Lecture 07

GEE Image Classification:

supervised & unsupervised classification

2024-04-15

Sébastien Valade

Previous lecture:

GEE image manipulation:

⇒ band arithmetic (spectral indices), thresholds, masks, reducers

Today:

GEE image classification:

⇒ assign a *class* to each pixel in the image

⇒ involves: selection of *training data* (supervised classification), selection of a *classifier*, and *training* of the classifier

Previous lecture:

GEE image manipulation:

⇒ band arithmetic (spectral indices), thresholds, masks, reducers

Today:

GEE image classification:

⇒ assign a *class* to each pixel in the image

⇒ involves: selection of *training data* (supervised classification), selection of a *classifier*, and *training* of the classifier

Table of Contents

Image Classification:

⇒ **Image classification** in remote sensing, is a task that involves categorizing all pixels in an image into a finite number of **classes**

⇒ its most common application is *land cover & land use classification*, whereby all pixels are categorized in predefined land cover classes (e.g., water, forest, urban, etc.)

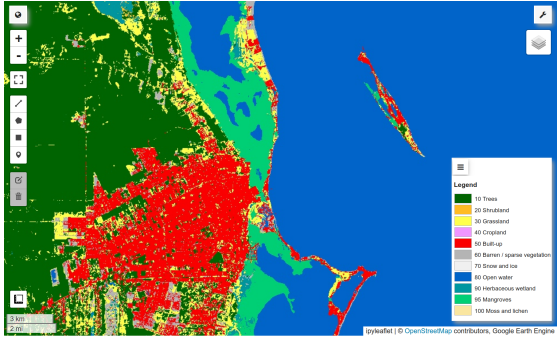
Image Classification:

⇒ **Image classification** in remote sensing, is a task that involves categorizing all pixels in an image into a finite number of **classes**

⇒ its most common application is *land cover & land use classification*, whereby all pixels are categorized in predefined land cover classes (e.g., water, forest, urban, etc.)

Natural color image

Land cover classification (ESA World Cover)



⇒ Several techniques exist to classify the image (Li et al. 2014, Kavzoglu et al. 2009 (ed2), 2025 (ed3)):

- **Pixel-based techniques**: classify each pixel individually based on spectral properties
- **Object-based techniques**: classify groups of pixels (objects) based on spectral, spatial, and cor
- **Deep Learning techniques**: use neural networks to learn features and classify images

⇒ These techniques generally fall in the broad field of **Machine Learning**, whereby *statistical algorithms are able to learn from data, and generalize to unseen data*, thus performing tasks (i.e., classification) without explicit instructions

⇒ The learning approach can be either **supervised** or **unsupervised**:

- **Supervised classification**: requires *training data* (labeled data)
- **Unsupervised classification**: does *not* require *training data*

⇒ In this lecture, we will focus on the *implementation in GEE* of **pixel-based classification** using both **supervised and unsupervised** techniques

⇒ Several techniques exist to classify the image (Li et al. 2014, Kavzoglu et al. 2009 (ed2), 2025 (ed3)):

- **Pixel-based techniques**: classify each pixel individually based on spectral properties
- **Object-based techniques**: classify groups of pixels (objects) based on spectral, spatial, and cor
- **Deep Learning techniques**: use neural networks to learn features and classify images

⇒ These techniques generally fall in the broad field of **Machine Learning**, whereby *statistical algorithms are able to learn from data, and generalize to unseen data*, thus performing tasks (i.e., classification) without explicit instructions

⇒ The learning approach can be either **supervised** or **unsupervised**:

- **Supervised classification**: requires *training data* (labeled data)
- **Unsupervised classification**: does *not* require *training data*

⇒ In this lecture, we will focus on the *implementation in GEE* of **pixel-based classification** using both **supervised and unsupervised** techniques

⇒ Several techniques exist to classify the image (Li et al. 2014, Kavzoglu et al. 2009 (ed2), 2025 (ed3)):

- **Pixel-based techniques**: classify each pixel individually based on spectral properties
- **Object-based techniques**: classify groups of pixels (objects) based on spectral, spatial, and cor
- **Deep Learning techniques**: use neural networks to learn features and classify images

⇒ These techniques generally fall in the broad field of **Machine Learning**, whereby *statistical algorithms are able to learn from data, and generalize to unseen data*, thus performing tasks (i.e., classification) without explicit instructions

⇒ The learning approach can be either **supervised** or **unsupervised**:

- **Supervised classification**: requires *training data* (labeled data)
- **Unsupervised classification**: does *not* require *training data*

⇒ In this lecture, we will focus on the *implementation in GEE* of **pixel-based classification** using both **supervised and unsupervised** techniques

