Project Matsu: Machine Learning of Spectral Features for Multispectral and Hyperspectral Images Version 1.3

Open Cloud Consortium
Open Data Group

November 28, 2012

Materials for Matsu

Matsu is a public project. Code and documentation are maintained on Github. The Open Cloud Consortium has an account on Github where projects may be released from time-to-time. The official Open Cloud Consortium account is `http://github.com/opencloudconsortium`. The Matsu project page is `http://github.com/opencloudconsortium/matsu-project`.

Information about the computing resources available from the OCC can be found at `http://opencloudconsortium.org`.

Summary

Matsu[1] is a project of the Open Cloud Consortium. A goal of the Matsu project is to use an open source cloud-based infrastructure to make high quality satellite image data accessible through an Open Geospatial Consortium (OGC)-compliant Web Map Service (WMS). Additionally, a workflow has been created using MapReduce which generalizes image processing and can be used for running distributed analytics over satellite images.

Matsu incorporates the following open source technologies and applications:

- GlusterFS;
- Hadoop and the Hadoop Streaming Interface;
- Avro;
- Accumulo;
- R;
- Python;
- Augustus; and
- PMML.

Contributors to the project have included the National Aeronautics and Space Administration (NASA), TexelTek, the University of Chicago, the Laboratory for Advanced Computing (LAC), and Open Data.

[1]Matsu, or Mazu, is a goddess of the sea said to protect fishermen and sailors. Project Matsu was initially a response to the 2010 Haiti earthquake and the name was chosen in the spirit of aiding those in need.

Contents

1 Introduction

One of the goals of the Matsu Project is to use an open source cloud-based infrastructure to make high quality satellite image data accessible through an Open Geospatial Consortium (OGC)-compliant Web Map Service (WMS). Another goal is to develop an open source cloud-based analytic framework for analyzing individual images, as well as collections of images. A third goal is to generalize this framework to manage and analyze other types of spatial-temporal data.

This report presents a comparison of three supervised machine learning techniques implemented in R and Python. The techniques are each applied to the task of learning a 4-category data set from known, labeled data. The data sets are derived from satellite image data, and the categories are various land cover types. Each procedure was employed in its default form, in order to make a fair comparison of the speed and accuracy of each method.

The water classifier runs as an optional module to the Matsu Web Tiling MapReduce code. All of the MapReduce modules piggyback on the web tiling procedure and create a web map tiles with the module's analytic instead of a RGB visible image. Currently, there are R and Python implementations of the module available on the GitHub site.

This report discusses possible algorithms which can be used for classifying pixels and not the MapReduce job, the spectral analysis, or feature extraction from binary data. It is intended to show how various models performed both in time and accuracy, to aid in the development of additional analytics to run in the Matsu framework.

2 Technique

For this analytic, we compared Support Vector Machines (SVM), Naïve Bayes, and Decision Trees. The implementations are from public R and Python packages.

The data is derived from individual pixels of satellite images. The images represent four distinct land cover types. For each pixel, we used nine values, corresponding to the reflectivity across nine spectral bands in the 433 nm to 2350 nm wavelength range.

The reflectivity data is projected into 9-dimensional feature space for classification by the machine learning algorithm, where similar land cover types are expected to have correspondingly similar locations. Training and verification data sets were constructed from this data.

2.1 Training

Hyperion and Ali images were hand classified. This process required that each pixel be labeled as one of (liquid) *water*, *cloud*, (dry) *land*, or (wet land) *forest*. This is time-intensive and subject to human error.

It should be noted that the classifications results we saw with the labeled data are very good. A reason for the high proportion of successfully identified pixels may be that even though distinct data sets were used for training and verification, the number of actual images hand scored was low. Also to make the hand scoring more accurate, images taken during the day where used. This removes challenges to accurately classify a pixel due to day / night, for example, discrepancies, but we are limited by how many images can be realistically labeled and how well.

3 Methods

The methods used for classification included Support Vector Machines, Naïve Bayes, and Decision Trees. The R implementations for SVM and Naïve Bayes are from the package **e1071**[2]. Decision Trees uses **rpart**[3]. The Python implements are from the package **Scikit-learn**[4].

The comparison of the three methods was performed using a similar routine (implemented by R or Python) in which a random selection of a random number of training data sets were concatenated and used to train a model, followed by a similar random construction of a test set used for classification using these models. This was done using a test harness and ran outside of MapReduce.

