DISTRIBUTED BOOSTING FOR CLOUD DETECTION

M. Le Goff [(1,2)], *J.-Y. Tourneret*[(1)], *H. Wendt*[(1)], *M. Ortner*[(2)], *M. Spigai*[(2)]

[(1)] IRIT/ENSEEIHT/TéSA, University of Toulouse and [(2)] IRT Saint Exupery, Toulouse, France

ABSTRACT

The SPOT 6-7 satellite ground segment includes a systematic and automatic cloud detection step in order to feed a catalogue with a binary cloud mask and an appropriate confidence measure. In order to significantly improve the SPOT cloud detection and get rid of frequent manual re-labelings, we study a new automatic cloud detection technique that is adapted to large datasets. The proposed method is based on a modified distributed boosting algorithm. Experiments conducted using the framework Apache Spark on a SPOT 6 image database with various landscapes and cloud coverage show promising results.

***Index Terms*—** Cloud detection, remote sensing, big data, distributed processing, boosting

1. INTRODUCTION

The generation of cloud masks associated with remote sensing images is an important issue in order to feed catalogues not only with images but also with cloud information. This problem has received considerable interest in the literature, cf., e.g., [1, 2]. Cloud information is of central importance for image catalogue customers as well as for operators in order to perform fast reprogramming/rescheduling in case of too cloudy acquisitions. SPOT 6-7 satellites belong to high resolution (2.5m) optical Earth observation systems from the SPOT satellite family. Upon request, the SPOT catalogue interface returns product meta-information, a cloud mask obtained from a semi-automatic pipeline, and a low resolution version of the requested image, called album version. Currently existing cloud detectors are in majority based on morphological operations such as shadow matching [1] or on physical models specific to clouds [2]. However, it has been observed that these detectors lack generalization capabilities and robustness since they are satellite-dependent and can provide poor performance for specific images. Cloud detection methods based on raw data (instead of morphological features or physical models) are expected to be more robust, e.g., able to discriminate clouds from other classes for any satellite and for any scenario. However, a central aspect of any powerful automated cloud detection system is to rely on a large size labelled database containing all representative landscapes avoiding any expert intervention. The problem is therefore to learn a robust classifier from a large database containing ground truth masks. Most classical data mining approaches are designed to process data that can fit in the memory of one single machine. Thus, they cannot handle large-scale data sets, which require some form of distributed processing by multiple machines.

Recent research efforts have focussed on the development of distributed versions of standard classification methods [3–6]. These methods include support vector machines (SVMs) that have been widely used in many remote sensing applications. However, the hyperparameter choice is critical for a good classification performance with SVMs [6]. Random forests are also known as state-of-the-art algorithms for classification because of their simplicity and efficiency. However, their distributed implementation is complex because of its underlying assumptions [4]. Boosting algorithms require less operational assumptions and are known to provide high classification performance with a small computational cost. They are well suited to distributed implementations and several architectures have been proposed [3, 5, 7]. Yet, these architectures assume that the data are equally distributed among the different machines (i.e., each data subset is representative of the full data set), which is difficult to satisfy in practice.

The goal of this paper is to propose and study a novel distributed boosting algorithm that is capable of processing large-scale datasets in order to solve the cloud detection problem. The contributions are twofold. First, we address the challenge of handling large-scale databases (and the induced computational complexity) by defining a distributed boosting algorithm whose complexity scales linearly with the sample size. The proposed algorithm inherits the robustness of the original boosting algorithm, without any condition on the data distribution. Second, we study the application of the proposed algorithm to a large-scale database (i.e, containing more than 10000 images) of SPOT 6 album images with associated cloud masks. The paper is organized as follows. Section 2 presents the images and features considered in this study. Section 3 investigates the proposed distributed boosting algorithm. Simulation results are presented and discussed in Section 4, and conclusions are given in Section 5.

The study was conducted in the frame of OCE contract managed by IRT Saint Exupery, Toulouse (France) and sponsored by the following funding partners: Agence Nationale de la Recherche, Airbus Defence and Space, Thales Alenia Space, Airbus Defence and Space Geoinformation, Telespazio, Airbus Group Innovation, LAAS-CNRS, IRIT.

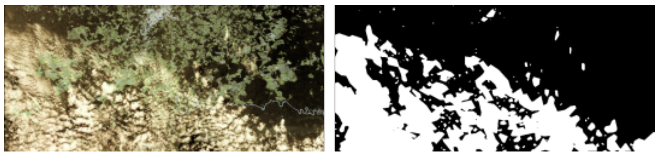

Fig. 1. Example of an image (left) and its cloud mask (right).

