0.4,0.8],[0.5,0.8],[0.6,0.8],[0.7,0.8],[0.8,0.8],[0.9,0.8],[1,0.8],
[0.1,0.9],[0.2,0.9],[0.3,0.9],[0.4,0.9],[0.5,0.9],[0.6,0.9],[0.7,0.9],[0.8,0.9],[0.9,0.9],[1,0.9],
[0.1,1],[0.2,1],[0.3,1],[0.4,1],[0.5,1],[0.6,1],[0.7,1],[0.8,1],[0.9,1],[0.1,1],
[0.1,1.1],[0.1,1.2],[0.1,1.3],[0.1,1.4],[0.1,1.5],[0.1,1.6],[0.1,1.7],[0.1,1.8],[0.1,1.9],[0.1,2],
[0.2,1.1],[0.2,1.2],[0.2,1.3],[0.2,1.4],[0.2,1.5],[0.2,1.6],[0.2,1.7],[0.2,1.8],[0.2,1.9],[0.2,2],
[0.3,1.1],[0.3,1.2],[0.3,1.3],[0.3,1.4],[0.3,1.5],[0.3,1.6],[0.3,1.7],[0.3,1.8],[0.3,1.9],[0.3,2],
[0.4,1.1],[0.4,1.2],[0.4,1.3],[0.4,1.4],[0.4,1.5],[0.4,1.6],[0.4,1.7],[0.4,1.8],[0.4,1.9],[0.4,2],
[0.5,1.1],[0.5,1.2],[0.5,1.3],[0.5,1.4],[0.5,1.5],[0.5,1.6],[0.5,1.7],[0.5,1.8],[0.5,1.9],[0.5,2],
[0.6,1.1],[0.6,1.2],[0.6,1.3],[0.6,1.4],[0.6,1.5],[0.6,1.6],[0.6,1.7],[0.6,1.8],[0.6,1.9],[0.6,2],
[0.7,1.1],[0.7,1.2],[0.7,1.3],[0.7,1.4],[0.7,1.5],[0.7,1.6],[0.7,1.7],[0.7,1.8],[0.7,1.9],[0.7,2],
[0.8,1.1],[0.8,1.2],[0.8,1.3],[0.8,1.4],[0.8,1.5],[0.8,1.6],[0.8,1.7],[0.8,1.8],[0.8,1.9],[0.8,2],
[0.9,1.1],[0.9,1.2],[0.9,1.3],[0.9,1.4],[0.9,1.5],[0.9,1.6],[0.9,1.7],[0.9,1.8],[0.9,1.9],[0.9,2],
[1,1.1], [1,1.2], [1,1.3], [1,1.4], [1,1.5], [1,1.6], [1,1.7], [1,1.8], [1,1.9], [1,2]
])

var accuracyTable = percc.map(function(perxx){
```

Random under sampling

```
        var DataMajority2 = DataMajority.filter(ee.Filter.lte('random',ee.List(perxx).get(0)))
```

Random over sampling

```
        var DataMinority2 =
DataMinority.merge(DataMinority.filter(ee.Filter.lte('random',ee.List(perxx).get(1))))
```

Merging them together to build final dataset

```
        var Data2 = DataMinority2.merge(DataMajority2.merge(DataMidle))


        var classifier = ee.Classifier.smileRandomForest({numberOfTrees: 500,variablesPerSplit: 4})
        .train({
        features: Data2,
        classProperty: 'landcover',
        inputProperties: bandsAll,
        });


        var classified = LL83.select(bandsAll).classify(classifier);


        var validData = classified.sampleRegions({
        collection: DataValid,
        properties: ['landcover'],
        scale: 30
        });
        var errorMatrix = validData.errorMatrix('landcover', 'classification');

        return  ee.Feature(null, {
    "SCC": perxx,
    "OA": errorMatrix.accuracy(),
    "UA": errorMatrix.consumersAccuracy().project([1]),
    "PA": errorMatrix.producersAccuracy().project([0]),
    "KAP": errorMatrix.kappa()
  })

});

print(ee.FeatureCollection(accuracyTable).toList(200).get(10))

Export.table.toDrive({
  collection: ee.FeatureCollection(accuracyTable),
  description: 'LL8_FracS1',
  folder: "AminLandsat8",
  fileNamePrefix: "LL8_FracS1",
  fileFormat: 'CSV'
});
```