LA-UR-17-21981
Approved for public release; distribution is unlimited.

Title: Discussion of CoSA: Clustering of Sparse Approximations

Author(s): Armstrong, Derek Elswick

Intended for: Presentation and internal distribution.

Issued: 2017-03-07

LA-UR-17-xxxx, xx/xx/2017

Discussion of CoSA: Clustering of Sparse Approximations

Derek Armstrong, XCP – 8

February 2017

Purpose of Presentation

The purpose of this talk is to discuss the possible applications of CoSA (Clustering of Sparse Approximations) to the exploitation of HSI (HyperSpectral Imagery) data. CoSA is presented by Moody et al. in the Journal of Applied Remote Sensing ("Land cover classification in multispectral imagery using clustering of sparse approximations over learned feature dictionaries", Vol. 8, 2014) and is based on machine learning techniques.

General Overview

- **Application is land cover classification of multispectral Worldview-2 data**
  - satellite data with 1.84m spatial resolution
  - 8-bands: coastal blue, blue, green, yellow, red, red-edge, NIR1, NIR2
- **Quote: "We present a technical solution for unsupervised classification of land cover in multispectral satellite imagery, using sparse representations in learned dictionaries…" (Moody et al.)**
  - *unsupervised classification* means classes not known in advance
  - *dictionary* is a basis / endmembers / feature vectors
  - *learned dictionary* implies the dictionary is determined with a "learning algorithm"
  - *sparse representation* means that pixel spectra are represented with a small subset of dictionary elements

LA-UR-17-xxxx, x/xx/2017

High Level Overview of CoSA Algorithm

- **Objective: Land cover classification from learned dictionary**
  - Find dictionary (matrix) $\mathbf{\Phi}$ such that $\mathbf{\Phi a}_i \approx \mathbf{x}_i$ for all spectra $\mathbf{x}_i$ in scene
  - Sparsity constraint applied to coefficient vectors $\mathbf{a}_i$
  - Clustering done in coefficient space, i.e., using **a**'s

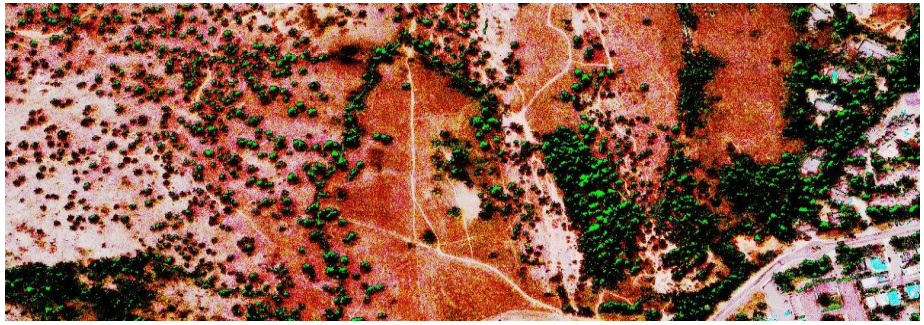


illustration representing false-color multi-band image

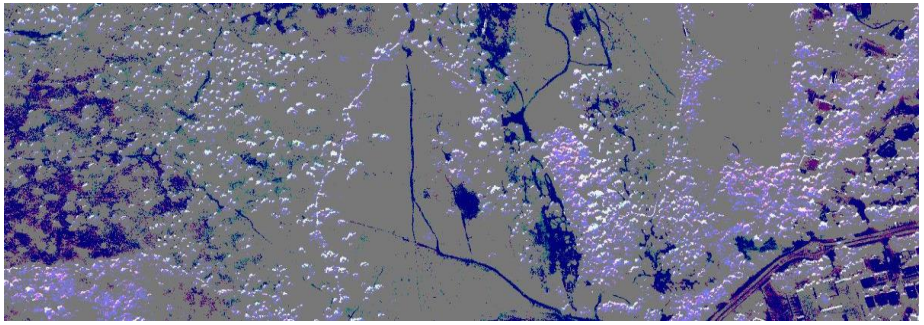


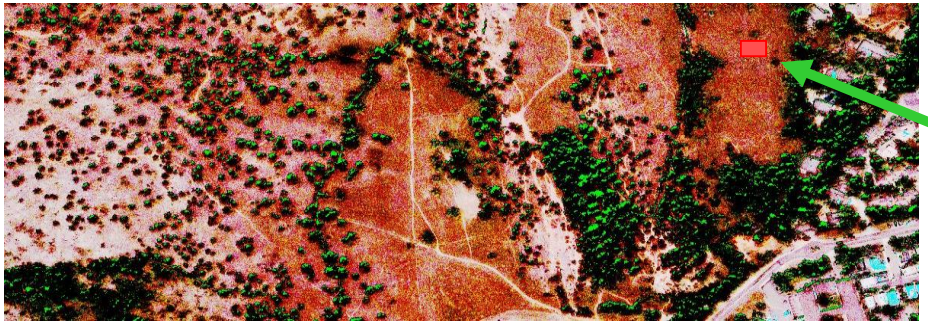
image after a clustering algorithm

- **Algorithm Procedures**
  - Apply K-means to scene and use centroids as initial dictionary
  - Optimize or learn dictionary with algorithm such as the Hebbian rule
  - Cluster scene using coefficients from fit of dictionary to scene spectra

