# TRABAJO FIN DE GRADO

## GRADO EN INGENIERÍA INFORMÁTICA

## Implementación de una interfaz para el algoritmo NSLVOrd en la biblioteca ORCA

## Manual de Código

**Autor**

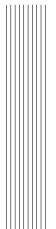Federico García-Arévalo Calles

**Directores**

Pedro Antonio Gutiérrez Peña

Juan Carlos Gámez Granados
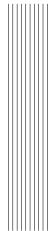
ESCUELA POLITÉCNICA SUPERIOR

—

Córdoba, Junio de 2020

# Índice General

# Índice de Código

# 1 Introducción

Este documento corresponde al manual de código del Trabajo Fin de Grado *"Implementación de una interfaz para el algoritmo NSLVOrd en la biblioteca ORCA"*, en el que se muestran los códigos que se han creado y modificado para este trabajo. Los ficheros se agruparán según al módulo que pertenecen:

- **ORCA:** este es el módulo principal del sistema que se encarga configurar el experimento, calcular las métricas de los resultados y exportar toda la información. Está codificado en lenguaje Matlab.

- **Read File:** este módulo se encarga de leer los ficheros de entrenamiento y test; devolviendo los datos separando las entradas y salidas. Está codificado en lenguaje Matlab.

- **Algorithms:** este módulo se encarga de contener diferentes algoritmos y de la ejecución de estos. Está codificado en lenguaje Matlab.

- **NSLVOrd:** este módulo es el que se encarga de ejecutar el algoritmo NSLVOrd. Está codificado en Java.

- **Rule View:** este módulo es el que se encarga de mostrar las reglas generadas por el algoritmo en una ventana de tal forma que cualquier tipo de usuario pueda entenderlas. Está codificado en Java.

- **JFML:** este módulo es el encargado de exportar las reglas generadas por el algoritmo a ficheros XML en formato JFML y PMML en un directorio especificado por el usuario. Está codificado en Java.

# 2 Módulo ORCA

## 2.1. Clase Utilities

La clase *Utilities* es la principal del sistema y se encarga de la configuración del experimento como de guardar los resultados de este. En el Código 2.1 se muestra la implementación de esta clase.

```matlab
1 classdef Utilities < handle
2     %UTILITIES Static class that contains several methods for
            configurating
3     %   and running the experiments. It allows experiments CPU
            parallelization.
4     %   Examples of integration with HTCondor are provided src/condor
            folder.
5     %
6     %   UTILITIES methods:
7     %      runExperiments            - setting and running experiments
8     %      runExperimentFold         - Launchs a single experiment fold
9     %      configureExperiment       - sets configuration of the several
            experiments
10    %      results                   - creates experiments reports
11    %
12    %   This file is part of ORCA: https://github.com/ayrna/orca
13    %   Original authors: Pedro Antonio Gutiérrez, María Pérez Ortiz,
            Javier Sánchez Monedero
14    %   Citation: If you use this code, please cite the associated paper
            http://www.uco.es/grupos/ayrna/orreview
15    %   Copyright:
16    %       This software is released under the The GNU General Public
            License v3.0 licence
```

```matlab
17      %         available at http://www.gnu.org/licenses/gpl-3.0.html
18
19      properties
20
21      end
22
23
24      methods (Static = true)
25          function [logsDir] = runExperiments(expFile, varargin)
26              %RUNEXPERIMENTS Function for setting and running the
                    experiments
27              %    [LOGSDIR] = RUNEXPERIMENTS(EXPFILE) runs
28              %    experiments described in EXPFILE and returns the folder
29              %    name LOGSDIR that stores all the results. LOGSDIR is
30              %    generated based on the date and time of the system.
31              %
32              %    [LOGSDIR] = RUNEXPERIMENTS(EXPFILE, options) runs
33              %    experiments described in EXPFILE and returns the folder
34              %    name LOGSDIR that stores all the results. Options are:
35              %        - 'parallel': 'false' or 'true' to activate CPU
                    parallel
36              %          processing of databases's folds. Default is 'false'
37              %        - 'numcores': default maximum number of cores or
                    desired
38              %          number. If parallel = 1 and numcores <2 it sets the
                    number
39              %          to maximum number of cores.
40              %        - 'closepool': whether to close or not the pool after
41              %          experiments. Default 'true'. Disabling it can speed
42              %          up consecutive calls to runExperiments.
43              %
44              %  Examples:
45              %
46              %  Runs parallel folds with 3 workers:
47              %   Utilities.runExperiments('tests/cvtests-30-holdout/kdlor.
                    ini', 'parallel', 1, 'numcores', 3)
48              %  Runs parallel folds with max workers:
49              %   Utilities.runExperiments('tests/cvtests-30-holdout/kdlor.
                    ini', 'parallel', 1)
50              %  Runs parallel folds with max workers and do not close the
51              %  pool:
52              %   Utilities.runExperiments('tests/cvtests-30-holdout/kdlor.
                    ini', 'parallel', 1, 'closepool', false)
53              %   Utilities.runExperiments('tests/cvtests-30-holdout/svorim.
                    ini', 'parallel', 1, 'closepool', false)
54              %
55              addpath(fullfile(fileparts(which('Utilities.m')),'Measures'));
56              addpath(fullfile(fileparts(which('Utilities.m')),'Algorithms'))
                    ;
57
58              disp('Setting up experiments...');
59
60              %TODO: move ID generation to configureExperiment?
```

4

## 2.1. Clase Utilities

```matlab
61                    c = clock;
62                    dirSuffix = [num2str(c(1)) '-' num2str(c(2)) '-'  num2str(c(3))
                          '-' num2str(c(4)) '-' num2str(c(5)) '-' num2str(uint8(c(6)
                          ))];
63                    logsDir = Utilities.configureExperiment(expFile,dirSuffix);
64                    expFiles = dir([logsDir '/' 'exp-*']);
65
66                    % Parse options.
67                    op = Utilities.parseParArgs(varargin);
68                    myExperiment = Experiment;
69
70                    if op.parallel
71                        Utilities.preparePool(op.numcores)
72                        if (exist ('OCTAVE_VERSION', 'builtin') > 0)
73                            logsCell = cell(numel(expFiles),1);
74                            logsCell(:) = logsDir;
75                            parcellfun(op.numcores,@(varargin) Utilities.
                                  octaveParallelAuxFunction(varargin{:}),num2cell(
                                  expFiles),logsCell);
76                        else
77                            parfor i=1:numel(expFiles)
78                                if ~strcmp(expFiles(i).name(end), '~')
79                                    disp(['Running experiment ', expFiles(i).name])
                                          ;
80                                    myExperiment.launch([logsDir '/' expFiles(i).
                                          name]);
81                                end
82                            end
83                        end
84
85                        Utilities.closePool()
86                    else
87                        for i=1:numel(expFiles)
88                            if ~strcmp(expFiles(i).name(end), '~')
89                                disp(['Running experiment ', expFiles(i).name]);
90                                myExperiment.launch([logsDir '/' expFiles(i).name])
                                      ;
91                            end
92                        end
93                    end
94
95                    disp('Calculating results...');
96                    % Train results (note last argument)
97
98                    Utilities.results([logsDir '/' 'Results'],'report_sum',
                          myExperiment.report_sum, 'train', true);
99                    % Test results
100                   Utilities.results([logsDir '/' 'Results'], 'report_sum',
                          myExperiment.report_sum);
101                   %rmpath('Measures');
102                   %rmpath('Algorithms');
103
104         end
```

5

```
105
106      function octaveParallelAuxFunction(experimentToRun,logsDir)
107          %OCTAVEPARALLELAUXFUNCTION Function for running one experiment
                 file
108          %   It is used in Octave because it Octave does not have parfor
109          %   OCTAVEPARALLELAUXFUNCTION(EXPERIMENT,LOGSDIR) run the
                 experiment
110          %   named EXPERIMENT and contained in the folder LOGSDIR
111          if ~strcmp(experimentToRun.name(end), '~')
112              myExperiment = Experiment;
113              disp(['Running experiment ', experimentToRun.name]);
114              myExperiment.launch([logsDir '/' experimentToRun.name]);
115          end
116      end
117
118      function results(experiment_folder,varargin)
119          %RESULTS Function for computing the results
120          %   RESULTS(EXPERIMENT_FOLDER) computes results of predictions
121          %   stored in EXPERIMENT_FOLDER. It generates CSV files with
122          %   several performance metrics of the testing (generalization)
123          %   predictions.
124          %       * |mean-results_test.csv|: CSV file with datasets in
                 files
125          %       and performance metrics in columns. For each metric two
                  columns
126          %       are created (mean and standard deviation considering
                 the _k_ folds
127          %       of the experiment).
128          %       * |mean-results_matrices_sum_test.csv|: CSV file with
129          %       performance metrics calculated using the sum of all the
130          %       confussion matrices of the _k_ experiments (as Weka
                 does). Each column
131          %       presents the performance of this single matrix.
132          %
133          %   RESULTS(EXPERIMENT_FOLDER, 'TRAIN', true) same as
134          %   RESULTS(EXPERIMENT_FOLDER) but calculates performance in
                 train
135          %   data. It can be usefull to evaluate overfitting.
136          %
137          %   See also MEASURES/MZE, MEASURES/MAE, MEASURES/AMAE,
                 MEASURES/CCR,
138          %   MEASURES/MMAE, MEASURES/GM, MEASURES/MS, MEASURES/Spearman,
139          %   MEASURES/Tkendall, MEASURES/Wkappa
140
141          addpath(fullfile(fileparts(which('Utilities.m')),'Measures'));
142          addpath(fullfile(fileparts(which('Utilities.m')),'Algorithms'))
                 ;
143
144          opt.train = false;
145          opt.report_sum = false;
146
147          opt = parsevarargs(opt, varargin);
148
```

```matlab
149                experiments = dir(experiment_folder);
150
151            for i=1:numel(experiments)
152                if ~(any(strcmp(experiments(i).name, {'.', '..'}))) &&
                       experiments(i).isdir
153                    disp([experiment_folder '/' experiments(i).name '/' '
                            dataset'])
154                    fid = fopen([experiment_folder '/' experiments(i).name
                            '/' 'dataset'],'r');
155                    datasetPath = fgetl(fid);
156                    fclose(fid);
157
158                    if opt.train
159                        predicted_files = dir([experiment_folder '/'
                                experiments(i).name '/' 'Predictions' '/' '
                                train_*']);
160                    else
161                        predicted_files = dir([experiment_folder '/'
                                experiments(i).name '/' 'Predictions' '/' '
                                test_*']);
162                    end
163                    % Check if we have a missing fold experiment.
164                    %-2 is to compensate . and ..
165                    predicted_files_train = dir([experiment_folder '/'
                            experiments(i).name '/' 'Predictions' '/' 'train_*'
                            ]);
166                    predicted_files_test = dir([experiment_folder '/'
                            experiments(i).name '/' 'Predictions' '/' 'test_*'
                            ]);
167
168                    if (numel(predicted_files_train)+numel(
                            predicted_files_test)) ~= numel(dir(datasetPath))
                            -2
169                        warning(sprintf('\n *********** \n The execution of
                                 some folds failed. Number of experiments
                                differs from number of train-test files. \n
                                *********** \n'))
170                    end
171                    time_files = dir([experiment_folder '/' experiments(i).
                            name '/' 'Times' '/' '*.*']);
172                    hyp_files = dir([experiment_folder '/' experiments(i).
                            name '/' 'OptHyperparams' '/' '*.*']);
173
174                    if opt.train
175                        guess_files = dir([experiment_folder '/'
                                experiments(i).name '/' 'Guess' '/' 'train_*'])
                                ;
176                    else
177                        guess_files = dir([experiment_folder '/'
                                experiments(i).name '/' 'Guess' '/' 'test_*']);
178                    end
179
180                    % Discard "." and ".."
```

7

```
181                    if ~(exist ('OCTAVE_VERSION', 'builtin') > 0)
182                        time_files = time_files(3:numel(time_files));
183                        hyp_files = hyp_files(3:numel(hyp_files));
184                    end
185
186                    if opt.train
187                        real_files = dir([datasetPath '/' 'train_*']);
188                    else
189                        real_files = dir([datasetPath '/' 'test_*']);
190                    end
191
192                    act = cell(1, numel(predicted_files));
193                    pred = cell(1, numel(predicted_files));
194                    proj = cell(1, numel(guess_files));
195
196                    times = zeros(3,numel(predicted_files));
197                    param = [];
198
199                    for j=1:numel(predicted_files)
200                        pred{j} = importdata([experiment_folder '/'
                                experiments(i).name '/' 'Predictions' '/'
                                predicted_files(j).name]);
201                        times(:,j) = importdata([experiment_folder '/'
                                experiments(i).name '/' 'Times' '/' time_files(
                                j).name]);
202                        proj{j} = importdata([experiment_folder '/'
                                experiments(i).name '/' 'Guess' '/' guess_files
                                (j).name]);
203
204                        if ~isempty(hyp_files)
205                            struct_hyperparams(j) = importdata([
                                    experiment_folder '/' experiments(i).name '
                                    /' 'OptHyperparams' '/' hyp_files(j).name],
                                    ',');
206                            for z = 1:numel(struct_hyperparams(j).data)
207                                param(z,j) = struct_hyperparams(j).data(z);
208                            end
209                        end
210
211
212                        actual = TFGFileReadClass().ReadFile(datasetPath,
                                real_files(j).name,0);
213                        act{j} = actual.targets;
214                    end
215
216                    names = {'Dataset', 'Acc', 'GM', 'MS', 'MAE', 'AMAE', '
                            MMAE','RSpearman', 'Tkendall', 'Wkappa', 'TrainTime
                            ', 'TestTime', 'CrossvalTime'};
217
218                    if ~isempty(hyp_files)
219                        for j=1:numel(struct_hyperparams(1).textdata)
220                            names{numel(names)+1} = struct_hyperparams(1).
                                    textdata{j};
```

```matlab
221                          end
222                      end
223
224                      if exist ('OCTAVE_VERSION', 'builtin') > 0
225                          accs = cell2mat(cellfun(@(varargin) CCR.
                                 calculateMetric(varargin{:}), act, pred, '
                                 UniformOutput', false)) * 100;
226                          gms = cell2mat(cellfun(@(varargin) GM.
                                 calculateMetric(varargin{:}), act, pred, '
                                 UniformOutput', false)) * 100;
227                          mss = cell2mat(cellfun(@(varargin) MS.
                                 calculateMetric(varargin{:}), act, pred, '
                                 UniformOutput', false)) * 100;
228                          maes = cell2mat(cellfun(@(varargin) MAE.
                                 calculateMetric(varargin{:}), act, pred, '
                                 UniformOutput', false));
229                          amaes = cell2mat(cellfun(@(varargin) AMAE.
                                 calculateMetric(varargin{:}), act, pred, '
                                 UniformOutput', false));
230                          maxmaes = cell2mat(cellfun(@(varargin) MMAE.
                                 calculateMetric(varargin{:}), act, pred, '
                                 UniformOutput', false));
231                          spearmans = cell2mat(cellfun(@(varargin) Spearman.
                                 calculateMetric(varargin{:}), act, pred, '
                                 UniformOutput', false));
232                          kendalls = cell2mat(cellfun(@(varargin) Tkendall.
                                 calculateMetric(varargin{:}), act, pred, '
                                 UniformOutput', false));
233                          wkappas = cell2mat(cellfun(@(varargin) Wkappa.
                                 calculateMetric(varargin{:}), act, pred, '
                                 UniformOutput', false));
234                      else
235                          accs = cell2mat(cellfun(@CCR.calculateMetric, act,
                                 pred, 'UniformOutput', false)) * 100;
236                          gms = cell2mat(cellfun(@GM.calculateMetric, act,
                                 pred, 'UniformOutput', false)) * 100;
237                          mss = cell2mat(cellfun(@MS.calculateMetric, act,
                                 pred, 'UniformOutput', false)) * 100;
238                          maes = cell2mat(cellfun(@MAE.calculateMetric, act,
                                 pred, 'UniformOutput', false));
239                          amaes = cell2mat(cellfun(@AMAE.calculateMetric, act
                                 , pred, 'UniformOutput', false));
240                          maxmaes = cell2mat(cellfun(@MMAE.calculateMetric,
                                 act, pred, 'UniformOutput', false));
241                          spearmans = cell2mat(cellfun(@Spearman.
                                 calculateMetric, act, pred, 'UniformOutput',
                                 false));
242                          kendalls = cell2mat(cellfun(@Tkendall.
                                 calculateMetric, act, pred, 'UniformOutput',
                                 false));
243                          wkappas = cell2mat(cellfun(@Wkappa.calculateMetric,
                                 act, pred, 'UniformOutput', false));
244                      end
```

```matlab
245
246                          results_matrix = [accs; gms; mss; maes; amaes; maxmaes;
                                 spearmans; kendalls; wkappas; times(1,:); times
                                 (2,:); times(3,:)];
247                      if ~isempty(hyp_files)
248                          for j=1:numel(struct_hyperparams(1).textdata)
249                              results_matrix = [results_matrix ; param(j,:)
                                     ];
250                          end
251                      end
252
253                      results_matrix = results_matrix';
254
255                      % Results for the independent dataset
256                      if opt.train
257                          fid = fopen([experiment_folder '/' experiments(i).
                                 name '/' 'results_train.csv'],'w');
258                      else
259                          fid = fopen([experiment_folder '/' experiments(i).
                                 name '/' 'results_test.csv'],'w');
260                      end
261
262                      for h = 1:numel(names)
263                          fprintf(fid, '%s,', names{h});
264                      end
265                      fprintf(fid,'\n');
266
267                      for h = 1:size(results_matrix,1)
268                          fprintf(fid, '%s,', real_files(h).name);
269                          for z = 1:size(results_matrix,2)
270                              fprintf(fid, '%f,', results_matrix(h,z));
271                          end
272                          fprintf(fid,'\n');
273                      end
274                      fclose(fid);
275
276                      % Confusion matrices and sum of confusion matrices
277                      if opt.report_sum
278                          if opt.train
279                              fid = fopen([experiment_folder '/' experiments(
                                     i).name '/' 'matrices_train.txt'],'w');
280                          else
281                              fid = fopen([experiment_folder '/' experiments(
                                     i).name '/' 'matrices_test.txt'],'w');
282                          end
283
284                          J = length(unique(act{1}));
285                          cm_sum = zeros(J);
286                          for h = 1:size(results_matrix,1)
287                              fprintf(fid, '%s\n————————\n', real_files(h).
                                     name);
288                              cm = confusionmat(act{h},pred{h});
289                              cm_sum = cm_sum + cm;
```

```matlab
290                            for ii = 1:size(cm,1)
291                                for jj = 1:size(cm,2)
292                                    fprintf(fid, '%d ', cm(ii,jj));
293                                end
294                                fprintf(fid, '\n');
295                            end
296                        end
297                        fclose(fid);
298
299                        % Calculate metrics with the sum of confusion
                                matrices
300                        accs_sum = CCR.calculateMetric(cm_sum) * 100;
301                        gms_sum = GM.calculateMetric(cm_sum) * 100;
302                        mss_sum = MS.calculateMetric(cm_sum) * 100;
303                        maes_sum = MAE.calculateMetric(cm_sum);
304                        amaes_sum = AMAE.calculateMetric(cm_sum);
305                        maxmaes_sum = MMAE.calculateMetric(cm_sum);
306                        spearmans_sum = Spearman.calculateMetric(cm_sum);
307                        kendalls_sum = Tkendall.calculateMetric(cm_sum);
308                        wkappas_sum = Wkappa.calculateMetric(cm_sum);
309                        results_matrix_sum = [accs_sum; gms_sum; mss_sum;
                                maes_sum; amaes_sum; maxmaes_sum; spearmans_sum
                                ; kendalls_sum; wkappas_sum; sum(times(1,:));
                                sum(times(2,:)); sum(times(3,:))];
310
311                        results_matrix_sum = results_matrix_sum';
312                    end
313
314
315                    means = mean(results_matrix,1);
316                    stdev = std(results_matrix,0,1);
317
318                    if opt.train
319                        if ~exist([experiment_folder '/' 'mean-
                                results_train.csv'],'file')
320                            add_head = 1;
321                        else
322                            add_head = 0;
323                        end
324                        fid = fopen([experiment_folder '/' 'mean-
                                results_train.csv'],'at');
325                    else
326                        if ~exist([experiment_folder '/' 'mean-results_test
                                .csv'],'file')
327                            add_head = 1;
328                        else
329                            add_head = 0;
330                        end
331                        fid = fopen([experiment_folder '/' 'mean-
                                results_test.csv'],'at');
332                    end
333
334
```

