Project Matsu:
Building Training Sets for the Water Classifier Analytic

Open Cloud Consortium
Open Data

January 1, 2013
Version 2.0

Materials for Matsu

Matsu is a public project. Code and documentation are maintained on Github. The Open Cloud Consortium has an account on Github where projects may be released from time-to-time. The official Open Cloud Consortium account is `http://github.com/opencloucdonsortium`. The project page is `http://github.com/opencloudconsortium/matsu-project`.

Information about the computing resources available from the OCC can be found at `http://opencloudconsortium.org`.

Summary

Matsu is a project of the Open Cloud Consortium's Open Science Data Cloud (OSDC). A goal of the Matsu project is to use an open source cloud-based infrastructure to make high quality satellite image data accessible through an Open Geospatial Consortium (OGC)-compliant Web Map Service (WMS). Additionally, a workflow has been created using MapReduce which generalizes image processing and can be used for running distributed analytics over satellite images.

Matsu incorporates the following open source technologies and applications:

- GlusterFS;
- Hadoop and the Hadoop Streaming Interface;
- Avro;
- Accumulo;
- R;
- Python;
- Augustus; and
- PMML.

Contributors to the project have included the National Aeronautics and Space Administration (NASA), TexelTek, the University of Chicago, the Laboratory for Advanced Computing (LAC), and Open Data.

More information about the OSDC can be found at `https://www.opensciencedatacloud.org`.

Contents

1 Introduction

The water classifier uses machine learning techniques to classify satellite imagery from known, labeled data. Pixels are classified based on their radiance values accross the range of spectral bands of the instrument. A set of known, labeled data is required to classify the images using supervised machine learning techniques. Training data sets for the alogrithm are also derived from satellite image data, and the labels correrspond to various land cover types.

Typically, the radiance values are aggregated in some way to construct the features that are used for classification. The sample training set on the GitHub site uses nine features for each pixel, corresponding to the reflectivity across nine groups of spectral bands in the 433 nm to 2350 nm wavelength range. Pixels were classified into four distinct land cover types.

The hand labeling process used was to:

1. Extract pixels (taking the largest contiguous set of pixels possible) from portions of images representing homogeneous land cover type;
2. Create a table where each row corresponds to a pixel; and
3. Label the row with an integer corresponding to its land cover type.

If a user wishes to use the water classifier with images from instruments other than EO-1's Hyperion or ALI, or if a different selection of bands is required, it is necessary to build a new training set. This may also require a new library of labeled data.

1.1 Creating a Training set

Classification of images using supervised machine learning requires a set of training data in order to construct the model which can be used to classify other image data.

A first step in classifying an image using radiance values is to decide on how many dimensions or features to use in the classification. A feature may be composed of the radiance value from one band, or several bands may be grouped together to form a feature.

Training data is a set of data for which the input values and corresponding class labels are known. The training data set is ideally constructed from images of known land cover types. In practice, as a surrogate for *a priori* knowledge of image land cover, pixels from images that represent obvious land cover are used.

1.2 Training library

A library of training data from which the user can construct training sets must be constructed.

Radiance values from pixels of a given land cover type are extracted from an image, hand-labeled, and added to a training directory containing data of that land cover type.

Ideally, the training data should be recorded by the same camera from images which are similar to the ones which will be classified by the algorithm. If care is taken to convert radiance values to their true reflectivity in both training and classification, images from different cameras may be used to compile training libraries. This has been shown to work between Hyperion and ALI images.

2 Training Set Script

A training set module written in Python (`trainingSet.py`) allows the user to construct a training set from hand-labeled image data. The module takes the location of the data data, such as `/home/username/training_data` along with a list of the names of land cover types, such as `[cloud, land, water]` to be used in the classification (the names of land cover types must correspond to sub-directories in this location). In addition, the user may optionally specify a list of bands to be used for classification, where the structure of the list should correspond to the grouping of bands into features, e.g.,

```
[[B009,B010],[B011,B012,B013], ... ]
```

along with a list of the desired weightings to be used for the bands within each feature, e.g.

```
[[1/2,1/2],[1/3,1/3,1/3], ... ]
```

Running the training module produces a text file called `trainingSet.txt`, stored in the directory where the code was run. The user can specify a set of bands without specifying the desired weightings, and the training set module will simply average the bands being accumulated in the given sub-band. If the user specifies no bands, the default (hard-coded) is to use a grouping of bands which loosely follows the band ranges of the ALI imager [1].

If a requested band is missing from the training data, the code will report the missing band to stderr and continue without the band (Note: If the user specifies a list of bands to be used for training, but some of those bands are not present, undesired results can occur).

3 Training Script Usage

The algorithm which is used to construct a training set can be called from a Python shell as:

```
trainingSet.main(training_directory_string,land_cover_lst, \
    band_lst, band_weight_lst])
```

or as part of the test *harness/wrapper.py* that runs a full classification, as:

```
wrapper.main(image name,classification type,data type, \
    program type,True)
```

where “True” indicates to the wrapper that the training script should be executed, and the wrapper should be modified to contain the relevant directives to the training script.

References

[1] K. G. Nikolakopoulos, *et. al.*, Proc SPIE, 7110, 1-12, 2008.