openPDC DM-Tools Usage Examples

Authors: Josh Patterson(jpatterson@floe.tv), Galen Riley(galenriley@gmail.com)

Summary

- Example guide on how to use the openPDC datamining tools with Hadoop
- Covers 3 sample cases scanning both openPDC archive data and CSV data

Introduction

This is a quick guide on how to use the datamining tools included with the openPDC Hadoop utilities. Most of the tools included in this package are in a **beta** stage at this point and should be treated accordingly. In this guide we'll take a look at how to classify time series data from both demo CSV data and the openPDC's time series archive format.

These tools are meant to serve as a basic framework on how to scan specific classes of time series patterns and classify them accordingly. In some cases they are useful for specific purposes such as finding "unbounded oscillations" in phasor measurement unit (PMU) data. They are still in a developmental stage and have not yet been widely tested. However they are quite suitable to be used as a basis for development and refinement in specific time series domains.

These tools will be incredibly useful for any engineer or computer scientist who wishes to work in the domains of:

- Phasor Measurement Unit (PMU) data
- Arbitrary sensor data collected with the openPDC
- Time series applications in other domains as illustrated by Dr Keogh
  - Image recognition
  - Shape Matching
  - Medical Data
  - Log Data

We hope to continue this work further and as sensor data collection accelerates we believe this work to be a good building block to work from. Please feel free to email the authors with questions or comments.

Example Case 1

- Input Archive Type: openPDC historian format
- Window Size: 10 seconds
- Window Step Size: 5 seconds

Extract Training Data From Archives

The openPDC stores time series data from multiple sensors in a single archive. Each “point block” in the archive is identified by a “point ID”. In order to create training data for a particular pattern, we extract a sensor (or sensors) data and create training samples for our classifier from those data points. Our training instances can be setup multiple ways but the primary variable is “window size”. The window size of the training samples should match the window size used in the Map Reduce classification job.

We want to start out by finding a range of data that contains enough instances of each class of pattern we would like to classify. This is a particularly time consuming process but making sure that you have quality training samples is a critical aspect to creating a good classifier.

The training instance creation command is of the following structure:

```
$ bin/dm-tools Instances <input path> <output path> <point ID to extract> <points per window> <window delta>
```

Input and output paths can be either directories or single files. For this example case our parameters would be:

```
$ bin/dm-tools Instances data/openpdc_archive.d training_data/train.csv 1603 300 150
```

Our openPDC archive input data is specified by “data/archive.d” and our output training instances are specified by “training_data/train.csv”. We also specify that we only want data from point ID 1603. We want instances that are 10 seconds of data in length. The instance generator only supports a number of points in a window, but fortunately we know that points in an archive file are recorded 30 times per second. This value would be 300.

Window delta is calculated based on the start of the previous window. For our example, this value is 150. With these values, each window begins with points from second half of the previous window.

The output of this command is a CSV file containing 300 points from the openPDC archive. An additional column is added to the end with the value of 0, representing an as-yet-unclassified sample.

Examine Training Data / Prepare For Classifier

Its always a good idea to examine your data to get a visual feel for what you are dealing with. With our sample PMU data, we have an “unbounded oscillation” that exists in the included sample openPDC archive (this type of power grid event is particularly rare so in this toy example we only have one to work with). Below in Figure 1 we show 10 (out of 967) instances graphed in Microsoft Excel.

