	Document:	IMAS practical work
	Course:	ETSE-URV, 2019-20

Release	Date	Description
1.0	09-SEP-2019	First release of the document

Table of contents

General overview

The main goal of the practical work of this course is to create an **agent-based decision support system (A-DSS)**. In this system, there are three main entities (agents): one user, one manager and a set of classifiers.

The system will process data about one particular domain. In this sense, all data will be picked up from a well-known repository as UCI Machine Learning[1].

For instance, we consider the dataset *Fertility*[2]. It consists in 100 rows with 10 numerical attributes defining each example. All examples were (manually) classified as normal (N) or altered (O). Taking into account these data, suppose that we have 3 classifiers that handle with the decision tree algorithm (J48). Imagine that we use 95 rows to train the agents, and the rest to predict the output. In this scenario, all classifiers will work with the same precision and the behaviour will be the same most of the times with the 5 instances used to predict (figure 1a). Now, we will change the settings and suppose that the first classifier uses 35 training rows, the second uses 48 training rows and the third uses 67 training rows. In this scenario, the prediction can differ from one agent to another, and we will require a consensus among all the answers received from all classifiers (figure 1b).

Instance to classify	Data	Classified as	Classification by Agent1	Precision Agent 1	Classification by Agent2	Precision Agent 2	Classification by Agent3	Precision Agent 3	Final Decision
1	-1,0.78,1,1,0,1,0.6,-1,0.38	N	N	100% 95/95	N	100% 95/95	N	100% 95/95	N
2	-1,0.78,1,0,1,0,1,-1,0.25	N	N	100% 95/95	N	100% 95/95	N	100% 95/95	N
3	-1,0.56,1,0,1,0,1,-1,0.63	N	N	100% 95/95	N	100% 95/95	N	100% 95/95	N
4	-1,0.67,0,0,1,0,0.6,0,0.5	O	N	100% 95/95	N	100% 95/95	N	100% 95/95	N
5	-1,0.69,1,0,0,0,1,-1,0.31	N	N	100% 95/95	N	100% 95/95	N	100% 95/95	N

(a) All classifiers with the same features

Instance to classify	Data	Classified as	Classification by Agent1	Precision Agent 1	Classification by Agent2	Precision Agent 2	Classification by Agent3	Precision Agent 3	Final Decision
1	-1,0.78,1,1,0,1,0.6,-1,0.38	N	N	36.84% 35/95	N	50.52% 48/95	N	70.52% 67/95	N
2	-1,0.78,1,0,1,0,1,-1,0.25	N	O	36.84% 35/95	N	50.52% 48/95	N	70.52% 67/95	N
3	-1,0.56,1,0,1,0,1,-1,0.63	N	N	36.84% 35/95	N	50.52% 48/95	N	70.52% 67/95	N
4	-1,0.67,0,0,1,0,0.6,0,0.5	O	O	36.84% 35/95	O	50.52% 48/95	N	70.52% 67/95	O
5	-1,0.69,1,0,0,0,1,-1,0.31	N	N	36.84% 35/95	N	50.52% 48/95	N	70.52% 67/95	N

(b) All classifiers with different features

Figure 1: Classification scenarios

Agents

As we previously introduced, the system include three different agents named *user* agent, *manager* agent and *classifier* agent (see figure 2).

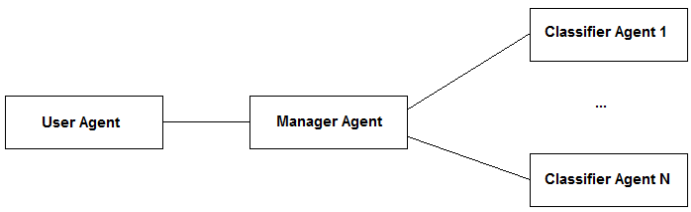

Figure 2: Agents of the system

[1] URL: https://archive.ics.uci.edu/ml/index.php (last access: 29-SEP-2019)
[2] URL: https://archive.ics.uci.edu/ml/datasets/Fertility (last access: 29-SEP-2019)

The user agent is able to handle the two main uses cases of the system, starting with the *training* phase and executing the *prediction* phase.

