D

Python Scripts

This appendix provides installation instructions and documentation for the Python scripts implementing the algorithms described in the text. The programs were tested with 32-bit Python 2.7 on MS Windows 7.0, and with Python 2.7 on 32 and 64-bit Linux (Ubuntu 12.04).

D.1   Installation

The Python interpreter is pre-installed on Mac OS and Linux. The latest Python 2.7x can be obtained for most operating systems from `http://www.python.org/`

D.1.1   Required packages

The following is a list of the Python extension packages which are imported in the various scripts, together with the URLs at which they can be obtained.[*]

> `numpy:` `http://www.numpy.org/` (numerical Python)
>
> `scipy:` `http://www.scipy.org/` (scientific Python)
>
> `matplotlib:` `http://matplotlib.org/` (2D plotting library)
>
> `gdal:` `https://pypi.python.org/pypi/GDAL/` (geo-spatial data abstraction library)
>
> `mlpy:` `http://mlpy.sourceforge.net/` (machine learning Python)
>
> `ctypes:` `http://python.net/crew/theller/ctypes/` (foreign function interface)
>
> `shapely:` `https://pypi.python.org/pypi/Shapely` (manipulation and analysis of planar geometric objects)

---

[*]Windows users can obtain pre-compiled binaries for most of these at `http://www.lfd.uci.edu/~gohlke/pythonlibs/`

    `opencv:` `http://sourceforge.net/projects/opencvlibrary/` (open source computer vision library)

    `spy:` `http://spectralpython.sourceforge.net/` (spectral Python)

    `multyvac:` `https://www.multyvac.com/` (high-performance cloud computing platform)

    `auxil` (additional auxiliary routines, included with the software on the author's website)

All Python scripts which accompany this book, including the `auxil` package, can be downloaded or cloned from the GitHub repository, see the links on

    `http://ms-image-analysis.appspot.com/static/homepage/software.html`

or on

    `http://mcanty.homepage.t-online.de/software.html`

The IR-MAD script `iMad.py` requires the dynamic library `prov_means.dll` (Windows) or `libprov_means.so` (Linux, Mac OS). Compiled versions for 32-bit and 64-bit Python (Windows) and Python on 32-bit and 64-bit Linux are included with the software, together with the source code `prov_means.c`. The libraries should be placed in the OS path, e.g., `C:\Windows` or `/usr/lib`.

    To install the `auxil` package, open a console in the unpacked directory and type the following:

    `python setup.py install`

    The scripts themselves are organized according to book chapter and can be run from the command line. Convenient environments are `idle` (included in most Python distributions) and `ipython`; see `http://ipython.org/`.

D.1.2   Eclipse

For those who wish to program the examples given in the exercises, or modify/improve the scripts provided here, Eclipse (`http://www.eclipse.org/`) together with the plug-in `Pydev` (`http://pydev.org/`) provide an excellent, platform-independent Python programming development environment, very similar to that for IDL. The environment includes syntax highlighting, code completion and debugging.

D.1.3   Docker containers

Docker containers which package command-line versions several of the Python scripts are also available. They eliminate the need to install the Python environment described above entirely. Only a 64-bit Linux OS and Docker (`https://www.docker.com/`) is needed. On Windows or Mac OS one can use

the freely available `VMWare Player` to host a Linux virtual machine and run Docker or, alternaively, install boot2docker from the Docker website.

From Ubuntu, for example, assuming Docker is installed, run

```
sudo docker run -d -p433:8888 -v my-dir:/crc/imagery
        --name=crc mort/crcdocker
```

to download a Docker image and start a container in daemon mode serving the iPython notebook kernel. Here `my-dir` is the location of your images on your host machine. The `-v` option shares this directory with the directory `/crc/imagery` in the running container. Any changes made in the container will be immediately reflected in the host directory.