11

```matlab
335                    if add_head
336                        fprintf(fid, 'Dataset-Experiment,');
337
338                        for h = 2:numel(names)
339                            fprintf(fid, 'Mean%, Std%,', names{h},names{h
                                });
340                        end
341                        fprintf(fid,'\n');
342                    end
343
344
345
346                    fprintf(fid, '%,', experiments(i).name);
347                    for h = 1:numel(means)
348                        fprintf(fid, '%,%f,', means(h), stdev(h));
349                    end
350                    fprintf(fid,'\n');
351                    fclose(fid);
352
353
354                    % Confusion matrices and sum of confusion matrices
355                    if opt.report_sum
356                        if opt.train
357                            fid = fopen([experiment_folder '/' 'mean-
                                results_matrices_sum_train.csv'],'at');
358                        else
359                            fid = fopen([experiment_folder '/' 'mean-
                                results_matrices_sum_test.csv'],'at');
360                        end
361
362                        if add_head
363                            fprintf(fid, 'Dataset-Experiment,');
364
365                            for h = 2:numel(names)
366                                fprintf(fid, '%,', names{h});
367                            end
368                            fprintf(fid,'\n');
369                        end
370
371                        fprintf(fid, '%,', experiments(i).name);
372                        for h = 1:numel(results_matrix_sum)
373                            fprintf(fid, '%,', results_matrix_sum(h));
374                        end
375                        fprintf(fid,'\n');
376                        fclose(fid);
377                    end
378
379                end
380
381            end
382            rmpath(fullfile(fileparts(which('Utilities.m')),'Measures'));
383            rmpath(fullfile(fileparts(which('Utilities.m')),'Algorithms'));
384
```

## 2.1. Clase Utilities

```matlab
385
386          end
387
388          function logsDir = configureExperiment(expFile,dirSuffix)
389              %CONFIGUREEXPERIMENT Function for setting the configuration of
                     the
390              %  different experiments.
391              %  LOGSDIR = CONFIGUREEXPERIMENT(EXPFILE,DIRSUFFIX) parses
                   EXPFILE and
392              %      generates single experiment files describing individual
                    experiment
393              %      of each fold. It also creates folders to store
                   predictions
394              %      and models for all the partitions. All the resources
                   are
395              %      created int exp-DIRSUFFIX folder.
396              if( ~(exist(expFile,'file')))
397                  error('The file %s does not exist\n',expFile);
398              end
399
400              logsDir = ['Experiments' '/' 'exp-' dirSuffix];
401              resultsDir = [logsDir '/' 'Results'];
402              if ~exist('Experiments','dir')
403                  mkdir('Experiments');
404              end
405              mkdir(logsDir);
406              mkdir(resultsDir);
407
408              %Load and parse conf file
409              cObj = Config(expFile);
410
411              num_experiment = numel(cObj.exps);
412              for e = 1:num_experiment
413                  expObj = cObj.exps{e};
414
415                  id_experiment = expObj.expId;
416                  directory = expObj.general('basedir');
417                  if ~(exist(directory,'dir'))
418                      error('Datasets directory "%s" does not exist',
                           directory)
419                  end
420
421                  archive = 'matlab';
422                  if isKey(expObj.general,'archive')
423                      archive = lower(expObj.general('archive'));
424                  end
425
426                  TFGFileReadClass().Valid_archive(archive);
427
428                  datasets = expObj.general('datasets');
429                  conf_file = [logsDir '/' 'exp-' id_experiment];
430                  [matchstart,matchend,tokenindices,matchstring,tokenstring,
                       tokenname,datasetsList] = regexpi(datasets,',');
```

13

```matlab
431                    % Check that all datasets partitions are accesible
432                    % The method checkDatasets calls error
433                    Utilities.checkDatasets(directory, datasets, archive);
434
435                    [train, test] = Utilities.processDirectory(directory,
                           datasetsList, archive);
436
437                    % Generate one config file and corresponding directories
438                    % for each fold.
439                    for i=1:numel(train)
440                        aux_directory = [resultsDir '/' datasetsList{i} '-'
                               id_experiment];
441                        mkdir(aux_directory);
442
443                        mkdir([aux_directory '/' 'OptHyperparams']);
444                        mkdir([aux_directory '/' 'Times']);
445                        mkdir([aux_directory '/' 'Models']);
446                        mkdir([aux_directory '/' 'Predictions']);
447                        mkdir([aux_directory '/' 'Guess']);
448
449                        file = [resultsDir '/' datasetsList{i} '-'
                               id_experiment '/' 'dataset'];
450                        fich = fopen(file,'w');
451                        fprintf(fich, [directory '/' datasetsList{i} '/'
                               archive]);
452                        fclose(fich);
453
454                        runfolds = numel(train{i});
455                        for j=1:runfolds
456                            iniFile = [conf_file '-' datasetsList{i} '-'
                                   num2str(j) '.ini'];
457
458                            expObj.general('directory') = [directory '/'
                                   datasetsList{i} '/' archive];
459                            expObj.general('train') = train{i}(j).name;
460                            expObj.general('test') = test{i}(j).name ;
461                            expObj.general('results') = [resultsDir '/'
                                   datasetsList{i} '-' id_experiment];
462
463                            expObj.writeIni(iniFile);
464                        end
465                    end
466                end
467            end
468
469        function runExperimentFold(confFile)
470            %RUNEXPERIMENTFOLD(CONFFILE) launch a single experiment
                   described in
471            %    file CONFFILE
472            addpath(fullfile(fileparts(which('Utilities.m')),'Measures'));
473            addpath(fullfile(fileparts(which('Utilities.m')),'Algorithms'))
                   ;
474
```

```matlab
475                 auxiliar = Experiment;
476                 auxiliar.launch(confFile);
477
478                 rmpath(fullfile(fileparts(which('Utilities.m')),'Measures'));
479                 rmpath(fullfile(fileparts(which('Utilities.m')),'Algorithms'));
480
481            end
482        end
483
484        methods(Static = true, Access = private)
485
486            function [trainFileNames, testFileNames] = processDirectory(
                    directory, dataSetNames, archive)
487                %PROCESSDIRECTORY Function to get all the train and test pair
                      of
488                %   files of dataset's folds
489                %   [TRAINFILENAMES, TESTFILENAMES] = PROCESSDIRECTORY(
                      DIRECTORY, DATASETNAMES)
490                %   process comma separated list of datasets names in
                      DATASETNAMES.
491                %   All the dataset's folders need to be stored in DIRECTORY.
492                %   Returns all the pairs of train-test files in TRAINFILENAMES
                       and
493                %   TESTFILENAMES.
494                %   [TRAINFILENAMES, TESTFILENAMES] = PROCESSDIRECTORY(
                      DIRECTORY,
495                %   'all') process all datasets in DIRECTORY.
496                dbs = dir(directory);
497                dbs(2) = [];
498                dbs(1) = [];
499                validDataSets = 1;
500
501                trainFileNames = cell(numel(dataSetNames),1);
502                testFileNames = cell(numel(dataSetNames),1);
503                for j=1:numel(dataSetNames)
504                    dsdirectory = [directory '/' dataSetNames{j}];
505                    if(isdir(dsdirectory))
506                        [trainFileNames{validDataSets},testFileNames{
                            validDataSets}] = ...
507                        TFGFileReadClass().TFGFileName(dsdirectory, archive
                            , dataSetNames{j});
508
509                        validDataSets = validDataSets + 1;
510                    end
511                end
512            end
513
514            function checkDatasets(basedir, datasets, archive)
515                %CHECKDATASETS Test datasets are accessible and with expected
516                %names. Launch error in case a dataset is not found.
517                %   CHECKDATASETS(BASEDIR, DATASETS) tests all DATASETS (comma
518                %   separated list of datasets) in directory BASEDIR.
519
```

```matlab
520                 if ~exist(basedir,'dir')
521                     error('Datasets directory "%s" does not exist', basedir)
522                 end
523
524             dsdirsCell = regexp(datasets, '((\w|-|_)+(\w*))','tokens');
525             for i=1:length(dsdirsCell) % skip . and ..
526                 dsName = dsdirsCell{i};
527                 dsName = dsName{:};
528                 if ~exist([basedir '/' dsName],'dir')
529                     error('Dataset directory "%s" does not exist', [basedir
                             '/' dsName])
530                 end
531
532                 datasetPath = [basedir '/' dsName '/' archive];
533                 dsTrainFiles = dir([datasetPath '/train*']);
534
535                 % Test every train file has a test file
536                 for f=1:length(dsTrainFiles)
537
538                     trainName = dsTrainFiles(f).name;
539                     testName = strrep(trainName, 'train', 'test');
540
541                     cellData = TFGFileReadClass().ReadFile(datasetPath,
                             trainName,0);
542                     trainData = [cellData.patterns cellData.targets];
543                     cellData = TFGFileReadClass().ReadFile(datasetPath,
                             testName,0);
544                     testData = [cellData.patterns cellData.targets];
545
546                     if size(trainData,2) ~= size(testData,2)
547                         error('Train and test data dimensions do not agree
                                 for dataset "%s"', dsName)
548                     end
549
550                 end
551             end
552
553         end
554
555         function preparePool(numcores)
556             %PREPAREPOOL(NUMCORES) creates a pool of workers. Function to
557             %abstract code from different matlab versions. Adapt the pool
558             %to the desired number of cores. If there is a current pool
                     with
559             %desired number of cores do not open again to save time
560             if (exist ('OCTAVE_VERSION', 'builtin') > 0)
561                 maximum_ncores = nproc;
562             else
563                 maximum_ncores = feature('numCores');
564             end
565
566             % Adjust number of cores
567             if numcores > maximum_ncores
```

```
568                    disp (['Number of cores was too high and was set up to the
                          maximum available: ' num2str(feature('numCores')) ])
569                    numcores = maximum_ncores;
570               end
571
572            % Check size of the pool
573            if (exist ('OCTAVE_VERSION', 'builtin') > 0)
574                pkg load parallel;
575            else
576                if verLessThan('matlab', '8.3')
577                    poolsize = matlabpool('size');
578                    if poolsize > 0
579                        if poolsize ~= numcores
580                            matlabpool close;
581                            matlabpool(numcores);
582                        end
583                    else
584                        matlabpool(numcores);
585                    end
586                else
587                    poolobj = gcp('nocreate'); % If no pool, do not create
                          new one.
588                    if ~isempty(poolobj)
589                        if poolobj.NumWorkers ~= numcores
590                            numcores = poolobj.NumWorkers;
591                            delete(gcp('nocreate'))
592                            parpool(numcores);
593                        end
594                    else
595                        parpool(numcores);
596                    end
597                end
598            end
599        end
600
601        function closePool()
602            if (exist ('OCTAVE_VERSION', 'builtin') > 0)
603                pkg unload parallel;
604            else
605                if verLessThan('matlab', '8.3')
606                    isOpen = matlabpool('size') > 0;
607                    if isOpen
608                        matlabpool close;
609                    end
610                else
611                    delete(gcp('nocreate'))
612                end
613            end
614        end
615
616        function options = parseParArgs(varargin)
617            %OPTIONS = PARSEPARARGS(VARARGIN) parses parallelization
618            %options with are:
```

17

```
619              %- 'parallel': 'false' or 'true' to activate, default 'false'
620              %- 'numcores': default maximum number of cores or desired
621              %    number. If parallel = 1 and numcores <2 it sets the number
622              %    to maximum number of cores.
623              %- 'closepool': whether to close or not the pool after
624              %    experiments. Default 'true'
625              % Solution adapted from https://stackoverflow.com/questions
                     /2775263/how-to-deal-with-name-value-pairs-of-function-
                     arguments-in-matlab#2776238
626
627              if (exist ('OCTAVE_VERSION', 'builtin') > 0)
628                  maximum_ncores = nproc;
629              else
630                  maximum_ncores = feature('numCores');
631              end
632
633              options = struct('parallel', false, 'numcores', maximum_ncores, '
                     closepool', true);
634
635              varargin = varargin{:};
636              if ~isempty(varargin)
637                  options = parsevarargs(options, varargin);
638                  if options.parallel && options.numcores <2
639                      disp('Number of cores to low, setting to default number
                             of cores')
640                      options.numcores = maximum_ncores;
641                  end
642              end
643          end
644
645      end
646 end
```

CÓDIGO 2.1: Archivo *Utilities.m* correspondiente al módulo ORCA

## 2.2.   Clase Dataset

La clase *Dataset* se encarga de la lectura y el preprocesamiento de los datos. En el Código 2.2 se muestra la implementación de esta clase.

```matlab
1 classdef DataSet < handle
2     %DATASET Class to specify the name of the datasets and perform data
          preprocessing
3     %
4     %   This file is part of ORCA: https://github.com/ayrna/orca
5     %   Original authors: Pedro Antonio Gutiérrez, María Pérez Ortiz,
          Javier Sánchez Monedero
6     %   Citation: If you use this code, please cite the associated paper
          http://www.uco.es/grupos/ayrna/orreview
7     %   Copyright:
8     %       This software is released under the The GNU General Public
          License v3.0 licence
9     %       available at http://www.gnu.org/licenses/gpl-3.0.html
10
11    properties
12        directory = '';
13        train = '';
14        test = '';
15        standarize = true;
16        dataname = '';
17        nOfFolds = 5;
18    end
19
20    methods
21        function obj = dataSet(direct)
22            if(nargin ~= 0)
23                obj.directory = direct;
24            end
25        end
26
27
28        function obj = set.directory(obj,direc)
29            if isdir(direc)
30                obj.directory = direc;
31            else
32                error('%s --> Not a directory', direc);
33            end
34        end
35
36        function [trainSet, testSet] = preProcessData(obj,method)
37        %PREPROCESSDATA preprocess a data partition, i.e., deletes the
               constant
38        %   and non numerical atributes and standarize the data. Test set
39        %   is standardised using train mean and standard error.
```

```matlab
40            %    [TRAINSET, TESTSET] = PREPROCESSDATA() preprocess dataset and
41            %    returns the preprocessed patterns in TRAINSET and TESTSET.
42
43                if(exist([obj.directory '/' obj.train], 'file') && exist([obj.
                      directory '/' obj.test], 'file'))
44                    obj.dataname = strrep(obj.train, 'train_', '');
45                    trainSet = TFGFileReadClass().ReadFile(obj.directory,obj.
                          train,method.categ);
46                    testSet = TFGFileReadClass().ReadFile(obj.directory,obj.
                          test,method.categ);
47
48                    if(obj.standarize)
49                        %[trainSet, testSet] = obj.deleteNonNumericValues(
                              trainSet, testSet);
50                        [trainSet, testSet] = obj.deleteConstantAtributes(
                              trainSet,testSet);
51                        [trainSet, testSet] = obj.standarizeData(trainSet,
                              testSet);
52                        %[trainSet, testSet] = obj.scaleData(trainSet,testSet);
53                    end
54
55
56                    datasetname=[obj.directory '/' obj.train];
57                    [matchstart,matchend] = regexpi(datasetname,'/');
58                    trainSet.name = datasetname(matchend(end)+1:end);
59
60                    datasetname=[obj.directory '/' obj.test];
61                    [matchstart,matchend] = regexpi(datasetname,'/');
62                    testSet.name = datasetname(matchend(end)+1:end);
63                else
64                    error('Can not find the files');
65                end
66        end
67
68
69    end
70    methods (Static = true)
71
72        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
73        %
74        % Function: standarizeData (static)
75        % Description:
76        % Type: It returns the standarized patterns (train and test)
77        % Arguments:
78        %           trainSet--> Array of training patterns
79        %           testSet--> Array of testing patterns
80        %
81        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
82
83        function [trainSet, testSet] = standarizeData(trainSet,testSet)
84        %STANDARIZEDATA standarizes a set of training and testing patterns
               .
85        %    [TRAINSET, TESTSET] = STANDARIZEDATA(TRAINSET,TESTSET)
```

```matlab
86              %    standarizes TRAINSET and TESTSET with TRAINSET mean and std.
87              [trainSet.patterns, trainMeans, trainStds] = DataSet.
                    standarizeFunction(trainSet.patterns);
88              testSet.patterns = DataSet.standarizeFunction(testSet.patterns,
                    trainMeans, trainStds);
89          end
90
91          function [XN, XMeans, XStds] = standarizeFunction(X,XMeans,XStds)
92          %STANDARIZEFUNCTION standardises data with patterns stored in rows
                .
93          %    [XN, XMeans, XStds] = standarizeFunction(X) standardises X
94          %    using X mean and std. Returns normalised data in XN and
95          %    calculated mean and std in XMEANS and XSTDS respectively
96          %    [XN, XMeans, XStds] = standarizeFunction(X,XMeans,XStds)
                standardises X
97          %    using XMeans as mean and XStds as std.
98
99              if (nargin<3)
100                 XStds = std(X);
101             end
102             if (nargin<2)
103                 XMeans = mean(X);
104             end
105             XN = zeros(size(X));
106             for i=1:size(X,2)
107                 XN(:,i) = (X(:,i) - XMeans(i)) / XStds(i);
108             end
109         end
110
111         function [trainSet, testSet] = scaleData(trainSet,testSet)
112         %SCALEDATA scales a set of training and testing patterns.
113         %    [TRAINSET, TESTSET] = SCALEDATA(TRAINSET,TESTSET)
114         %    scales TRAINSET and TESTSET.
115             for i = 1:size(trainSet.patterns,1)
116                 for j = 1:size(trainSet.patterns,2)
117                     trainSet.patterns(i,j) = 1/(1+exp(-trainSet.patterns(i,
                        j)));
118                 end
119             end
120             for i = 1:size(testSet.patterns,1)
121                 for j = 1:size(testSet.patterns,2)
122                     testSet.patterns(i,j) = 1/(1+exp(-testSet.patterns(i,j)
                        ));
123                 end
124             end
125         end
126
127         function [trainSet, testSet] = deleteNonNumericValues(trainSet,
                testSet)
128         %DELETENONNUMERICVALUES This function deletes non numerical values
129         %    in the data, as NaN or Inf.
130         %    [TRAINSET, TESTSET] = DELETENONNUMERICVALUES(TRAINSET,TESTSET)
131         %    performs data cleaning on arrays of patterns TRAINSET and
```

21

```matlab
            TESTSET. Returns
132     %   processed matrices.

133

134         [fils,cols]=find(isnan(trainSet.patterns) | isinf(trainSet.
                patterns));
135         cols = unique(cols);
136         for a = size(cols):-1:1
137             trainSet.patterns(:,cols(a)) = [];
138         end

139

140         [fils,cols]=find(isnan(trainSet.targets) | isinf(trainSet.
                targets));
141         cols = unique(cols);
142         for a = size(cols):-1:1
143             trainSet.patterns(:,cols(a)) = [];
144         end

145

146         [fils,cols]=find(isnan(testSet.patterns) | isinf(testSet.
                patterns));
147         cols = unique(cols);
148         for a = size(cols):-1:1
149             testSet.patterns(:,cols(a)) = [];
150         end

151

152         [fils,cols]=find(isnan(testSet.targets) | isinf(testSet.targets
                ));
153         cols = unique(cols);
154         for a = size(cols):-1:1
155             testSet.patterns(:,cols(a)) = [];
156         end

157

158     end

159

160     function [trainSet,testSet] = deleteConstantAtributes(trainSet,
            testSet)
161     %DELETECONSTANTATRIBUTES This function deletes constant variables
162     %   [TRAINSET, TESTSET] = DELETECONSTANTATRIBUTES(TRAINSET,TESTSET)
163     %   performs data cleaning on arrays of patterns TRAINSET and
            TESTSET. Returns
164     %   processed matrices.

165

166         all = [trainSet.patterns ; testSet.patterns];

167

168         minvals = min(all);
169         maxvals = max(all);

170

171         r = 0;
172         for k=1:size(trainSet.patterns,2)
173             if minvals(k) == maxvals(k)
174                 r = r + 1;
175                 index(r) = k;
176             end
177         end
```

```matlab
178
179              if r > 0
180                  r = 0;
181                  for k=1:size(index,2)
182                      trainSet.patterns(:,index(k)-r) = [];
183                      testSet.patterns(:,index(k)-r) = [];
184                      r = r + 1;
185                  end
186              end
187          end
188      end
189 end
```