2. IMAGES AND FEATURES

Album SPOT 6 images consist of 4-channel images acquired in the blue, green, red and near infrared wavelength domains. Their spatial resolution is significantly smaller than those of the full resolution images in order to reduce memory requirements, while the radiometric resolution is preserved at 12 bits. A database of more than 10000 SPOT 6 album images, containing a large and representative variety of cloud coverages and landscapes, has been provided by Airbus Defense and Space. The images have been corrected for radial distorsion, internal sensor geometry and radiometric distorsion.

As a preprocessing step, image features are computed from the 4 channels of the album images. Here, we mainly focus on the classification algorithm part and therefore consider simple and well-established pixel-wise descriptors defined as band ratios (i.e., the ratio of the image intensities of two channels). Band ratios have been considered previously for remote sensing applications and in particular for cloud detection [8]. These features have the advantage of being simple to compute and are independent of illumination conditions, which may vary significantly within each image. For the 4 channels of the SPOT 6 images, $\binom{4}{2} = 6$ band ratios can be computed. Moreover, we propose here to use multi-scale features obtained by computing band-ratios for 3 different spatial resolutions (60m, 120m and 240m), leading to a total of $d = 18$ features per pixel.[1] The use of additional features will be studied in future work, including, for instance, pixel-wise descriptors such as NDCI and NDSI [8], texture features or object-level features (at the price, though, of additional parameter tuning).

3. PROPOSED BOOSTING ALGORITHM

3.1. Machine learning problem and notations

Denote as $\mathcal{S} = \{(\boldsymbol{x}_i, y_i), i = 1, ..., N\}$ a training set containing N feature vectors $\boldsymbol{x}_i \in \mathbb{R}^d$ (here, normalized to $[-1, 1]^d$) and their corresponding labels $y_i \in \pm 1$ (where $y_i = 1$ means that the pixel corresponds to a cloud). To each training sample $(\boldsymbol{x}_i, y_i)$, we associate a weight w_i which is used to quantify its relevance. The weights are concatenated into a weight vector $\boldsymbol{w} = (w_1, ..., w_N)$. The goal of a supervised machine learning algorithm is to identify the classifier $f : \mathbb{R}^d \mapsto \pm 1$ from a (possibly infinite) set of binary classifiers (detectors) $\mathcal{F}$ that minimizes the training error (or empirical risk) defined as

$$\epsilon(\mathcal{S}, f, \boldsymbol{w}) = \sum\nolimits_{i=1}^{N} w_i L(y_i, f(\boldsymbol{x}_i)) \qquad (1)$$

where L is a given loss function (here, the $0/1$ loss defined as $L(y_i, f(\boldsymbol{x}_i)) = 1$ if $y_i \neq \boldsymbol{x}_i$ and $L(y_i, f(\boldsymbol{x}_i)) = 0$ otherwise). The criterion that is used to build the decision rule is hence the empirical risk ϵ. Note that the classification performance obtained for test examples unseen during the training phase is referred to as generalization performance.

3.2. The boosting algorithm

The distributed algorithm described in this section is a modification of the classical boosting algorithm, which has shown very good performance in many practical applications (see, e.g., [9] for details). It is sketched in Algo. 1. The goal of boosting is to construct a decision function f as a linear combination of weak learners[2] $h(\boldsymbol{x})$

$$f_T(\boldsymbol{x}) = \text{sign}\left(\sum\nolimits_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})\right).$$

Here, we consider weak learners defined as the following "decision stumps"

$$h^{j,\gamma}(\boldsymbol{x}_i) = 1 \text{ if } \boldsymbol{x}_i^j \geq \gamma \text{ and } h^{j,\gamma}(\boldsymbol{x}_i) = 0 \text{ otherwise} \qquad (2)$$

