UNIVERSITÀ DEGLI STUDI DI SALERNO

IOT DATA ANALYTICS

# MEASURING THE PERFORMANCES OF ONLINE ANALYSIS FOR FOREST COVER TYPE CLASSIFICATION

Professors: Giuseppe Polese, Genoveffa Tortora
Project coordinator: Stefano Cirillo
Student: Michelangelo Esposito – 0522500982

Academic year 2020/2021

## INDEX

## *INTRODUCTION*

Most of the times, machine learning is associated with batch data: a dataset is obtained from a source of some sort, and then it is used to train a classifier. As a consequence, most machine learning and data mining approaches assume that the elements of the dataset are independent, identically distributed and generated from a stationary distribution. This process is generally adaptable to most "static" circumstances and there are a variety of machine learning models and networks that employ it.

Sometimes, however, the data may not be available all at once from the start, or the hardware on which these machine learning processes run may not satisfy the requirements to run traditional batch learning. Nowadays, we are faced with a tremendous amount of distributed data that can be generated from the ever-increasing number of smart devices. In an IoT scenario, for example, dynamic streams of data are being constantly emitted from countless sensors, and resource-limited devices; in most of cases, this data is transient and may be stored for a brief period of time before getting discarded. In this small fraction of time, we are often required to perform analysis operations on the data we receive; therefore, there is the need to adapt the learning process to a workflow that is able to satisfy limited hardware requirements and to dynamically adapt to the continuous flow of data.

Online learning is a method of machine learning in which data becomes available sequentially and is used to update the predictor for future data at each step. It can be data efficient since once an entry has been processed, it is no longer required; therefore, storing previously seen data is not necessary. This approach is also highly adaptable to evolving circumstances since it makes no assumptions about the distribution of the data: as the distribution morphs or drifts, the model can adapt on-the-fly to keep pace with trends in real-time. In order to do something similar with traditional offline learning, one would have to create a sliding window for the data and retrain it every time.

There are various reasons to use online learning over batch learning, among these:

- The whole dataset does not fit in memory.
- The distribution of the data is expected to morph over time.
- The data itself is a function of time (i.e., stock prices).

A very noticeable difference between online and batch learning resides in the training and testing phases of the classifier. Traditionally, there are various ways for evaluating a batch classifier, such as reserving a percentage of the data for testing, or performing k-fold cross validation, a technique by which the data is split into k groups and for each group an evaluation is performed by considering that group as the testing one and the others as the training ones.

In online learning there is no way of reserving a portion of the data for testing purposes, given the volatile nature of the stream, therefore a new approach becomes mandatory. In data stream classification, the most used evaluation technique is the prequential one, where instances are first used to test, and then to train the model.

## *CONTEXT OF THE PROJECT*

The main focus of this project is to employ online machine learning techniques to analyse the performances of various classifiers on a data stream. This stream does not produce real-time data, but it is generated from a csv/arff file in order to use the online approach.

The platform chosen for the initial analysis is MOA, Massive Online Analysis. MOA is an open-source framework designed specifically for data stream mining with concept drift. It is written in Java and developed at the University of Waikato, New Zealand. It allows to build and run experiments of machine learning or data mining on evolving data streams;  it includes a set of learners and stream generators that can be used from the Graphical User Interface (GUI), the command-line, and the Java API.

First of all, both the MOA GUI and the Java API were used to perform three parallel classification tasks on the dataset; these results acted as a baseline for any future run.

At this point, a first comparison among the three classifiers was made, in order to analyse their performances and costs. We then decided to repeat the same process in Python, to further elaborate on MOA’s efficiency. Finally, a traditional batch approach was employed to additionally assess the potential gap with online learning.

## *FOREST COVER TYPE DATASET*

The dataset used for this study is available for free on the UCI Machine Learning Repository website and its employment revolves around predicting forest cover type from cartographic variables only; no images or other kinds of data are present in this dataset. The actual forest cover type for a given observation (30 x 30 meters cell) was determined from US Forest Service (USFS) Region 2 Resource Information System (RIS) data. Independent variables were derived from data originally obtained from US Geological Survey (USGS) and USFS data. The data was not originally scaled and contains binary columns of data for qualitative independent variables, such as wilderness areas and soil types. A total of over 500,000 entries are present in the dataset, each with the following attributes:

- Elevation in meters.
- Aspect in degrees azimuth.
- Slope in degrees.
- Horizontal distance to nearest surface water.
- Vertical distance to nearest surface water.
- Horizontal distance to nearest roadway.
- Hill shade index at 9am, summer solstice.
- Hill shade index at noon, summer solstice.
- Hill shade index at 3am, simmer solstice.
- Horizontal distance to nearest wildfire ignition points.
- Wilderness area designation.
- Soil type designation.
- Forest cover type designation (dependent attribute). The seven cover types, with the respective number of entries, are:
    - Spruce/Fir: 211840.
    - Lodgepole Pine: 283301.
    - Ponderosa Pine: 35754.
    - Cottonwood/Willow: 2747.
    - Aspen: 9493.
    - Douglas-fir: 17367.
    - Krummholz: 20510.

## *ONLINE DATASET ANALYSIS*

In order to use this dataset in online fashion, and simulate real-time learning, its entries are streamed from the arff file to MOA. Three classifiers were used for this analysis: adaptive random forest, naïve bayes and hoeffding tree. Among these three, the random forest produces the best result, however its execution time is significantly slower.
Each of the three runs produced a set of parameters on which to evaluate the model; furthermore, every 100,000 processed instances, intermediary results were logged, in order to measure the growth of the classifier over time.
For each task run on the dataset, the following attributes have been measured:

- Classified instances: the number of entries in the dataset that has been processed so far by the classifier.
- Time: the execution time, in seconds, necessary for the classification.
- Correct predictions: percentage of entries successfully classified by the model so far.
- Kappa score: a statistic that is used to measure inter-rater reliability, a score of how much homogeneity exists in the classification.
- Precision: the ratio of correctly predicted positive observations to all positive observations in the positive class
- Recall: the ratio of correctly predicted positive observations to all observations in the positive class
- F1-score: a weighted average of precision and recall. It takes into account both false positives and false negatives.