⇒ Several techniques exist to classify the image (Li et al. 2014, Kavzoglu et al. 2009 (ed2), 2025 (ed3)):

- **Pixel-based techniques**: classify each pixel individually based on spectral properties
- **Object-based techniques**: classify groups of pixels (objects) based on spectral, spatial, and cor
- **Deep Learning techniques**: use neural networks to learn features and classify images

⇒ These techniques generally fall in the broad field of **Machine Learning**, whereby *statistical algorithms are able to learn from data, and generalize to unseen data*, thus performing tasks (i.e., classification) without explicit instructions

⇒ The learning approach can be either **supervised** or **unsupervised**:

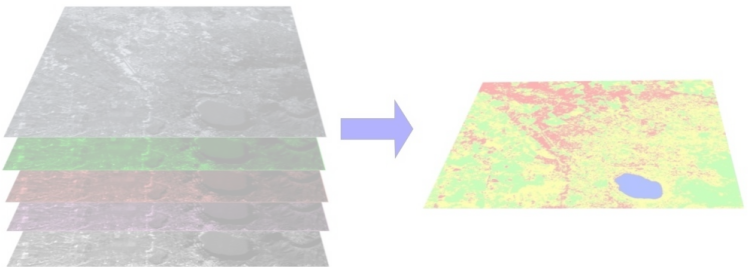
- **Supervised classification**: requires *training data* (labeled data)
- **Unsupervised classification**: does *not* require *training data*

⇒ In this lecture, we will focus on the *implementation in GEE* of **pixel-based classification** using both **supervised and unsupervised** techniques

Pixel-based classification:

⇒ With the pixel-based classification method, pixels are classified *individually*, i.e. without spatial context from the neighboring pixels

⇒ Classification is based on the *pixel spectral properties* (i.e., reflectance values in each band), and/or on their transformations (i.e., spectral indices, principal components, etc.)



Pixel-based classification:

⇒ With the pixel-based classification method, pixels are classified *individually*, i.e. without spatial context from the neighboring pixels

⇒ Classification is based on the *pixel spectral properties* (i.e., reflectance values in each band), and/or on their transformations (i.e., spectral indices, principal components, etc.)

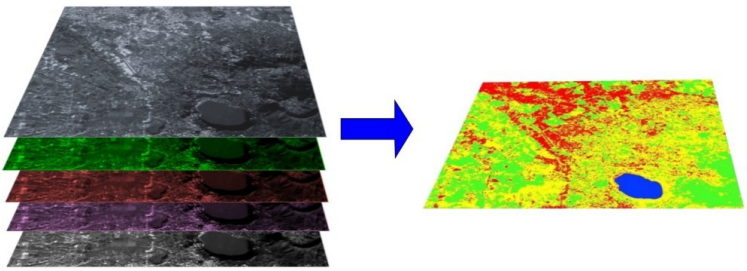
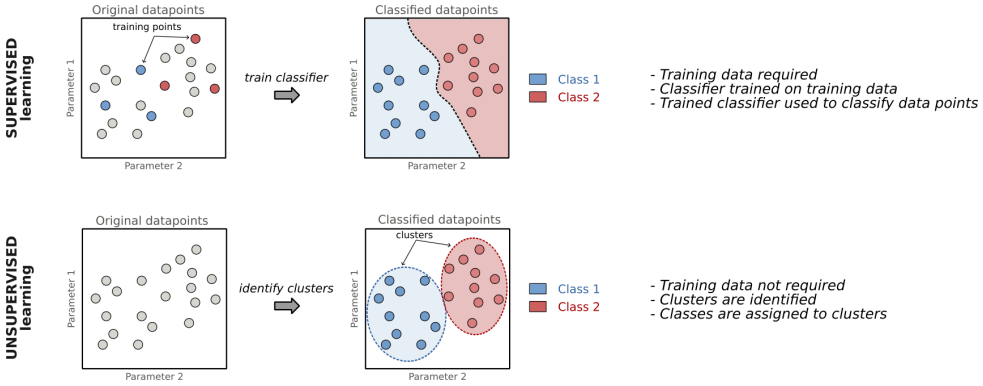


Image Source: link

Pixel-based classification:

⇒ **Supervised** vs. **Unsupervised** learning:



Workflow of supervised classification

1. **Select image to classify**
2. **Collect training data**
   ⇒ a *training dataset* is collection of *labeled data*, that is input-output pairs, where the *input* is the data provided to the model, and the *output* is the corresponding target or label that the model is expected to predict.
3. **Select prediction bands**
   ⇒ *prediction bands* correspond to the bands used to extract the spectral information to classify each pixels.