Código 2.2: Archivo *Dataset.m* correspondiente al módulo ORCA

## 2.3. Clase Experiment

La clase *Experiment* se encarga de la ejecución del experimento y guardar sus resultados. En el Código 2.3 se muestra la implementación de esta clase.

```
1  % File changed for TFG
2  classdef Experiment < handle
3      %EXPERIMENT creates an experiment to run an ORCA's experiment which
4      %   consist on optimising and running a method in fold (a pair of train
            −test
5      %   dataset partition). Theexperiment is described by a configuration
            file.
6      %   This class is used by Utilities to launch a set of experiments
7      %
8      %   EXPERIMENT properties:
9      %       data                − DataSet object to store the train/test data
10     %       method              − Method to learn and classify data
11     %       cvCriteria          − Metric to guide the grid search for
            parameters optimisation
12     %       resultsDir          − Directory to store performance reports and
            learned models
13     %       seed                − Seed to be used for random number
            generation
14     %       crossvalide         − Activate corssvalidation
15     %
16     %   EXPERIMENT methods:
17     %       launch              − Launch experiment described in file
18     %
19     %   This file is part of ORCA: https://github.com/ayrna/orca
20     %   Original authors: Pedro Antonio Gutiérrez, María Pérez Ortiz,
            Javier Sánchez Monedero
21     %   Citation: If you use this code, please cite the associated paper
            http://www.uco.es/grupos/ayrna/orreview
22     %   Copyright:
23     %       This software is released under the The GNU General Public
            License v3.0 licence
24     %       available at http://www.gnu.org/licenses/gpl−3.0.html
25     %
26     properties
27         data = DataSet;
28         method = Algorithm;
29         cvCriteria = MAE;
30         crossvalide = 0;
31         resultsDir = '';
32         % calculate metrics with the sum of matrices (only suitable for
33         %k−fold experimental design)
34         report_sum = 0;
35         seed = 1;
36         parameters; % parameters to optimize
```

```matlab
37        end
38
39        properties (SetAccess = private)
40
41            logsDir
42        end
43
44        methods
45            function obj = launch(obj,expFile)
46                %LAUNCH Launch experiment described in file.
47                %   EXPOBJ = LAUNCH(EXPFILE) parses EXPFILE and run the
                           experiment
48                %   described on it. It performs the following steps:
49                %   # Preprocess data cleaning and standardization (option need
                            to be actived in configuration file)
50                %   # Optimize parameters by performing a grid search (if
                           selected
51                %   in configuration file)
52                %   # Run algorithm with optimal parameters (if crossvalidation
                            was
53                %   selected)
54                %   # Save experiment results for the fold
55                obj.process(expFile);
56                obj.run();
57            end
58        end
59
60        methods(Access = private)
61
62            function obj = run(obj)
63                %RUN do experiment steps: data cleaning and standardization,
                           parameters
64                %   optimization and save results
65                [train,test] = obj.data.preProcessData(obj.method); %Modificado
                           TFG añadido el parametro obj.method
66
67                if obj.crossvalide
68                    c1 = clock;
69                    Optimals = obj.crossValideParams(train);
70                    c2 = clock;
71                    crossvaltime = etime(c2,c1);
72                    totalResults = obj.method.runAlgorithm(train, test,
                               Optimals);
73                    totalResults.crossvaltime = crossvaltime;
74                else
75                    totalResults = obj.method.runAlgorithm(train, test);
76                end
77
78                obj.saveResults(totalResults);
79            end
80
81            function obj = process(obj,fname)
82                %PROCESS parses experiment described in FNAME
```

```matlab
83                cObj = Config(fname);
84                expObj = cObj.exps{:};
85                % Copy ini values to corresponding object properties
86
87                % General experiment properties
88                if expObj.general.isKey('num_folds')
89                    obj.data.nOfFolds = str2num(expObj.general('num_folds'));
90                end
91                if expObj.general.isKey('standarize')
92                    obj.data.standarize = str2num(expObj.general('standarize'))
                        ;
93                end
94                if expObj.general.isKey('cvmetric')
95                    met = upper(expObj.general('cvmetric'));
96                    eval(['obj.cvCriteria = ' met ';']);
97                end
98                if expObj.general.isKey('seed')
99                    obj.seed = str2num(expObj.general('seed'));
100               end
101               if expObj.general.isKey('report_sum')
102                   obj.report_sum = str2num(expObj.general('report_sum'));
103               end
104
105               try
106                   obj.data.directory = expObj.general('directory');
107                   obj.data.train = expObj.general('train');
108                   obj.data.test = expObj.general('test');
109                   obj.resultsDir = expObj.general('results');
110               catch ME
111                   error('Configuration file % does not have mininum fields.
                        Exception %', fname, ME.identifier)
112               end
113
114               % Algorithm properties are transformed to varargs ('key',value)
115               varargs = obj.mapsToCell(expObj.algorithm);
116               alg = expObj.algorithm('algorithm');
117               obj.method = feval(alg, varargs);
118
119               % Parameters to be optimized
120               if ~isempty(expObj.params)
121                   pkeys = expObj.params.keys;
122                   for p=1:cast(expObj.params.Count,'int32')
123                       %isfield(obj.parameters.' pkeys{p})
124                       eval(['obj.parameters.' pkeys{p} ' = [' expObj.params(
                            pkeys{p}) '];']);
125                       obj.crossvalide = 1;
126                   end
127               end
128           end
129
130           function obj = saveResults(obj,TotalResults)
131               %SAVERESULTS saves the results of the experiment and
132               % the best hyperparameters.
```

```matlab
133
134              par = obj.method.getParameterNames();
135          if ~isempty(par)
136              outputFile = [obj.resultsDir filesep 'OptHyperparams'
                      filesep obj.data.dataname ];
137              fid = fopen(outputFile,'w');
138
139              for i=1:(numel(par))
140                  value = getfield(TotalResults.model.parameters,par{i});
141                  fprintf(fid,'%s,%f\n', par{i},value);
142              end
143
144              fclose(fid);
145          end
146
147          outputFile = [obj.resultsDir filesep 'Times' filesep obj.data.
                  dataname ];
148          fid = fopen(outputFile,'w');
149          if obj.crossvalide
150              fprintf(fid, '%f\n%f\n%f', TotalResults.trainTime,
                      TotalResults.testTime, TotalResults.crossvaltime);
151          else
152              fprintf(fid, '%f\n%f\n%f', TotalResults.trainTime,
                      TotalResults.testTime, 0);
153          end
154          fclose(fid);
155
156
157          outputFile = [obj.resultsDir filesep 'Predictions' filesep obj.
                  data.train ];
158          dlmwrite(outputFile, TotalResults.predictedTrain);
159          outputFile = [obj.resultsDir filesep 'Predictions' filesep obj.
                  data.test ];
160          dlmwrite(outputFile, TotalResults.predictedTest);
161
162          model = TotalResults.model;
163          % Write complete model
164          outputFile = [obj.resultsDir filesep 'Models' filesep obj.data.
                  dataname '.mat'];
165          save(outputFile, 'model');
166
167          outputFile = [obj.resultsDir filesep 'Guess' filesep obj.data.
                  train ];
168          dlmwrite(outputFile, TotalResults.projectedTrain, 'precision',
                  '%.15f');
169
170          outputFile = [obj.resultsDir filesep 'Guess' filesep obj.data.
                  test ];
171          dlmwrite(outputFile, TotalResults.projectedTest, 'precision', '
                  %.15f');
172
173          if obj.method.export
174              obj.method.export_rules([obj.resultsDir filesep 'Models']);
```

```matlab
175                end
176
177                if obj.method.visual
178                    obj.method.visual_rules();
179                end
180         end
181
182         function optimals = crossValideParams(obj,train)
183             %CROSSVALIDEPARAMS Function for performing the crossvalidation
                      in a specific train partition.
184             %
185             %   OPTIMALS = CROSSVALIDEPARAMS(TRAIN) Divides the data in k-
                      folds
186             %   (k defined by 'num fold' in configuration file). Returns
187             %    structure OPTIMALS with optimal parameter(s)
188             optimals = paramopt(obj.method,obj.parameters,train, 'metric',
                      obj.cvCriteria ,...
189                                 'nfolds', obj.data.nOfFolds, 'seed', obj.
                                     seed);
190
191         end
192
193     end
194
195     methods (Static = true)
196
197         function varargs = mapsToCell(aObj)
198             %varargs = mapsToCell(mapObj) returns key value pairs in a
                      comma separated
199             %    string. Example: "'kernel', 'rbf', 'c', 0.1"
200
201             % If there are no parameters return empty cell
202             if aObj.Count == 1
203                 varargs = cell(1,1);
204                 return
205             end
206
207             mapObj = containers.Map(aObj.keys,aObj.values);
208             mapObj.remove('algorithm');
209             pkeys = mapObj.keys;
210             varargs = cell(1,cast(mapObj.Count, 'int32')*2);
211
212             for p=1:2:(cast(mapObj.Count, 'int32')*2)
213                 p = cast(p, 'int32');
214                 keyasstr = pkeys(p/2);
215                 keyasstr = keyasstr{:};
216                 value = mapObj(keyasstr);
217                 varargs{1,p} = sprintf('%s', pkeys{p/2});
218                 % Check numerical values
219                 valuenum = str2double(value);
220                 if isnan(valuenum) % we have a string
221                     varargs{1,p+1} = sprintf('%s', value);
222                 else % we have a number
```

```
223                        varargs {1,p+1} = valuenum ;
224               end
225            end
226         end
227      end
228
229
230 end
```

CÓDIGO 2.3: Archivo *Dataset.m* correspondiente al módulo ORCA

# 3 Módulo Read File

## 3.1. Clase TFGFileReadClass

La clase *TFGFileReadClass* sirve de intermediario entre el módulo ORCA y el módulo Read File. En el Código 3.1 se muestra la implementación de esta clase.

```matlab
1  classdef TFGFileReadClass
2      properties (Access = private)
3          path = fullfile(fileparts(which('TFGFileReadClass.m')),'
             TFGReadFiles');
4      end
5
6      methods
7          function Valid_archive(obj,archive)
8              addpath(obj.path);
9
10             if exist(fullfile(obj.path,[lower(strtrim(archive)) '.m']),'
                  file') ~= 2 || strcmpi(archive,'ReadFileCommon')
11                 error('"%s" unsupported file type', archive)
12             end
13
14             rmpath(obj.path);
15         end
16
17         function datas = ReadFile(obj,directory,file,cat)
18             addpath(obj.path);
19
20             folders = strsplit(directory,'/');
21             archive = char(folders(end));
```

```matlab
22
23                  TFGReadFiles = feval(archive);
24                  raw = [directory '/' file];
25                  datas = TFGReadFiles.ReadFileFunction(raw,cat);
26
27                  rmpath(obj.path);
28          end
29
30          function [trainFileName,testFileName] = TFGFileName(obj,dsdirectory
                    , archive, dataSetName)
31                  addpath(obj.path);
32
33                  format = feval(archive);
34                  [trainFileName,testFileName] = format.FormatFile(dsdirectory,
                        archive,dataSetName);
35
36                  rmpath(obj.path);
37          end
38      end
39
40      methods (Static, Access = private)
41          function cols = SearchInvalidValue(datas)
42                  [~,cols] = find(isnan(datas) | isinf(datas));
43          end
44      end
45 end
```

CÓDIGO 3.1: Archivo *TFGFileReadClass.m* correspondiente al módulo Read File

## 3.2.    Clase ReadFileCommon

La clase *ReadFileCommon* es una clase abstracta que define los ajustes para los métodos de lectura de datos. En el Código 3.2 se muestra la implementación de esta clase.

```matlab
1 classdef ReadFileCommon < handle
2     properties
3         info = [];
4         categ_att = [];
5     end
6
7     properties (Access = protected)
8         categ = 0;
9     end
10
11    methods
12        function [trainFileName,testFileName] = FormatFile(obj,dsdirectory,
               archive,dataSetName)
13            [file_train_expr, file_test_expr] = obj.Format(dataSetName);
14
15            file_expr = [dsdirectory '/' archive '/' file_train_expr];
16            trainFileName = dir(file_expr);
17            file_expr = [dsdirectory '/' archive '/' file_test_expr];
18            testFileName = dir(file_expr);
19        end
20
21        function [file_train_expr, file_test_expr] = Format(obj,dataSetName
               )
22            error('format should be implemented in all subclasses');
23        end
24
25        function datas = ReadFileFunction(obj,file,cat)
26            obj.categ = cat;
27            try
28                datas = obj.ReadFile(file);
29            catch ME
30                error('Cannot read file "%s" \n %s', file, ME.message)
31            end
32            datas = obj.deleteNonNumericValues(datas);
33
34            datas.info.personal = obj.info;
35            datas.info.utilities.type = class(obj);
36            datas.info.utilities.categ_att = obj.categ_att;
37        end
38
39        function datas = ReadFile(obj,file)
40            error('ReadFile method should be implemented in all subclasses'
               );
```

```
41          end
42      end
43
44      methods (Access = private)
45          function datas = deleteNonNumericValues(obj,datas)
46              % Search invalid data on targets
47              [fils,cols] = find(isnan(datas.targets) | isinf(datas.targets))
                    ;
48              del = fils;
49
50              if obj.categ == 0 || ~iscell(datas.patterns)
51                  % Search invalid data on patterns
52                  [fils,cols] = find(isnan(datas.patterns) | isinf(datas.
                        patterns));
53                  del = unique([del;fils]);
54
55              else
56                  for i = 1:size(datas.patterns,1)
57                      for j = 1:size(datas.patterns,2)
58                          if ~ischar(datas.patterns{i,j}) && isnan(datas.
                                patterns{i,j})
59                              del = [del;i];
60                          end
61                      end
62                  end
63                  del = unique(del);
64              end
65
66              if isempty(del)
67                  return;
68              end
69
70              % Delete lines whit invalid data
71              datas.targets(del,:) = [];
72              datas.patterns(del,:) = [];
73          end
74      end
75 end
```

CÓDIGO 3.2: Archivo *ReadFileCommon.m* correspondiente al módulo Read File

## 3.3.   Clase matlab

La clase *matlab* es la que se encarga de la lectura de datos de los ficheros de tipo 'matlab'. En el Código 3.3 se muestra la implementación de esta clase.

```matlab
1 classdef matlab < ReadFileCommon
2     methods
3         function [file_train_expr, file_test_expr] = Format(obj,dataSetName)
4             file_train_expr = ['train_' dataSetName '.*'];
5             file_test_expr = ['test_' dataSetName '.*'];
6         end
7
8         function datas = ReadFile(obj,file)
9             raw = load(file);
10
11            datas.targets = raw(:,end);
12            datas.patterns = raw(:,1:end-1);
13        end
14     end
15 end
```

CÓDIGO 3.3: Archivo *matlab.m* correspondiente al módulo Read File

## 3.4. Clase weka

La clase *weka* es la que se encarga de la lectura de datos de los ficheros de tipo 'weka'. En el Código 3.4 se muestra la implementación de esta clase.

```matlab
1  classdef weka < ReadFileCommon
2      properties
3          attrs = [];
4      end
5
6      methods
7          function [file_train_expr, file_test_expr] = Format(obj,dataSetName)
8              file_train_expr = ['train_' dataSetName '-*.arff'];
9              file_test_expr = ['test_' dataSetName '-*.arff'];
10         end
11
12         function datas = ReadFile(obj,file)
13             [datas.patterns, datas.targets] = obj.ReadWekaFile(file);
14             obj.info.attrs = obj.attrs;
15         end
16     end
17
18     methods (Access = private)
19         function [patterns,targets] = ReadWekaFile(obj,file_name)
20             file = fopen(file_name, 'rt');
21
22             % Read header
23             obj.ReadHeader(file);
24
25             % Read datas
26             [patterns,targets] = obj.ReadDatas(file);
27
28             fclose(file);
29         end
30
31         function ReadHeader(obj,file)
32             while ~feof(file)
33                 line = fgetl(file);
34
35                 if ~isempty(line)
36                     vec = strsplit(line,' ');
37                     if strcmpi(vec(1),'@attribute')
38                         % Check if attribute have a name
39                         % and type
40                         if length(vec) < 3
41                             error('Attribute incorrect.');
42                         end
43
```

```matlab
44                        % Read name and type
45                        name = vec(2);
46                        aux = strcat(name,{' '});
47                        aux = aux{1};
48                        ini = length(aux) + 12;
49                        type = lower(line(ini:end));
50
51                        % Add attribute
52                        obj.NewAttribute(name,type);
53                  elseif strcmpi(vec(1),'@data')
54                        if length(obj.attrs) < 2
55                            error('Need more attributes.');
56                        end
57                        return;
58                  end
59              end
60          end
61          error('Unrecognized as WEKA format.')
62      end
63
64      function NewAttribute(obj,name,type)
65          % Comprobar que no exista el nombre en otro atributo
66          if ~isempty(obj.attrs)
67              if ismember(name,[obj.attrs.name])
68                  error('Attributes with same name.');
69              end
70          end
71
72          % Comprobar el tipo de atributo
73          info = [];
74          if ~strcmp(type,'numeric')
75              indexL = strfind(type,'{');
76              indexR = strfind(type,'}');
77              if length(indexL) == 1 && length(indexR) == 1 && indexL ==
                      1 && indexR == length(type)
78                  type = strrep(type(2:length(type)-1),' ','');
79                  [info,num] = strsplit(type,',');
80                  for i = num
81                      if length(i{1}) ~= 1
82                          error('Categoric attribute without type.');
83                      end
84                  end
85                  type = 'categoric';
86              else
87                  error('Attributes should be numeric or categoric.');
88              end
89          end
90
91          % Guardar los datos
92          aux.name = name;
93          aux.type = type;
94          aux.info = info;
95          obj.attrs = [obj.attrs;aux];
```

```
96              end
97
98          function [patterns,targets] = ReadDatas(obj,file)
99              % Leer los datos
100             datas = [];
101             while ~feof(file)
102                 line = fgetl(file);
103                 line = strrep(line,' ','');
104                 if ~isempty(line)
105                     att_datas = strsplit(line,',');
106                     datas = [datas;att_datas];
107                 end
108             end
109             datas = lower(datas);
110
111             if isempty(datas)
112                 error('No data found.');
113             end
114
115             % Guardar las entradas
116             patterns_aux = datas(:,1:end-1);
117             patterns = [];
118             att_aux = [];
119             for i = 1:size(patterns_aux,2)
120                 if strcmp(obj.attrs(i).type,'categoric')
121                     if obj.categ == 0
122                         [patt,atti] = obj.ToOneHot(patterns_aux(:,i),obj.
                                attrs(i));
123                         patterns = [patterns patt];
124                         att_aux = [att_aux;atti];
125                     else
126                         aux = patterns_aux(:,i);
127                         elements = obj.attrs(i).info;
128                         [k,~] = obj.Categoric_to_Numeric(aux,elements);
129                         aux(find(isnan(k),1)) = {NaN};
130
131                         patterns = [patterns aux];
132                         att_aux = [att_aux;obj.attrs(i)];
133                     end
134                 elseif strcmp(obj.attrs(i).type,'numeric')
135                     line_aux = zeros(length(patterns_aux(:,i)),1);
136                     for j = 1:length(line_aux)
137                       line_aux(j) = str2double(patterns_aux(j,i));
138                     end
139                     if obj.categ == 0
140                         patterns = [patterns line_aux];
141                         att_aux = [att_aux;obj.attrs(i)];
142                     else
143                         aux = patterns_aux(:,i);
144                         aux(find(isnan(line_aux),1)) = {NaN};
145                         patterns = [patterns aux];
146                         att_aux = [att_aux;obj.attrs(i)];
147                     end
```

```matlab
148                    else
149                        error('Attribute type no valid.');
150                    end
151                end
152                obj.attrs = [att_aux;obj.attrs(end)];
153
154                % Gardar las salidas
155                datas = datas(:,end);
156                att = obj.attrs(end);
157                [targets,obj.attrs(end)] = obj.ToNumeric(datas,att);
158            end
159
160            function [datas,attnew] = ToOneHot(obj,patterns,att)
161                % Convert datas
162                datas = zeros(size(patterns,1),size(att.info,2));
163                for i = 1:size(datas,1)
164                    for j = 1:size(datas,2)
165                        datas(i,j) = double(strcmp(patterns{i},att.info{j}));
166                    end
167                end
168
169                % Check all values are valids
170                ind = ~sum(datas,2);
171                datas(ind,:) = NaN;
172
173                % Update attribute
174                attnew = [];
175                for i = 1:size(att.info,2)
176                    att_aux.type = 'categoric';
177                    att_aux.name = strcat(att.name,'_',int2str(i));
178                    att_aux.info = ['0' '1'];
179                    attnew = [attnew;att_aux];
180                end
181            end
182
183            function [datas,att] = ToNumeric(obj,datas,att)
184                if strcmpi(att.type,'numeric')
185                    line_aux = zeros(size(datas));
186                    for j = 1:size(line_aux,1)
187                        line_aux(j) = str2double(datas(j));
188                    end
189                    datas = line_aux;
190                elseif strcmpi(att.type,'categoric')
191                    elements = att.info;
192                    [datas,convert] = obj.Categoric_to_Numeric(datas,elements);
193                    att.info = convert;
194                end
195            end
196
197            function [final_datas,targets_type] = Categoric_to_Numeric(obj,
                     datas,elements)
198                % Apuntar la conversion
199                targets_type.cat = elements;
```

```
200                    targets_type.num = 1:length(elements);
201
202                    % Convertir los datos
203                    final_datas = zeros(size(datas,1),size(targets_type.cat,2));
204                    for i = 1:size(final_datas,1)
205                        for j = 1:size(final_datas,2)
206                            final_datas(i,j) = double(strcmp(datas{i},targets_type.
                                 cat{j}));
207                        end
208                    end
209                    final_datas = final_datas * targets_type.num';
210
211                    % Comprobar que ninguno sea un valor no valido
212                    ind = ~sum(final_datas,2);
213                    final_datas(ind,:) = NaN;
214              end
215          end
216 end
```