The following tables display the progression of the three classifiers during their execution:

**Adaptive Random Forest**

| **Classified instances** | **Time (s)** | **Correct pred(%)** | **Kappa** | **Kappa temp.** | **Kappa M** | **Precision** | **Recall** | **F1** |
|---|---|---|---|---|---|---|---|---|
| 100,000 | 347 | 93.80 | 93.8 | -58 | 93.8 | | | |
| 200,000 | 732 | 98.20 | 98.2 | 43 | 98.2 | | | |
| 300,000 | 1135 | 93.80 | 93.8 | -40 | 93.8 | | | |
| 400,000 | 1578 | 91.20 | 91.2 | -95 | 91.2 | | | |
| 500,000 | 1966 | 94.39 | 94.39 | -19 | 94.39 | | | |
| 581,012 | 2232 | 98.4 | 98.4 | 20 | 98.4 | | | |

The Adaptive Random Forest classifier is an adaptation of the original Random Forest algorithm which has been successfully applied to various machine learning tasks. The "Adaptive" part comes from its mechanisms to adapt to different kinds of concept drifts, given the same hyper-parameters. This classifier is the one that produced the highest accuracy among the three, however its execution time was significantly slower compared to both Naïve Bayes and Hoeffding Tree. Its accuracy reaches the lowest value at around 400,000 classified instances but manages to regrow by the end of the process.

**Naïve Bayes**

| **Classified instances** | **Time (s)** | **Correct pred(%)** | **Kappa** | **Kappa temp.** | **Kappa M** | **Precision** | **Recall** | **F1** |
|---|---|---|---|---|---|---|---|---|
| 100,000 | 1.75 | 78.90 | 78.90 | -441 | 78.90 | | | |
| 200,000 | 3.46 | 52.60 | 52.60 | -1381 | 52.60 | | | |
| 300,000 | 5.12 | 64.30 | 64.30 | -711 | 64.30 | | | |
| 400,000 | 6.79 | 31.00 | 31.00 | -1433 | 31.00 | | | |
| 500,000 | 8.46 | 41.00 | 41.00 | -1155 | 41.00 | | | |
| 581,012 | 9.85 | 63.80 | 63.80 | -1709 | 63.80 | | | |

This classifier performs a classic Bayesian prediction while making naïve assumption that all inputs are independent. It is a very simple algorithm with low computational costs, and this is reflected in its execution time, but also in its underwhelming accuracy when it comes to the considered problem.

**Hoeffding Tree**

| **Classified instances** | **Time (s)** | **Correct pred(%)** | **Kappa** | **Kappa temp.** | **Kappa M** | **Precision** | **Recall** | **F1** |
|---|---|---|---|---|---|---|---|---|
| 100,000 | 1.92 | 90.90 | 90.90 | -133 | 90.90 | | | |
| 200,000 | 3.45 | 83.20 | 83.20 | -424 | 83.20 | | | |
| 300,000 | 5.09 | 75.20 | 75.20 | -463 | 75.20 | | | |
| 400,000 | 6.82 | 68.50 | 68.50 | -599 | 68.50 | | | |
| 500,000 | 8.67 | 75.60 | 75.60 | -419 | 75.60 | | | |
| 581,012 | 10.18 | 67.30 | 67.30 | -1534 | 67.30 | | | |

A Hoeffding tree is an incremental decision tree induction algorithm that is capable of learning from massive data streams, assuming that the distribution of the data does not change significantly over time. After the first 300,000 instances, we can see the accuracy oscillating. This suggests us that the classifier might have needed more samples before stabilizing its accuracy value.

## *COMPARISON WITH OTHER TECHNIQUES*

To further expand on the analysis of the goodness of the results obtained in MOA, we decided to compare these findings with other analysis techniques, using both online and batch learning. In particular, the focus shifted towards one of the most widely used libraries for machine learning, python's scikit learn, and its online counterpart, scikit multiflow. First of all, a direct online comparison is due: the scikit multiflow library allows us to design and run experiments by making use of powerful stream learning methods and evaluators. To better compare the results, the same three classifiers have been used.

The next table summarizes the final measurements obtained with all of the employed techniques:

| | Accuracy | Precision | Recall | F1-score | Kappa | Time(s) |
|---|---|---|---|---|---|---|
| **MOA** | | | | | | |
| Adaptive Random Forest | 0.98 | | | | 0.98 | 2400 |
| Naïve Bayes | 0.63 | | | | 0.63 | 9.39 |
| Hoeffding Tree | 0.78 | | | | 0.78 | 9.25 |
| | | | | | | |
| **Scikit multiflow** | | | | | | |
| Adaptive Random Forest | 0.94 | 0.79 | 0.76 | 0.77 | 0.90 | 7386 |
| Naïve Bayes | 0.57 | 0.43 | 0.36 | 0.36 | 0.27 | 1408 |
| Hoeffding Tree | 0.82 | 0.64 | 0.62 | 0.63 | 0.71 | 1122 |
| | | | | | | |
| **Scikit learn** | | | | | | |
| Random Forest | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 163 |
| Naïve Bayes | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 2.77 |
| Decision Tree | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 3.06 |

## ***CONCLUSION***