   EX: use the bands provided in the product (including bands in the visible range, and possibly in the infrared range), and possibly derived bands (e.g., spectral indices, or principal components derived from a PCA analysis)
4. **Select a classifier and train it on the training data**
   ⇒ a *classifier* is a *statistical model* that learns to map input pixels to output classes based on the provided labels.

   EX: commonly used classifiers are Random Forest, Support Vector Machine, K-Nearest Neighbors, CART, etc.
5. **Classify the image using the *trained classifier***
   The trained classifier is applied to the image, and each pixel is assigned a class based on the classifier's prediction.

Workflow of supervised classification

1. **Select image to classify**
2. **Collect training data**
   ⇒ a *training dataset* is collection of *labeled data*, that is input-output pairs, where the *input* is the data provided to the model, and the *output* is the corresponding target or label that the model is expected to predict.
3. **Select prediction bands**
   ⇒ *prediction bands* correspond to the bands used to extract the spectral information to classify each pixels.

   EX: use the bands provided in the product (including bands in the visible range, and possibly in the infrared range), and possibly derived bands (e.g., spectral indices, or principal components derived from a PCA analysis)
4. **Select a classifier and train it on the training data**
   ⇒ a *classifier* is a *statistical model* that learns to map input pixels to output classes based on the provided labels.

   EX: commonly used classifiers are Random Forest, Support Vector Machine, K-Nearest Neighbors, CART, etc.
5. **Classify the image using the *trained classifier***
   The trained classifier is applied to the image, and each pixel is assigned a class based on the classifier's prediction.

Workflow of supervised classification

1. **Select image to classify**

2. **Collect training data**
   ⇒ a *training dataset* is collection of *labeled data*, that is input-output pairs, where the *input* is the data provided to the model, and the *output* is the corresponding target or label that the model is expected to predict.

3. **Select prediction bands**
   ⇒ *prediction bands* correspond to the bands used to extract the spectral information to classify each pixels.

   EX: use the bands provided in the product (including bands in the visible range, and possibly in the infrared range), and possibly derived bands (e.g., spectral indices, or principal components derived from a PCA analysis)

4. **Select a classifier and train it on the training data**
   ⇒ a *classifier* is a *statistical model* that learns to map input pixels to output classes based on the provided labels.

   EX: commonly used classifiers are Random Forest, Support Vector Machine, K-Nearest Neighbors, CART, etc.

5. **Classify the image using the *trained classifier***
   The trained classifier is applied to the image, and each pixel is assigned a class based on the classifier's prediction.

Workflow of supervised classification

1. **Select image to classify**

2. **Collect training data**
   ⇒ a *training dataset* is collection of *labeled data*, that is input-output pairs, where the *input* is the data provided to the model, and the *output* is the corresponding target or label that the model is expected to predict.

3. **Select prediction bands**
   ⇒ *prediction bands* correspond to the bands used to extract the spectral information to classify each pixels.

   EX: use the bands provided in the product (including bands in the visible range, and possibly in the infrared range), and possibly derived bands (e.g., spectral indices, or principal components derived from a PCA analysis)

4. **Select a classifier and train it on the training data**
   ⇒ a *classifier* is a *statistical model* that learns to map input pixels to output classes based on the provided labels.

   EX: commonly used classifiers are Random Forest, Support Vector Machine, K-Nearest Neighbors, CART, etc.

5. **Classify the image using the *trained classifier***
   The trained classifier is applied to the image, and each pixel is assigned a class based on the classifier's prediction.

Workflow of supervised classification

1. **Select image to classify**

2. **Collect training data**
   ⇒ a *training dataset* is collection of *labeled data*, that is input-output pairs, where the *input* is the data provided to the model, and the *output* is the corresponding target or label that the model is expected to predict.

3. **Select prediction bands**
   ⇒ *prediction bands* correspond to the bands used to extract the spectral information to classify each pixels.

   EX: use the bands provided in the product (including bands in the visible range, and possibly in the infrared range), and possibly derived bands (e.g., spectral indices, or principal components derived from a PCA analysis)

4. **Select a classifier and train it on the training data**
   ⇒ a *classifier* is a *statistical model* that learns to map input pixels to output classes based on the provided labels.

   EX: commonly used classifiers are Random Forest, Support Vector Machine, K-Nearest Neighbors, CART, etc.

5. **Classify the image using the *trained classifier***
   The trained classifier is applied to the image, and each pixel is assigned a class based on the classifier's prediction.