Once the manager agent has been created, it waits the requests from its user agent and acts accordingly. When the user agent starts the training phase, the manager agent creates the required classifiers and it sends to them the data to start the activities. Later, when the set of classifiers are ready, the user agent can start the prediction phase in order to know in which instance a set of instances will be classified.

Machine learning methods: Weka

Weka[3] is a collection of *open source machine learning algorithms* for data mining tasks. It contains tools for data preparation, classification, regression, clustering, association rules mining, and visualization. It could be used as standalone application, or by calling it through its API functions. You can learn/test the features of the machine learning algorithms with the standalone mode, but the A-DSS will use the java-based API calls. All required JAR files are provided with this ZIP package on the *lib* folder.

First of all, Weka uses a particular format of data called ARFF, which includes two main parts, the first one that defines the set of attributes of the dataset, and the second one, lists the set of data rows (see figure 3). All ARFF files required for the practical course are provided with this ZIP package on the *data* folder.

Weka implements a set of classifier methods, such as decision trees (J48), nearest neighbour (IBk), neural networks (MLP), as well as, it permits to create new ones. In this practical work, we will only work with one machine learning method.

```
% Fertility Data Set
   %
   % David Gil,
   % dgil '@' dtic.ua.es,
   % Lucentia Research Group, Department of Computer Technology, University of Alicante
   %
   % Jose Luis Girela,
   % girela '@' ua.es,
   % Department of Biotechnology, University of Alicante
   %
   @RELATION fertility

   @ATTRIBUTE Season  NUMERIC
   @ATTRIBUTE Age  NUMERIC
   @ATTRIBUTE disease  NUMERIC
   @ATTRIBUTE Accident  NUMERIC
   @ATTRIBUTE Intervention  NUMERIC
   @ATTRIBUTE Highfever  NUMERIC
   @ATTRIBUTE alcohol  NUMERIC
   @ATTRIBUTE smoking  NUMERIC
   @ATTRIBUTE sitting  NUMERIC
   @ATTRIBUTE output {N,O}

@DATA
-0.33,0.75,1,1,1,1,0.8,1,0.25,N
-1,0.67,1,0,0,0,1,-1,0.5,N
-0.33,0.56,1,0,0,0,1,-1,0.63,N
…
```

Figure 3: Sample of an ARFF file

[3] URL: https://www.cs.waikato.ac.nz/~ml/weka/index.html (last access 29-SEP-2019). The current stable release of this software is v3.8, but you can use older releases like v3.6.

System architecture

In the following, all steps of the ADSS will be explained in detail.

- The first step is to create the two first (and required) agents, one user agent and one manager agent.
- The user agent reads a configuration file that specifies the main parameters of the simulation (see appendix 1). Meanwhile, the manager agent is waiting for the first message from its user agent.
- The user agent permits to specify the action to perform from a real user (human): *training* (T) or executing the *predictions* (P).
- The human, through his/her user interface, orders the action T. The user agent begins the transaction with the manager agent attaching the details of the simulation.
- The manager agent receives the T-order with the particular settings of the simulation. First of all, it creates the required classifier agents and then loads the file with the data (ARFF). It stores a set of instances to be classified (the number depends on the configuration file). And finally, it sends the training data to each agent with the algorithm to use.
- The classifiers after being created are able to receive an array of data and use it to train a required machine learning algorithm. When they have finished the training phase, they have to send an acknowledge to their manager.
- The manager agent waits all acknowledgements from all classifiers, and when these are received, it should send a final acknowledgement to its user agent. At this point, the human user (through his/her agent) is able to start the second phase, the prediction.

- Now, the human, through his/her user interface, orders the action P. The user agent begins the transaction with the manager agent.
- The manager receives the P-order from its user agent. Now, it starts a coordination mechanism between classifiers in order to classify the set of stored instances.
- The coordination mechanism between the manager and classifiers ends up with the composition of the final decision for each instance to be classified (aggregation). At the end, the manager returns to the user the set of instances and the set of final taken decisions.

- At this point, if the user demands another T-order, the process of training will begin again with new settings to train and predict the data. However, the creation of classifiers is not required because the agents have been created before.

