Análise de Imagens de Sensoriamento Remoto - 2018.2
Assignment #2

Introduction

In these experiments we will use the *Indian Pines* and the *Pavia University* hyperspectral datasets. You may search in the Internet for information about those datasets.

You should run the `MCRF LVC` package to execute the MRF and the CRF models to classify images. In the compressed file you will find the `MCRF.exe` executable (for Windows only, sorry!); the MATLAB script `ReadClassificationResults.m`, which has functions to read the classification results, compute accuracy statistics and write the classification outcome (as an image); and the data files.

Notice that not all pixels are labeled in the datasets, and training, as well test samples are in two separate files.

Your report must present the classification results as label images, and report accuracy metrics (overall and average per class accuracies; and F1-scores).

Results of the following configurations must be reported:

(1) Using only the unary term in MRF (delivered by a Naïve Bayes classifier).
(2) Using the unary term in MRF (delivered by a Naïve Bayes classifier) as well as a binary/pairwise term (delivered by the simple Potts model). In this case you have to tune the weight manually.
(3) Using only the unary in CRF (delivered by a Random Forest classifier).
(4) Using the unary term in CRF (delivered by a Random Forest classifier) as well as a binary/pairwise term (delivered by the Contrast Sensitive Potts model). In this case you have to tune the weight manually.

You should refer to the slides presented in class about Directed and Undirected Graphical models, and to the slides attached to this task (`AISR20182_Assignment_2_slides.pdf`) for information about the models' parameters.

Análise de Imagens de Sensoriamento Remoto - 2018.2
Assignment #2

Details

- In the `MCRF_LVC` package, verify the following file distributions in the `Data` folder: for both datasets: `Indian_Pines` and `PaviaUniversity`:
  - `NodeTrainerParams_NB`: Parameters for Naïve Bayes classifier for the unary term.
  - `NodeTrainerParams_RF`: Parameters for Random Forest classifier for the unary term.
  - `RawData/Image.mat`: MAT file containing the image data with dimensions $rows \times cols \times bands$.
  - `RawData/Train.bmp`: Image with labels corresponding to pixels for training.
  - `RawData/Test.bmp`: Image with labels corresponding to pixels for testing.
  - `TrainEdgeContrastSensitivePotts.par`: Parameter file for the Contrast-sensitive Potts model for the binary/pairwise term.
  - `TrainEdgePotts.par`: Parameter file for the Potts model for the binary/pairwise term.

  Additionally, in the main folder, there is a file called: `ReadClassificationResults.m`, which can be used to read the classification results generated in the folder `Predictions_MonoCRF`.

- To run the `MCRF_LVC` package, it is necessary to pre-process the input data in the `./Data/DATASET/RawData/` folder to create the following files: `SpatialConnectivity`, `Features`, `Labels`, `TrainTestMask`. Notice that the path to the folders those files are stored in are parameters of the configuration file (see Figure 4). All these files must be binary files (extension `.bin`), which can be created with any language. Using MATLAB, these files can be created as follows:

```
% To save a 'Matrix' with 'float' precision as Binary File
fileID = fopen('BinaryFileName.bin','w');
fwrite(fileID,single(Matrix),'float'); %'float' or 'double' for instance
fclose(fileID);
```

- The `SpatialConnectivity` indicates the connections of a labeled pixel to the neighbor pixels. This matrix is `double` precision with dimensions $3 \times N_{pixels}$, where N_{pixels} is the number of labeled pixels (foreground) in the image and each row of this matrix contains the *left*, *up* and *up-left* indices of the neighbor pixels according to Figure 1. Notice that the pixels are indexed by column in the figure. The blue squares represent labeled pixels and the white squares are background pixels (which will not be used for training and testing). If there is not a labeled pixel on those positions, an index equal to -1 must be selected. The filename should be like: `01_Filename1.bin`

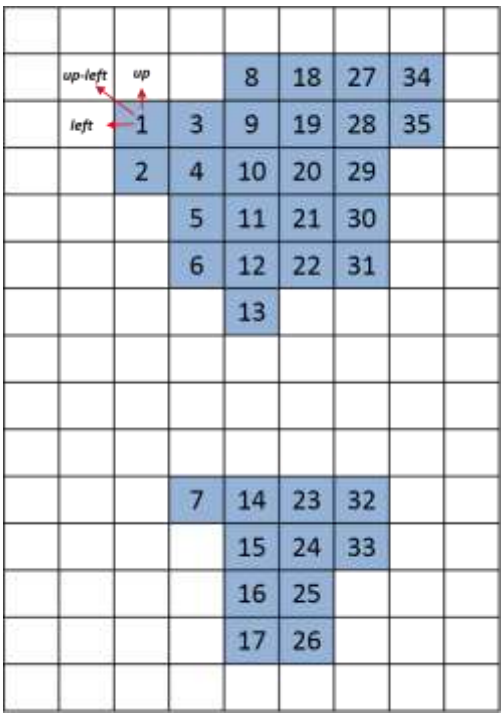

Figure 1. Pixels indexed by column where blue squares represent labeled pixels and the white squares represent background.

For instance, for the first pixels from Figure 1, the `SpatialConnectivity` matrix will be as follows:

Index by column	1	2	3	4	5	6	7	…
left	-1	-1	1	2	-1	-1	-1	…
up	-1	1	-1	3	4	5	-1	…
up-left	-1	-1	-1	1	2	-1	-1	…

Figure 2. Example of the `SpatialConnectivity` matrix.

