*Proceedings of the 17th International Conference
on Computational and Mathematical Methods
in Science and Engineering, CMMSE 2017
4–8 July, 2017.*

Cloud implementation of logistic regression for hyperspectral image classification

Juan Mario Haut[1], Mercedes Eugenia Paoletti[1], Abel Paz-Gallardo[2], Javier Plaza[1] and Antonio Plaza[1]

[1] *Department of Technology of Computers and Communications, University of Extremadura, Escuela Politecnica, Avda. de la Universidad s/n, Cáceres*

[2] *Extremadura Research Centre for Advanced Technologies, CETA-CIEMAT, Calle Sola, 1, 10200, Trujillo, Cáceres*

emails: juanmariohaut@unex.es, mpaolett@alumnos.unex.es, abelfrancisco.paz@ciemat.es, jplaza@unex.es, aplaza@unex.es

Abstract

Classification of remotely sensed hyperspectral images is a challenging task due the enormous amount of information comprised in these images, that contain hundreds of continuous spectral bands. This creates a need to develop new techniques for hyperspectral classification using high performance computing architectures. Despite the availability of multiple algorithms adapted to parallel environments (such as multicore computers or accelerators like field programmable gate arrays or graphics processing units, the application of cloud computing techniques has not been as widespread, although there are many potential advantages in exploiting cloud computing architectures for distributed hyperspectral image analysis. In this paper, we present a cloud implementation (developed using Apache Spark) of a successful technique for hyperspectral image classification: the multinomial logistic regression probabilistic classifier. Our experimental results suggest that cloud computing architectures allow for the efficient classification of large hyperspectral image data sets.

Key words: Hyperspectral imaging, multinomial logistic regression, cloud computing, Apache Spark.

1 Introduction

Remotely sensed hyperspectral imaging is a popular technique for Earth observation (EO) [1], which allows for the simultaneous collection of images (at different wavelength channels) for the same area on the surface of the Earth. A characteristic of hyperspectral imagers is that they can collect data in thousands of narrow, contiguous spectral bands [2], providing so-called hyperspectral image data cubes [3].

An important property of hyperspectral instruments is their ability to acquire a complete reflectance spectrum for each pixel in the image (called contiguous spectral curves or spectral signatures). These signatures allow us to accurately distinguish different physical materials. For instance, the NASA's Jet Propulsion Laboratory's Airbone Visible/Infrared Imaging Spectrometer (AVIRIS) [4] measures the solar reflected spectrum from 0.4μm to 2.5μm at intervals of 0.01μm. The EO-1 Hyperion imaging spectrometer also collects bands in the range of 0.4μm to 2.5μm (more than 200 bands in both cases) [5, 6]. Several new satellite mission that will be soon operative and ready to collect data in a very similar spectral range. For instance, the German Environmental Mapping and Analysis Program (EnMAP [7]) is expected to collect data in the range 0.42μm to 2.45μm, as well as the Italian PRISMA program [8]. Other spectrometers acquire hyperspectral images in other regions of the spectrum, for instance the Reflective Optics System Imaging Spectrometer (ROSIS) takes images with a spectral range from 0.43μm to 0.96μm [9].[2].

Hyperspectral imaging has proved to be useful over a wide range of applications, such as agriculture, forestry, geology, ecological monitoring and disaster monitoring [10, 6]. However, due to the great dimensionality of hyperspectral data cubes, analysis techniques exhibit significant requirements in terms of storage and data processing [11, 12]. Therefore, the development of techniques that are computationally efficient becomes critical [6, 13, 14, 15].

Many efforts have been made within the field of hyperspectral image classification, both supervised and unsupervised [16]. Supervised techniques have been generally more popular due to their higher classification accuracy, but they require sufficient training information in order to perform properly. One of the supervised classifiers that can perform more accurately in the presence of limited training samples is the multinomial logistic regression (MLR) [17]. However, this classifier is computationally expensive, and available implementations have not considered the possibility of using cloud computing architectures [18]. These platforms can be greatly beneficial for hyperspectral image classification due to their advanced capabilities for internet-scale, service-oriented and high-performance computing. Specifically, the use of cloud computing for the classification of large hyperspectral data repositories can be considered a natural solution and an evolution of previously developed techniques for other kinds of computing platforms [19]. Still, there are few efforts in the recent literature oriented to the exploitation of cloud computing infrastructure for hyperspectral imaging techniques.