where $\boldsymbol{x} = (\boldsymbol{x}_1, ..., \boldsymbol{x}_d)$, $\gamma \in \Gamma^j$ is a threshold for feature j. At each iteration t, the algorithm selects the weak learner h_t that has minimal empirical risk ϵ_t and adds it to the decision function weighted by α_t that reflects the overall performance of the weak learner (cf., lines 2–4 of Algo. 1). In addition, the weights w_i^t are updated depending on how $\boldsymbol{x}_i$ is difficult to classify (cf., lines 5–6 of Algo. 1), i.e., the weights used for the following iteration $t+1$ are increased for misclassified samples and decreased for correctly classified samples.

```
  Data: $\mathcal{S} = \{(\boldsymbol{x}_i, y_i), i = 1, ..., N\}$
  Result: $f_T(\cdot) = \text{sign}\big(\sum_{t=1}^{T} \alpha_t h_t(\cdot)\big)$
1 while $t \leq T$ do
2     $h_t = \underset{j \in \{1,...,d\}, \gamma \in \Gamma^j}{\text{argmin}} \epsilon(S, h^{j,\gamma}, \boldsymbol{w}^t)$;
3     $\epsilon_t = \sum_{i=1}^{N} w_i^t L(y_i, h_t(\boldsymbol{x}_i))$;
4     $\alpha_t = 0.5 \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$;
5     $z_i^{t+1} = w_i^t \exp[\alpha_t y_i h_t(\boldsymbol{x}_i)]$;   $Z_{t+1} = \sum_{i=1}^{N} z_i^{t+1}$;
6     $w_i^{t+1} = z_i^{t+1} / Z_{t+1}$;
7 end
```
Algorithm 1: Boosting algorithm.

The main motivations for using the boosting algorithm are its ability to control the training error, which is upper bounded

[1]The resolution of the album images is 60m, and the 120m and 240m resolution images have been obtained by downsampling.

[2]By "weak learners", we mean classifiers that perform (slightly) better than random guessing.

by an exponential decrease at each iteration [9], and also its generalization performance since boosting performs a margin optimisation [10].

3.3. A distributed boosting algorithm

In order to distribute the computations on a cluster of machines, the classical boosting algorithm needs to be modified. One of the most efficient strategies is to test classifiers and aggregate their performance [3]. We propose here to define a large finite set of weak learners $h^{j,\gamma}$ induced by using a finite set of discrete thresholds belonging to Γ^j in (2) (in contrast to the the original algorithm where there are as many thresholds as samples). Here, the thresholds are equally-spaced in the interval $[-1, 1]$. With this modification, the training error of each weak learner for the training set $\mathcal{S}$ can, at each iteration, be determined by aggregating the training errors computed and broadcast for sub-sets of $\mathcal{S}$ in a distributed architecture. More precisely, the dataset is split and distributed among n machines. Denote as $\mathcal{S}^k$ the sub-datasets with $N_k = \text{card}(\mathcal{S}^k)$ elements associated with machine $\#k$, $k = 1, \ldots, n$, where $\mathcal{S} = \bigcup_{k=1,\ldots,n} \mathcal{S}^k$ and $\mathcal{S}^k \cap \mathcal{S}^l = \emptyset$ for $k \neq l$. Each machine k has a local set of weights, denoted as $\boldsymbol{w}_k^t$, that is associated with the samples in $\mathcal{S}^k$. At each iteration t, the distributed boosting algorithm computes for each machine k (independently of the other machines) the training error of each classifier $h^{j,\gamma}$, denoted as $\epsilon(\mathcal{S}^k, h^{j,\gamma}, \boldsymbol{w}_k^t)$. These training errors are then communicated and aggregated to evaluate the global performance of each classifier $h^{j,\gamma}$

$$\epsilon(\mathcal{S}, h^{j,\gamma}, \boldsymbol{w}^t) = \frac{1}{N} \sum\nolimits_{k=1}^{n} N^k \epsilon(\mathcal{S}^k, h^{j,\gamma}, \boldsymbol{w}_k^t) \qquad (3)$$

which is in turn used to select the classifier h_t as

$$h_t = \underset{j \in \{1,\ldots,d\}, \gamma \in \Gamma^j}{\text{argmin}} \epsilon(S, h^{j,\gamma}, \boldsymbol{w}^t)$$

cf., line 2 in Algorithm 2. The remaining steps of the algorithm consist of standard boosting weight update steps for each data subset $\mathcal{S}^k$ (cf., lines 4-6 of Algo. 1), which are again performed on each machine k independently. The resulting distributed boosting architecture is sketched in Fig. 2.

