s c c h
software competence center
hagenberg

FUZZY.LOGIC.LAB.LINZ

Advances in Knowledge-Based Technologies

Proceedings of the Master and PhD Seminar Summer term 2015, part 2

Softwarepark Hagenberg
SCCH, Room 0/2
7 July 2015

Software Competence Center Hagenberg
Softwarepark 21
A-4232 Hagenberg
Tel. +43 7236 3343 800
Fax +43 7236 3343 888
www.scch.at

Fuzzy Logic Laboratorium Linz
Softwarepark 21
A-4232 Hagenberg
Tel. +43 7236 3343 431
www.flll.jku.at

Program

Chair: Susanne Saminger-Platz

9:00 Werner Reisner:
An Evolutionary Framework for Coil Slitting Optimization
9:30 Roland Richter:
Classifiers at the Lab

An Evolutionary Framework for Coil Slitting Optimization

Werner Reisner

Software Competence Center Hagenberg GmbH

Abstract - A successful realization of an evolutionary based framework for slitting core laminations for power transformers is proposed. The slitting of metal coils for the power transformer requires great amount of steel (up to 200 tons for a single transformer). Depending on the properties, e.g. width, length or electrical conductivity, of the used metal coils, the produced transformer shows different characteristics, e.g. power loss. Based on simple mutations of slitting plans for coils, our evolutionary framework computes an optimized slitting plan w.r.t. garbage reduction, production times and some special electrical properties. Although the multi-objective optimization problem is a non-linear integer problem with many constraints, the realization of the framework, developed by the Software Competence Center Hagenberg GmbH (SCCH), shows comparably low runtime. Experimental results show the advantages of the framework compared to an older deterministic optimization algorithm of a SCCH partner from industry, based on different ground truth data of successfully produced power transformers.

Classifiers at the Lab

Roland Richter, Fuzzy Logic Laboratorium Linz

AdvKBT SS15-2, July 2015

A typical task at our lab is to train several different types of classifiers on large data sets provided by our industrial partners, with the aim to use these classifiers within several customer-specific software packages. Hence, there is the need for tools to read and write data files, create, parametrize and train classifiers, save them, and re-use them other environments.

The development of such a software package was started at the Fuzzy Logic Laboratorium Linz back in 2006. In this talk, I will report on the current state of the project.

From 2006 to 2009, a software package called *FLLLame* (FLLL Advanced Machine Learning Environment) was created. Back then, several important design decisions were made which are still in effect:

- all classifiers are part of an object-oriented hierarchy in C++
- a C API allows to use classifiers as DLLs
- data formats CSV and Orange tab[1] are supported

The *FLLLame* framework was reported by Ulrich Brandstätter and me at the AdvKBT-W0809-2 workshop in February 2009.

Since 2013, the framework is developed further within project *UseML*. Whereas some of the old features were dropped (e.g., the graphical user interface and the support of the Weka ARFF file format), many new features were added:

- the Classifier C API was reworked (#481[2])
- unit tests were added (#494[3], #482[4])
- several command line tools were added (#536[5], #539[6], . . .)
- support for interpretable classifiers was introduced (#627[7])

As of July 2015, the complete package consist of about 17000 lines of code (C, C++, CMake, DOS batch), and can be compiled to Windows 32bit and 64bit binaries.

Derived from base class `Classifier`, there are five sub-classes, each of which provides a specific machine learning algorithm:

[1] http://docs.orange.biolab.si/reference/rst/Orange.data.formats.html#tab-delimited
[2] https://pm.flll.jku.at/issues/481
[3] https://pm.flll.jku.at/issues/494
[4] https://pm.flll.jku.at/issues/482
[5] https://pm.flll.jku.at/issues/536
[6] https://pm.flll.jku.at/issues/539
[7] https://pm.flll.jku.at/issues/627

- `CARTClassifier`: Classification and Regression Tree, own port
- `EFSClassifier`: Evolving Fuzzy Systems, by E. Lughofer
- `FANNClassifier`: Fast Artificial Neural Networks, wraps FANN[8]; *currently broken*
- `RFClassifier`: Random Forrests, wraps OpenCV[9]
- `SVMClassifier`: Support Vector Machines, wraps libsvm[10]

Note that `EFSClassifier` is the only classifier which allows incremental training, and provides interpretable explanations.