Step 1: select the image to classify

```
# Select region of interest (lon, lat)
roi = ee.Geometry.Point(-86.85, 21.17)

# Filter the Sentinel-2 collection and select the least cloudy image
image = (ee.ImageCollection('COPERNICUS/S2_SR_HARMONIZED')
            .filterBounds(roi)
            .filterDate('2020-01-01', '2024-10-01')
            .filter(ee.Filter.lt('CLOUD_COVERAGE_ASSESSMENT', 10))
            .sort("CLOUD_COVERAGE_ASSESSMENT")
            .first()
            )
```

Step 2: collect training samples

NB: the built-in tool in `GEEMAP` called "Collect training samples" is suffering bugs in the Google Colab environment (in particular, it does not store the "property" & "value" fields in the `user_rois` object). The approach suggested below is a workaround to collect training samples.

⇒ For each *land cover class* (see table below), repeat the following:

1. *Select* training pixels using the `Draw a marker` tool on the interactive map
2. *Convert* the collected samples (`Map.user_rois`) to a `GeoPandasDataframe`, and create a new column called "class" storing the numeric value corresponding to the land cover class
3. *Export* the `GeoPandasDataframe` to a `GeoJson` file on your Google Drive for future use
4. *Combine* all collected samples in a unique `FeatureCollection`: once steps 1-3 have been performed for each class, *import* all `GeoJson` files and combine in a unique `FeatureCollection`, which will store all the collected samples along with their class

⇒ In this example, we will be using the following *land cover classes* (feel free to adapt to your image):

Class	*Description*
0	Vegetation
1	Urban
2	Water
3	Grassland

Step 2: collect training samples

NB: the built-in tool in `GEEMAP` called "Collect training samples" is suffering bugs in the Google Colab environment (in particular, it does not store the "property" & "value" fields in the `user_rois` object). The approach suggested below is a workaround to collect training samples.

⇒ For each *land cover class* (see table below), repeat the following:

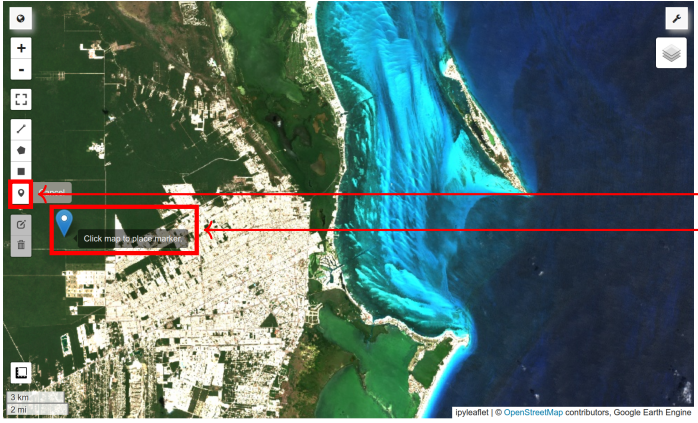
1. *Select* training pixels using the `Draw a marker` tool on the interactive map
2. *Convert* the collected samples (`Map.user_rois`) to a `GeoPandasDataframe`, and create a new column called "class" storing the numeric value corresponding to the land cover class
3. *Export* the `GeoPandasDataframe` to a `GeoJson` file on your Google Drive for future use
4. *Combine* all collected samples in a unique `FeatureCollection`: once steps 1-3 have been performed for each class, *import* all `GeoJson` files and combine in a unique `FeatureCollection`, which will store all the collected samples along with their class

⇒ In this example, we will be using the following *land cover classes* (feel free to adapt to your image):

Class	*Description*
0	Vegetation
1	Urban
2	Water
3	Grassland

Step 2: collect training samples

1. *Select* training pixels using the `Draw a marker` tool on the interactive map



a. select tool "Draw a marker"

b. pick on pixel corresponding to the "land cover class" (e.g., vegetation)

Step 2: collect training samples

2. *Convert* the collected samples (`Map.user_rois`) to a `GeoPandasDataframe`, and create a new column called "class" storing the numeric value corresponding to the land cover class

```
# === Convert collected training samples to GeoPandasDataframe

# Set class of the collected smaples
class_value, class_label = 0, 'vegetation'

# Convert ROIs to geodataframe
gdf = geemap.ee_to_gdf(Map.user_rois)

# Add class column (will be saved in "properties" field in json)
gdf['class'] = class_value
```

Step 2: collect training samples

3. *Export* the `GeoPandasDataframe` to a `GeoJson` file on your Google Drive for future use