4 Results

The procedure was repeated 100 times for 4-classification using each of the three methods. The average across the 100 trials is given as confusion matrices in tables 3 - 8. The overall accuracy for the implementations of the three algorithms is given in table 4 below:

Decision trees have very similar accuracy as applied to the task of 4-classification of the image data sets. Naïve Bayes performs nearly 2 times

[2]e1071: Misc Functions of the Department of Statistics (e1071), TU Wien, `http://cran.r-project.org/web/packages/e1071/index.html`

[3]rpart: Recursive Partitioning, `http://cran.r-project.org/web/packages/rpart/index.html`

[4]scikit-learn: machine learning in Python, `http://scikit-learn.org`

	Python	R
SVM	75.1	86.7
RPART	79.5	80.2
Naïve Bayes	50.5	30.5

Table 1: Accuracy, Python and R 4-classifier

better implemented in Python than in R. SVM does not perform as well in the Python implementation as in R, though similar performance can be obtained with the Python SVM algorithm by using a nonlinear radial basis function kernel and performing minimal tuning.

4.1 Run Times

The different classification algorithms were evaluated by run time as well as accuracy. One of the goals of Matsu is to run MapReduce tilings and analytics each day as new satellite images are received. While there is not a hard limit on the runtime of an analytic, performance is a concern. The average run times for the three algorithms over the hand classified test pixels are given in table 2:

	Python	R
SVM	46 sec	85 sec
Decision Tree	42 sec	69 sec
Naïve Bayes	45 sec	3445 sec

Table 2: Timing, Python and R 4-classifier

The Python SVM implementation from Scikit-learn is currently used in the Matsu project. The amount of time spent classifying the pixels of a full satellite image, which consists of millions of pixels, is approximately two minutes. The run time of the entire process is (reading the file, creating web tiles, and storing in Accumulo) takes longer.

5 False-Color Images

The output of the water classifier module are false-color web tiles giving the classification of each pixel. An example of a classified Ali image (not

	Predicted			
	Cloud	Land	Water	Forest
Cloud	**500**	0	7	0
Land	0	**500**	259	0
Water	0	0	**234**	0
Forest	0	0	0	**500**

Table 3: Confusion Matrix: 4-classifier Support Vector Machines in R

	Predicted			
	Cloud	Land	Water	Forest
Cloud	**500**	0	11	0
Land	0	**500**	487	0
Water	0	0	**2**	0
Forest	0	0	0	**500**

Table 4: Confusion Matrix: 4-classifier Support Vector Machines in Python

	Predicted			
	Cloud	Land	Water	Forest
Cloud	**85**	0	0	0
Land	0	**500**	486	489
Water	415	0	**14**	0
Forest	0	0	0	**11**

Table 5: Confusion Matrix: 4-classifier Naïve Bayes in R

	Predicted			
	Cloud	Land	Water	Forest
Cloud	**500**	0	0	0
Land	0	**500**	500	490
Water	0	0	**0**	0
Forest	0	0	0	**10**

Table 6: Confusion Matrix: 4-classifier Naïve Bayes in Python

	Predicted			
	Cloud	Land	Water	Forest
Cloud	**500**	0	13	0
Land	0	**491**	253	0
Water	0	0	**133**	0
Forest	0	9	122	**500**

Table 7: Confusion Matrix: 4-classifier Regression Trees in R

	Predicted			
	Cloud	Land	Water	Forest
Cloud	**500**	0	12	0
Land	0	**487**	252	0
Water	0	0	**103**	0
Forest	0	13	132	**500**

Table 8: Confusion Matrix: 4-classifier Regression Trees in Python

web tiled) obtained from running both implementations of the three algorithms is below. The image covers a portion of Namibia which is prone to flooding. Figure 1 shows the classification of an Ali image showing part of Namibia using the Python implementations and figure 2 shows the same image classified using the R implementations.

Qualitatively, all methods show comparable classification performance when using either the Python or R implementation. The performance across methods varies.