Requirements

The A-DSS is open to include different design decisions. In the following, a list of minimum requirements to accept the practical work is listed:

- The system should be able to interact with a human user (through keyword) in order to specify which action wants to run, training (T) or prediction (P).
- The A-DSS has to be configured through a configuration file. Classifier agents should be dynamically created as required.
- It is required to analyse the performance of the A-DSS with the two configurations (basic and complete) provided in appendix 1.
- The A-DSS should implement at least one machine learning algorithm, such as J48.
- The A-DSS should provide a coordination mechanism between the manager and the classifiers in order to compose the final decision in the prediction stage.

Useful links and recommended readings

Information about Weka:

- In this link, you can find an introduction to all different concepts introduced in Weka: https://www.cs.auckland.ac.nz/courses/compsci367s1c/tutorials/IntroductionToWeka.pdf
- Documentation about how to use/include Weka in your Java code, https://waikato.github.io/weka-wiki/use_weka_in_your_java_code/ .

The configuration files could be written using an XML-based notation or using java-like properties:

- In this link, you can find information about how to handle properties' files, https://www.mkyong.com/java/java-properties-file-examples/.
- There are some alternatives to handle XML files. In previous courses of IMAS, the JAXB was used (https://docs.oracle.com/javaee/7/api/javax/xml/bind/JAXBContext.html).

As you work in a team, it is useful to use well-known and widely-used tools such as:

- As IDE, Eclipse (https://www.eclipse.org/) is a good alternative. Any version since Oxygen would be valid.
- JDK 1.6 or newer (https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html).
- It is recommended the use of Maven (https://maven.apache.org/) to create the workspace. Weka (https://mvnrepository.com/artifact/nz.ac.waikato.cms.weka/weka-stable/3.8.0) and JADE (https://jade.tilab.com/developers/maven/) are supported.
- It is highly recommended to use a logging library to trace all the entities of the A-DSS. In the following, two alternatives are listed:
    - The Apache Log4j (or its new release log4j 2) (https://logging.apache.org/log4j/2.x/).
    - The Java Logger (https://examples.javacodegeeks.com/core-java/util/logging/java-util-logging-example/).

It is mandatory to document all changes on the practical work using a Git server. You can use GitHub (https://github.com/github), GitLab (https://about.gitlab.com/) or BitBucket (https://bitbucket.org/).

You can find the skeletons of past practical works at https://gitlab.com/disern/imas-skeleton-18-19 or https://gitlab.com/FadiHassan/imasproject. It is recommended to see these examples to analyse how an agent creates other agents, to see how agents handle with communication behaviours, to read XML-based configuration files, to pass information between agents using serialized objects, as well as to see a proposed organization of the code in packages.

Appendix 1: configuration file

The configuration contains all the parameters required in each simulation. The name of this file should be *imas.settings* and it could be written using an XML-notation or using a properties-based notation. Both alternatives are accepted.

Basic simulation: 2 classifiers using J48, 5 instances to classify, 80 training rows per classifier, and working on the fertility dataset

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<SimulationSettings>
   <title>IMAS_basic_simulation</title>
   <algorithm>J48</algorithm>
   <classifiers>2</classifiers>
   <trainingSettings>80,80</trainingSettings>
   <classifyInstances>5</classifyInstances>
   <file>fertility_Diagnosis_train.arff</file>
 </SimulationSettings>
```

```
urv.imas.title=IMAS_basic_simulation
urv.imas.algorithm=J48
urv.imas.classifiers=2
urv.imas.trainingSettings=80,80
urv.imas.classifyInstances=5
urv.imas.file=fertility_Diagnosis_train.arff
```

Complete simulation: 5 classifiers using J48, 10 instances to classify, different training rows per classifier, and working on the segment-test dataset

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<SimulationSettings>
   <title>IMAS_complete_simulation</title>
   <algorithm>J48</algorithm>
   <classifiers>5</classifiers>
   <trainingSettings>100,200,300,400,500</trainingSettings>
   <classifyInstances>10</classifyInstances>
   <file>segment-test.arff</file>
</SimulationSettings>
```

```
urv.imas.title= IMAS_complete_simulation
urv.imas.algorithm=J48
urv.imas.classifiers=5
urv.imas.trainingSettings=100,200,300,400,500
urv.imas.classifyInstances=10
urv.imas.file=fertility_Diagnosis_train.arff
```