Point your browser to http://localhost:433 to see the iPython notebook home page. Then open a new notebook and get help for the various scripts with

- `run dispms -h` (displaying a ms-image in RGB)

- `run pca -h` (principal components analysis)

- `run iMad -h` (IR-Mad change detection)

- `run radcal -h` (relative radiometric normalization)

- `run em -h` (multi-scale, spatial/spectral Gaussian mixture clustering)

- `run register -h` (frequency domain image-image registration)

- `run atwt -h` (a trous transform panchromatic sharpening)

- `run dwt -h` (discrete wavelet transform panchromatic sharpening)

- `run c_corr -h` (empirical cosine correction for solar illumination in rough terrain)

- etc ...

The bash script `/crc/normalize` in the container can be used for automatic radiometric normalization of Landsat TM full scenes. The script takes four required parameters and one optional one:

1. the spectral band to use for warping (registering) the target image to the reference image

2. the spectral subset (bands) to include in the calculation

3. the filename of the reference image

4. the filename of the target image

5. (optional) the spatial subset to use for registering and IR-MAD

For example:

```
!./ normalize 4 [1,2,3,4,5,7] reference.tif target.tif [500,500,2000,2000]
```

Similarly, the bash script `c-correction.sh` will run solar illumination correction with the c-correction method discussed in Chapter 5. The input parameters are:

1. the spatial subset to include in the calculation

2. the spectral subset (bands) to include in the calculation

3. the number of land cover classes to assume

4. the solar azimuth angle in degrees

5. the solar elevation angle in degrees

6. the filename of the multispectral image

7. the filename of the associated DEM (must have the same projection as the ms image)

For example:

```
!./ c-correction.sh [0,0,1000,1000] [1,2,3,4,5,7] 3 135 57 ms-image dem-image
```

If you stop the container, you can start it again with

```
sudo docker start crc
```

D.2   Documentation

D.2.1   Utilities

The `auxil` package contains the following modules:

`auxil.auxil.py`

A collection of auxiliary routines for processing multispectral imagery.

`auxil.congrid.py`

Arbitrary re-sampling of an array to new dimension sizes. Mimics the `CONGRID()` function in IDL.

`auxil.header.py`

Defines an object class representing the text fields of an ENVI format header.

`auxil.png.py`

A pure Python PNG coder/decoder, see `http://pythonhosted.org/pypng/png.html`

`auxil.polsar.py`

Defines an object class to store fully polarimetric SAR data in multilook covariance matrix form.

`auxil.supervisedclass.py`

Defines object classes for supervised image classification: Bayes maximum likelihood, neural networks with backpropagation and scaled conjugate gradient training, and a support vector machine.

D.2.2   Scripts for Chapter 1

`dispms.py`

A command-line oriented routine to display any three spectral bands of a multispectral image as an RGB composite. Usage:

```
python dispms [-f filename] [-p pos] [-d dims] [- e enhancement]
```

The RGB band positions and spatial dimensions are quoted lists, e.g.,

```
-p "[0,1,3]" -d "[0,0,400,400]"
```

The dimensions list `dims` is of form `[X0,Y0,samples,lines]`. The enhancement modes are: $1 =$ linear byte stretch, $2 =$ linear stretch, $3 =$ linear 2% stretch, $4 =$ histogram equalization. Information not supplied on the command line is queried interactively.

D.2.3   Scripts for Chapter 4

D.2.3.1   Nonlinear principal components analysis (Section 4.4.2)

`kpca.py`

The user is first queried for a working directory, image filename and a training sample size. If the latter is 0, then 100 representative training pixel vectors are chosen by the k-means algorithm. Otherwise, a random sample of the desired size is used. Next, the number of kernel principal components to retain is entered and the destination output file selected. The user can then choose between a linear or Gaussian kernel. The Gaussian kernel parameter γ is calculated as $\gamma = 1/(2\sigma^2)$, where $\sigma = \langle \|\boldsymbol{g}(\nu) - \boldsymbol{g}(\nu')\| \rangle$ is the average Euclidean distance between the training observations. Finally, the output destination is selected. After centering on the training data and diagonalizing the kernel matrix, a plot of the eigenvalues is displayed and the projected image is stored to disk.