This work explores the possibility of using a distributed framework for classification of massive hyperspectral images based on cloud computing architectures. In particular we have focused on the discriminative MLR classifier [17] to demonstrate the applicability of utilizing cloud computing technologies to efficiently perform distributed classification of hyperspectral data.

The remainder of the paper is organized as follows. Section 2 first presents the theoretical principles of the MLR method (section2.1). Then, it describes our distributed framework design for this classifier (section 2.2). Finally, it describes our cloud implementation in detail (section 2.3). Section 3 validates the proposed cloud MLR algorithm by comparing it with other implementations. Finally, section 4 concludes with some remarks and hints at plausible future research lines.

2 Methodology

2.1 Multinomial Logistic Regression

To understand the operations of the MLR, we first we need to describe how logistic regression (LR) works. Given a collection of n linear-separable numeric samples $X = \{x_1, ..., x_n\}$ where each $x_i \in \mathbb{R}^d$, $x_i = [x_{i,1}, ..., x_{i,d}]$, the goal of classification methods is to categorize each x_i into a class or category y_i of those available in $Y = \{y_1, ..., y_k\}$, with $k < n$. But, in contrast to other classification methods, LR does not try to predict the value of a x_i given a set of inputs. Instead, the output is a probability that the input x_i belongs to a certain class y_i. It would be 0 when x_i does not belong to y_i and 1 if x_i belongs to y_i. Suppose that $x_i = [x_{i,1}, ..., x_{i,d}]$ and $Y = \{-, +\}$, LR assumes that the input d-space can be separated into two regions by a linear boundary: $\beta_0 + \beta_1 x_{i,1} + ... + \beta_d x_{i,d}$. This function outputs a value in $(-\infty, \infty)$ given an input data point, x_i[1]. To map the label probabilities with boundary values, LR applies log-odds functions[2] and calculates the predicted probabilities as $P(y_i = +|x_i, \beta) = \frac{\exp(\beta_0 + \sum_{j=1}^{d} \beta_j x_{i,j})}{1+\exp(\beta_0 + \sum_{j=1}^{d} \beta_j x_{i,j})}$. The goal of LR is to estimate the coefficients $\beta = \{\beta_0, \beta_1, ..., \beta_d\}$ through *maximum likelihood estimation* (MLE), that optimizes β in order to maximize the *log likelihood* (LL, i.e. the log odds).

MLR extends the binary problem of LR to any number of classes, $k > 2$. Specifically,

[1] If x_i lies in the region defined by the $+$ class, the function's value is positive in the range $(0, +\infty)$ and its probability $P(y_i = +|x_i, \beta)$ is in $(0.5, 1]$. If x_i lies in the region defined by the $-$ class, the function's value is negative in the range $(-\infty, 0)$ and its probability $P(y_i = -|x_i, \beta) = 1 - P(y_i = +|x_i, \beta)$ is in $[0, 0.5)$. Finally if we do not know whats x_i is, the function's value is 0, and probabilities of being $+$ or $-$ are exactly 0.5

[2] Given a probability function $P(x) \in [0, 1]$, the odds ratio is defined as $OR(x) = \frac{P(x)}{1-P(x)} \in [0, +\infty)$. Applying the logarithm to $OR(x)$ we obtain $\log(OR(x)) \in (-\infty, \infty)$. If the log-odds is $\log(\frac{P(x)}{1-P(x)}) = a + bx$, we can transform it into $\frac{P(x)}{1-P(x)} = \exp(a + bx) \rightarrow P(x) = \frac{\exp(a+bx)}{1+\exp(a+bx)}$, i.e the logistic function

