## "Machine Learning and Computational Statistics"

Project

Introduction

This project is devoted to Hyperspectral Images (HSI) processing.

Hyperspectral images (HSIs): An HSI depicts a specific scene at several (L) narrow continuous spectral bands (actually, they visualize the reflectance of the depicted scene in various spectral bands). It can be represented by a $MxNxL$ three-dimensional cube, where the first two dimensions correspond to the spatial information, while the third corresponds to the spectral information. Thus, the (i,j) pixel in such an image, $i=1,…,M$, $j=1,…,N$, is represented by an L-dimensional vector (constituted by the corresponding spectral bands), called **spectral signature** of the pixel.
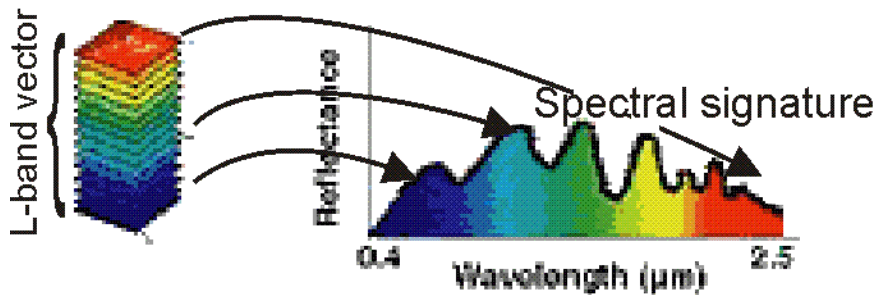
In several remote sensing applications, the HSIs (taken from satellites) that depict specific scenes of the earth surface at a specific spatial resolution (that is, a single pixel may represent an area from 3x3 m^2, to, e.g., 100x100m^2 or more). That is, each pixel is likely to depict more than one materials depicted in the corresponding area of the scene. Such pixels are called **mixed pixels** and they are the vast majority of the pixels in the image. On the other hand, there are (usually) a few pixels that depict a single material. These are called **pure pixels**.
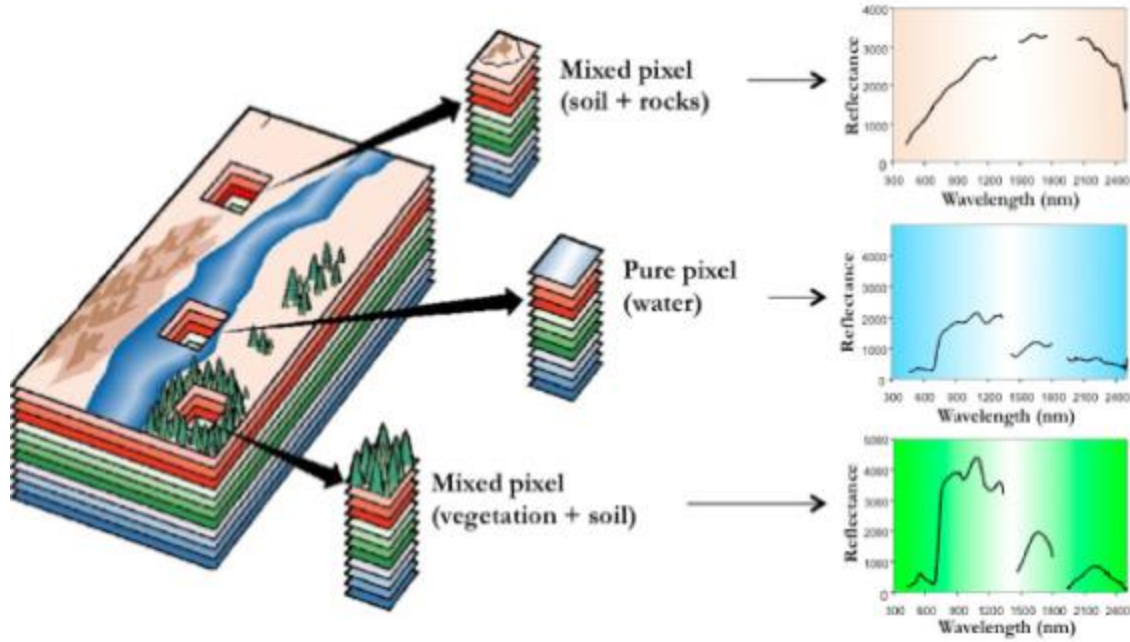
Processing in HSIs: The usual processing procedures in HSIs follow two main directions, namely, the ***spectral unmixing*** and the ***classification*** (supervised, unsupervised).

(a) <u>**Spectral unmixing (SU):**</u> The problem here is stated as follows: Assume that a set of m spectral signatures corresponding to the pure pixels in the HSI under study is given[1]. For a given pixel in the image, the aim is to determine the percentage (***abundance***) to which each pure material contributes in its formation. It is clear, that SU provides ***sub-pixel information*** for a given pixel. Speaking in mathematical terms, let

(i) y be the (column L-dimensional) spectral signature of the pixel under study,

(ii) $x_1, \ldots x_m$, be the spectral signatures (column L-dimensional vectors) of the pure pixels in the image (each one corresponding to a pure material met in the image) and

(iii) θ, the m-dimensional ***abundance vector*** of the pixel (each q-th coordinate corresponds to the percentage to which the q-th pure pixel contributes to the formation of the pixel under study).