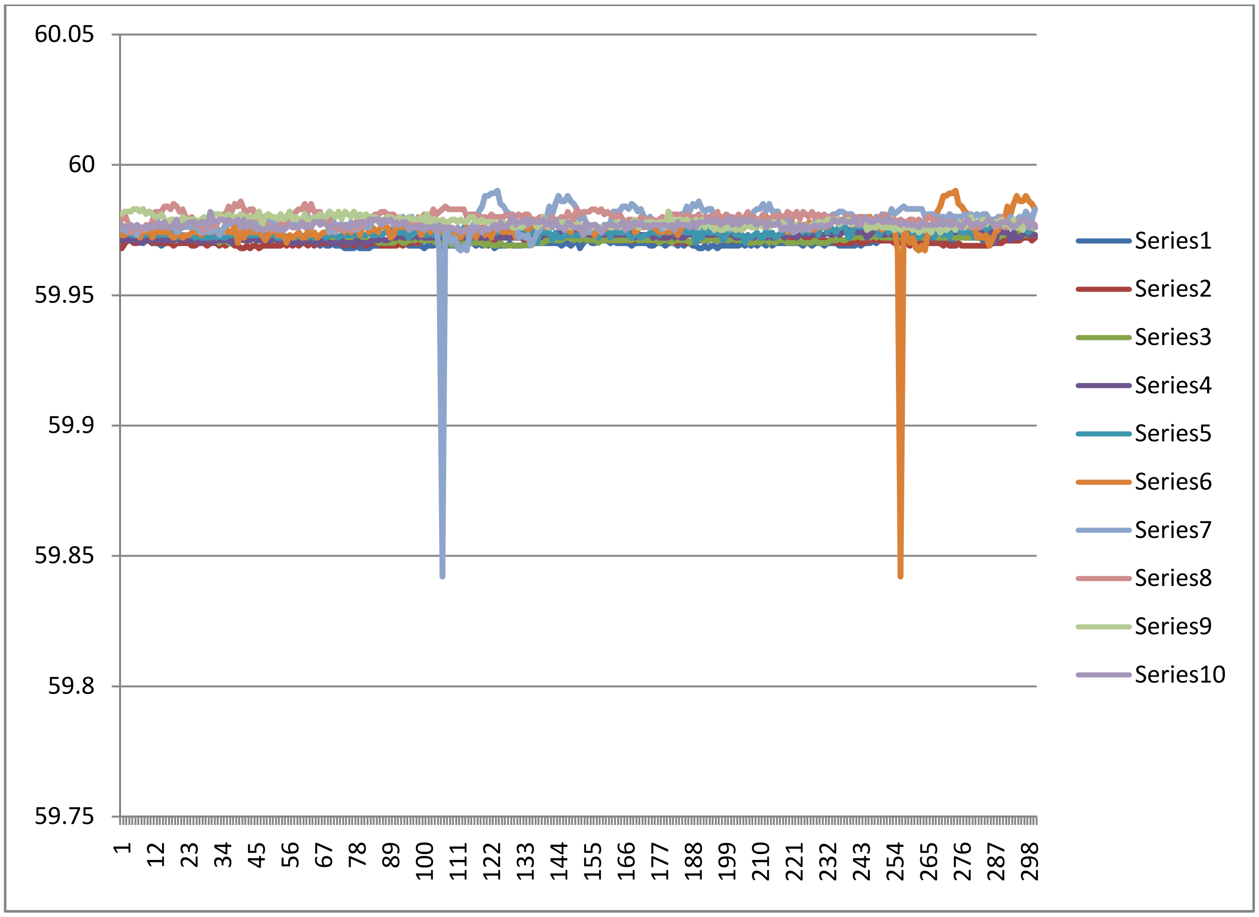

Figure 1. *Graphed data from openPDC archive*

Excel is a good tool to use for this since we can easily graph selected ranges of instances and we can also quickly tag instance rows with the correct classification. Unfortunately at this point there is no easy way to classify your training instances other than by hand. In the future we’d like to see tools that make this process much easier and efficient.

Excel should open the CSV directly as a worksheet and allow you to work with the instances. The last column in each row is marked as “UNCLASSIFED”. This is the column you should change to be a numeric class ID. Typically we graph blocks of instances and then hand classify each instance visually. We repeat this process until we’ve classified all of the instances. Once all instances have been classified we then export this newly classified training instances CSV file. We now want to move this set of classified training instance to HDFS so the Map Reduce classifier can find it during the classification pass. This CSV file will be used by the 1NN classifier as the training set.

In this particular example we want to mark the two instances of the (same) unbounded oscillation with the class of 1 and all other instances with the class of 0. In a real scenario we ideally would like to have a proportionate number of instances from each class. This example should be considered a toy example due to the fact that there are only two instances of our anomaly.

Run Map Reduce Job

Once we have the training instances on HDFS, we can run our Map Reduce job. In the companion document "openPDC Datamining Tools Guide" we cover the various aspects involved in basic 1NN time series classification. Below is an example of how to run a Map Reduce classification job with the code included in the openPDC project.

```
bin/hadoop
jar {jar_name}.jar
TVA.Hadoop.MapReduce.Datamining.SAX.SlidingTSClassifier_kNN
-pointTypeID 1603
-windowSize 10000
-windowStepSize 5000
-trainingSetCSV {hdfs_path_to_your_training_instances}
-samplesPerRawInstance 300
-saxDim 20
-saxCard 10
{hdfs_path_to_openpdc_archive_to_scan}
{hdfs_path_to_results_directory}
```

Results

If you check the output file in HDFS you should see a file with two lines in it, each being a long number representing the same time offset. These time offsets should be the front end of where the pattern occurs. Depending on which file you were scanning, the file should look something like:

```
1       1212440905001
1       1212440910001
```