MLR selects one category as the baseline, e.g. the k-th class, and calculates the regression coefficients for the $l = 1, ..., k-1$ non-baseline categories $(\beta^{(1)}, ..., \beta^{(k-1)}$ with $\beta^{(l)} = \{\beta_0^{(l)}, ..., \beta_d^{(l)}\})$ against the baseline class. The predicted probabilities are extended to $P(y_i = l|x_i, B) = \frac{\exp\left(\beta_0^{(l)} + \sum_{j=1}^{d} \beta_j^{(l)} x_{i,j}\right)}{1 + \sum_{l'=1}^{k-1}\left(\exp\left(\beta_0^{(l')} + \sum_{j=1}^{d} \beta_j^{(l')} x_{i,j}\right)\right)}$, where B is the $(d+1) \times (k-1)$ matrix of all the regression coefficients. The goal of MLR is then to estimate B given the samples dataset X and the categories Y, by minimizing the optimization function:

$$f(B; X, Y) = -\sum_{i=1}^{n} \log P(y_i|x_i B) + \frac{\lambda}{2} \sum_{j=1}^{d} \sum_{l=1}^{k-1} |\beta_j^{(l)}|^2, \tag{1}$$

where λ is a regularization term added in order to mitigate the overfitting problem.

2.2 Distributed framework design

To create our distributed environment, two frameworks have been used: 1) *OpenStack*[3] and 2) *Apache Spark*[4]. Each one of them will be in charge of the correct execution of the architecture in two aspects:

- On the one hand, *OpenStack* provides Infrastructure as a Service (IaaS), abstracting and manages the physical machines that will give the support to the virtual machines. *OpenStack* works like a cloud operating system that controls large pools of compute, storage, and networking resources throughout a datacenter, all managed through a dashboard that gives administrators control while empowering their users to provision resources through a web interface.

- On the other hand, *Apache Spark* is a distributed in-memory processing framework that works over the virtual machines (managed by *OpenStack*) and allows to implement MapReduce[5] distributed programming model [18]. Also, *Apache Spark* implements a fault-tolerant abstraction for in-memory cluster computing, and provides fast and general data processing on large distributed platforms. It supports simple one-pass computations and can also be extended to the case of multi-pass, iterative algorithms.

[3] https://wiki.openstack.org/wiki/Main_Page

[4] http://spark.apache.org/

[5] The MapReduce model takes full advantage of the high-performance capabilities provided by cloud computing architectures. The operation is easy: a task is processed by two distributed operations, map and reduce. The datasets are organized as key/value pairs, and the map function processes a key/value pair to generate a set of intermediate pairs, dividing a task into several independent subtasks to be run in parallel. The reduce function is in charge of processing all intermediate values associated with the same intermediate key, then collecting all the subtask results to gather the result for the whole task.

The full architecture of our newly developed system for hyperspectral image classification is shown in Fig. 1. In Fig. 2 we display the services offered by *OpenStack* and the *Apache Spark* framework used.

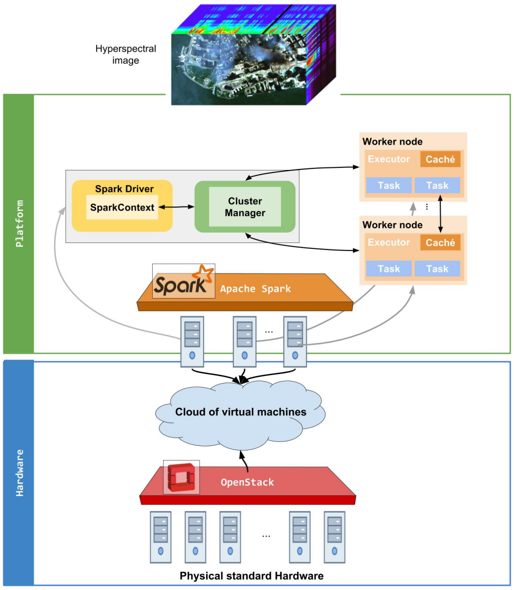


Figure 1: Integrated OpenStack and Apache Spark framework for Logistic Regression.

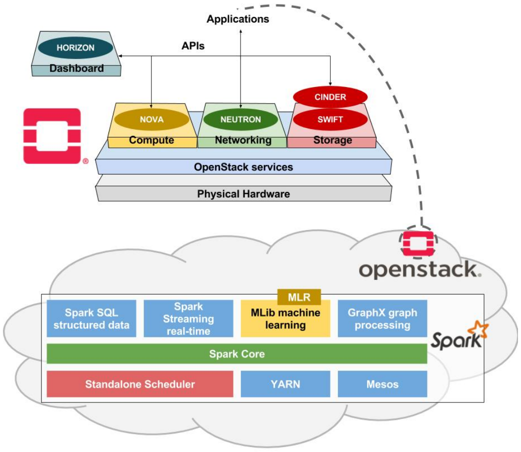


Figure 2: Description of the proposed Apache Spark and Open Stack architecture.