Código 3.4: Archivo *weka.m* correspondiente al módulo Read File

# 4 Módulo Algorithms

## 4.1. Clase Algorithm

La clase *Algorithm* es una clase abstracta que define los ajustes para los algoritmos que posee ORCA. En el Código 4.1 se muestra la implementación de esta clase.

```matlab
1 classdef Algorithm < handle
2     %ALGORITHM abstract interface class. Abstract class which defines the
3     % settings for the algorithms (common methods and variables).
4     %
5     %    This file is part of ORCA: https://github.com/ayrna/orca
6     %    Original authors: Pedro Antonio Gutiérrez, María Pérez Ortiz,
7             Javier Sánchez Monedero
8     %    Citation: If you use this code, please cite the associated paper
9             http://www.uco.es/grupos/ayrna/orreview
10    %    Copyright:
11    %        This software is released under the The GNU General Public
12            License v3.0 licence
13    %        available at http://www.gnu.org/licenses/gpl-3.0.html
14
15    properties
16        model = [];
17        categ = false;
18        export = false;
19        visual = false
20    end
21
22    methods
23        function mInf = runAlgorithm(obj, train, test, param)
```

```matlab
21              %RUNALGORITHM runs the corresponding algorithm, fitting the
22              % model and testing it in a dataset.
23              %   mInf = RUNALGORITHM(OBJ, TRAIN, TEST, PARAMETERS) learns a
24              %   model with TRAIN data and PARAMETERS as hyper-parameter
25              %   structure of values for the method. It tests the
26              %   generalization performance with TRAIN and TEST data and
27              %   returns predictions and model in mInf structure.
28              if nargin == 3
29                  param = [];
30              else
31                  % Mix parameters with default
32                  obj.setParam(param)
33              end
34              param = obj.parameters;
35
36              c1 = clock;
37              [mInf.projectedTrain, mInf.predictedTrain] = obj.fit(train,
                    param);
38              % Save the model type
39              obj.model.algorithm = class(obj);
40              c2 = clock;
41              mInf.trainTime = etime(c2,c1);
42
43              c1 = clock;
44              [mInf.projectedTest, mInf.predictedTest] = obj.predict(test.
                    patterns);
45              c2 = clock;
46              mInf.testTime = etime(c2,c1);
47              mInf.model = obj.model;
48          end
49
50          function [projectedTrain, predictedTrain] = fit( obj,train,param)
51          %FIT trains the model for the Algorithm method with TRAIN data and
52          % vector of parameters PARAMETERS. Returns the projection of
53          % patterns (only valid for thresshold models) and the predictel
                labels.
54          % The model can be accessed thourgh getModel() method.
55              if nargin < 3
56                  param = [];
57                  if nargin < 2
58                      error('Please provide training data')
59                  end
60              end
61              if ~all(isfield(train, {'patterns','targets'}))
62                  error('Please provide a structure with train patterns and
                        targets')
63              end
64              % check that dimensions agree
65              if ~size(train.patterns,1) == size(train.targets,1)
66                  error('Number of train patterns and targets must agree')
67              end
68
69              [projectedTrain, predictedTrain] = obj.privfit(train, param);
```

```matlab
70              end
71
72          function [projected, predicted]= predict(obj,test)
73          %PREDICT predicts labels of TEST patterns labels using fitted
                   MODEL.
74              % Check if there is a model
75              if isempty(obj.model)
76                  error('The object does not have a fitted model')
77              end
78              % Avoid typicall error of passing a structure instead of
79              % the matrix of independent variables
80              if ~obj.categ && ~isa(test,'double')
81                  error('test parameter has to be a matrix')
82              end
83
84              [projected, predicted]= privpredict(obj,test);
85          end
86
87      % Abstract methods: they have been implemented in this way to
88      % ensure compatibility with Octave. An error is thrown if the
               method
89      % is not implemented in child class.
90
91          function [projectedTrain, predictedTrain] = privfit( obj,train,
                   param)
92              %PRIVFIT trains the model for the Algorithm method. It is
                       called by
93              % super-class Algorithms's 'fit' function. This method is
                       public, but
94              % should not be called by the user.
95              error('train method should be implemented in all subclasses');
96          end
97
98          function [projected, predicted]= privpredict(obj,test)
99              %PREDICT predicts labels of TEST patterns labels using fitted
                       MODEL.
100             % It is called by super-class Algorithms's 'predict' function.
101             % This method is public, but should not be called by the user.
102             error('test method should be implemented in all subclasses');
103         end
104
105         function parseArgs(obj,varargin)
106             %PARSEARGS(VARARGIN) parses a pair of keys-values in matlab
107             % style format. It throws different exceptions if the field
                       does
108             % not exits on the class or if the type assignment is not
                       consistent.
109             if ~isempty(varargin) && ~isempty(varargin{1})
110                 while iscell(varargin{1})
111                     varargin = varargin{1};
112                     if isempty(varargin{1})
113                         return
114                     end
```

```matlab
115                  end
116
117                  %# read the acceptable names
118                  optionNames = fieldnames(obj);
119
120                  %# count arguments
121                  nArgs = length(varargin);
122                  if mod(nArgs,2)
123                      error('parseParArgs needs propertyName/propertyValue
                              pairs')
124                  end
125
126                  for pair = reshape(varargin,2,[]) % pair is {propName;
                          propValue}
127                      inpName = pair{1}; % make case insensitive
128
129                      if any(strcmp(inpName,optionNames))
130                          % overwrite properties.
131                          % check type
132                          if strcmp(class(obj.(inpName)), class(pair{2}))
133                              obj.(inpName) = pair{2};
134                          else
135                              % Check boolean
136                              if islogical(obj.(inpName)) && ...
137                                      (strcmp(pair{2},'true') || strcmp(pair
                                          {2},'false'))
138                                  obj.(inpName) = eval(pair{2});
139                              else
140                                  msg = sprintf('Data type of property ''%''
                                      (%s) not compatible with data type (%s
                                      ) of assigned value in configuration
                                      file', ...
141                                      inpName, class(obj.(inpName)), class(
                                          pair{2}));
142                                  error(msg);
143                              end
144                          end
145                      else
146                          error('Error ''%'' is not a recognized class
                              property name',inpName)
147                      end
148                  end
149              end
150          end
151
152          function setParam(obj,param)
153              %SETPARAM(PARAM) set parameters contained in param and keep
                      default
154              % values of class parameters field. It throws different
                      exceptions if
155              % the field does not exits on the class or if the type
                      assignment is not consistent.
156              % paramNames = fieldnames(obj.parameters);
```

```matlab
157
158             % Ignore empty argument
159             if isempty(param)
160                 return
161             end
162             if ~isstruct(param)
163                 error('parameters variable have to be a structure')
164             end
165             paramNames = fieldnames(param);
166
167             for i = 1:length(paramNames)
168                 inpName = paramNames{i};
169                 if isfield(obj.parameters,inpName)
170                     % check type
171                     if strcmp(class(obj.parameters.(inpName)), class(param
                            .(inpName)))
172                         obj.parameters.(inpName) = param.(inpName);
173                     else
174                         % Check boolean
175                         if islogical(obj.parameters.(inpName)) && ...
176                                 (strcmp(param.(inpName),'true') || strcmp(
                                    param.(inpName),'false'))
177                             obj.parameters.(inpName) = eval(pair{2});
178                         else
179                             msg = sprintf('Data type of property ''%s'' (%s
                                ) not compatible with data type (%s) of
                                assigned value in configuration file', ...
180                                 inpName, class(obj.parameters.(inpName)),
                                    class(param.(inpName)));
181                             error(msg);
182                         end
183                     end
184                 else
185                     error('Error ''%s'' is not a recognized class parameter
                        name',inpName)
186                 end
187             end
188         end
189
190         function m = getModel(obj)
191             m = obj.model;
192         end
193
194         function m = setModel(obj, m)
195             obj.model = m;
196         end
197
198         function name_parameters = getParameterNames(obj)
199             if ~isempty(obj.parameters)
200                 name_parameters = sort(fieldnames(obj.parameters));
201             else
202                 name_parameters = [];
203             end
```

```
204            end
205
206            function export_rules(obj,dir)
207                disp('Class ' + class(obj) + ' can not export rules.')
208            end
209
210            function visual_rules(obj)
211                disp('Class ' + class(obj) + ' can not see rules.')
212            end
213        end
214
215 end
```

CÓDIGO 4.1: Archivo *Algorithm.m* correspondiente al módulo Algorithms

## 4.2.   Clase NSLVOrd

La clase *NSLVOrd* es la que contiene los métodos para ejecutar el algoritmo
NSLVOrd. En el Código 4.2 se muestra la implementación de esta clase.

```matlab
1 classdef NSLVOrd < Algorithm
2     properties
3         description = 'Inclusion del algoritmo NSLVOrd como TFG de Federico
                Garcia-Arevalo Calles';
4         % Parameters to optimize and default value
5         parameters = struct('Seed', 1286082570, 'LabelsInputs', 5, '
              LabelsOutputs', 5,...
6                             'Shift', 35, 'Alpha', 0.5, 'Population', -1, '
                                 MaxIteration', 500,...
7                             'IniProbBin', 0.9, 'CrosProbBin', 0.25, '
                                 MutProbBin', 0.5, 'MutProbEachBin',
                                 0.17,...
8                             'IniProbInt', 0.5, 'CrosProbInt', 0.0, '
                                 MutProbInt', 0.5,'MutProbEachInt', 0.01,...
9                             'IniProbReal', 0.0, 'CrosProbReal', 0.25,'
                                 MutProbReal', 0.5, 'MutProbEachReal',
                                 0.14,...
10                            'SeeRules', 0, 'ExportRules', 0);
11
12    end
13
14    methods (Access = private, Static)
15        function param_java = initParameters(param)
16            param_java = {...
17                num2str(param.Seed),...
18                num2str(param.LabelsInputs),...
19                num2str(param.LabelsOutputs),...
20                num2str(param.Shift),...
21                num2str(param.Alpha),...
22                num2str(param.Population),...
23                num2str(param.MaxIteration),...
24                num2str(param.IniProbBin),...
25                num2str(param.CrosProbBin),...
26                num2str(param.MutProbBin),...
27                num2str(param.MutProbEachBin),...
28                num2str(param.IniProbInt),...
29                num2str(param.CrosProbInt),...
30                num2str(param.MutProbInt),...
31                num2str(param.MutProbEachInt),...
32                num2str(param.IniProbReal),...
33                num2str(param.CrosProbReal),...
34                num2str(param.MutProbReal),...
35                num2str(param.MutProbEachReal)};
36        end
```

47

```matlab
37
38          function header = getHeader(datas)
39              header = '@relation NSLVOrd';
40              if strcmp(datas.info.utilities.type,'weka')
41                  for i = 1:length(datas.info.personal.attrs)-1
42                      line = strcat('@attribute',{' '},datas.info.personal.
                             attrs(i).name,{' '});
43                      if strcmp(datas.info.personal.attrs(i).type,'numeric')
44                          line = strcat(line,datas.info.personal.attrs(i).
                                 type);
45                      elseif strcmp(datas.info.personal.attrs(i).type,'
                             categoric')
46                          line = strcat(line,'{',datas.info.personal.attrs(i)
                                 .info(1));
47                          for j = 2:length(datas.info.personal.attrs(i).info)
48                              line = strcat(line,',',datas.info.personal.
                                     attrs(i).info(j));
49                          end
50                          line = strcat(line,'}');
51                      else
52                          error('error');
53                      end
54                      header = [header;line];
55                  end
56
57                  line = strcat('@attribute',{' '},datas.info.personal.attrs(
                         end).name,{' '});
58                  if strcmp(datas.info.personal.attrs(end).type,'numeric')
59                      error('In NSLVOrd output should be categoric');
60                  elseif strcmp(datas.info.personal.attrs(end).type,'
                         categoric')
61                      line = strcat(line,'{',datas.info.personal.attrs(end).
                             info.cat(1));
62                      for j = 2:length(datas.info.personal.attrs(end).info.
                             cat)
63                          line = strcat(line,',',datas.info.personal.attrs(
                                 end).info.cat(j));
64                      end
65                      line = strcat(line,'}');
66                  else
67                      error('In NSLVOrd output should be categoric');
68                  end
69                  header = [header;line];
70              else
71                  for i = 1:size(datas.patterns,2)
72                      line = strcat('@attribute x',{int2str(i)},' numeric');
73                      header = [header;line];
74                  end
75                  aux = unique(datas.targets);
76                  line = strcat('@attribute y {',num2str(aux(1)),{''});
77                  for i = 2:size(aux,1)
78                      line = strcat(line,',',num2str(aux(i)));
79                  end
```

```matlab
80                    line = strcat(line,'}');
81                    header = [header;line];
82                end
83                header = [header;'@data'];
84            end
85
86            function datas_java = getDatas(datas)
87                [a,b] = size(datas);
88                datas_java = [];
89                for i = 1:a
90                    aux = '';
91                    for j = 1:b-1
92                        dat = datas(i,j);
93                        if strcmpi(class(dat),'double')
94                            dat = num2str(dat);
95                        end
96                        aux = strcat(aux,dat,',');
97                    end
98                    aux = strcat(aux,datas(i,b));
99                    datas_java = [datas_java;aux];
100               end
101           end
102
103           function [targets,output] = ConvertTargetsToCategoric(train)
104               if strcmp(train.info.utilities.type,'weka')
105                   trans = train.info.personal.attrs(end).info;
106                   targets_m = repmat(train.targets,1,length(trans.num));
107                   num_m = repmat(trans.num,length(train.targets),1);
108                   a = (targets_m == num_m) * [1:length(trans.num)]';
109                   targets = trans.cat(a)';
110                   output.cat = trans.cat;
111                   output.num = trans.num;
112               else
113                   aux = unique(train.targets);
114                   aux_cat = [];
115                   for i = 1:length(aux)
116                       aux_cat = [aux_cat {num2str(aux(i))}];
117                   end
118                   output.cat = aux_cat;
119                   output.num = unique(train.targets)';
120                   targets = cellstr(num2str(train.targets));
121               end
122           end
123
124           function patterns = ConvertPatternsToChar(patterns)
125               patt_aux = [];
126               for i = 1:size(patterns,2)
127                   aux = patterns(:,i);
128                   if strcmpi(class(aux),'double')
129                       patt_aux = [patt_aux cellstr(num2str(aux))];
130                   else
131                       patt_aux = [patt_aux aux];
132                   end
```

```matlab
133                    end
134                patterns = patt_aux;
135            end
136
137        function targets = ConvertCategoricToTargets(result,trans)
138            a = zeros(size(result,1),size(trans.cat,2));
139            for i = 1:size(a,1)
140                for j = 1:size(a,2)
141                    a(i,j) = double(strcmp(result(i),trans.cat{j}));
142                end
143            end
144            a = a * [1:length(trans.cat)]';
145            targets = trans.num(a)';
146        end
147
148        function res = toChar(param)
149            res = [];
150            for i = 1:size(param,1)
151                a1 = param(i);
152                res1 = [];
153                for j = 1:size(a1,1)
154                    a2 = a1(j);
155                    res2 = [];
156                    for k = 1:size(a2,1)
157                        a3 = char(a2(k));
158                        res2 = [res2,{a3}];
159                    end
160                    res1 = [res1;{res2}];
161                end
162                res = [res;{res1}];
163            end
164        end
165
166        function res = toCell(param)
167            res = [];
168            for i = 1:size(param,1)
169                res = [res;{char(param(i))}];
170            end
171        end
172
173        function res = toJavaString(param)
174            res = javaArray('java.lang.String[][]',size(param,1));
175            for i = 1:size(param,1)
176                a1 = param{i};
177                res1 = [];
178                for j = 1:size(a1,1)
179                    a2 = a1{j};
180                    res2 = [];
181                    for k = 1:size(a2,2)
182                        a3 = java.lang.String(a2{k});
183                        res2 = [res2,a3];
184                    end
185                    res1 = [res1;res2];
```

```
186                   end
187                   res(i) = [res1;[]];
188              end
189         end
190    end
191
192    methods
193        function obj = NSLVOrd(~,varargin)
194            % Process key-values pairs of parameters
195            obj.parseArgs(varargin);
196            obj.categ = true;
197        end
198
199        function [projectedTrain, predictedTrain] = privfit(obj, train,
                   param)
200            % fit the model and return prediction of train set. It is
                       called by
201            % super class Algorithm.fit() method.
202
203            % Convertir los datos a objetos Java
204            param_java = obj.initParameters(param);
205
206            header = obj.getHeader(train);
207            [targets,output] = obj.ConvertTargetsToCategoric(train);
208            patterns = obj.ConvertPatternsToChar(train.patterns);
209            datas = [patterns targets];
210            datas = obj.getDatas(datas);
211
212            % NSLVOrd Java
213            algorithmPath = fullfile(fileparts(which('Algorithm.m')),'
                   NSLVOrd');
214            jarfolder = fullfile(algorithmPath,'NSLVOrdJava.jar');
215            javaaddpath(jarfolder);
216
217            nslvord = javaObject('NSLVOrdJava.NSLVOrdJava');
218            result = javaMethod('Train', nslvord, header,datas,param_java);
219            knowledgebase = javaMethod('get_knowledge_base', nslvord);
220            rulebase = javaMethod('get_rule_base', nslvord);
221            rules = javaMethod('get_rules', nslvord);
222
223            clear nslvord;
224            javarmpath(jarfolder);
225
226            % Process output
227            %if strcmpi(train.info.utilities.type,'weka')
228            targets = obj.ConvertCategoricToTargets(result,output);
229            projectedTrain = targets;
230            predictedTrain = targets;
231
232            % Save the model
233            try
234                model.name = train.name;
235            catch
```

```matlab
236            end
237            model.output = output;
238            model.knowledgebase =  obj.toCell(knowledgebase);
239            model.rulebase = obj.toCell(rulebase);
240            model.rules = obj.toCell(rules);
241            model.type = train.info.utilities.type;
242            model.header = header;
243            model.parameters = param;
244            obj.model = model;
245
246            % Active export and see rules
247            obj.export = param.ExportRules;
248            obj.visual = param.SeeRules;
249        end
250
251        function [projected, predicted] = privpredict(obj, patterns)
252            % predict unseen patterns with 'obj.model' and return
                    prediction and
253            % projection of patterns (for threshold models)
254            % It is called by super class Algorithm.predict() method.
255
256            % Convert inputs to java objects
257            patterns = obj.ConvertPatternsToChar(patterns);
258            datas = obj.getDatas(patterns);
259
260            % NSLVOrd Java
261            algorithmPath = fullfile(fileparts(which('Algorithm.m')),'
                    NSLVOrd');
262            jarfolder = fullfile(algorithmPath,'NSLVOrdJava.jar');
263            javaaddpath(jarfolder);
264
265            nslvord = javaObject('NSLVOrdJava.NSLVOrdJava');
266            javaMethod('LoadModel', nslvord, obj.model.knowledgebase,obj.
                    model.rulebase);
267            result = javaMethod('Test', nslvord, obj.model.header,datas);
268
269            clear nslvord;
270            javarmpath(jarfolder);
271
272            % Process output
273            output = obj.model.output;
274            targets = obj.ConvertCategoricToTargets(result,output);
275            projected = targets;
276            predicted = targets;
277        end
278
279        function visual_rules(obj)
280            visual = RulesVisual;
281
282            % RuleBase
283            rb = obj.model.rules;
284            numr = str2num(rb{1});
285            finr = 1;
```

```
286                % Rule
287                for i = 1:numr
288                    inir = finr + 1;
289                    finr = inir + 2 + str2num(rb{inir + 2});
290                    r = rb(inir:finr);
291                    rname = r{1};
292                    rweight = str2double(r{2});
293                    numant = str2num(r{4}) - 1;
294                    con = r(end-1:end);
295                    ant = getant(obj,r(5:end-2),numant);
296                    visual.new_rule(rname,rweight);
297                    for j = 1:size(ant,1)
298                        visual.add_antecedent(ant(j).name,ant(j).values);
299                    end
300                    visual.new_consequent(con{1},con{2});
301                end
302
303                % Visual
304                visual.visual_rules(obj.model.name);
305            end
306
307        function export_rules(obj,dir)
308                export = RulesExport;
309
310                % KnowledgeBase
311                kb = obj.model.knowledgebase;
312                numfv = str2num(kb{5});
313                finfv = 5;
314                % FuzzyVariable
315                for i = 1:numfv
316                    inifv = finfv + 1;
317                    numft = str2num(kb{inifv + 8});
318                    finfv = inifv + 8 + 7 * numft;
319                    fv = kb(inifv:finfv);
320                    export.new_variable(fv{1},fv{5},fv{6});
321                    finft = 9;
322                    % FuzzyTerm
323                    for j = 1:numft
324                        inift = finft + 1;
325                        finft = inift + 6;
326                        ft = fv(inift:finft);
327                        export.add_terms(ft{1},ft{2},ft{3},ft{4},ft{5});
328                    end
329                end
330
331                % RuleBase
332                rb = obj.model.rules;
333                numr = str2num(rb{1});
334                finr = 1;
335                % Rule
336                for i = 1:numr
337                    inir = finr + 1;
338                    finr = inir + 2 + str2num(rb{inir + 2});
```

```
339                     r = rb(inir:finr);
340                     rname = r{1};
341                     rweight = str2double(r{2});
342                     numant = str2num(r{4}) - 1;
343                     con = r(end-1:end);
344                     ant = subrules(obj,r(5:end-2),numant);
345                     for j = 1:size(ant,1)
346                         acname = rname;
347                         if size(ant,1) > 1
348                             acname = [rname '_' num2str(j)];
349                         end
350                         export.new_rule(acname,rweight);
351                         antr = ant{j};
352                         for k = 1:2:size(antr,2)
353                             export.add_antecedent(antr{k},antr{k+1});
354                         end
355                         export.new_consequent(con{1},con{2});
356                     end
357                 end
358
359             % Export
360             export.export_rules(dir,obj.model.name);
361         end
362
363         function ant = subrules(obj,r,numant)
364             [ant_aux,comb] = obj.getant(r,numant);
365             ant = [];
366             if ~isempty(ant_aux)
367                 ant = cell(comb,1);
368                 for i = 1:comb
369                     start = comb;
370                     index = i;
371                     for j = 1:size(ant_aux,1)
372                         aux = ant_aux(j);
373                         name = aux.name;
374                         aux = aux.values;
375                         start = start / size(aux,1);
376                         value = aux{floor((index-1)/start) + 1};
377                         index = index - start * floor((index-1)/start);
378                         ant{i} = [ant{i},{name},{value}];
379                     end
380                 end
381             end
382         end
383
384         function [ant,comb] = getant(obj,r,numant)
385             ant = [];
386             fin = 0;
387             comb = 1;
388             for i = 1:numant
389                 ini = fin + 1;
390                 aux.name = r{ini};
391                 numval = str2num(r{ini+1});
```