Usage of C API functions

To create a new classifier, one might either train a classifier from scratch, or load a classifier from file, by calling one of these functions which are part of the C API:

```
API_DECL ClfHandle clfTrain(const char* name,
                            /* ... */ );

API_DECL ClfHandle clfLoad(const char* pathname);
```

After that, classifiers are immutable, with one exception: if incremental training is available for the specific classifier, one can update the classifier. Note that this only works for `EFSClassifier` right now:

```
API_DECL BOOL clfUpdate(ClfHandle h,
                        /* ... */ );
```

The next step would be to classify new samples. It is worth mentioning here that, in our framework, to classify new samples actually means to obtain a confidence level for each possible label. Confidence levels are in [0, 1], and sum up to 1. Classification is done by calling this API function:

```
API_DECL BOOL clfClassify(ClfHandle h,
                        /* ... */ );
```

After classifying one sample, a call to `clfGetLastExplanation()` returns an explation, if available. We decided to provide explanations consisting of up to four lines, namely,

- Line 1: the best classification result, together with its confidence level
- Line 2 (optional): the second-best classification result, together with its confidence level, provided that its confidence level is greater than zero
- Line 3: the coverage of the sample
- Line 4: an explanation consisting of up to four terms in the form `<feature> is <term>`, where these terms are sorted w.r.t. the importance of the feature

As an example in German, consider this explanation for a well-known classification task:

[8] https://github.com/libfann/fann
[9] http://opencv.org/
[10] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

```
Iris-versicolor
# Das Beispiel ist mit sehr hoher Sicherheit (89 %) Iris-versicolor.
# Das Beispiel kann mit sehr niedriger Sicherheit (11 %) auch Iris-virginica sein.
# Der Bereich ist durch das Training gut (54 %) abgedeckt.
# Grund: sepal length ist mittel (5.5 +/- 0.4) und sepal width ist klein (2.6 +/- 0.3).
```

Finally, after all work is done, one has to clean up:

```
    API_DECL void clfDestroy(ClfHandle h);
```

There are several further functions to obtain information about trained classifiers, to do error handling, and so on.

Command line tools

Together with the aforementioned classifier DLLs, we provide a set of command line tools which can create or use these DLLs, namely:

- `clftrain`: train classifier with given training data
- `clfupdate`: update trained classifier with given training data
- `clfinfo`: get info about a trained classifier
- `clfclassify`: use trained classifier to classify given test data
- `clfresult`: show confusion matrix of classifier results

In a batch file, a command sequence to train, classify, get information, and show results might look like that:

```
clftrain    -p vigilance:0.2*sqrt(nFeatures)/sqrt(2) -r42
               %DLL_FILE%  %DATA_FILE%  %CLF_FILE%
clfclassify -e %CLF_FILE%  %DATA_FILE%  %PRED_FILE%
clfinfo     %CLF_FILE%
clfresult   %PRED_FILE%  %DATA_FILE%
```

Note that when reading CSV data files, all tools stick to the convention to ignore lines marked with #, to use the last line of the header comment as column names, to ignore columns in brackets, and to identify the class column as the one which is surrounded by exclamation marks. That is, a valid CSV file might look like that:

```
# This table has one ignored column, eight data columns, and the class column.
# The ignored column is indicated by (), the class column by !! markers (#643)
#
# (Sequence Name), mcg, gvh, alm, mit, erl, pox, vac, nuc, !Localization site!
ADT1_YEAST,0.58,0.61,0.47,0.13,0.50,0.00,0.48,0.22,MIT
ADT2_YEAST,0.43,0.67,0.48,0.27,0.50,0.00,0.53,0.22,MIT
ADT3_YEAST,0.64,0.62,0.49,0.15,0.50,0.00,0.53,0.22,MIT
AAR2_YEAST,0.58,0.44,0.57,0.13,0.50,0.00,0.54,0.22,NUC
AATM_YEAST,0.42,0.44,0.48,0.54,0.50,0.00,0.48,0.22,MIT
...
```