- The *Features* file contains the extracted features from each pixel (original bands, principal component bands, texture, etc.). This matrix is `single` precision with dimensions $N_{features} \times N_{pixels}$, where $N_{features}$ is the dimensionality of the features extracted from the image and N_{pixels} is the number of pixels in the image. Notice that the features extracted from the image should be normalized between 0 and 1. The filename should be like: `01_30_`$N_{features}$`_Filename2.bin`

- The *Labels* file contains the labels corresponding to all pixels in the image. This matrix is `single` precision with dimensions $N_{pixels} \times 1$. The filename should be like: `01_30_`$N_{features}$`_Filename3.bin`

- The `TrainTestMask` file contains two values, '1' for training and or '2' for testing, which indicates if a pixel will be used for training or testing. This matrix is `single` precision with dimensions $N_{pixels} \times 1$. The filename should be like: `01_30_`$N_{features}$`_Filename4.bin`

- Finally, a configuration file, to set up the parameters for an experiment, must be created (as examples of these files see `ConfigIndianPines_Seq_1_From_1_A1_SP1.par` and `ConfigPaviaUniversity_Seq_1_From_1_A1_SP1.par`, available in the package). The structure of the configuration file is given in the Figure 4. The program must be using `MCRF ConfigurationFile.par` in the *Command Prompt*.

- To test different models for the unary potential, the parameters `nodeTrainingType` and `nodeTrainingDataPath` should be changed to `RandomForestCV` and `NodeTrainerParams_RF` to use Random Forest as classifier; and to `NaiveBayesAdd` and `NodeTrainerParams_NB` to use Naïve Bayes as classifier. Notice that by simply changing the classifier, you will be running a MRF or a CRF model, that is: if you chose the Naïve Bayes (generative) classifier, you will be running a MRF model; if you chose the Random Forest (discriminative) classifier, you will be running a CRF model.

- To test different models for the binary/pairwise potential, the parameters `edgeSpatialTrainingType` and `edgeSpatialDataFile` should be changed to `ContrastPotts` and `TrainEdgeContrastSensitivePotts.par` for the Contrast-sensitive Potts model; and to Potts and `TrainEdgePotts.par` for Potts model.

- To analyze how the level of smoothing in both Potts and Contrast-sensitive Potts models influence the classification outcome, the parameter `edgeSpatialTrainingWeight` of the configuration file should take different values from one to any desired value. For instance, from 1 to 10. Referring to the slides attached to this task (`AISR20182_Assignment_2_slides.pdf`), the θ_P weight is equal to `nodeTrainingWeight` divided by `edgeSpatialTrainingWeight`.

- For Potts model the β term is equivalent to θ_P. For the Contrast-sensitive Potts model, the parameter p can be set in the `TrainEdgeContrastSensitivePotts.par` (field `l1`), which is by default set to 0.5. If you wish, you can work with the default value. (See Figure 3)

```
%YAML:1.0
weight: 1.
approachP: 1
l1: 0.50000000000000000e+000
l2: 400.
useList: !!opencv-matrix
   rows: 1
   cols: 3
   dt: u
   data: [ 1, 1, 1 ]
```

Figure 3. Example of the *TrainEdgeContrastSensitivePotts.par* file.

- To read the classification results, use the script `ReadClassificationResults.m`, where the variable `Database` can take the values `PaviaUniversity` or `Indian_Pines`. Notice that the results will be read from the folder `Predictions_MonoCRF`, which is created when an experiment is carried out.

- In order to run multiple experiments, many configuration files can be created varying the parameters. Then, a batch file can be used to list and run all the experiments.

- The parameters `initEpoch` and `sequenceLength` should be fixed to 1. These parameters should be changed only for a multitemporal analysis. The parameter `tileSize` could be reduce in order to take less pixels and reduce the computational cost.

```
%YAML:1.0
# Paths to Raster, Labels, Index Image and DL features
pathLabels: ./Data/PaviaUniversity/Labels/
pathTrainTest: ./Data/PaviaUniversity/TrainTestMask/
pathFeatures: ./Data/PaviaUniversity/Features/
# ----------------------------------------------------------------------
# Association Potential Parameters (links labels with pixel values)
# Classifier:
# RandomForestCV
# NaiveBayesAdd
nodeTrainingType: RandomForestCV
# Classifier parameters:
# NodeTrainerParams_RF
# NodeTrainerParams_NB
nodeTrainingDataPath: ./Data/PaviaUniversity/NodeTrainerParams_RF/
# Association Potential Weight
nodeTrainingWeight: 1.
# ----------------------------------------------------------------------
# Interaction Potential Parameters (links every pixel with each neighbor
# Interaction type model:
# ContrastPotts
# Potts
edgeSpatialTrainingType: Potts
# Feature engine:
edgeSpatialTrainingFeatureEngine: InteractionDistance
# Interaction type parameters
edgeSpatialDataFile: ./Data/PaviaUniversity/TrainEdgePotts.par
# Spatial Connectivity folder
edgeSpatialConnectivityPath: ./Data/PaviaUniversity/Connectivity_Spatial
# Interaction Potential Weight
edgeSpatialTrainingWeight: 1.
# ----------------------------------------------------------------------
# Inference
inferIterations: 15
# Tile Size
tileSize: 30000
# Experiment Configuration
initEpoch: 1
sequenceLength: 1
```

Figure 4. Structure of the Configuration File