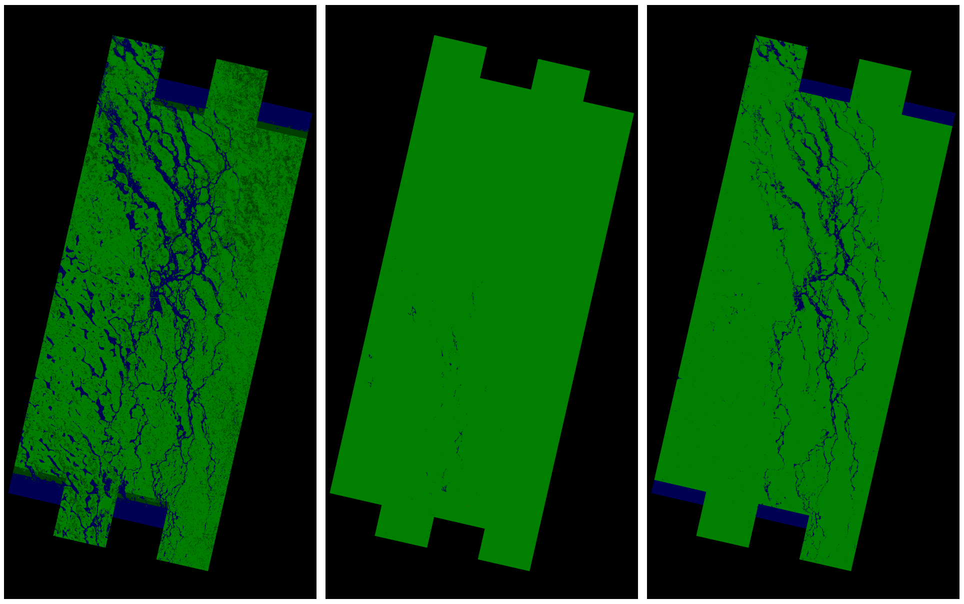

Figure 1: Namibia floods classified by Python. Left: SVM, Center: Naïve Bayes, Right: Decision Trees

6 Conclusion

The code for the water classifier module on Github supports the algorithms described in this report. For different types of images or features, one algorithm may be better suited than another. The R code was embedded into Python using rpy2 and this may account for the slightly longer runtime of the R implementations.

The classification performance with Decision Trees was nearly the same using the R and Python implementations and with Naïve Bayes, Python

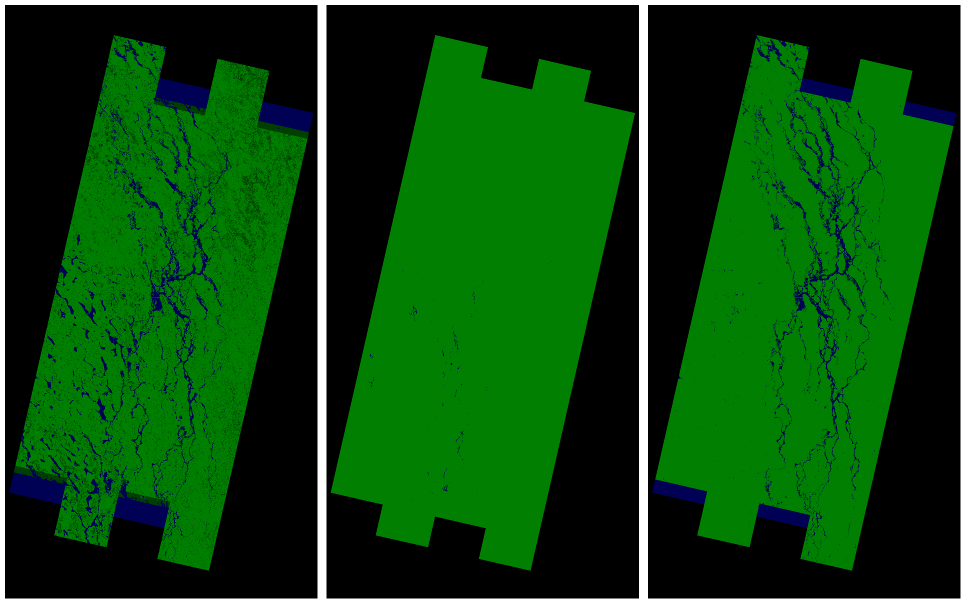

Figure 2: Namibia floods classified by R. Left: SVM, Center: Naïve Bayes, Right: Decision Trees

was clearly better. The algorithms were run "out of the box", but similar performance can be obtained with the Python SVM algorithm as the R version by using a nonlinear radial basis function kernel. This is not the default setting for the algorithm in Scikit-learn.

A test harness, which lets you apply the algorithms to individual images, is being released to GitHub. This allows for testing of various algorithm in a "desktop" manner without the MapReduce framework. It also assembles and saves the resulting classification as a (PNG) image.

7 About Project Matsu

Matsu is an Open Cloud Consortium (OCC)-sponsored project. The source code and documentation will be made available on GitHub under Open Cloud Consortium (`https://github.com/opencloudconsortium`) account.

The OCC is a not for profit that manages and operates cloud computing infrastructure to support scientific, environmental, medical and health care research. The OCC is focused on using this technology to make scientific advances by working with scientists in a variety of disciplines. Visit us at `http://opencloudconsortium.org` or, for more information, email `info {at} opencloudconsortium {dot} org`.

Open Data is a member of the OCC. Open Data began operations in 2001, specializes in building predictive models over big data, and is one of the pioneers using technologies such as Hadoop and NoSQL databases so that companies can build predictive models efficiently over all of their data. More information can be found at `http://opendatagroup.com` or by emailing `info {at} opendatagroup {dot} com`.