54

```
392                    fin = ini + 1 + numval * 7;
393                    vr = r(ini:fin);
394                    values = [];
395                    finv = 2;
396                    comb = comb * numval;
397                    for j = 1:numval
398                        iniv = finv + 1;
399                        finv = iniv + 6;
400                        tr = vr(iniv:finv);
401                        aux2 = {tr{1},str2double(tr{2}),str2double(tr{3}),
                                    str2double(tr{4}),str2double(tr{5}),str2num(tr{6}),
                                    str2num(tr{7})};
402                        values = [values;aux2];
403                    end
404                    aux.values = values;
405                    ant = [ant;aux];
406                end
407            end
408        end
409 end
```

CÓDIGO 4.2: Archivo *NSLVOrd.m* correspondiente al módulo Algorithms

# 5 Módulo NSLVOrd

## 5.1. Clase NSLVOrdJava

La clase *NSLVOrdJava* se encarga de la ejecución del algoritmo NSLVOrd. En el Código 5.1 se muestra la implementación de esta clase.

```java
1 package NSLVOrdJava;
2
3 // para la integracion en keel
4 import keel.Dataset.*;
5
6 import java.util.*;
7
8 /**
9  * @file NSLVOrd.java
10  * @brief main file of proyect
11  * @author Juan Carlos Gamez (original de Raul Perez)
12  * @version 1
13  * @date diciembre 2015
14  * @note Implement of NSLV algorithm for ordinal classification
15  */
16 public class NSLVOrdJava {
17
18     // habra 3 valores: indice 0->izda, indice 1->centro, indice 2->dcha
19     // static int numDesplazamientos=3;
20     static int numDesplazamientos=1;
21     static double[] time;
22     static int[] iter;
23
24     static FuzzyProblemClass[] fuzzyProblem;
25     static ExampleSetProcess[] E_par, E_par_test;
```

```
26      static RuleSetClass [] R;
27
28      static InstanceSet iSet;
29      static InstanceSet tSet;
30
31      static Random [] randomNum;
32
33      static String fileResultDebug;
34      static double [][] costMatrix;
35      static int seed;
36      static int numLabelsInputs;
37      static int numLabelsOutput;
38      static int shift;
39
40      static int homogeneousLabel=0; // se elimina el parametro de cuda para
            introducir la creacion de etiquetas homogeneas
41      // parametros de ponderacion de la caracteristica ordinal o nominal de
            la funcion fitness
42      // alpha * CCR y (1−alpha) * MAE
43      static double alpha=0.5;
44
45      static String [] poblationParam;
46
47      public static String [] Train(String [] _header, String [] _datas, String []
            args){
48          Attributes.clearAll();
49          initParameters(args);
50
51          // Aqui se inicializa Random
52          randomNum= new Random[numDesplazamientos];
53          iter= new int[numDesplazamientos];
54          time= new double[numDesplazamientos];
55          for (int i=0; i < numDesplazamientos; i++){
56            randomNum[i]= new Random(seed);
57          }
58
59          // obtener las instancias (ejemplos) de training y test y pasarlas
                a "los objetos de mis clases"
60          if (!ReadSet(_header, _datas, true)) return null;
61
62          if (!executeNSLVOrd(homogeneousLabel)) return null;
63
64          return Targets(E_par,1);
65
66      }
67
68      public static FuzzyProblemClass [] GetFuzzyProblem(){
69          return fuzzyProblem;
70      }
71
72      public static String [] get_knowledge_base(){
73          RuleSystem _exp = new RuleSystem(fuzzyProblem[0], R[0]);
74          String [] kb = _exp.Export_KnowledgeBase();
```

```java
75          return kb;
76      }
77
78      public static String[] get_rules(){
79          RuleSystem _exp = new RuleSystem(fuzzyProblem[0], R[0]);
80          String[] rules = _exp.Export_Rules();
81          return rules;
82      }
83
84      public static String[] get_rule_base(){
85          RuleSystem _exp = new RuleSystem(fuzzyProblem[0], R[0]);
86          String[] rb = _exp.Export_RuleBase();
87          return rb;
88      }
89
90      public static void LoadModel(String[] _fuzzyProblem, String[] _R){
91          fuzzyProblem = new FuzzyProblemClass[numDesplazamientos];
92          R= new RuleSetClass[numDesplazamientos];
93          E_par_test= new ExampleSetProcess[numDesplazamientos];
94
95          Load_KnowledgeBase(_fuzzyProblem);
96          Load_RuleBase(_R);
97      }
98
99      public static void Load_KnowledgeBase(String[] sfp){
100         // FUZZY PROBLEM
101         FuzzyProblemClass fp = new FuzzyProblemClass();
102         fp.setConsequentIndexOriginal(Integer.parseInt(sfp[0]));
103         fp.setShift(Integer.parseInt(sfp[1]));
104         fp.setDirection(Integer.parseInt(sfp[2]));
105         fp.setHomogeneousLabel(Integer.parseInt(sfp[3]));
106         fp.setFuzzyLinguisticVariableNum(Integer.parseInt(sfp[4]));
107
108         // FUZZY VARIABLE
109         FuzzyLinguisticVariableClass[] fv = new
                FuzzyLinguisticVariableClass[fp.getFuzzyLinguisticVariableNum()
                ];
110         int pos = 5;
111         for(int i = 0; i < fp.getFuzzyLinguisticVariableNum(); i++){
112             fv[i] = new FuzzyLinguisticVariableClass();
113             fv[i].setName(sfp[pos]); pos++;
114             fv[i].setUnit(Integer.parseInt(sfp[pos])); pos++;
115             fv[i].setNumTermAutomatic(Double.parseDouble(sfp[pos])); pos++;
116             fv[i].setVariableType(Integer.parseInt(sfp[pos])); pos++;
117             fv[i].setInfRange(Double.parseDouble(sfp[pos])); pos++;
118             fv[i].setSupRange(Double.parseDouble(sfp[pos])); pos++;
119             fv[i].setInfRangeIsInf(Integer.parseInt(sfp[pos])); pos++;
120             fv[i].setSupRangeIsInf(Integer.parseInt(sfp[pos])); pos++;
121             fv[i].setFuzzyLinguisticTermNum(Integer.parseInt(sfp[pos]));
                    pos++;
122
123             // FUZZY TERM
124             FuzzyLinguisticTermClass[] ft = new FuzzyLinguisticTermClass[fv
```

59

```
                        [i].getFuzzyLinguisticTermNum()];
125             for(int j = 0; j < fv[i].getFuzzyLinguisticTermNum(); j++){
126                 ft[j] = new FuzzyLinguisticTermClass();
127                 ft[j].setName(sfp[pos]); pos++;
128                 ft[j].setA(Double.parseDouble(sfp[pos])); pos++;
129                 ft[j].setB(Double.parseDouble(sfp[pos])); pos++;
130                 ft[j].setC(Double.parseDouble(sfp[pos])); pos++;
131                 ft[j].setD(Double.parseDouble(sfp[pos])); pos++;
132                 ft[j].setAbInf(Integer.parseInt(sfp[pos])); pos++;
133                 ft[j].setCdInf(Integer.parseInt(sfp[pos])); pos++;
134             }
135
136             fv[i].setFuzzyLinguisticTermList(ft);
137         }
138
139         fp.setFuzzyLinguisticVariableList(fv);
140
141         fuzzyProblem[0] = fp;
142     }
143
144     public static void Load_RuleBase(String[] srs){
145         // RULE SET
146         RuleSetClass rs = new RuleSetClass();
147         rs.setNumRules(Integer.parseInt(srs[0]));
148         rs.CCR = Double.parseDouble(srs[1]);
149         rs.SM = Double.parseDouble(srs[2]);
150         rs.TPR = Double.parseDouble(srs[3]);
151         rs.TNR = Double.parseDouble(srs[4]);
152         rs.FPR = Double.parseDouble(srs[5]);
153         rs.Kappa = Double.parseDouble(srs[6]);
154         rs.AUC = Double.parseDouble(srs[7]);
155         rs.MSE = Double.parseDouble(srs[8]);
156         rs.RMSE = Double.parseDouble(srs[9]);
157         rs.RMAE = Double.parseDouble(srs[10]);
158         rs.OMAE = Double.parseDouble(srs[11]);
159         rs.OMAENormalizado = Double.parseDouble(srs[12]);
160         rs.MMAE = Double.parseDouble(srs[13]);
161         rs.mMAE = Double.parseDouble(srs[14]);
162         rs.AMAE = Double.parseDouble(srs[15]);
163         rs.Spearman = Double.parseDouble(srs[16]);
164         rs.Kendall = Double.parseDouble(srs[17]);
165         rs.OC = Double.parseDouble(srs[18]);
166         rs.beta = Double.parseDouble(srs[19]);
167         rs.metric = Double.parseDouble(srs[20]);
168         rs.metricMedia = Double.parseDouble(srs[21]);
169         rs.Precision = Double.parseDouble(srs[22]);
170         rs.alphaMetric = Double.parseDouble(srs[23]);
171         rs.confusion = new double[Integer.parseInt(srs[24])][];
172         int pos = 25;
173         for(int i = 0; i < rs.confusion.length; i++){
174             rs.confusion[i] = new double[Integer.parseInt(srs[pos])]; pos
                    ++;
175             for(int j = 0; j < rs.confusion[i].length; j++){
```

```
176                        rs.confusion[i][j] = Double.parseDouble(srs[pos]); pos++;
177                }
178            }
179
180            // RULE
181            GenetCodeClass[] rul = new GenetCodeClass[rs.getNumRules()];
182            for(int i = 0; i < rul.length; i++){
183                // Binary elements
184                int binaryBlocs;
185                int[] sizeBinaryBlocs;
186                int[][] binaryMatrix;
187                binaryBlocs = Integer.parseInt(srs[pos]); pos++;
188                sizeBinaryBlocs = new int[binaryBlocs];
189                binaryMatrix = new int[binaryBlocs][];
190                for(int j = 0; j < binaryBlocs; j++){
191                    sizeBinaryBlocs[j] = Integer.parseInt(srs[pos]); pos++;
192                    binaryMatrix[j] = new int[sizeBinaryBlocs[j]];
193                    for(int k = 0; k < sizeBinaryBlocs[j]; k++){
194                        binaryMatrix[j][k] = Integer.parseInt(srs[pos]); pos++;
195                    }
196                }
197
198                // Integer elements
199                int integerBlocs;
200                int[] sizeIntegerBlocs;
201                int[][] integerMatrix;
202                int[] integerRange;
203                integerBlocs = Integer.parseInt(srs[pos]); pos++;
204                sizeIntegerBlocs = new int[integerBlocs];
205                integerMatrix = new int[integerBlocs][];
206                for(int j = 0; j < integerBlocs; j++){
207                    sizeIntegerBlocs[j] = Integer.parseInt(srs[pos]); pos++;
208                    integerMatrix[j] = new int[sizeIntegerBlocs[j]];
209                    for(int k = 0; k < sizeIntegerBlocs[j]; k++){
210                        integerMatrix[j][k] = Integer.parseInt(srs[pos]); pos
                            ++;
211                    }
212                }
213                integerRange = new int[Integer.parseInt(srs[pos])]; pos++;
214                for(int j = 0; j < integerRange.length; j++){
215                    integerRange[j] = Integer.parseInt(srs[pos]); pos++;
216                }
217
218                // Real elements
219                int realBlocs;
220                int[] sizeRealBlocs;
221                double[][] realMatrix;
222                double[] realInfRange;
223                double[] realSupRange;
224                realBlocs = Integer.parseInt(srs[pos]); pos++;
225                sizeRealBlocs = new int[realBlocs];
226                realMatrix = new double[realBlocs][];
227                for(int j = 0; j < realBlocs; j++){
```

```
228             sizeRealBlocs[j] = Integer.parseInt(srs[pos]); pos++;
229             realMatrix[j] = new double[sizeRealBlocs[j]];
230             for(int k = 0; k < sizeRealBlocs[j]; k++){
231                 realMatrix[j][k] = Double.parseDouble(srs[pos]); pos++;
232             }
233           }
234           realInfRange = new double[Integer.parseInt(srs[pos])]; pos++;
235           for(int j = 0; j < realInfRange.length; j++){
236               realInfRange[j] = Double.parseDouble(srs[pos]); pos++;
237           }
238           realSupRange = new double[Integer.parseInt(srs[pos])]; pos++;
239           for(int j = 0; j < realSupRange.length; j++){
240               realSupRange[j] = Double.parseDouble(srs[pos]); pos++;
241           }
242
243
244           rul[i] = new GenetCodeClass(binaryBlocs, integerBlocs, realBlocs,
245                     sizeBinaryBlocs, sizeIntegerBlocs, sizeRealBlocs,
246                     integerRange, realInfRange, realSupRange);
247           rul[i].setBinaryMatrix(binaryMatrix);
248           rul[i].setIntegerMatrix(integerMatrix);
249           rul[i].setRealMatrix(realMatrix);
250         }
251         rs.setRules(rul);
252
253         R[0] = rs;
254     }
255
256     public static String[] Test(String[] _header, String[] _datas){
257         Attributes.clearAll();
258         iter= new int[numDesplazamientos];
259         time= new double[numDesplazamientos];
260
261         // obtener las instancias (ejemplos) de training y test y pasarlas
262             a "los objetos de mis clases"
262         if(!ReadSet(_header,_datas,false)) return null;
263
264         if(!executeNSLVOrdPredict()) return null;
265
266         return Targets(E_par_test,1);
267     }
268
269     public static void initParameters(String[] param){
270         fuzzyProblem = new FuzzyProblemClass[numDesplazamientos];
271         E_par= new ExampleSetProcess[numDesplazamientos];
272         E_par_test= new ExampleSetProcess[numDesplazamientos];
273         R= new RuleSetClass[numDesplazamientos];
274
275         seed= Integer.parseInt(param[0]);
276         homogeneousLabel = 0;
277         numLabelsInputs= Integer.parseInt(param[1]);
278         numLabelsOutput= Integer.parseInt(param[2]);
279         shift= Integer.parseInt(param[3]);
```

```
280              alpha= Double.parseDouble(param[4]); // indica si realiza
                      clasificacion(1) o regresion(0)
281              if ((alpha + (1-alpha)) != 1){
282                  alpha= 0.5;
283              }
284
285              // el resto de parametros que corresponden a las probabilidades de
286              // inicializacion, cruce y mutacion de las subpoblaciones y de cada
287              // elemento de la subpoblacion
288              poblationParam = new String[14];
289              poblationParam[0]  = param[5];
290              poblationParam[1]  = param[6];
291              poblationParam[2]  = param[7];
292              poblationParam[3]  = param[8];
293              poblationParam[4]  = param[9];
294              poblationParam[5]  = param[10];
295              poblationParam[6]  = param[11];
296              poblationParam[7]  = param[12];
297              poblationParam[8]  = param[13];
298              poblationParam[9]  = param[14];
299              poblationParam[10] = param[15];
300              poblationParam[11] = param[16];
301              poblationParam[12] = param[17];
302              poblationParam[13] = param[18];
303          }
304
305          public static boolean ReadSet(String[] _header, String[] _datas,
                 boolean _train){
306              // obtener las instancias (ejemplos) de training y test y pasarlas
                      a "los objetos de mis clases"
307              InstanceSet _Set= new InstanceSet();
308              _Set.readSetTFG(_header, _datas, _train);
309              _Set.setAttributesAsNonStatic();
310
311              //si no hay ejemplos sale directamente
312              if (_Set.getNumInstances() == 0){
313                return false;
314              }
315
316              if(_train){
317                  iSet = _Set;
318              }else{
319                  tSet = _Set;
320              }
321
322              return true;
323          }
324
325          public static boolean executeNSLVOrd(int homogeneousLabel){
326              // parte de ejecucion en serie
327              randomNum[0]= new Random(seed);
328              return executeLearning(0, 0, 0, homogeneousLabel); // para probar
                      por ahora nada mas que con una ejecucion
```

```
329          // FIN − parte de ejecucion en serie
330      }
331
332      public static boolean executeNSLVOrdPredict() {
333          // parte de ejecucion en serie
334          return executePredict(0, 0, 0); // para probar por ahora nada mas
                 que con una ejecucion
335          // FIN − parte de ejecucion en serie
336      }
337
338      public static boolean executeLearning(int shift , int direction , int
             index , int homogeneousLabel){
339          iter[index]=0;
340          if (numLabelsInputs == −1 || numLabelsOutput == −1){
341              // constructor para la creacion de etiquetas no homogeneas (en
                     funcion del numero de individuos por etiqueta)
342              numLabelsInputs = 11;
343              numLabelsOutput = 11;
344              fuzzyProblem[index]= new FuzzyProblemClass(iSet ,
                     numLabelsInputs , numLabelsOutput , shift , direction ,
                     homogeneousLabel);
345          }
346          else{
347              // constructor original para la creacion de etiquetas
                     homogeneas
348              fuzzyProblem[index]= new FuzzyProblemClass(iSet ,
                     numLabelsInputs , numLabelsOutput , shift , direction ,
                     homogeneousLabel);
349          }
350          // pasar los ejemplos a "mis objetos"
351          E_par[index]= new ExampleSetProcess(fuzzyProblem[index], iSet);
352          String result= E_par[index].calcAdaptExVarLabTFG();
353          if (result.compareTo("") != 0){
354              return false;
355          }
356
357          // calcular las medidas de informacion para agilizar los calculos
358          E_par[index].calcInformationMeasures();
359
360          // crear el objeto para el algoritmo genetico
361          R[index]= new RuleSetClass(alpha);
362
363          //creacion del objeto genetico e inicializarlo
364          GeneticAlgorithmClass GA= new GeneticAlgorithmClass(
                 poblationParam , E_par[index]);
365          // inicializar la poblacion
366          GA.initPopulation(randomNum[index],E_par[index],costMatrix);
367
368
369          // BEGIN − aqui comenzaria el bloque de ejecuciones del algoritmo
                 genetico
370          Util.initStatisticalData(GA.getP() , fuzzyProblem[index]);
371
```