LA-UR-17-xxxx, x/xx/2017

Incorporating Spatial Information

- **Pixel patches used to include spatial information in dictionary**
  - Spatial region of *p x p* is reshaped into vector of length $N = p * p * 8$
  - In approximations $\mathbf{\Phi a} \approx \mathbf{x}$, column vectors of $\mathbf{\Phi}$ and vector $\mathbf{x}$ have length N
  - This is not relevant to clustering a library of signatures



Computations performed on pixel patches of size *p x p*

- **Dictionary (matrix) is size *N* x *K***
  - *K* is the number of dictionary elements; $K = 300$ is used
  - Different patch sizes are used (p = 5, 7, 9, and 11) resulting in $N = 200$, 392, 648, and 948
  - $K > N$ results in an over-complete dictionary

Evaluating the Dictionary

- **What is a good dictionary in this context?**

  Want the dictionary to provide a good fit along with a sparsity constraint:

  Minimal value of $\|\mathbf{x}_i - \Phi\mathbf{a}_i\|_2^2 + \lambda\|\mathbf{a}_i\|_0$ for all spectra $\mathbf{x}_i$

  Dictionary fit to spectra

  Sparsity; # of nonzero $\mathbf{a}_i$

- **Greedy matching pursuits used for computational efficiency**
  - Hard sparsity constraint used rather than a penalty term; e.g.,

    Minimize $\|\mathbf{x}_i - \Phi\mathbf{a}_i\|_2^2$ subject to $\|\mathbf{a}_i\|_0 \leq L$

  - Coefficients determined according to:
    - Find best fit between $\mathbf{x}_i$ and a single dictionary element
    - Subtract this fit from $\mathbf{x}_i$ and find best fit between residual and single dictionary element
    - Repeat until there are L elements in the fit

Updating the Dictionary

- **Algorithm is mini-batch; mixture of online and batch**
  - batch algorithm → changes (to dict.) accumulated over an entire presentation of the training data before being applied
  - online algorithm → changes (to dict.) are made after each input instance
  - mini-batch → changes are accumulated over some number of instances before being applied
- **Code uses a batch size of 5; dictionary updated after each time it is applied to 5 library spectra**
- **Why use online training / algorithms?**
  - Can be more efficient, especially if there is a lot of data
  - Can provide better results due to improved efficacy/efficiency
  - Drawback is that solution (dictionary) can oscillate

Gradient Descent to Update Dictionary

- **Gradient descent: move current solution (dictionary) in the direction of the negative gradient for minimization**
- **Recalling the objective:**

$$\text{Minimize value of } \|\mathbf{x}_i - \Phi\mathbf{a}_i\|_2^2 \text{ subject to } \|\mathbf{a}_i\|_0 \leq L$$

Taking the gradient of the above with respect to Φ, dictionary element φ_k is updated by

$$\Delta\varphi_k = \mathrm{a}_{ik}(\mathbf{x}_i - \Phi\mathbf{a}_i)$$

and

$$\varphi_k^{(c+1)} = \varphi_k^{(c)} + \eta\Delta\varphi_k$$

Here η is the learning rate; tend to small values (e.g., 0.01) to prevent over-stepping

Dictionary is normalized after updating for a batch

Algorithm Summary and Issues

- **Algorithm Summary:**
  - Dictionary is initialized with centroids from K-means; could be initialized with data / library spectra
  - Online batch Hebbian / Gradient descent is used to update dictionary
  - K-means applied to spectra in dictionary coefficient space
- **Algorithm Issues:**
  - Hebbian / Gradient descent may diverge; dependent on learning rate
  - In general, no known analytical way to set learning rate
- **Possible to apply method to atmospheric compensation**

LA-UR-17-xxxx, x/xx/2017

Application to Atmospheric Compensation

- **TEAS iteratively updates atmosphere:**
  - using all 128 pixels in each iteration,
  - using a total of approximately 80 iterations,
  - which means that SLiM-TES is performed 80*128 = **10240 times**
- **Alternate Method:**
  - select 10 pixels at each iteration,
  - apply SLiM-TES and update atmosphere as usual,
  - which potentially means 1000+ iterations could be performed while "using" thousands of different pixels