2.3 Cloud Implementation

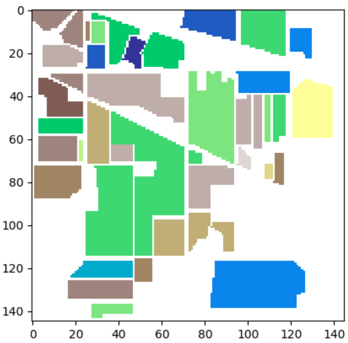
Proposed cloud MLR divides the execution between one master (Spark driver) and several slaves (Spark executors). The driver prepares the environment (reserves resources), launches the executors and initializes β. Each executor loads their corresponding patch of the hyperspectral image and for each data calculates the loss function and the gradient (Map) that are summed up (Resume) and sent back to the driver. The driver calculates the loss and the gradient of regularizer and plugs the gradient and loss of the model and the regularizer into optimizer to get the new β. If the loss is less than the stopping criterion, the algorithm ends.

To execute our cloud implementation of MLR we need several parameters: the number of classes (k), the input training data (a percentage of hyperspectral image's pixels with which MLR will train), the number of maximum iterations and the tolerance of the L-BFGS optimizer. On the other hand, the input data is regularized by L2.

3 Experimental results

3.1 Experimental Configuration

In order to evaluate the performance of the adopted MLR implementation, we use a hardware environment composed by a Intel(R) Xeon(R) CPUs E5430 @ 2.66GHz (8 cores), 16 GB RAM, Shared storage, NetApp FAS3140. Virtual nodes have two virtual CPUs, 4GB of RAM and 40 GB hard disk each. In addition, we have developed a parallel version of the algorithm for comparative purposes. This version has been implemented on a paltform with Intel(R) Core(TM) i7-4790 CPUs @ 3.60GHz (8 cores), 16 GB RAM, SanDisk SDSSDA240G. In our experiments, we used Ubuntu 14.04 x64 LTS as operating system. For the parallel version of MLR, a virtual machine of the cluster with 2 cores, 4GB of RAM and 40GB of hard disk, and the same software configuration has been used.



Color	Land cover type	Samples
	Background	10776
■	Alfalfa	46
■	Corn-notill	1428
■	Corn-min	830
■	Corn	237
■	Grass/Pasture	483
■	Grass/Trees	730
■	Grass/pasture-mowed	28
■	Hay-windrowed	478
■	Oats	20
■	Soybeans-notill	972
■	Soybeans-min	2455
■	Soybean-clean	593
■	Wheat	205
■	Woods	1265
■	Bldg-Grass-Tree-Drives	386
■	Stone-steel towers	93
	Total samples	21025

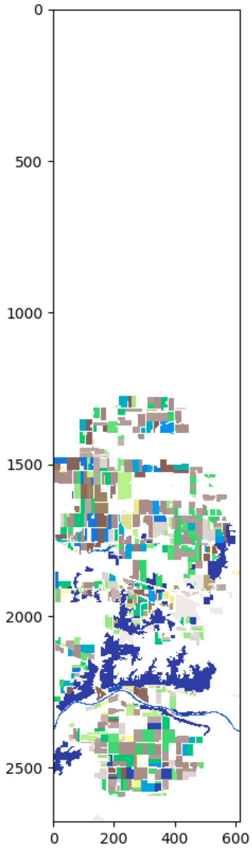
Figure 3: Original ground-truth of Small Indian Pines scene, with class labels and original number of samples per class.