D.2.4   Scripts for Chapter 5

D.2.4.1   Panchromatic sharpening of multispectral images using the discrete or à trous wavelet transform (Sections 5.3.4 and 5.3.5)

`dwt.py`
`atwt.py`

The user is queried for the working directory, the (spatial/spectral subset of the) multispectral image to be sharpened, the corresponding panchromatic or high-resolution image and the output file. The panchromatic image should overlap the multispectral image completely, so that the multispectral image defines the extent of the final pan-sharpened product. Then the MS to pan spatial resolution ratio (2 or 4), the MS band to be used for co-registration, and, in the case of DWT fusion, a fine adjustment parameter are queried. During the calculation the correlations of the wavelet coefficients for the low- vs. high-resolution bands are printed.

D.2.4.2   Conversion of SAR imagery to ENVI standard files

`polsaringest.py`

Geocoded, multi-look polarimetric SAR imagery suitable for processing with the filtering, classification and change detection algorithms described in the text may be obtained directly from the provider or generated from single-look complex (SLC) data. In the latter case, the open source software packages PolSARpro (European Space agency),

`http://earth.eo.esa.int/polsarpro/`

together with MapReady (Alaska Satellite Facility),

```
http://www.asf.alaska.edu/downloads/software_tools
```

are a good choice; see **?**. PolSARpro is first used to create multi-look images in covariance matrix format, which can then be exported to MapReady for georeferencing with or without a DEM. Alternative commercial solutions are the Gamma Software (Gamma Remote Sensing)

```
http://www.gamma-rs.ch/
```

and the SARscape add-on module for ENVI

http://www.exelisvis.com/ProductsServices/ENVI/ENVISARscape.aspx

The Python script `polsaringest.py` combines the outputs from the above preprocessing systems to a single, multi-band file in 32-bit floating point format. For full quad polarimetric data, it generates 9 bands, which are ordered as follows:

$$C11 = \langle |s_{hh}|^2 \rangle$$
$$C12\text{re} = \langle \sqrt{2} s_{hh} s_{hv}^* \rangle (\text{real part})$$
$$C12\text{im} = \langle \sqrt{2} s_{hh} s_{hv}^* \rangle (\text{imaginary part})$$
$$C13\text{re} = \langle s_{hh} s_{vv}^* \rangle (\text{real part})$$
$$C13\text{im} = \langle s_{hh} s_{vv}^* \rangle (\text{imaginary part})$$
$$C22 = \langle 2|s_{hv}|^2 \rangle$$
$$C23\text{re}) = \langle \sqrt{2} s_{hv} s_{vv}^* \rangle (\text{real part})$$
$$C23\text{im}) = \langle \sqrt{2} s_{hv} s_{vv}^* \rangle (\text{imaginary part})$$
$$C33 = \langle |s_{vv}|^2 \rangle.$$

For dual or single polarimetry, the bands corresponding to the missing matrix elements are not present. Thus, a dual polarimetric image will consist of four bands, e.g., C11, C12re, C12im, and C22; a single polarimetric image will consist of just one band, usually C11 or C33. The files generated by the script can be read by all of the SAR processing routines described below, and also by their ENVI/IDL counterparts described in Appendix C.

The user is prompted for the directory containing the georeferenced covariance matrix files (consisting of one file for each of the real and imaginary components), and then requested to choose (a spatial subset of) one of them. The other files are then read in automatically with the same spatial subset. Finally, an output filename and desired format are requested.

D.2.4.3 Multivariate estimation of equivalent number of looks for polarimetric SAR in covariance matrix format

```
enlml.py
lookup.txt
```

This method is not discussed in the text. It is a multivariate technique based on the maximum likelihood estimator explained in **?** and is reported to be superior to the standard univariate method discussed in Section 5.4.2. The polarimetric SAR image should be in the format generated by the script `polsaringest.py` described in Section D.2.4.2 above. The user is asked to select (a spatial subset of) the covariance matrix file, the desired window size (default 7×7) and an output filename and format. The local ENL values are estimated in a moving window and stored as a single-band, floating point raster image. A histogram of the ENL values is plotted.

D.2.4.4   Minimum mean square error filtering of polarimetric SAR imagery (Section 5.4.3.1)

`mmse_filter.py`

The polarimetric SAR image should be in the format generated by the script `polsaringest.py` described in Section D.2.4.2 above. The user is queried for the working directory, then for the input filename, the equivalent number of looks and, finally, the output filename. The filtering operation occurs in two steps: first the span image is processed to determine the filter weights, then all components of the covariance matrix are filtered separately.