Adopting the ***linear spectral unmixing hypothesis***, the above quantities are related as follows

$$y = X\,\theta + \eta,$$

---

[1] Actually, in most real cases, these pure pixels are lected from the image via a procedure called ***Endmember Extraction***.

where η is an *L*-dimensional i.i.d., zero mean Gaussian noise vector. Note that, physically, the entries of θ should be nonnegative and (ideally) they should sum to one.

(b) **(Supervised) classification:** In this case, the problem is stated as follows: Assume that all the pixels in the HSI under study are known to belong to one out of *m known classes*. Given a specific pixel, the aim is to determine the most suitable class to assign it.

All questions in this project refer to the so called "Salinas" HSI, which depicts an area of the Salinas valley in California, USA. It is a 150x150 spatial resolution HSI and consists of 204 spectral bands (from 0.2µm – 2.4µm) and its spatial resolution is 3.7m (that is, the HSI is a 150x150x204 cube). The data that will be used are in the files "*Salinas_cube.mat*" (the Salinas hypercube) and "*Salinas_ground_truth.mat*" (the class label for each pixel).

NOTES:

1. You will need first to run the attached python code to retrieve the quantities mentioned below.
2. Only the **pixels** with **nonzero class label** will be **taken into consideration** in this project.

Description of the project

Part 1 (spectral unmixing)

The aim here is to perform **unmixing** on **each one** of the **pixels** in the image with nonzero label, with respect to **5** endmembers (obtained after the execution of the attached python code) using the following four different spectral unmixing methods:

(a) Least squares (as it was taught in the class),
(b) Least squares imposing the sum-to-one constraint (state the problem explicitly and use the solvers.qp() function from the CVXOPT package, which should be downloaded)
(c) Least squares imposing the non-negativity constraint on the entries of θ (state the problem explicitly and use the solvers.qp() function from the CVXOPT package).
(d) Least squares imposing both the non-negativity and the sum-to-one constraint on the entries of θ
(e) LASSO, i.e., impose sparsity on θ via l_1 norm minimization.

** You may use either sklearn, scipy.optimize or the solvers.qp() function from the CVXOPT package, which should be downloaded. (state the problem explicitly and use the solvers.qp() function from the CVXOPT package).

(A) For **each method**:

(i) Derive the corresponding 5 abundance maps (one for each endmember/material)

(ii) Compute the **reconstruction error** (for each (**non-zero class label**) pixel y_i compute the quantity $(y_i - X\theta_i)^2$ and then take the average over those pixels).

(B) Compare the results obtained from the above four methods and comment briefly on them (utilize the class information given in "Salinas_ground_truth.mat").

Part 2 (classification)

In this case, we consider also the image pixels with non zero class label. The task is to assign each one of them to the most appropriate class, among the **5 known classes**. To this end three classifiers will be used: (i) the naïve Bayes classifier, (ii) the minimum Euclidean distance classifier and (iii) the k-nearest neighbor classifier. For each one of them the **same training**, **test** and **operation sets** will be used (obtained after the execution of the attached python code). Note that, **in practice**, we have at our disposal a set of points, which we split into the training and the test set. The operational set mentioned above corresponds to data presented to the classifiers after their training and evaluation, that is, during their **operational mode**.

(A) **For each classifier**:

(i) Train it based on the training set performing 10-fold cross validation (that is, the training procedure will be repeated 10 times). Report the estimated validation error as the mean of the ten resulting error values (compute also the associated standard deviation).

(ii) After that, use the whole training set to train the classifier and evaluate their performance on the test set as follows: First, **compute** the **confusion matrix** (a 5x5 matrix, whose (i,j) element is the number of pixels that belong to the i-class and are assigned from the classifier to the j-th class. Clearly, the "more diagonal" the matrix, the better the performance of the classifier is) and identify the classes that are not well separated by the classifier (if any). Then, **compute** the **success rate** of the classifier (sum of the diagonal elements of the confusion matrix and divide by the sum of all elements of the matrix).

(B) Compare the results of the classifiers and comment on them (Relate the confusion matrices obtained from each classifier to each other and pay attention to non-diagonal entries that are "not nearly zero". For example, a non-diagonal entry equal to 1 does not carry any significant information).

(C) Under what conditions do you believe that the **Bayesian classifier** where the classes are modeled by **normal distributions** could also be used in the framework of this application? (Consider the fact that the inverse of the covariance matrix for each class need to be estimated in this case).

NOTE: You can use any ready-to-use python implementations of the above classifiers, except the minimum distance classifier, which should be implemented by you. For the cross-validation, you can also use relevant products, ready-for-use (e.g. function *cross_val_score* from sklearn.model_selection tool).

Part 3 (combination)

Comment briefly on the possible correlation of the results obtained from the spectral unmixing procedure with those obtained from classification.