3.2 Hyperspectral data sets

In our experiments, we use two different hyperspectral images. The first one was collected by AVIRIS [4] in 1992 over a set of agricultural fields with regular geometry and irregular patches of forest in Northwestern Indiana (Indian Pines image). This scene has 145 × 145 pixels with 224 spectral bands in the range 0.4-2.5μm, with 0.01μm of spectral resolution, 0.020μm moderate spatial resolution and 16 bits of radiometric resolution. 4 zero bands plus 20 bands with lower signal-to-noise ratio (SNR) have been removed, retaining 200 spectral

ISBN: 978-84-617-8694-7

channels. The data has 16 ground-truth classes (Fig. 3). Also, we use a larger version of the Indian Pines scene, with a size of 2678 × 614 pixels. It was collected over the same area, but spanning a much larger extent. It contains 220 spectral bands an the total number of classes is 58 (Fig. 4).



Color	Land cover type	Samples	Color	Land cover type	Samples
	Background	1310047	⬜	BareSoil	57
⬜	Buildings	17195	⬜	Concrete/Asphalt	69
⬜	Corn	17783	⬜	Corn?	158
⬜	Corn-EW	514	🟫	Corn-NS	2356
🟫	Corn-CleanTill	12404	🟫	Corn-CleanTill-EW	26486
🟫	Corn-CleanTill-NS	39678	🟫	Corn-CleanTill-NS-Irrigated	800
🟫	Corn-CleanTilled-NS?	1728	🟫	Corn-MinTill	1049
🟫	Corn-MinTill-EW	5629	🟫	Corn-MinTill-NS	8862
🟫	Corn-NoTill	4381	🟫	Corn-NoTill-EW	1206
🟫	Corn-NoTill-NS	5685	🟫	Fescue	114
🟨	Grass	1147	🟨	Grass/Trees	2331
🟨	Grass/Pasture-mowed	19	🟨	Grass/Pasture	73
🟨	Grass-runway	37	🟨	Hay	1128
🟨	Hay?	2185	🟨	Hay-Alfalfa	2258
🟨	Lake	224	🟨	NotCropped	1940
🟨	Oats	1742	🟨	Oats?	335
🟩	Orchard	39	🟩	Pasture	10386
🟩	pond	102	🟩	Soybeans	9391
🟩	Soybeans?	894	🟩	Soybeans-NS	1110
🟩	Soybeans-CleanTill	5074	🟩	Soybeans-CleanTill?	2726
🟩	Soybeans-CleanTill-EW	11802	🟩	Soybeans-CleanTill-NS	10387
🟩	Soybeans-CleanTill-Drilled	2242	🟩	Soybeans-CleanTill-Weedy	543
🟩	Soybeans-Drilled	15118	🟩	Soybeans-MinTill	2667
🟦	Soybeans-MinTill-EW	1832	🟦	Soybeans-MinTill-Drilled	8098
🟦	Soybeans-MinTill-NS	4953	🟦	Soybeans-NoTill	2157
🟦	Soybeans-NoTill-EW	2533	🟦	Soybeans-NoTill-NS	929
🟦	Soybeans-NoTill-Drilled	8731	🟦	Swampy Area	583
🟦	River	3110	🟦	Trees?	580
🟦	Wheat	4979	🟦	Woods	63562
🟦	Woods?	144			
	Total samples 1644292				

Figure 4: Original ground-truth of Big Indian Pines scene, with class labels and original number of samples per class.

3.3 Performance Evaluation

To evaluate the performance of our cloud implementation of MLR, we make a comparison between the cloud version and a multi-core parallel implementation of the same MLR algorithm. Our experiments have been launched for each hyperspectral image, using different training percentages (15%, 25% and 50% of the training samples available in each class). For the cloud version we have considered 2, 4 and 8 distributed nodes. The optimal number of iterations and λ value are obtained by cross-validation. Each configuration has been repeated five times, and the results reported are the average across the executions for statistical consistency.

Training percentage	**Parallel**	**Distributed**		
		2 nodes	**4 nodes**	**8 nodes**
	Time Execution			
5%	1.82 (2.143)	16.34 (1.876)	16.90 (2.102)	19.32 (2.201)
15%	4.48 (2.051)	18.65 (2.039)	20.01 (1.872)	23.50 (2.231)
25%	6.84 (1.942)	21.04 (2.052)	22.61 (2.312)	25.32 (2.214)
50%	13.42 (2.161)	30.34 (1.911)	32.39 (1.891)	32.84 (1.857)
	Accuracy Results			
5%	68.25 (1.0)	67.11 (0.9)	68.19 (1.0)	67.02 (0.8)
15%	77.15 (0.8)	75.74 (0.8)	75.00 (1.0)	73.77 (0.9)
25%	79.58 (0.9)	77.27 (0.9)	76.69 (1.1)	76.58 (0.9)
50%	82.05 (1.4)	78.40 (1.1)	79.36 (1.2)	79.11 (1.0)

Table 1: Average processing time, classification accuracy (and standard deviation) for different implementations of multinomial logistic regression using the Small Indian Pines Image.