D.2.4.5   Gamma maximum a posteriori filtering of polarimetric SAR imagery (Section 5.4.3.2)

`gamma_filter.py`

The polarimetric SAR image should have the same format as in Section D.2.4.2 above. The user is queried for the working directory, then for the input filename, the equivalent number of looks, the number of iterations and, finally, the output filename. Only the diagonal elements of the covariance matrix are filtered.

D.2.5   Scripts for Chapter 6

D.2.5.1   Supervised classification of multispectral images with maximum likelihood, neural network and support vector machine (Sections 6.3, 6.5, 6.6 and Appendix B)

`classify.py`

The user is first queried for a working directory and input image filename and (optionally) a spectral subset. The image can be in any format recognized by GDAL, but must be geo-referenced. Then the user can choose between maximum likelihood, neural network (with backpropagation or scaled conjugate gradient training), or support vector machine algorithms. Next an ESRI shape file, which defines the regions of interest in the input image, and an output filename for the test results must be entered. Following this, the user must

enter the filename for the classification (thematic map) image and (option-
ally) a filename for the class membership probability image. The latter is not
available for the maximum likelihood classifier. Finally, if the neural network
classifier was selected, the number of hidden neurons must be entered. (A
cross-entropy plot will be displayed after training has completed.) If the out-
put format chosen is ENVI, then an ENVI header corresponding to the ENVI
filetype "classification" will be written. The test result file can be processed
with the scripts `ct.py` and `mcnemar.py` discussed in Section D.2.6.3 below.

D.2.6   Scripts for Chapter 7

D.2.6.1   Probabilistic label relaxation postprocessing (Section 7.1.2)

`plr.py`
`plr_reclass.py`

The script takes as input a class probability vector image generated by
most of the supervised classification extensions described in this appendix, as
well as from the clustering routine `em.py` described in Section D.2.7.2 below,
and generates a modified class probability vector image (rule image). At the
prompt, choose a class membership probabilities image and the number of
iterations (default = 3). Then choose a destination filename and format. The
generated rule image can then be processed with the script `plr_reclass.py` to
produce an improved classification image with better spatial coherence. At
the prompt, choose a class membership probabilities image. Then choose an
output filename and format. If the format is ENVI, then an ENVI classifica-
tion file will be written.

D.2.6.2   Neural network supervised classification with cross-valida-
tion (Section 7.2.2)

`ffncg.py`

This script requires registration on the Multyvac website

`https://www.multyvac.com/`

The user is first queried for a working directory, an input image filename and
the training data shapefile. The image can be in any format recognized by
GDAL, but must be georeferenced. Next, enter the number of hidden neurons
and the filename for the classification image (thematic map). Training and
classification take place on the host computer using 9/10th of the training
data. The classification method used is a two-layer feed-forward network with
scaled conjugate gradient training. Upon completion, a cross-entropy plot
characterizing the training phase is displayed. When the plot is closed, the
training data are uploaded to the cloud service and a 10-fold cross-validation
is carried through. The results (misclassification rate and standard deviation)
are printed to the standard output. **Note:** Presently Multyvac does not offer

parallelization, so that the script uses the ordinary Python `map()` function to emulate parallel processing on the cloud service.

D.2.6.3   Contingency tables and McNemar test (Section 7.2)

`ct.py`
`mcnemar.py`

The scripts `ct.py` and `mcnemar.py` are used to evaluate and compare test results generated by the supervised classifiers discussed in this appendix. They request input files with extension `tst` and generate their outputs (contingency tables, accuracies, test statistics, etc.) on the standard output.

D.2.6.4   Anomaly detection with the RX-algorithm (Section 7.5.4)

`rx.py`

The user is first queried for a working directory and input image filename. The image can be in any format recognized by GDAL. Then the output filename and desired format must be entered. The anomaly image (the Mahalanobis distance of each pixel vector to the mean image background) is written to disk.