64

```
372              // calcular la nueva regla
373              int ejemplosCubiertos=0, eliminadoReglas=0, newRule=1;
374              Util.numIterGenetic++;
375
376              //// AQUI PARA ANNADIR O NO LA REGLA POR DEFECTO AL COMIENZO ...
377              int addDefaultRule=0;
378              Util.classDefaultRule= GA.setDefaultRule(addDefaultRule,E_par[
                     index],R[index]);
379              if (addDefaultRule == 1){ // Si se ha includo la regla por
                     defecto al principio.
380                ejemplosCubiertos= E_par[index].calcCoveredTFG(R[index],GA.getP
                       (), fuzzyProblem[index]);
381              }
382
383              eliminadoReglas= 1;
384              while (eliminadoReglas == 1){
385                while (newRule == 1 && ejemplosCubiertos < E_par[index].
                       numExamples){
386                  iter[index]++;
387                  newRule= GA.findNewRuleTFG(randomNum[index],0,E_par[index],R[
                         index]); // en la version de homogeneousLabel se ha
                         eliminado la opcion de cuda
388                  Util.numIterGenetic++;
389
390                  ejemplosCubiertos= E_par[index].calcCoveredTFG(R[index],GA.
                         getP(),fuzzyProblem[index]);
391
392
393                }//while (newRule == 1){
394                  eliminadoReglas= R[index].removeRulesForImproveMetricTFG(
                         E_par[index],GA, fuzzyProblem[index]); // probar a quitar
                          reglas y ver si mejora la precision
395                  if (eliminadoReglas == 1){
396                    newRule=1;
397                    ejemplosCubiertos= E_par[index].calcCoveredTFG(R[index], GA
                           .getP(), fuzzyProblem[index]);
398                  }
399              }// while (eliminadoReglas != 1){
400
401
402              if (addDefaultRule == 0){ // No se ha includo la regla por
                     defecto al principio. Se debe incluir al final
403                  R[index].addRule(Util.DefaultRule, Util.DefaultRule.
                         getRealMatrix(Util.classDefaultRule, 4), E_par[index]);
404              }
405
406              return true;
407      }
408
409      public static boolean executePredict(int shift, int direction, int
             index){
410              iter[index]=0;
411
```

```
412            // pasar los ejemplos a "mis objetos"
413            E_par_test[index]= new ExampleSetProcess(fuzzyProblem[index],
                   tSet);
414            String result= E_par_test[index].calcAdaptExVarLabTFG();
415            if (result.compareTo("") != 0){
416                return false;
417            }
418
419            return true;
420     }
421
422     public static String[] Targets(ExampleSetProcess[] _par, int numShifts){
423         int indexRule=0;
424         int claseInference;
425         int varCons= _par[0].getProblemDefinition().consequentIndex();
426         double valueReglaCombinado;
427         int[] indRegla;
428         String[] _Resultado;
429
430         indRegla= new int[numShifts];
431         _Resultado = new String[_par[0].getNumExamples()];
432
433         for (int i=0; i < _par[0].getNumExamples(); i++){
434             for (int d=0; d < numShifts; d++){
435                 indexRule= R[d].inference(_par[d],i);
436                 indRegla[d]= indexRule;
437             }
438             claseInference= R[0].getRules(indexRule).getIntegerMatrix(0,0);
439             valueReglaCombinado= Util.getCentralValue(claseInference,
                   varCons, _par[0]);
440
441             _Resultado[i] = _par[0].getProblemDefinition().
                   getFuzzyLinguisticVariableList(varCons).
                   getFuzzyLinguisticTermList((int)valueReglaCombinado).
                   getName();
442         }
443
444         return _Resultado;
445     }
446 }
```

Código 5.1: Archivo *NSLVOrdJava.java* correspondiente al módulo NSLVOrd

## 5.2.   Clase RuleSystem

La clase *RuleSystem* se encarga de configurar la base de conocimiento y de reglas para ser exportadas a ORCA. En el Código 5.2 se muestra la implementación de esta clase.

```java
1 package NSLVOrdJava;
2
3 import java.util.ArrayList;
4 import jfml.FuzzyInferenceSystem;
5
6 public class RuleSystem {
7     static private FuzzyInferenceSystem _f;
8     static private FuzzyProblemClass _fuzzyProblem;
9     static private RuleSetClass _R;
10
11     public RuleSystem(FuzzyProblemClass fuzzyProblem, RuleSetClass R){
12         _f = new FuzzyInferenceSystem();
13         _fuzzyProblem = fuzzyProblem;
14         _R = R;
15     }
16
17     public String[] Export_KnowledgeBase(){
18         ArrayList export_aux = new ArrayList();
19
20         // FUZZY PROBLEM
21         export_aux.add(String.valueOf(_fuzzyProblem.
                getConsequentIndexOriginal()));
22         export_aux.add(String.valueOf(_fuzzyProblem.getShift()));
23         export_aux.add(String.valueOf(_fuzzyProblem.getDirection()));
24         export_aux.add(String.valueOf(_fuzzyProblem.getHomogeneousLabel()))
                ;
25         export_aux.add(String.valueOf(_fuzzyProblem.
                getFuzzyLinguisticVariableNum()));
26
27         // FUZZY VARIABLE
28         for (FuzzyLinguisticVariableClass auxLinguisticVar : _fuzzyProblem.
                getFuzzyLinguisticVariableList()){
29             export_aux.add(auxLinguisticVar.getName());
30             export_aux.add(String.valueOf(auxLinguisticVar.getUnit()));
31             export_aux.add(String.valueOf(auxLinguisticVar.
                    getNumTermAutomatic()));
32             export_aux.add(String.valueOf(auxLinguisticVar.getVariableType
                    ()));
33             export_aux.add(String.valueOf(auxLinguisticVar.getInfRange()));
34             export_aux.add(String.valueOf(auxLinguisticVar.getSupRange()));
35             export_aux.add(String.valueOf(auxLinguisticVar.getInfRangeIsInf
                    ()));
36             export_aux.add(String.valueOf(auxLinguisticVar.getSupRangeIsInf
```

```
               ())) ;
37             export_aux . add ( String . valueOf ( auxLinguisticVar .
                   getFuzzyLinguisticTermNum () ) ) ;
38
39             // FUZZY TERM
40             for ( FuzzyLinguisticTermClass auxLinguisticTerm  :
                   auxLinguisticVar . getFuzzyLinguisticTermList () ) {
41                 export_aux . add ( auxLinguisticTerm . getName () ) ;
42                 export_aux . add ( String . valueOf ( auxLinguisticTerm . getA () ) ) ;
43                 export_aux . add ( String . valueOf ( auxLinguisticTerm . getB () ) ) ;
44                 export_aux . add ( String . valueOf ( auxLinguisticTerm . getC () ) ) ;
45                 export_aux . add ( String . valueOf ( auxLinguisticTerm . getD () ) ) ;
46                 export_aux . add ( String . valueOf ( auxLinguisticTerm . getAbInf () )
                       ) ;
47                 export_aux . add ( String . valueOf ( auxLinguisticTerm . getCdInf () )
                       ) ;
48             }
49         }
50
51         // To String vector
52         String [] export = new String [ export_aux . size () ] ;
53         for ( int i = 0 ; i < export_aux . size () ; i++) {
54             export [ i ] = ( String ) export_aux . get ( i ) ;
55         }
56
57         return export ;
58     }
59
60     public String [] Export_RuleBase () {
61         ArrayList export_aux = new ArrayList () ;
62
63         // RULE SET
64         export_aux . add ( String . valueOf ( _R . getNumRules () ) ) ;
65         export_aux . add ( String . valueOf ( _R . CCR ) ) ;
66         export_aux . add ( String . valueOf ( _R . SM ) ) ;
67         export_aux . add ( String . valueOf ( _R . TPR ) ) ;
68         export_aux . add ( String . valueOf ( _R . TNR ) ) ;
69         export_aux . add ( String . valueOf ( _R . FPR ) ) ;
70         export_aux . add ( String . valueOf ( _R . Kappa ) ) ;
71         export_aux . add ( String . valueOf ( _R . AUC ) ) ;
72         export_aux . add ( String . valueOf ( _R . MSE ) ) ;
73         export_aux . add ( String . valueOf ( _R . RMSE ) ) ;
74         export_aux . add ( String . valueOf ( _R . RMAE ) ) ;
75         export_aux . add ( String . valueOf ( _R . OMAE ) ) ;
76         export_aux . add ( String . valueOf ( _R . OMAENormalizado ) ) ;
77         export_aux . add ( String . valueOf ( _R . MMAE ) ) ;
78         export_aux . add ( String . valueOf ( _R . mMAE ) ) ;
79         export_aux . add ( String . valueOf ( _R . AMAE ) ) ;
80         export_aux . add ( String . valueOf ( _R . Spearman ) ) ;
81         export_aux . add ( String . valueOf ( _R . Kendall ) ) ;
82         export_aux . add ( String . valueOf ( _R . OC ) ) ;
83         export_aux . add ( String . valueOf ( _R . beta ) ) ;
84         export_aux . add ( String . valueOf ( _R . metric ) ) ;
```

```
85              export_aux.add(String.valueOf(_R.metricMedia));
86              export_aux.add(String.valueOf(_R.Precision));
87              export_aux.add(String.valueOf(_R.alphaMetric));
88              export_aux.add(String.valueOf(_R.confusion.length));
89          for (double[] confusion : _R.confusion) {
90              export_aux.add(String.valueOf(confusion.length));
91              for (double val : confusion) {
92                  export_aux.add(String.valueOf(val));
93              }
94          }
95
96          // RULE
97          for(GenetCodeClass auxGenetCode : _R.getRules()){
98              // Binary elements
99              export_aux.add(String.valueOf(auxGenetCode.getBinaryBlocs()));
100             for(int i = 0; i < auxGenetCode.getBinaryBlocs(); i++){
101                 export_aux.add(String.valueOf(auxGenetCode.
                        getSizeBinaryBlocs(i)));
102                 for(int j = 0; j < auxGenetCode.getSizeBinaryBlocs(i); j++)
                        {
103                     export_aux.add(String.valueOf(auxGenetCode.
                            getBinaryMatrix(i,j)));
104                 }
105             }
106
107             // Integer elements
108             export_aux.add(String.valueOf(auxGenetCode.getIntegerBlocs()));
109             for(int i = 0; i < auxGenetCode.getIntegerBlocs(); i++){
110                 export_aux.add(String.valueOf(auxGenetCode.
                        getSizeIntegerBlocs(i)));
111                 for(int j = 0; j < auxGenetCode.getSizeIntegerBlocs(i); j
                        ++){
112                     export_aux.add(String.valueOf(auxGenetCode.
                            getIntegerMatrix(i,j)));
113                 }
114             }
115             export_aux.add(String.valueOf(auxGenetCode.getIntegerRange().
                    length));
116             for(int i : auxGenetCode.getIntegerRange()){
117                 export_aux.add(String.valueOf(i));
118             }
119
120             // Real elements
121             export_aux.add(String.valueOf(auxGenetCode.getRealBlocs()));
122             for(int i = 0; i < auxGenetCode.getRealBlocs(); i++){
123                 export_aux.add(String.valueOf(auxGenetCode.getSizeRealBlocs
                        (i)));
124                 for(int j = 0; j < auxGenetCode.getSizeRealBlocs(i); j++){
125                     export_aux.add(String.valueOf(auxGenetCode.
                            getRealMatrix(i,j)));
126                 }
127             }
128             export_aux.add(String.valueOf(auxGenetCode.getRealInfRange().
```

```
                        length ) ) ;
129             for (double i : auxGenetCode.getRealInfRange()){
130                 export_aux.add(String.valueOf(i));
131             }
132             export_aux.add(String.valueOf(auxGenetCode.getRealSupRange().
                        length));
133             for (double i : auxGenetCode.getRealSupRange()){
134                 export_aux.add(String.valueOf(i));
135             }
136         }
137
138         // To String vector
139         String[] export = new String[export_aux.size()];
140         for (int i = 0; i < export_aux.size(); i++){
141             export[i] = (String) export_aux.get(i);
142         }
143
144         return export;
145     }
146
147     public String[] Export_Rules(){
148         ArrayList export_aux = new ArrayList();
149
150         // RULES
151         int numRules= _R.getNumRules();
152         export_aux.add(String.valueOf(numRules));
153         for (int i = 0; i < numRules; i++){
154             int classR= _R.getRules(i).getIntegerMatrix(0,0);
155
156             // DATA RULE
157             export_aux.add("R" + i);
158             export_aux.add(String.valueOf(_R.getRules(i).getRealMatrix(2+
                        classR,4)));
159
160             // ANTECEDENT
161             ArrayList ant = Export_Antecedents(i);
162             export_aux.add(String.valueOf(ant.size() + 2));
163             for (Object ant1 : ant) {
164                 export_aux.add((String) ant1);
165             }
166
167             // CONSEQUENT
168             int conseqIndex = _fuzzyProblem.consequentIndex();
169             export_aux.add(_fuzzyProblem.getFuzzyLinguisticVariableList(
                        conseqIndex).getName());
170             export_aux.add(_fuzzyProblem.getFuzzyLinguisticVariableList(
                        conseqIndex).getFuzzyLinguisticTermList(_R.getRules(i).
                        getIntegerMatrix(0,0)).getName());
171         }
172
173         // To String vector
174         String[] export = new String[export_aux.size()];
175         for (int i = 0; i < export_aux.size(); i++){
```

```
176                export[i] = (String) export_aux.get(i);
177          }
178
179          return export;
180     }
181
182     public ArrayList Export_Antecedents(int rule){
183          // Get antecedents
184          ArrayList validTerm = new ArrayList();
185          int numVariables = _fuzzyProblem.getFuzzyLinguisticVariableNum();
186          int start = 0;
187          int tamBloc = _R.getRules(rule).getSizeRealBlocs(0);
188          int conseqIndex = _fuzzyProblem.consequentIndex();
189          int numT = 0;
190          double infMeasureClass = _R.getRules(rule).getRealMatrix(0, tamBloc
                 -1);
191          for (int j=0; j < numVariables-1; j++){
192              ArrayList aux = new ArrayList();
193              int numLabels = _fuzzyProblem.getFuzzyLinguisticVariableList(j)
                     .getFuzzyLinguisticTermNum();
194              double actInfMeasure = _R.getRules(rule).getRealMatrix(0, j);
195              if ((_R.getRules(rule).binaryMatrix0AllToOne(start,numLabels)
                     != 1) &&
196                  (j != conseqIndex && actInfMeasure >= infMeasureClass)){
197                  for (int k=0; k < numLabels; k++){
198                      int valueLabel = _R.getRules(rule).getBinaryMatrix(0,
                         start+k);
199                      if (valueLabel == 1){
200                          aux.add(_fuzzyProblem.
                             getFuzzyLinguisticVariableList(j).
                             getFuzzyLinguisticTermList(k).getName());
201                          aux.add(String.valueOf(_fuzzyProblem.
                             getFuzzyLinguisticVariableList(j).
                             getFuzzyLinguisticTermList(k).getA()));
202                          aux.add(String.valueOf(_fuzzyProblem.
                             getFuzzyLinguisticVariableList(j).
                             getFuzzyLinguisticTermList(k).getB()));
203                          aux.add(String.valueOf(_fuzzyProblem.
                             getFuzzyLinguisticVariableList(j).
                             getFuzzyLinguisticTermList(k).getC()));
204                          aux.add(String.valueOf(_fuzzyProblem.
                             getFuzzyLinguisticVariableList(j).
                             getFuzzyLinguisticTermList(k).getD()));
205                          aux.add(String.valueOf(_fuzzyProblem.
                             getFuzzyLinguisticVariableList(j).
                             getFuzzyLinguisticTermList(k).getAbInf()));
206                          aux.add(String.valueOf(_fuzzyProblem.
                             getFuzzyLinguisticVariableList(j).
                             getFuzzyLinguisticTermList(k).getCdInf()));
207                      }
208                  }
209                  if (!aux.isEmpty()){
210                      aux.add(_fuzzyProblem.getFuzzyLinguisticVariableList(j)
```

71

```
                              . getName ( ) ) ;
211                    validTerm . add ( aux ) ;
212                    numT++;
213                }
214            }
215            start= start+numLabels ;
216        }
217
218        // Save antecedents
219        ArrayList ant = new ArrayList ( ) ;
220        ant . add ( String . valueOf ( numT+1 ) ) ;
221        for ( Object next : validTerm ) {
222            ArrayList aux = ( ArrayList ) next ;
223            ant . add ( ( String ) aux . get ( aux . size ( )−1 ) ) ;
224            int numTerm = ( aux . size ( ) − 1 ) / 7 ;
225            ant . add ( String . valueOf ( numTerm ) ) ;
226            for ( int i = 0 ; i < numTerm ; i++){
227                int num = i ∗ 7 ;
228                ant . add ( ( String ) aux . get ( num ) ) ;
229                ant . add ( ( String ) aux . get ( num + 1 ) ) ;
230                ant . add ( ( String ) aux . get ( num + 2 ) ) ;
231                ant . add ( ( String ) aux . get ( num + 3 ) ) ;
232                ant . add ( ( String ) aux . get ( num + 4 ) ) ;
233                ant . add ( ( String ) aux . get ( num + 5 ) ) ;
234                ant . add ( ( String ) aux . get ( num + 6 ) ) ;
235            }
236        }
237
238        return ant ;
239    }
240 }
```

CÓDIGO 5.2: Archivo *RuleSystem.java* correspondiente al módulo NSLVOrd

# 6 Módulo Rule View

## 6.1. Clase RulesVisual

La clase *RulesVisual* configura las reglas del algoritmo para su visualización. En el Código 6.1 se muestra la implementación de esta clase.

```matlab
1 classdef RulesVisual < handle
2     properties
3         rules = [];
4     end
5
6     methods
7         function visual_rules(obj,name)
8             if isempty(obj.rules)
9                 error('Rules System is empty');
10            end
11
12            addpath(fullfile(fileparts(which('RulesVisual.m')),'VisualRules
                '));
13
14            Visual(name,obj.rules);
15
16            rmpath(fullfile(fileparts(which('RulesVisual.m')),'VisualRules'
                ));
17        end
18
19        function num = detect_number(~,num)
20            if iscell(num)
21                if ~isempty(find(size(num) ~= 1,1))
22                    num = NaN;
23                else
```

```matlab
24                      num = num{1};
25                  end
26              end
27
28              switch class(num)
29                  case 'double'
30                      if ~isempty(find(size(num) ~= 1,1))
31                          num = NaN;
32                      end
33                  case 'char'
34                      num = str2double(num);
35                  otherwise
36                      num = NaN;
37              end
38          end
39
40          function new_rule(obj,name,weight)
41              obj.rules = [obj.rules,struct('Name',name,'Weight',num2str(
                    weight),'Antecedent',[],'Consequent',[])];
42          end
43
44          function add_antecedent(obj,variable,term)
45              if isempty(obj.rules)
46                  error('Rules System is empty');
47              end
48
49              term_aux = [];
50              if ~iscell(term)
51                  term = {term};
52              end
53              for i = 1:size(term,1)
54                  try
55                      for j = 1:7
56                          term_aux = [term_aux;{num2str(term{i,j})}];
57                      end
58                  catch
59                      error('Terms no valids');
60                  end
61              end
62              term = term_aux;
63
64              obj.rules(length(obj.rules)).Antecedent = [obj.rules(length(obj
                    .rules)).Antecedent;{variable term}];
65          end
66
67          function new_consequent(obj,variable,term)
68              if isempty(obj.rules)
69                  error('Rules System is empty');
70              end
71
72              obj.rules(length(obj.rules)).Consequent = {variable term};
73          end
74      end
```

```
75 end
```

CÓDIGO 6.1: Archivo *RulesVisual.m* correspondiente al módulo Rule View

## 6.2. Función Visual

La función *Visual* es la que se encarga de ejecutar la visualización de las reglas. En el Código 6.2 se muestra la implementación de esta clase.

```matlab
1 function Visual(name,rules)
2     % Load VisualRules.jar
3     algorithmPath = fileparts(which('Visual.m'));
4     jarfolder = fullfile(algorithmPath,'VisualRules.jar');
5     javaaddpath(jarfolder);
6
7     % Initialize VisualRules
8     try
9         visual = javaObject('visualrules.VisualRules');
10    catch
11        disp('*************************');
12        disp('See rules is not possible.');
13        disp('*************************');
14
15        % Clear Java
16        clear visual;
17        javarmpath(jarfolder);
18
19        return;
20    end
21
22    % Add rules
23    for i = 1:size(rules,2)
24        namer = rules(i).Name;
25        weight = rules(i).Weight;
26        ant = rules(i).Antecedent;
27        con = rules(i).Consequent;
28
29        % Rule
30        javaMethod('new_rule',visual,namer,weight);
31
32        % Antecedent
33        for j = 1:size(ant,1)
34            namea = ant(j,1);
35            values = ant(j,2);
36            javaMethod('new_antecedent',visual,namea{1},values{1});
37        end
38
39        % Consequent
40        javaMethod('new_consequent',visual,con{1},con{2});
41    end
42
43    % Activate panel
44    javaMethod('SeeRules',visual,name);
```

```
45
46     % Clear  Java
47     clear  visual ;
48     javarmpath ( jarfolder );
49 end
```