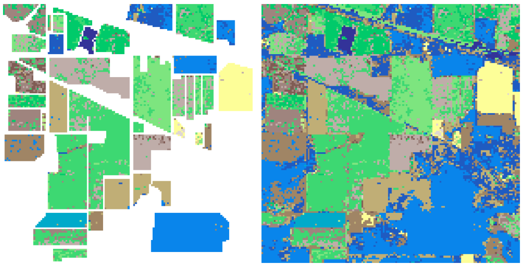


Figure 5: Classification results for the Small Indian Pines image: classification map without background (left) and classification map with background (right), obtained using 15% training.

Table 1 shows the results obtained by different implementations of the MLR using the Small Indian Pines dataset. The classification accuracies are worse as we add nodes to the cluster, due to the lack of data within the nodes, and the processing times tend to be worse too. This is because the nodes have not enough data to optimize. Fig. 5 shows the obtained classification result. As mentioned before, the processing times increase slightly as we add nodes to the distributed environment (with 8 nodes the weight of the communication prevents to improve the speed up). However, the increase of training samples affects non-uniformly the obtained classification results (this depends on the calculation of

λ).

Training percentage	**Parallel**	**Distributed**		
		2 nodes	**4 nodes**	**8 nodes**
	Time Execution			
5%	163.83 (2.620)	169.12 (2.527)	106.22 (1.403)	79.33 (2.300)
15%	400.20 (9.601)	364.00 (4.341)	218.46 (5.231)	127.36 (3.053)
25%	598.31 (3.310)	556.48 (3.254)	327.106 (1.900)	181.65 (1.024)
50%	1208.98 (11.708)	1087 (9.865)	584.65 (5.651)	374.42 (3.643)
	Accuracy Results			
5%	44.52 (1.1)	42.24 (0.9)	42.47 (0.9)	42.48 (1.0)
15%	45.39 (1.2)	43.06 (1.1)	43.02 (1.3)	43.88 (1.2)
25%	45.52 (1.1)	43.54 (1.2)	43.63 (0.9)	43.95 (1.1)
50%	45.65 (1.3)	44.29 (1.3)	44.34 (1.2)	44.96 (1.3)

Table 2: Average processing time, classification accuracy (and standard deviation) for different implementations of multinomial logistic regression using the Big Indian Pines Image.

