Automatization of quality assurance process for ALMA data using supervised learning algorithms

Alexandros Chalevin (etamax space GmbH, European Southern Observatory, Germany)

Introduction

In the ALMA data processing workflow, the stage of quality analysis of the pipeline/manually processed data before its delivery to PI is called QA2 stage (Chavan et al. 2016).

For the pipeline processed data (Humphreys, E., et al. 2016) the QA decisions are based on the number of parameters like scores of the execution of pipeline tasks, angular resolution and the sensitivity achieved for a representative frequency and some other criteria. In total, we have about two dozen incoming criteria which result in the delivery data to PI, sending data to reprocess and marking data as needed more observations.

Approximately 75 % of ALMA data is being processed by pipeline software, but the final QA2 decisions are always human made.

1. Data structure

As input parameters for our algorithm we were using the scores of CASA/ALMA Pipeline tasks

hifa_importdata	hifa_flagdata	hifa_fluxcalflag
hif_rawflagchans	hif_refant	h_tsyscal
hifa_tsysflag	hifa_antpos	hifa_wvrgcalflag
hif_lowgainflag	hif_setmodels	hifa_bandpassflag
hifa_spwphaseup	hifa_gfluxscaleflag	hifa_gfluxscale
hifa_timegaincal	hif_applycal	hif_findcont
hif_makeimages	hif_makeimlist	hif_uvcontfit
hifa_imageprecheck	hif_checkproductsize	hifa_exportdata
hif_mstransform	hifa_flagtargets	hif_uvcontsub

Each score is ranged between 0 (the worst) and 1 (the best). The sensitivity parameter was calculated as $RMS_{expexted}/RMS_{observed}$. The output decisions array includes “Pass”, ”Semi-Pass”, ”Fail (re-observe)”, ”PL Calibrate”, “PL Image”, “PL Cal&Img”, “Manually calibrate”, “Manually Image” and “Manual Cal&Img”. On this stage of development these cases can be reduced to two possible outcome classes: the “Happy path” and the opposite.

So, our data set was labelled as having only two outcomes: 0 and 1.

2. Data processing

In the algorithm we were using an open source library Eclipse Deeplearning4j, having API for Java, Python and C++. This library has integration with Hadoop and Apache Spark and supports distributed CPU and GPU computations.

The example of the code used for computations is presented below.

```
MultiLayerConfiguration conf = new NeuralNetConfiguration.Builder()
        .seed(seed) Builder
        .updater(new Nesterovs(learningRate, momentum: 0.9)) Builder
        .list() ListBuilder
        .layer( ind: 0, new DenseLayer.Builder().nIn(numInputs).nOut(numHiddenNodes)
                .weightInit(WeightInit.XAVIER)
                .activation(Activation.RELU)
                .build()) ListBuilder
        .layer( ind: 1, new DenseLayer.Builder().activation(Activation.RELU)
                .nIn(numHiddenNodes)
                .nOut(numHiddenNodes)
                .build()) ListBuilder
.layer( ind: 2, new OutputLayer.Builder(LossFunctions.LossFunction.XENT)
        .weightInit(WeightInit.XAVIER)
        .activation(Activation.SIGMOID)
        .nIn(numHiddenNodes).nOut(numOutputs).build()) ListBuilder
.backpropType(BackpropType.Standard) Builder
.build();
```

From the code one can see that for the current computations we were using the neural network with one hidden dense layer, Xavier Initialization of weights and linear unit rectifier (ReLU). For the output layer activation, the Sigmoid function was used. We are still in the process of adjusting different neural network parameters and improvement of the input data as well. For the current computations we used 6669 input sets. 75% of this data was used for training and 25% was used for testing. Evaluation results are presented below:

```
========================Evaluation Metrics========================
 # of classes:    2
 Accuracy:        0.7570
 Precision:       0.6473
 Recall:          0.6234
 F1 Score:        0.8465
Precision, recall & F1: reported for positive class (class 1 - "1") only


=========================Confusion Matrix=========================
    0    1
-----------
  145  244 | 0 = 0
  161 1117 | 1 = 1

Confusion matrix format: Actual (rowClass) predicted as (columnClass) N times
==================================================================
```

As one can see, with the current level of accuracy, this algorithm cannot be used for automatization of QA2 process, but we have not included yet all the parameters used by data reducers to make decisions and the parameters of the neural network also will be optimized in the future computations.

References

1. Humphreys, E., et al. The ALMA Science Pipeline: Current Status, Proceedings of the 2016 ALMA Conference, "Half a Decade of ALMA: Cosmic Dawn Transformed", Sept. 20-23, 2016.
2. Chavan, A.M., et al. ALMA quality assurance: concepts, procedures, and tools. Proceedings of the SPIE, Volume 9910, 2016.
3. Deep learning for Java. https://deeplearning4j.org/