CÓDIGO 6.2: Archivo *Visual.java* correspondiente al módulo Rule View

## 6.3.  Clase VisualRules

La clase *VisualRules* se encarga de crear la ventana en la que se visualizan las reglas. En el Código 6.3 se muestra la implementación de esta clase.

```
 1 /*
 2  * To change this license header, choose License Headers in Project
        Properties.
 3  * To change this template file, choose Tools | Templates
 4  * and open the template in the editor.
 5  */
 6 package visualrules;
 7
 8 import java.util.ArrayList;
 9 import java.util.Arrays;
10
11 /**
12  *
13  * @author Federico Garcia-Arevalo Calles
14  */
15 public class VisualRules extends javax.swing.JFrame {
16     /**
17      * Creates new form VisualRules
18      */
19     private final ArrayList rules;
20
21     public VisualRules() {
22         initComponents(0,0);
23         this.rules = new ArrayList();
24         pack();
25     }
26
27     public void new_rule(String name, String weight){
28         if(!this.rules.isEmpty()) CreateRules();
29         this.rules.add(name);
30         this.rules.add(weight);
31         this.rules.add("");
32         this.rules.add("");
33     }
34
35     public void new_antecedent(String name, String[] values){
36         if(this.rules.isEmpty()) return;
37         this.rules.add(name);
38         this.rules.add(String.valueOf(values.length/7));
39         this.rules.addAll(Arrays.asList(values));
40     }
41
42     public void new_consequent(String name, String value){
43         if(this.rules.isEmpty()) return;
```

```
44            this.rules.set(2,name);
45            this.rules.set(3,value);
46        }
47
48        /**
49         * This method is called from within the constructor to initialize the
                  form.
50         * WARNING: Do NOT modify this code. The content of this method is
                  always
51         * regenerated by the Form Editor.
52         */
53        @SuppressWarnings("unchecked")
54        // <editor-fold defaultstate="collapsed" desc="Generated Code">
55        private void initComponents(int w, int h) {
56
57            jPanel1 = new javax.swing.JPanel();
58            jLabel1 = new javax.swing.JLabel();
59            _actual_num_row = new javax.swing.JTextField();
60            _OK = new javax.swing.JButton();
61            _info = new javax.swing.JTextPane();
62            _cont = new javax.swing.JScrollPane();
63            _lista = new javax.swing.JPanel();
64
65            setDefaultCloseOperation(javax.swing.WindowConstants.
                  DISPOSE_ON_CLOSE);
66            setMinimumSize(new java.awt.Dimension(860, 642));
67
68            jLabel1.setText("Num of variable per row:");
69
70            _actual_num_row.setHorizontalAlignment(javax.swing.JTextField.
                  CENTER);
71            _actual_num_row.setMaximumSize(new java.awt.Dimension(45, 23));
72            _actual_num_row.setMinimumSize(new java.awt.Dimension(45, 23));
73            _actual_num_row.setPreferredSize(new java.awt.Dimension(45, 23));
74
75            _OK.setText("OK");
76            _OK.setFocusPainted(false);
77            _OK.setMaximumSize(new java.awt.Dimension(60, 23));
78            _OK.setMinimumSize(new java.awt.Dimension(60, 23));
79            _OK.setPreferredSize(new java.awt.Dimension(60, 23));
80            _OK.addActionListener(new java.awt.event.ActionListener() {
81                public void actionPerformed(java.awt.event.ActionEvent evt) {
82                    _OKActionPerformed(evt);
83                }
84            });
85
86            _info.setText("Zoom an area of graph: click left and mark the area
                  to right.\n" +
87                "See all graph: click left and move to left.");
88            _info.setDisabledTextColor(new java.awt.Color(0, 0, 0));
89            _info.setEnabled(false);
90            _info.setOpaque(false);
91
```

79

```
92          javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout
                (jPanel1);
93          jPanel1.setLayout(jPanel1Layout);
94          jPanel1Layout.setHorizontalGroup(
95              jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.
                    Alignment.LEADING)
96              .addGroup(jPanel1Layout.createSequentialGroup()
97                  .addContainerGap()
98                  .addComponent(jLabel1, javax.swing.GroupLayout.
                        PREFERRED_SIZE, 139, javax.swing.GroupLayout.
                        PREFERRED_SIZE)
99                  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
                        .UNRELATED)
100                 .addComponent(_actual_num_row, javax.swing.GroupLayout.
                        PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
                        javax.swing.GroupLayout.PREFERRED_SIZE)
101                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
                        .RELATED)
102                 .addComponent(_OK, javax.swing.GroupLayout.PREFERRED_SIZE,
                        60, javax.swing.GroupLayout.PREFERRED_SIZE)
103                 .addGap(18, 18, 18)
104                 .addComponent(_info))
105         );
106         jPanel1Layout.setVerticalGroup(
107             jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.
                    Alignment.LEADING)
108             .addGroup(jPanel1Layout.createSequentialGroup()
109                 .addContainerGap()
110                 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
                        GroupLayout.Alignment.BASELINE)
111                     .addComponent(jLabel1, javax.swing.GroupLayout.
                            DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
                            Short.MAX_VALUE)
112                     .addComponent(_actual_num_row, javax.swing.GroupLayout.
                            DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
                            Short.MAX_VALUE)
113                     .addComponent(_OK, javax.swing.GroupLayout.DEFAULT_SIZE
                            , javax.swing.GroupLayout.DEFAULT_SIZE, Short.
                            MAX_VALUE))
114                 .addContainerGap())
115             .addGroup(jPanel1Layout.createSequentialGroup()
116                 .addComponent(_info, javax.swing.GroupLayout.PREFERRED_SIZE
                        , 43, javax.swing.GroupLayout.PREFERRED_SIZE)
117                 .addGap(0, 0, Short.MAX_VALUE))
118         );
119
120         javax.swing.GroupLayout _listaLayout = new javax.swing.GroupLayout(
                _lista);
121         _lista.setLayout(_listaLayout);
122         _listaLayout.setHorizontalGroup(
123             _listaLayout.createParallelGroup(javax.swing.GroupLayout.
                    Alignment.LEADING)
124             .addGap(0, w, Short.MAX_VALUE)
```

```
125            );
126            _listaLayout.setVerticalGroup(
127                _listaLayout.createParallelGroup(javax.swing.GroupLayout.
                       Alignment.LEADING)
128                .addGap(0, h, Short.MAX_VALUE)
129            );
130
131            _cont.setViewportView(_lista);
132
133            javax.swing.GroupLayout layout = new javax.swing.GroupLayout(
                   getContentPane());
134            getContentPane().setLayout(layout);
135            layout.setHorizontalGroup(
136                layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
                       LEADING)
137                .addComponent(_cont)
138                .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
                       javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
139            );
140            layout.setVerticalGroup(
141                layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
                       LEADING)
142                .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.
                       createSequentialGroup()
143                    .addComponent(jPanel1, javax.swing.GroupLayout.
                           PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
                           javax.swing.GroupLayout.PREFERRED_SIZE)
144                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
                           .RELATED)
145                    .addComponent(_cont))
146            );
147
148        }// </editor-fold>
149
150        private void _OKActionPerformed(java.awt.event.ActionEvent evt) {
151            // TODO add your handling code here:
152            String text = _actual_num_row.getText();
153            if(text.length() < 1 || text.length() > 2) return;
154
155            try{
156                int num = Integer.parseInt(text);
157                if(num < 1 || num > 99) return;
158
159                change_num_rules_in_row(num);
160            }catch(NumberFormatException e){
161
162            }
163        }
164
165        public void SeeRules(String name){
166            this.setTitle(name);
167            CreateRules();
168            int num = 5;
```

```
169            change_num_rules_in_row(num);
170            _actual_num_row.setText("" + num);
171            this.setVisible(true);
172        }
173
174    private void CreateRules(){
175            if(this.rules.isEmpty()) return;
176
177            Rule rule = new Rule();
178
179            if(this.rules.size() > 2){
180                // ANTECEDENT
181                rule = getAntecedent(this.rules,rule);
182
183                // CONSEQUENT
184                String consequent_variable = (String) this.rules.get(2);
185                String consequent_term = (String) this.rules.get(3);
186                rule.consequent(consequent_variable,consequent_term);
187            }
188
189            // ADD RULE
190            rule.weight(Float.parseFloat((String) this.rules.get(1)));
191            rule.number((String) this.rules.get(0));
192            _lista.add(rule);
193
194            this.rules.clear();
195    }
196
197    private void change_num_rules_in_row(int num){
198            int h = 6;
199            int w = 0;
200            Rule aux[] = new Rule[_lista.getComponentCount()];
201            for(int i = 0; i < _lista.getComponentCount(); i++){
202                aux[i] = (Rule) _lista.getComponent(i);
203                aux[i].regroup_components(num);
204                aux[i].setSize(aux[i].getPreferredSize());
205                aux[i].setLocation(6,h);
206                h += aux[i].getHeight() + 6;
207                if(aux[i].getWidth() > w) w = aux[i].getWidth();
208            }
209
210            this.remove(jPanel1);
211            this.remove(jLabel1);
212            this.remove(_actual_num_row);
213            this.remove(_OK);
214            this.remove(_info);
215            this.remove(_cont);
216            this.remove(_lista);
217
218            initComponents(w,h);
219
220            for (Rule rule : aux) {
221                _lista.add(rule);
```

```
222            }
223        }
224
225      private static Rule getAntecedent(ArrayList aux, Rule rule) {
226          // Obtener las variables y terminos que van en la regla
227          int act = 4;
228          while(act  < aux.size()){
229              String name = (String) aux.get(act); act++;
230              int numLabels = Integer.parseInt((String) aux.get(act)); act++;
231              int type_variable = 0;
232              ArrayList aux_2 = new ArrayList();
233              for(int k = 0; k < numLabels; k++){
234                  String[] data = new String[7];
235                  data[0] = (String) aux.get(act); act++;
236                  data[1] = (String) aux.get(act); act++;
237                  data[2] = (String) aux.get(act); act++;
238                  data[3] = (String) aux.get(act); act++;
239                  data[4] = (String) aux.get(act); act++;
240                  data[5] = (String) aux.get(act); act++;
241                  data[6] = (String) aux.get(act); act++;
242                  if(data[1].equals(data[2]) && data[1].equals(data[3]) &&
                         data[1].equals(data[4]) && type_variable != 2){
243                      type_variable = 1;
244                  }else{
245                      type_variable = 2;
246                  }
247                  aux_2.add(data);
248              }
249              if(type_variable == 1){
250                  String[] terms = new String[aux_2.size()];
251                  for(int i = 0; i < terms.length; i++){
252                      String[] data = (String[]) aux_2.get(i);
253                      terms[i] = data[0];
254                  }
255                  rule.add_categoric_antecedent(name,terms);
256              }else if(type_variable == 2){
257                  double[][][] series = new double[aux_2.size()][4][2];
258                  for(int i = 0; i < series.length; i++){
259                      String[] data = (String[]) aux_2.get(i);
260                      series[i][0][0] = Double.parseDouble(data[1]);
261                      series[i][0][1] = Double.parseDouble(data[5]);
262                      series[i][1][0] = Double.parseDouble(data[2]);
263                      series[i][1][1] = 1;
264                      series[i][2][0] = Double.parseDouble(data[3]);
265                      series[i][2][1] = 1;
266                      series[i][3][0] = Double.parseDouble(data[4]);
267                      series[i][3][1] = Double.parseDouble(data[6]);
268                  }
269                  rule.add_fuzzy_antecedent(name,series);
270              }
271          }
272          return rule;
273      }
```

83

```
274
275      // Variables declaration − do not modify
276      private javax.swing.JButton _OK;
277      private javax.swing.JTextField _actual_num_row;
278      private javax.swing.JScrollPane _cont;
279      private javax.swing.JTextPane _info;
280      private javax.swing.JPanel _lista;
281      private javax.swing.JLabel jLabel1;
282      private javax.swing.JPanel jPanel1;
283      // End of variables declaration
284 }
```

CÓDIGO 6.3: Archivo *VisualRules.java* correspondiente al módulo Rule View

## 6.4.  Clase Rule

La clase *Rule* sirve para representar una regla de forma individual. En
el Código 6.4 se muestra la implementación de esta clase.

```
 1 /∗
 2  ∗ To change this license header , choose License Headers in Project
        Properties .
 3  ∗ To change this template file , choose Tools | Templates
 4  ∗ and open the template in the editor .
 5  ∗/
 6 package visualrules ;
 7
 8 import java . awt . Component ;
 9
10 /∗∗
11  ∗
12  ∗ @author Federico Garcia−Arevalo Calles
13  ∗/
14 public class Rule extends javax . swing . JPanel {
15
16     /∗∗
17      ∗ Creates new form Rules
18      ∗/
19     public Rule () {
20         initComponents () ;
21         _num_rules = 0;
22         _IF . setSize ( _IF . getPreferredSize () ) ;
23         _num_row = −1;
24         _weight = 0;
25         _consequent = new ConditionCategoric () ;
26         _consequent . setLocation (46 ,0) ;
27         _thenpanel . add ( _consequent ) ;
28     }
29
30     public void weight ( float weight ) {
31         if ( weight < 0) weight = 0;
32         _weight = weight ;
33     }
34
35     public void number ( String name ) {
36         _num . setText ( name + " :  (WEIGHT = " + _weight + " )" ) ;
37     }
38
39     public void regroup_components ( int num ) {
40         if ( _num_rules <= 1 || num == _num_row ) return ;
41         _num_row = num ;
42         if ( num < 1) num = 1;
43         Component a = _ifpanel . getComponent (1) ;
```

```
44          int h,w,line;
45          w = a.getX() + a.getWidth() + 6;
46          java.awt.Dimension tam = new java.awt.Dimension(w,a.getHeight());
47          h = 0;
48          for(int i = 2; i < _num_rules*2; i = i + 2){
49              // Cambio de linea
50              if ((i/2) %num == 0){
51                  w = 30;
52                  h = tam.height + 6;
53              }
54
55              // Recolocacion AND
56              a = _ifpanel.getComponent(i);
57              a.setLocation(w,h + 11);
58              w += a.getWidth() + 6;
59
60              // Recolocacion antecedente
61              a = _ifpanel.getComponent(i+1);
62              a.setLocation(w,h);
63              w += a.getWidth() + 6;
64
65              // Cambiar tamanyo
66              if(tam.height < a.getHeight() + h) tam.height = a.getHeight() +
                      h;
67              if(tam.width < w) tam.width = w;
68          }
69
70      // Aplicar cambios en los paneles
71      update_panel(tam);
72  }
73
74  private void update_panel(java.awt.Dimension tam){
75      // Actualizar el panel con las reglas
76      _ifpanel.setPreferredSize(tam);
77      _ifpanel.setSize(tam);
78
79      this.updateUI();
80  }
81
82  private void add_antecedent(Component antecedent) {
83      Component ult = _ifpanel.getComponent(_ifpanel.getComponents().
              length - 1);
84      int w = ult.getX() + ult.getWidth() + 6; // 6: el espacio entre
              componentes
85      int h = 0;
86      int lon = _ifpanel.getHeight();
87      int aux_lon;
88
89      // Anyadir un JLabel con el texto AND entre reglas
90      if(_num_rules != 0){
91          javax.swing.JLabel andText = new javax.swing.JLabel();
92          andText.setText("AND");
93          andText.setSize(andText.getPreferredSize());
```

```
 94              //if((float) (_num_rules + 1) % (_num_row + 1) == 0){ // Cambio
                    de linea
 95              //     w = 30;
 96              //     h = lon + 6;
 97              //}
 98              andText.setLocation(w,h+11);
 99              w += andText.getWidth() + 6; // 16: la altura del componente
                    JLabel
100                                         // 6: el espacio entre componentes
101              _ifpanel.add(andText);
102          }
103
104          // Anyadir el antecedente
105          antecedent.setSize(antecedent.getPreferredSize());
106          antecedent.setLocation(w,h);
107          w += antecedent.getWidth() + 6;
108          _ifpanel.add(antecedent);
109
110          // Aplicar cambios en los paneles
111          aux_lon = antecedent.getHeight() + antecedent.getY();
112          if(aux_lon > lon) lon = aux_lon;
113          if(_ifpanel.getWidth() > w) w = _ifpanel.getWidth();
114          update_panel(new java.awt.Dimension(w, lon));
115
116          _num_rules++;
117      }
118
119      public void add_categoric_antecedent(String variable, String[] terms) {
120          ConditionCategoric antecedent = new ConditionCategoric();
121
122          antecedent.setVariable(variable);
123
124          for(String term : terms) {
125              antecedent.addLabel(term);
126          }
127
128          add_antecedent(antecedent);
129      }
130
131      public void add_fuzzy_antecedent(String variable, double[][][] series)
             {
132          ConditionFuzzyLogic antecedent = new ConditionFuzzyLogic();
133
134          antecedent.setVariable(variable);
135
136          for(double[][] serie : series) {
137              antecedent.new_serie();
138              for(double[] point : serie) {
139                  antecedent.add(point[0], point[1]);
140              }
141          }
142
143          antecedent.createGraph();
```

87

```
144
145          add_antecedent ( antecedent ) ;
146      }
147
148      public void consequent ( String variable , String term ) {
149          _consequent . setVariable ( variable ) ;
150          _consequent . addLabel ( term ) ;
151          _consequent . setSize ( _consequent . getPreferredSize ( ) ) ;
152          java . awt . Dimension tam = new java . awt . Dimension (46 + _consequent .
                 getWidth ( ) , _consequent . getHeight ( ) ) ;
153          _thenpanel . setPreferredSize ( tam ) ;
154          _thenpanel . setSize ( tam ) ;
155      }
156
157      /**
158       * This method is called from within the constructor to initialize the
               form .
159       * WARNING: Do NOT modify this code . The content of this method is
               always
160       * regenerated by the Form Editor .
161       */
162      @SuppressWarnings ( "unchecked" )
163      // <editor−fold defaultstate="collapsed" desc="Generated Code">//GEN−
              BEGIN: initComponents
164      private void initComponents ( ) {
165
166          _num = new javax . swing . JLabel ( ) ;
167          _ifpanel = new javax . swing . JPanel ( ) ;
168          _IF = new javax . swing . JLabel ( ) ;
169          _thenpanel = new javax . swing . JPanel ( ) ;
170          _THEN = new javax . swing . JLabel ( ) ;
171
172          _num . setText ( "RULE X:  (WEIGHT = Y)" ) ;
173
174          _IF . setText ( "IF" ) ;
175
176          javax . swing . GroupLayout _ifpanelLayout = new javax . swing .
                 GroupLayout ( _ifpanel ) ;
177          _ifpanel . setLayout ( _ifpanelLayout ) ;
178          _ifpanelLayout . setHorizontalGroup (
179              _ifpanelLayout . createParallelGroup ( javax . swing . GroupLayout .
                     Alignment .LEADING )
180              . addGroup ( _ifpanelLayout . createSequentialGroup ( )
181                  . addContainerGap ( )
182                  . addComponent ( _IF )
183                  . addContainerGap (197 , Short .MAX_VALUE ) )
184          ) ;
185          _ifpanelLayout . setVerticalGroup (
186              _ifpanelLayout . createParallelGroup ( javax . swing . GroupLayout .
                     Alignment .LEADING )
187              . addGroup ( _ifpanelLayout . createSequentialGroup ( )
188                  . addContainerGap ( )
189                  . addComponent ( _IF )
```

```
190                    . addContainerGap ( javax . swing . GroupLayout . DEFAULT_SIZE ,
                            Short . MAX_VALUE ) )
191            ) ;
192
193          _THEN . setText ( "THEN" ) ;
194
195          javax . swing . GroupLayout  _thenpanelLayout = new javax . swing .
                  GroupLayout ( _thenpanel ) ;
196          _thenpanel . setLayout ( _thenpanelLayout ) ;
197          _thenpanelLayout . setHorizontalGroup (
198              _thenpanelLayout . createParallelGroup ( javax . swing . GroupLayout .
                      Alignment . LEADING )
199              . addGroup ( _thenpanelLayout . createSequentialGroup ( )
200                  . addContainerGap ( )
201                  . addComponent ( _THEN )
202                  . addContainerGap ( 181 ,  Short . MAX_VALUE ) )
203          ) ;
204          _thenpanelLayout . setVerticalGroup (
205              _thenpanelLayout . createParallelGroup ( javax . swing . GroupLayout .
                      Alignment . LEADING )
206              . addGroup ( _thenpanelLayout . createSequentialGroup ( )
207                  . addContainerGap ( )
208                  . addComponent ( _THEN )
209                  . addGap ( 11 ,  11 ,  11 ) )
210          ) ;
211
212          javax . swing . GroupLayout  layout = new javax . swing . GroupLayout ( this ) ;
213          this . setLayout ( layout ) ;
214          layout . setHorizontalGroup (
215              layout . createParallelGroup ( javax . swing . GroupLayout . Alignment .
                      LEADING )
216              . addGroup ( layout . createSequentialGroup ( )
217                  . addContainerGap ( )
218                  . addGroup ( layout . createParallelGroup ( javax . swing .
                          GroupLayout . Alignment . LEADING )
219                      . addComponent ( _num )
220                      . addGroup ( layout . createSequentialGroup ( )
221                          . addGap ( 10 ,  10 ,  10 )
222                          . addGroup ( layout . createParallelGroup ( javax . swing .
                                  GroupLayout . Alignment . LEADING )
223                              . addComponent ( _ifpanel ,  javax . swing . GroupLayout
                                      . PREFERRED_SIZE ,  javax . swing . GroupLayout .
                                      DEFAULT_SIZE ,  javax . swing . GroupLayout .
                                      PREFERRED_SIZE )
224                              . addComponent ( _thenpanel ,  javax . swing .
                                      GroupLayout . PREFERRED_SIZE ,  javax . swing .
                                      GroupLayout . DEFAULT_SIZE ,  javax . swing .
                                      GroupLayout . PREFERRED_SIZE ) ) ) )
225                  . addContainerGap ( javax . swing . GroupLayout . DEFAULT_SIZE ,
                          Short . MAX_VALUE ) )
226          ) ;
227          layout . setVerticalGroup (
228              layout . createParallelGroup ( javax . swing . GroupLayout . Alignment .
```

```
                            LEADING )
229                 . addGroup ( j a v a x . swing . GroupLayout . Alignment .TRAILING ,  l a y o u t .
                            createSequentialGroup ( )
230                      . addContainerGap ( )
231                      . addComponent ( _num )
232                      . addPreferredGap ( j a v a x . swing . LayoutStyle . ComponentPlacement
                              .RELATED)
233                      . addComponent ( _ i f p a n e l ,  j a v a x . swing . GroupLayout .
                              PREFERRED_SIZE ,  j a v a x . swing . GroupLayout .DEFAULT_SIZE,
                              j a v a x . swing . GroupLayout .PREFERRED_SIZE)
234                      . addPreferredGap ( j a v a x . swing . LayoutStyle . ComponentPlacement
                              .RELATED)
235                      . addComponent ( _thenpanel ,  j a v a x . swing . GroupLayout .
                              PREFERRED_SIZE ,  j a v a x . swing . GroupLayout .DEFAULT_SIZE,
                              j a v a x . swing . GroupLayout .PREFERRED_SIZE)
236                      . addContainerGap ( j a v a x . swing . GroupLayout .DEFAULT_SIZE,
                              Short .MAX_VALUE) )
237            ) ;
238     }// </ editor−fold >//GEN−END: initComponents
239
240     private int  _num_row ;
241     private int  _num_rules ;
242     private float  _weight ;
243     private ConditionCategoric  _consequent ;
244
245     // Variables  declaration − do  not  modify //GEN−BEGIN: variables
246     private j a v a x . swing . JLabel  _IF ;
247     private j a v a x . swing . JLabel  _THEN;
248     private j a v a x . swing . JPanel  _ifpanel ;
249     private j a v a x . swing . JLabel  _num ;
250     private j a v a x . swing . JPanel  _thenpanel ;
251     // End  of  variables  declaration //GEN−END: variables
252 }
```