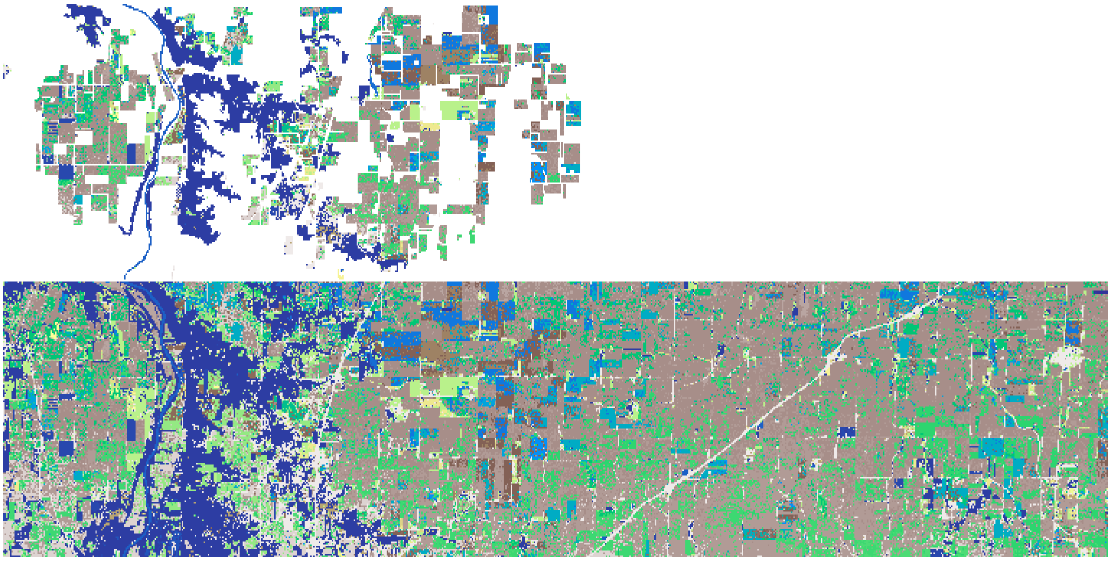


Figure 6: Classification results for the Big Indian Pines image: classification map without background (top) and classification map with background (bottom), obtained using 15% training.

On the other hand, Table 2 shows the results obtained by MLR using the Big Indian Pines dataset. As we can see, as we increase the number of nodes, time decreases. The highest speed up is achieved with 8 nodes (a 3.29), while the accuracy results are quite acceptable given the complexity of this scene (although is less than in the parallel version). These results reveal that our cloud implementation benefits from the availability of large data volumes and complex analysis scenarios, such as the one given by the Big Indian Pines scene. The complexity of the classification of this scene can be appreciated in Fig. 6.

4 Conclusions and Future Lines

In this paper, we have discussed the possibility of exploiting cloud computing architectures for hyperspectral image classification. As a case study, we have presented a cloud computing implementation of the multinomial logistic regression classifier (a technique that has been used successfully for hyperspectral data interpretation) on the Apache Spark and Openstack platforms. Our experimental results show the effectiveness of the proposed distributed implementation with large hyperspectral datasets (i.e., the proposed technique provides satisfactory results with very large images and complex analysis scenarios given by a large numbers of samples and classes). As future work, we will implement other techniques for hyperspectral image classification using cloud computing platforms, as it is our feeling that there are many open and unexplored possibilities for the exploitation of these kind of platforms in remotely sensed hyperspectral imaging.

Acknowledgements

This work has been supported by Ministerio de Educación (Resolución de 26 de diciembre de 2014 y de 19 de noviembre de 2015, de la Secretaría de Estado de Educación, Formación Profesional y Universidades, por la que se convocan ayudas para la formación de profesorado universitario, de los subprogramas de Formación y de Movilidad incluidos en el Programa Estatal de Promoción del Talento y su Empleabilidad, en el marco del Plan Estatal de Investigación Científica y Técnica y de Innovación 2013-2016). This work has also been supported by Junta de Extremadura (decreto 297/2014, ayudas para la realización de actividades de investigación y desarrollo tecnológico, de divulgación y de transferencia de conocimiento por los Grupos de Investigación de Extremadura, Ref. GR15005).

This work was partially supported by the computing facilities of Extremadura Research Centre for Advanced Technologies (CETA-CIEMAT), funded by the European Regional Development Fund (ERDF). CETA-CIEMAT belongs to CIEMAT and the Government of Spain.

References

[1] D. Chutia, D. K. Bhattacharyya, K. K. Sarma, R. Kalita, and S. Sudhakar. Hyperspectral Remote Sensing Classifications: A Perspective Survey. *Transactions*

in GIS, 20(4):463–490, 2016.

[2] Alexander F H Goetz, Gregg Vane, Jerry E Solomon, and Barrett N Rock. Imaging Spectrometry for Earth Remote Sensing. *Science*, 228(4704):1147–1153, 1985.

[3] Chein-I Chang. *Hyperspectral Imaging: Techniques for Spectral Detection and Classification*. Springer US, 2003.

[4] Robert O. Green, Michael L. Eastwood, Charles M. Sarture, Thomas G. Chrien, Mikael Aronsson, Bruce J. Chippendale, Jessica A. Faust, Betina E. Pavri, Christopher J. Chovit, Manuel Solis, Martin R. Olah, and Orlesa Williams. Imaging spectroscopy and the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS). *Remote Sensing of Environment*, 65(3):227–248, 1998.