Remarks. The algorithm has an extremely simple and easy to implement structure, in contrast to, e.g., distributed random forests [4] or SVM [6]. Moreover, it is scalable since its overall complexity is linear with respect to the training set size (indeed, each single data instance will at most be considered $N_c \times T$ times, where $N_c = \prod_{j=1}^{d} \text{card}(\Gamma^j)$ is the total number of stumps). In addition, it yields a classifier that is independent of the composition of the sub-datasets $\mathcal{S}^k$ (which do hence not need to be representative of the full data set $\mathcal{S}$; indeed, each $\mathcal{S}^k$ could, e.g., contain only one single image without altering the final detector f_T since each weak learner is nonetheless tested on all training examples, cf., (3)). This is in contrast with and a major advantage over previously proposed distributed boosting algorithms: Among them, our algorithm is close to the PreWeak algorithm [3], which relies however on the independent execution of Adaboost on each individual machine in order to determine the thresholds (whose number hence grows linearly with the size of the data set, yielding an overall quadratic complexity). Similarly, the Distboost algorithm [5] makes use of a majority vote of decision trees that are learnt on each machine independently, which requires that each machine holds a representative sample of the full dataset. This is also the case for the Adasampling algorithm [3], in which Adaboost is executed independently on each machine in order to select a subset of "important" examples that are then fed to a centrally executed machine learning algorithm.

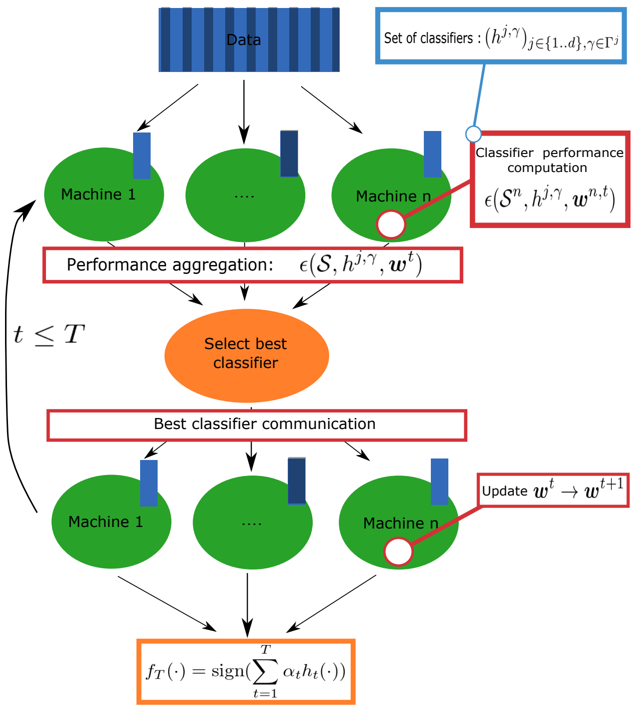
Data
Set of classifiers : $(h^{j,\gamma})_{j\in\{1..d\},\gamma\in\Gamma^j}$
Machine 1
....
Machine n
Classifier performance computation
$\epsilon(\mathcal{S}^n, h^{j,\gamma}, \boldsymbol{w}^{n,t})$
Performance aggregation: $\epsilon(\mathcal{S}, h^{j,\gamma}, \boldsymbol{w}^t)$
$t \leq T$
Select best classifier
Best classifier communication
Machine 1
....
Machine n
Update $\boldsymbol{w}^t \to \boldsymbol{w}^{t+1}$
$f_T(\cdot) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(\cdot))$

Fig. 2. Distributed boosting architecture.

4. EXPERIMENTS

The algorithm was implemented using the Python API of the framework Apache Spark. Spark enables efficient parallel computations with a set of high level operations (that automatically handle, e.g., work distribution and fault tolerance). An emulated cluster on a single workstation was used in our first experiments (40 cores, 128 GB RAM). However, the proposed implementation is scalable and independent of the cluster size and could be executed on a cluster in the cloud.

Simulation scenario. The training and generalization performances of the proposed distributed boosting algorithm have been computed using randomly selected subsets of 100 training images and 80 test images. The sizes of the training and

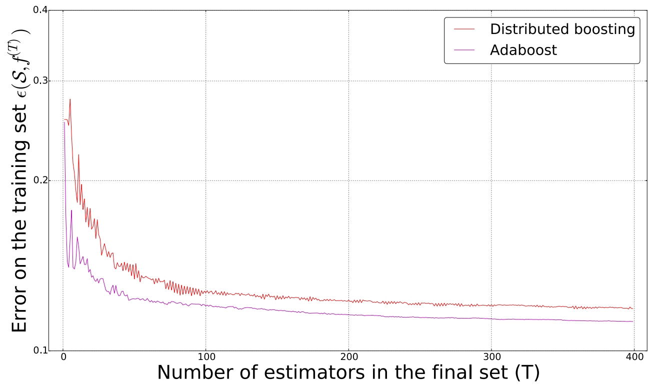

Fig. 3. Training error obtained on a training set of 100 images for different learning algorithms.