CÓDIGO 6.4: Archivo *Rule.java* correspondiente al módulo Rule View

# 6.5.  Clase ConditionCategoric

La clase *ConditionCategoric* sirve para representar un antecedente de tipo categórico. En el Código 6.5 se muestra la implementación de esta clase.

```
1 /*
2  * To change this license header, choose License Headers in Project
      Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6 package visualrules;
7
8 import java.awt.Color;
9
10 /**
11  *
12  * @author Federico Garcia-Arevalo Calles
13  */
14 public class ConditionCategoric extends javax.swing.JPanel {
15
16     /**
17      * Creates new form ConditionCategoric
18      */
19     public ConditionCategoric() {
20         initComponents();
21         _num_labels = 0;
22     }
23
24     /**
25      * This method is called from within the constructor to initialize the
            form.
26      * WARNING: Do NOT modify this code. The content of this method is
            always
27      * regenerated by the Form Editor.
28      */
29     @SuppressWarnings("unchecked")
30     // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-
            BEGIN:initComponents
31     private void initComponents() {
32
33         _variable = new javax.swing.JLabel();
34         _is = new javax.swing.JLabel();
35         _label = new javax.swing.JPanel();
36         _parenthesis = new javax.swing.JLabel();
37
38         _variable.setForeground(new java.awt.Color(0, 0, 204));
39         _variable.setText("NameVariable");
```

```
40
41          _is.setText("IS");
42
43          _label.setLayout(new java.awt.FlowLayout(java.awt.FlowLayout.CENTER
                , 0, 0));
44
45          _parenthesis.setText("(");
46          _parenthesis.setVisible(false);
47          _label.add(_parenthesis);
48
49          javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
50          this.setLayout(layout);
51          layout.setHorizontalGroup(
52              layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
                    LEADING)
53              .addGroup(layout.createSequentialGroup()
54                  .addContainerGap()
55                  .addComponent(_variable)
56                  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
                        .UNRELATED)
57                  .addComponent(_is)
58                  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
                        .RELATED)
59                  .addComponent(_label, javax.swing.GroupLayout.
                        PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
                        javax.swing.GroupLayout.PREFERRED_SIZE)
60                  .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
                        Short.MAX_VALUE))
61          );
62          layout.setVerticalGroup(
63              layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
                    LEADING)
64              .addGroup(layout.createSequentialGroup()
65                  .addContainerGap()
66                  .addGroup(layout.createParallelGroup(javax.swing.
                        GroupLayout.Alignment.LEADING)
67                      .addGroup(layout.createParallelGroup(javax.swing.
                            GroupLayout.Alignment.BASELINE)
68                          .addComponent(_variable)
69                          .addComponent(_is))
70                      .addComponent(_label, javax.swing.GroupLayout.
                            PREFERRED_SIZE, javax.swing.GroupLayout.
                            DEFAULT_SIZE, javax.swing.GroupLayout.
                            PREFERRED_SIZE))
71                  .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
                        Short.MAX_VALUE))
72          );
73      }// </editor-fold>//GEN-END:initComponents
74
75      public void setVariable(String variable) {
76          _variable.setText(variable);
77      }
78
```

```java
79      public void addLabel(String label) {
80          javax.swing.JLabel new_label = new javax.swing.JLabel(label);
81          if(_num_labels > 1) {
82              _label.remove(_label.getComponentCount() - 1);
83          }
84          if(_num_labels > 0) {
85              _label.add(new javax.swing.JLabel("  OR  "));
86              _parenthesis.setVisible(true);
87          }
88          new_label.setForeground(Color.BLUE);
89          _label.add(new_label);
90          if(_num_labels > 0) {
91              _label.add(new javax.swing.JLabel(")"));
92          }
93          _num_labels++;
94      }
95
96      int _num_labels;
97
98      // Variables declaration - do not modify//GEN-BEGIN:variables
99      private javax.swing.JLabel _is;
100     private javax.swing.JPanel _label;
101     private javax.swing.JLabel _parenthesis;
102     private javax.swing.JLabel _variable;
103     // End of variables declaration//GEN-END:variables
104 }
```

CÓDIGO 6.5: Archivo *ConditionCategoric.java* correspondiente al módulo Rule View

## 6.6. Clase ConditionFuzzyLogic

La clase *ConditionFuzzyLogic* sirve para representar un antecedente al que se le ha aplicado lógica difusa. En el Código 6.6 se muestra la implementación de esta clase.

```
 1 /*
 2  * To change this license header, choose License Headers in Project
        Properties.
 3  * To change this template file, choose Tools | Templates
 4  * and open the template in the editor.
 5  */
 6 package visualrules;
 7
 8 import java.util.ArrayList;
 9 import org.jfree.chart.ChartFactory;
10 import org.jfree.chart.ChartFrame;
11 import org.jfree.chart.ChartPanel;
12 import org.jfree.chart.JFreeChart;
13 import org.jfree.chart.plot.PlotOrientation;
14 import org.jfree.data.xy.XYSeries;
15 import org.jfree.data.xy.XYSeriesCollection;
16
17 /**
18  *
19  * @author Federico Garcia−Arevalo Calles
20  */
21 public class ConditionFuzzyLogic extends javax.swing.JPanel {
22
23     private static ArrayList _series;
24     private static int _num_series;
25
26     /**
27      * Creates new form ConditionFuzzyLogic
28      */
29     public ConditionFuzzyLogic() {
30         initComponents();
31         _series = new ArrayList();
32         _num_series = 0;
33     }
34
35     /**
36      * This method is called from within the constructor to initialize the
            form.
37      * WARNING: Do NOT modify this code. The content of this method is
            always
38      * regenerated by the Form Editor.
39      */
40     @SuppressWarnings("unchecked")
```
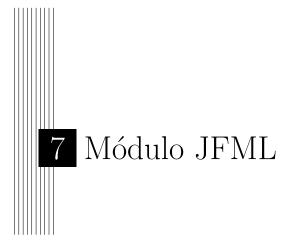
## 6.6. Clase ConditionFuzzyLogic

```
41      // <editor−fold defaultstate="collapsed" desc="Generated Code">//GEN−
            BEGIN: initComponents
42      private void initComponents() {
43
44          _variable = new javax.swing.JLabel();
45          _is = new javax.swing.JLabel();
46          _label = new javax.swing.JPanel();
47
48          _variable.setForeground(new java.awt.Color(0, 0, 204));
49          _variable.setText("NameVariable");
50
51          _is.setText("IS");
52
53          javax.swing.GroupLayout _labelLayout = new javax.swing.GroupLayout(
                _label);
54          _label.setLayout(_labelLayout);
55          _labelLayout.setHorizontalGroup(
56              _labelLayout.createParallelGroup(javax.swing.GroupLayout.
                    Alignment.LEADING)
57              .addGap(0, 393, Short.MAX_VALUE)
58          );
59          _labelLayout.setVerticalGroup(
60              _labelLayout.createParallelGroup(javax.swing.GroupLayout.
                    Alignment.LEADING)
61              .addGap(0, 263, Short.MAX_VALUE)
62          );
63
64          javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
65          this.setLayout(layout);
66          layout.setHorizontalGroup(
67              layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
                    LEADING)
68              .addGroup(layout.createSequentialGroup()
69                  .addContainerGap()
70                  .addComponent(_variable)
71                  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
                        .UNRELATED)
72                  .addComponent(_is)
73                  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
                        .UNRELATED)
74                  .addComponent(_label, javax.swing.GroupLayout.
                        PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
                        javax.swing.GroupLayout.PREFERRED_SIZE)
75                  .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
                        Short.MAX_VALUE))
76          );
77          layout.setVerticalGroup(
78              layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
                    LEADING)
79              .addGroup(layout.createSequentialGroup()
80                  .addContainerGap()
81                  .addGroup(layout.createParallelGroup(javax.swing.
                        GroupLayout.Alignment.LEADING)
```

95

```
82                        . addGroup ( layout . createParallelGroup ( javax . swing .
                              GroupLayout . Alignment . BASELINE)
83                          . addComponent ( _variable )
84                          . addComponent ( _is ) )
85                        . addComponent ( _label , javax . swing . GroupLayout .
                              PREFERRED_SIZE, javax . swing . GroupLayout .
                              DEFAULT_SIZE, javax . swing . GroupLayout .
                              PREFERRED_SIZE) )
86                      . addContainerGap ( javax . swing . GroupLayout . DEFAULT_SIZE,
                          Short .MAX_VALUE) )
87          );
88      }// </editor−fold >//GEN−END: initComponents
89
90      public void new_serie ( ) {
91          _series . add (new XYSeries ("" + _num_series ) );
92          _num_series++;
93      }
94
95      public void add (Number X, Number Y) {
96          (( XYSeries )  _series . get ( _num_series − 1) ) . add (X,Y) ;
97      }
98
99      public void setVariable ( String variable ) {
100          _variable . setText ( variable ) ;
101      }
102
103      public void createGraph ( ) {
104          _label . setSize ( _label . getPreferredSize () ) ;
105
106          XYSeriesCollection  dataset = new XYSeriesCollection () ;
107          for ( Object serie :  _series ){
108              dataset . addSeries (( XYSeries )  serie ) ;
109          }
110
111          JFreeChart chart = ChartFactory . createXYAreaChart ("" ,"" ,"" , dataset ,
112                                  PlotOrientation .VERTICAL, false , false , false )
                                        ;
113
114          ChartFrame frame = new ChartFrame ("" , chart ) ;
115          frame . pack () ;
116          frame . setSize ( _label . getSize () ) ;
117          frame . setResizable ( false ) ;
118
119          ChartPanel panel = frame . getChartPanel () ;
120          panel . setPopupMenu ( null ) ;
121          // panel . setDomainZoomable ( false ) ;
122          // panel . setRangeZoomable ( false ) ;
123
124          _label . add ( panel ) ;
125      }
126
127      // Variables declaration − do not modify //GEN−BEGIN: variables
128      private javax . swing . JLabel  _is ;
```

```
129        private javax.swing.JPanel _label;
130        private javax.swing.JLabel _variable;
131        // End of variables declaration//GEN-END:variables
132 }
```

CÓDIGO 6.6: Archivo *ConditionFuzzyLogic.java* correspondiente al módulo
Rule View

# 7 Módulo JFML

## 7.1. Clase RulesExport

La clase *RulesExport* configura las reglas del algoritmo para su exportación.
En el Código 7.1 se muestra la implementación de esta clase.

```matlab
1  classdef RulesExport < handle
2      properties
3          rules = [];
4          knowledge_base = [];
5      end
6
7      methods
8          function export_rules(obj,dir,name)
9              if isempty(obj.rules) || isempty(obj.knowledge_base)
10                 error('Rules or Knowledge base is empty')
11             end
12
13             addpath(fullfile(fileparts(which('RulesExport.m')),'JFML'));
14
15             JFML(dir,name,obj.knowledge_base,obj.rules);
16
17             rmpath(fullfile(fileparts(which('RulesExport.m')),'JFML'));
18         end
19
20         function new_variable(obj,name,domain_left,domain_right)
21             domain_left = obj.detect_number(domain_left);
22             if isnan(domain_left)
23                 error('Domain left is not detected like a number.')
24             end
25
```

```matlab
26              domain_right = obj.detect_number(domain_right);
27              if isnan(domain_right)
28                  error('Domain right is not detected like a number.')
29              end
30
31              obj.knowledge_base = [obj.knowledge_base, struct('Name',name,'
                    DomainLeft',domain_left,'DomainRight',domain_right,'Terms'
                    ,[])];
32          end
33
34          function add_terms(obj,name,p1,p2,p3,p4)
35              if isempty(obj.knowledge_base)
36                  error('Knowledge base is empty');
37              end
38
39              p1 = obj.detect_number(p1);
40              p2 = obj.detect_number(p2);
41              p3 = obj.detect_number(p3);
42              p4 = obj.detect_number(p4);
43              if isnan(p1) || isnan(p2) || isnan(p3) || isnan(p4)
44                  error('A value is not detected like a number.')
45              end
46
47              obj.knowledge_base(length(obj.knowledge_base)).Terms = [obj.
                    knowledge_base(length(obj.knowledge_base)).Terms;{name p1
                    p2 p3 p4}];
48          end
49
50          function new_rule(obj,name,weight)
51              weight = obj.detect_number(weight);
52              if isnan(weight)
53                  error('Weight is not detected like a number.')
54              end
55
56              obj.rules = [obj.rules,struct('Name',name,'Weight',weight,'
                    Antecedent',[],'Consequent',[])];
57          end
58
59          function add_antecedent(obj,variable,term)
60              if isempty(obj.rules)
61                  error('Rules base is empty');
62              end
63
64              obj.rules(length(obj.rules)).Antecedent = [obj.rules(length(obj
                    .rules)).Antecedent;{variable term}];
65          end
66
67          function new_consequent(obj,variable,term)
68              if isempty(obj.rules)
69                  error('Rules base is empty');
70              end
71
72              obj.rules(length(obj.rules)).Consequent = {variable term};
```

100

```
73          end
74      end
75
76      methods( Access = private , Static )
77          function num = detect_number (num)
78              if iscell (num)
79                  if ~isempty ( find ( size (num)  ~= 1 ,1 ) )
80                      num = NaN;
81                  else
82                      num = num{1};
83                  end
84              end
85
86              switch class (num)
87                  case 'double'
88                      if ~isempty ( find ( size (num)  ~= 1 ,1 ) )
89                          num = NaN;
90                      end
91                  case 'char'
92                      num = str2double (num) ;
93                  otherwise
94                      num = NaN;
95              end
96          end
97      end
98 end
```

CÓDIGO 7.1: Archivo *RulesExport.m* correspondiente al módulo JFML

## 7.2.   Función JFML

La función *JFML* es la que se encarga de ejecutar la exportación de las reglasn. En el Código 7.2 se muestra la implementación de esta clase.

```matlab
1 function JFML(dir,name,knowledge_base,rules)
2     % Load JFML.jar
3     algorithmPath = fileparts(which('JFML.m'));
4     jarfolder = fullfile(algorithmPath,'JFML.jar');
5     javaaddpath(jarfolder);
6
7     % Initialize FuzzyInferenceSystem
8     f = javaObject('jfml.FuzzyInferenceSystem');
9
10    %KNOWLEDGE BASE
11    KnowledgeBase(f,knowledge_base);
12
13    %RULE BASE
14    RuleBase(f,rules);
15
16    %WRITTING INTO AN XML FILE
17    WriteFile(f,dir,name);
18
19    % Clear java
20    clear f;
21    javarmpath(jarfolder);
22 end
23
24 function KnowledgeBase(f,knowledge_base)
25    % Initialize KnowledgeBaseType
26    kb = javaObject('jfml.knowledgebase.KnowledgeBaseType');
27    javaMethod('setKnowledgeBase',f,kb);
28
29    %FUZZY VARIABLE
30    for i = 1:length(knowledge_base)-1
31        s = javaObject('jfml.knowledgebase.variable.FuzzyVariableType',
                knowledge_base(i).Name,knowledge_base(i).DomainLeft,
                knowledge_base(i).DomainRight);
32
33        %FUZZY TERM
34        for j = 1:size(knowledge_base(i).Terms,1)
35            term = knowledge_base(i).Terms(j,:);
36            st = javaObject('jfml.term.FuzzyTermType',term{1},7,[term{2},
                    term{3},term{4},term{5}]);
37
38            javaMethod('addFuzzyTerm',s,st);
39        end
40
41        javaMethod('addVariable',kb,s);
```

```matlab
42      end
43
44      %OUTPUT CLASS
45      s = javaObject('jfml.knowledgebase.variable.FuzzyVariableType',
            knowledge_base(length(knowledge_base)).Name,knowledge_base(length(
            knowledge_base)).DomainLeft,knowledge_base(length(knowledge_base)).
            DomainRight);
46      javaMethod('setType',s,'output');
47
48      % FUZZY TERM OUTPUT CLASS
49      for j = 1:size(knowledge_base(length(knowledge_base)).Terms,1)
50          term = knowledge_base(length(knowledge_base)).Terms(j,:);
51
52          st = javaObject('jfml.term.FuzzyTermType',term{1},7,[term{2},term
                {3},term{4},term{5}]);
53
54          javaMethod('addFuzzyTerm',s,st);
55      end
56
57      javaMethod('addVariable',kb,s);
58 end
59
60 function RuleBase(f,rules)
61      % Initialize MamdaniRuleBaseType
62      rb = javaObject('jfml.rulebase.MamdaniRuleBaseType','');
63      kb = javaMethod('getKnowledgeBase',f);
64
65      % RULES
66      for i = 1:length(rules)
67          %ANTECEDENT
68          ant = javaObject('jfml.rule.AntecedentType');
69          for j = 1:size(rules(i).Antecedent,1)
70              ant_aux = rules(i).Antecedent(j,:);
71              a = javaMethod('getVariable',kb,ant_aux{1});
72              b = javaMethod('getTerm',a,ant_aux{2});
73              javaMethod('addClause',ant,javaObject('jfml.rule.ClauseType',a,
                    b));
74          end
75
76          % CONSEQUENT
77          con = javaObject('jfml.rule.ConsequentType');
78          a = javaMethod('getVariable',kb,rules(i).Consequent{1});
79          b = javaMethod('getTerm',a,rules(i).Consequent{2});
80          javaMethod('addThenClause',con,a,b);
81
82          %ADD RULE
83          r = javaObject('jfml.rule.FuzzyRuleType',rules(i).Name,'and','MIN',
                javaObject('java.lang.Float',rules(i).Weight));
84          javaMethod('setAntecedent',r,ant);
85          javaMethod('setConsequent',r,con);
86
87          javaMethod('addRule',rb,r);
88      end
```

```matlab
89
90       javaMethod('addRuleBase',f,rb);
91 end
92
93 function  WriteFile(f,dir,name)
94       % Configure directory
95       if ~exist(dir,'dir')
96           mkdir(dir);
97       end
98
99       % Export JFML
100      try
101          disp('Export JFML...');
102          a = javaObject('jfml.JFML');
103          file = javaObject('java.io.File',[dir '/' name '_JFML.xml']);
104          javaMethod('writeFSTtoXML',a,f,file);
105      catch
106          disp('JFML can not export');
107      end
108
109      % Export PMML
110      try
111          disp('Export PMML...');
112          b = javaObject('jfml.compatibility.ExportPMML');
113          file = [dir '/' name '_PMML.xml'];
114          javaMethod('exportFuzzySystem',b,f,file);
115      catch
116          disp('PMML can not export');
117      end
118 end
```

Código 7.2: Archivo *JFML.m* correspondiente al módulo JFML