[5] Amin Beiranvand Pour and Mazlan Hashim. ASTER, ALI and Hyperion sensors data for lithological mapping and ore minerals exploration. *SpringerPlus*, 3(1):130, 2014.

[6] A. Plaza, J. Plaza, A. Paz, and S. Sanchez. Parallel Hyperspectral Image and Signal Processing. *IEEE Signal Processing Magazine*, 28(3):119–126, 2011.

[7] H Kaufmann, L Guanter, K Segl, S Hofer, K.-P Foerster, T Stuffler, A Mueller, R Richter, H Bach, and P Hostert. Environmental Mapping and Analysis Program (EnMAP) – Recent Advances and Status. *IEEE International Geoscience & Remote Sensing Symposium, IGARSS*, 4:109–112, 2008.

[8] C. Galeazzi, A. Sacchetti, A. Cisbani, and G. Babini. The PRISMA Program. In *IGARSS 2008 - 2008 IEEE International Geoscience and Remote Sensing Symposium*, pages IV – 105–IV – 108, 2008.

[9] B. Kunkel, F. Blechinger, R. Lutz, R. Doerffer, and H. van der Piepen. ROSIS (Reflective Optics System Imaging Spectrometer) - A candidate instrument for polar platform missions. In J. Seeley and S. Bowyer, editors, *Optoelectronic technologies for remote sensing from space*, pages 134–141, 1988.

[10] Mustafa Teke, Hüsne Seda Deveci, Onur Haliloğlu, Sevgi Zübeyde Gürbüz, and Ufuk Sakarya. A Short Survey of Hyperspectral Remote Sensing Applications in Agriculture. In *Recent Advances in Space Technologies (RAST)*, 2013.

[11] Antonio Plaza, Javier Plaza, and David Valencia. Impact of platform heterogeneity on the design of parallel algorithms for morphological processing of high-dimensional image data. *Journal of Supercomputing*, 40(1):81–107, 2007.

[12] Antonio Plaza, Jon Atli Benediktsson, Joseph W Boardman, Jason Brazile, Lorenzo Bruzzone, Gustavo Camps-Valls, Jocelyn Chanussot, Mathieu Fauvel, Paolo Gamba, Anthony Gualtieri, Mattia Marconcini, James C Tilton, and Giovanna Trianni. Recent advances in techniques for hyperspectral image processing. *Remote Sensing of Environment*, 113(1):S110–S122, 2009.

[13] Javier Setoain, Manuel Prieto, Christian Tenllado, and Francisco Tirado. GPU for Parallel On-Board Hyperspectral Image Processing. *International Journal of High Performance Computing Applications*, 2008.

[14] Carlos González, Sergio Sánchez, Abel Paz, Javier Resano, Daniel Mozos, and Antonio Plaza. Use of FPGA or GPU-based architectures for remotely sensed hyperspectral image processing. *Integration, the VLSI Journal*, 46(2):89 – 103, 2013.

[15] Antonio J Plaza, Chein-I Chang, Liping Di, and Yuqi Bai. *High Performance Computing in Remote Sensing Book Review Book Review.* Chapman & Hall/CRC Press, Computer & Information Science Series, Boca Raton, Florida, 2008.

[16] P. Ghamisi, J. Plaza, Y. Chen, J. Li, and A. Plaza. Advanced Supervised Spectral Classifiers for Hyperspectral Images: A Review. *IEEE Geoscience and Remote Sensing Magazine*, 5(1):8–32, 2017.

[17] Balaji Krishnapuram, Lawrence Carin, Mário A T Figueiredo, and Alexander J. Hartemink. Sparse multinomial logistic regression: Fast algorithms and generalization bounds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6):957–968, 2005.

[18] Zebin Wu, Yonglong Li, Antonio Plaza, Jun Li, Fu Xiao, and Zhihui Wei. Parallel and Distributed Dimensionality Reduction of Hyperspectral Data on Cloud Computing Architectures. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 9(6):2270–2278, 2016.

[19] J. A. Martínez, E. M. Garzón, A. Plaza, and I. García. Automatic tuning of iterative computation on heterogeneous multiprocessors with ADITHE. *Journal of Supercomputing*, 58(2):151–159, 2011.