Learning model	Adaboost	Random Forest	Distributed boosting
Mean error on test data	27.6%	27.8%	28%

Table 1. Average generalization performance on a test set of 80 images (in percent of misclassified pixels).

test sets were chosen in order to meet our memory requirements and to be able to compare with the classical Adaboost algorithm [9] and the random forest algorithm [11] (here, random forests of depth 2 are used, since other depths lead to similar generalization performance for the data set and features considered here). Note, however, that 100 images already correspond to a set of $N = 5 \cdot 10^6$ pixels. The classification performance has been evaluated in terms of average pixel misclassification rate. All algorithms have used the $d = 18$ features described in Section 2.

Results. First, we study the performance of the proposed distributed boosting algorithm on the training set as a function of the number of weak learners and compare it to Adaboost [9]. The results are plotted in Fig. 3 and indicate that our algorithm closely reproduces the performance of standard boosting. The slight difference in performance is due to the fact that, in contrast to Adaboost, a relatively small number of fixed thresholds (100 thresholds per feature) has been used in the distributed boosting algorithm.

In a second step, we evaluated the different classification algorithms on the same sets of test images to assess their generalization performance. The results are shown in Tab. 1. We observe that all 3 algorithms yield nearly equivalent generalization performance, with on average $\approx 28\%$ error on the test data. Note that this is below the state-of-the-art performance of $15 - 20\%$ error for SPOT6 images, which is a direct consequence of the fact that only few and very simple features have been used. To conclude, the proposed algorithm enables the detection of clouds in SPOT 6 images with the same performance as Adaboost, yet can be executed in a distributed computing environment and could hence be applied to larger image data bases.

5. CONCLUSION

This paper studied a new architecture for distributed boosting. The resulting algorithm was applied to cloud detection on a large database of SPOT 6 images with ground truth provided by human experts. The processing of large sets of images is a critical factor for obtaining a robust automatic cloud detection and naturally needs distributed processing. Our results indicate that the proposed algorithm performs as well as other classical machine learning algorithms. However, it has the advantage of being scalable and being easily integrated in distributed (cloud) computing environments such as Apache Spark. First tests on the full SPOT 6 album image database using a large distributed computing environment are currently being conducted. Future work will include the study of an optimal discretization scheme for the features and an automatic procedure for handling outliers by defining appropriate weights in the boosting algorithm.

6. REFERENCES

[1] A. Fisher, "Cloud and cloud-shadow detection in SPOT5 HRG imagery with automated morphological feature extraction," *Remote Sens.*, vol. 6, no. 1, pp. 776–800, 2014.

[2] C. Panem, S. Baillarin, C. Latry, H. Vadon, and P. Dejean, "Automatic cloud detection on high resolution images," in *Proc. IEEE IGARSS*, Seoul, South Korea, July 2005, pp. 506–509.

[3] J. Cooper and L. Reyzin, "Improved algorithms for distributed boosting," in *NIPS Workshop on Distributed Machine Learning and Matrix Computations*, Montreal, Canada, Dec. 2014.

[4] B. Panda, J. Herbach, S. Basu, and R. Bayardo, "Planet: massively parallel learning of tree ensembles with MapReduce," *Proc. of the Vldb Endowment*, vol. 2, no. 2, pp. 1426–1437, 2009.

[5] A. Lazarevic and Z. Obradovic, "The distributed boosting algorithm," in *Proc. ACM SIGKDD Int. Conf. on Knowledge discovery and data mining*, San Francisco, USA, August 2001, pp. 311–316.

[6] K. Zhu, H. Wang, H. Bai, J. Li, Z. Qiu, H. Cui, and E. Y. Chang, "Parallelizing support vector machines on distributed computers," in *Adv. in Neural Inf. Process. Syst. 2007*, J.C. Platt, D. Koller, Y. Singer, and S.T. Roweis, Eds., pp. 257–264. Curran Associates, Inc., 2008.

[7] M. Collins, R. Schapire, and Y. Singer, "Logistic regression, AdaBoost and Bregman distances," *Machine Learning*, vol. 48, no. 1-3, pp. 253–285, 2002.

[8] R. R. Irish, "Landsat 7 automatic cloud cover assessment," in *Proc. SPIE 4049*, 2000, pp. 348–355.

[9] J. Friedman, T. Hastie, R. Tibshirani, et al., "Additive logistic regression: a statistical view of boosting," *Annals of Statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[10] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods," *Annals of statistics*, vol. 26, no. 5, pp. 1651–1686, 1998.

[11] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.