D.2.7   Scripts for Chapter 8

D.2.7.1   Kernel K-means clustering of multispectral imagery (Section 8.2. 2)

`kkmeans.py`

The user is first queried for a working directory and input image filename. The image can be in any format recognized by GDAL. Then spatial and/or spectral subsets can be entered along with a training data sample size to estimate the kernel and the desired number of clusters. Finally, the output filename and format and the kernel type (Gaussian or linear) must be chosen. If the output format is ENVI, then an ENVI classification file header is generated.

D.2.7.2   Gaussian mixture clustering using the expectation maximization algorithm (Section 8.3)

`em.py`

Clustering occurs optionally at different scales, and both simulated annealing and spatial memberships can be included if desired. The user is queried for a working directory and input image filename. The image may be in any format recognized by GDAL. Then, at the corresponding prompts, spectral and/or spatial subsets may be chosen, followed by the number of clusters, the

number of compressions (initial and final pyramid depths), initial annealing temperature (zero for no annealing), and spatial membership parameter `beta` (zero for no spatial memberships). Finally, the user is prompted for the desired output format and filename for the classification image and (optionally) for a probability image. If ENVI format is chosen, an ENVI classification file will be written.

D.2.8 Scripts for Chapter 9

D.2.8.1 Iteratively re-weighted multivariate alteration detection (Section 9.4)

`iMad.py`

When running the `iMad.py` script, you are first prompted for an input directory and then for the (spectral and spatial subsets of) the first and second input images. The input subsets must be co-registered and have the same spatial/spectral dimensions. They can be in any format recognized by GDAL. The following prompts are for a regularization or penalization factor (default 0) and for an output filename, which again can be in any GDAL multispectral format. The script prints the convergence criterion `Delta` and the canonical correlations for each iteration. Computation terminates when `Delta` < 0.001 or after 100 iterations. The output consists of the stacked MAD variates in order of decreasing variance followed by the chi-square image.

D.2.8.2 Polarimetric SAR change detection

`wishartchange.py`

The two polarimetric SAR images should have the same format as in Section D.2.4.2 above and be geo-referenced. The user is asked to choose a working directory, and then to enter the input filename and the equivalent number of looks for each image successively. The images must have the same polarimetry, which can be any of the following:

- full polarimetry
- dual polarimetry
- single polarization.

Next a significance threshold for the change map image is queried, and finally, an output filename and desired format. The images are co-registered automatically and the second image is clipped to have the same spatial extent as the first. Note: this will only work reliably when the original spatial dimensions of the two images are similar. After completion, a 2-band image consisting of the test statistic $-2\rho \ln Q$, and the associated change probability $\Pr(-2\rho \ln Q) \leq z)$ is written to the chosen output filename. The latter can be

thresholded to obtain a change map at any desired significance level (e.g., at 0.99 for changes at the 1% significance level.) The change map image, consisting of the changes at the chosen significance level in red superimposed onto the first band (C11) of the first image, is written to the the chosen output filename with `_cmap` appended.

D.2.8.3 IR-MAD relative radiometric normalization (Section 9.8)

`radcal.py`

This script takes advantage of the linear and affine invariance of the MAD transformation to perform a relative radiometric normalization of the images involved in the transformation.

The routine first prompts for an input directory, (spectral/spatial subsets of) the reference and target images, as well as for the spatial subset of the iMad image generated according to Section D.2.8.1 above (the chi-square band is found automatically). The spatial subsets of all three input images must have the same size and the spectral subsets of the reference and target images must match. After the prompt for an output filename, which can have any of the formats recognized by GDAL, the filename of a larger (e.g., full) scene can be (optionally) entered. This file must have the same spectral dimensions as the reference and target files and will be normalized with the regression coefficients determined by them. The result will be stored in the same format and with the same root name as the specified output filename, but with `_norm` appended. Finally the user is prompted for a minimum no-change probability threshold (default 0.95). The script prints, band-wise, the regression coefficients, and the results of statistical tests for equal means and variances of the reference and normalized target image bands. The tests are evaluated on the basis of the hold-out test pixels (train:test = 2:1). The script also plots the orthogonal regression lines for up to 10 spectral bands.