NB: to mount your Google Drive in the Google Colab jupyter environment, click on the "Folder" icon in the vertical tool on the left-hand side of the screen, then click on the icon representing a folder with the Google Drive logo inside.

```
# === Export GeoDataFrame to GeoJson

p_geojson = '/content/drive/MyDrive/training_samples/'
f_geojson = 'training_samples_class-{}_{}.geojson'.format(class_value, class_label)
gdf.to_file(p_geojson+f_geojson, driver='GeoJSON')
```

Step 2: collect training samples

4. *Combine* all collected samples in a unique `FeatureCollection`: once steps 1-3 have been performed for each class, *import* all `GeoJson` files and combine in a unique `FeatureCollection`, which will store all the collected samples along with their class

```
# Import GeoJson as individual feature collections
p_geojson = '/content/drive/MyDrive/training_samples/'
vegetation = geemap.geojson_to_ee(p_geojson+'training_samples_class-0_vegetation.geojson')
urban = geemap.geojson_to_ee(p_geojson+'training_samples_class-1_urban.geojson')
water = geemap.geojson_to_ee(p_geojson+'training_samples_class-2_water.geojson')
grass = geemap.geojson_to_ee(p_geojson+'training_samples_class-3_grass.geojson')

# Display the FeatureCollections (optional)
Map.addLayer(vegetation, {'color':'green'}, "vegetation")
Map.addLayer(urban, {'color':'red'}, "urban")
Map.addLayer(water, {'color':'blue'}, "water")
Map.addLayer(grass, {'color':'orange'}, "grass")

# Combine all feature collections in a unique training FeatureCollection
trainingFeatures = (ee.FeatureCollection([
    vegetation, urban, water, grass
    ]).flatten())
```

Step 3: select prediction bands

⇒ *Select the prediction bands* to be used for the classification. In the example below, we only use bands from the Sentinel-2 product, but you could also use derived bands (e.g., spectral indices, principal components, etc.)

⇒ *Sample the training data points* using these bands: use the sampleRegions method to sample the bands information at each training data point

NB: use `display(classifierTraining)` to visualize the content of the object: by expanding the first feature (i.e. first training data point), you should see 1) the data of the prediction bands for that pixel, and 2) the class information you assigned to it.

```
# Select prediction bands (Sentinel-2 product)
predictionBands = [
    'B1', 'B2', 'B3', 'B4', 'B5', 'B6', 'B7', 'B8', 'B8A', 'B9', 'B11', 'B12'
]

# Sample bands at each training point
classifierTraining = (image.select(predictionBands)
                      .sampleRegions(
                          collection=trainingFeatures,
                          properties=['class'],
                          scale=20
                      ))
```

Step 4: select and train the classifier

⇒ The classifier is a *statistical model* which contains mathematical rules linking the pixel's spectral information to its class

⇒ Selecting the appropriate classifier can be tricky. Various classifiers are available in GEE, e.g.: Random Forest, CART (Classification and Regression Trees), etc.

NB: use the command `display(classifier)` to display the basic characteristics of the classifier (bands, properties, and classifier name), and the command `display(classifier.explain())` to display the decision rules of the trained classifier (for the CART classifier, see the property "tree").

```
# Instantiate and train a CART classifier
classifier = (ee.Classifier.smileCart()
                .train(features=classifierTraining,
                    classProperty='class',
                    inputProperties=predictionBands))
```

Step 5: classify the image using the trained classifier

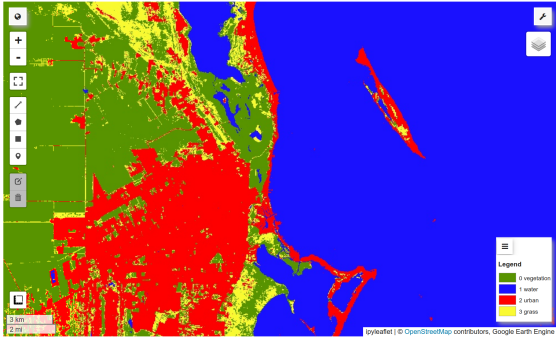
⇒ once the classifier has been trained, you can use it to predict the class of each pixel in the image

```
# Classify the image using the trained classifier
classified = image.select(predictionBands).classify(classifier)

# Visualize the classified image and add custom legend to the map
classificationVis = {'min': 0, 'max': 3, 'palette': ['589400', 'ff0000', '1a11ff', 'd0741e']}
Map.addLayer(classified, classificationVis, 'CART classified')

legend_dict = {
    '0 vegetation': '589400',
    '1 water': '1a11ff',
    '2 urban': 'ff0000',
    '3 grass': 'd0741e'
}
Map.add_legend(legend_title="Land Cover Classification", legend_dict=legend_dict)
Map
```

Classification results

Natural color image

Land cover classification (CART)



Next class

Next class!