Departamento de Informática
Universidad Carlos III de Madrid

Bachelor in Data Science and Engineering

Machine Learning I
Course 2022-2023

Assignment 1: Classification and Prediction

March 7, 2023

This assignment consists of the application of Machine Learning techniques to carry out prediction and classification tasks in the Pac-Man using Weka. In addition, with the generated models, an agent will be built that automatically controls Pacman.

1 Introduction

In the Pac-Man platform, the objective is to eat all the ghosts in the shortest time possible, maximizing the score obtained. For the development of this practice, the simulator provided will be used that allows you to control Pac-Man automatically and collect information from the environment. This information (number of ghosts, distance to ghosts, distance to the nearest food point, etc.) can be used to apply different Machine Learning techniques and make Pac-Man able to solve a task automatically.

To do this, a set of training examples (or learning experiences) must be generated that will be used as input for the classification and prediction techniques (regression). More precisely, this assignment consists of:

- **Instances collection**: collect a set of instances by playing the game with different Pac-Man agents on different maps.
- **Classification**: generate a classification model that allows to classify which action should Pac-Man take given a game state.
- **Intelligent agent**: employ one of the generated models to build an automatic Pac-Man agent.
- **Prediction**: generate a regression model that allows to predict the score in the next tick given a game state and the action Pac-Man has just taken.

The student is asked to do at least two iterations of the whole process, until the Pac-Man agent model is improved and a definitive candidate is obtained. Please, describe this process.

2 Phase 1: Instances collection

In this first part you will need to slightly modify the *printLineData()* function you created in Tutorial 1 and Tutorial 3. This function should (1) generate .arff files that Weka can load; and (2) store information related to future ticks.

This extraction function will serve as the basis for this assignment and all others, so it is vitally important that it is well programmed as soon as possible. To do this, the following steps must be followed:

1. Modify the *printLineData()* function to generate .arff files that Weka can load. Be very careful with the header definition, the type of each attribute, etc.

2. Each line on the file will be now an instance that Weka can interpret. Each instance should have all the information about the state you consider relevant, and also (as the last attribute), the action Pac-Man has just executed. This will be the class we will try to predict in Phase 2.

3. You also need to modify this function to store information related to future ticks. In Phase 3 we will use the score in the current tick to predict the score in the next one. Hence, **an instance should contain**: the **information about the state** you consider relevant, the **score in the current tick**, the **action** Pac-Man has just taken, and the **score in the next tick**.

4. Play with BustersKeyBoardAgent (the one by default) to generate the training and test data. You should have around 80% training instances and around 20% test instances. Test instances cannot be a copy of the training ones. **You should obtain a collection of at least this 3 files**:

   - **training_keyboard.arff**: training instances generated by manually playing with BustersKeyBoardAgent on 5 maps, combining games where the ghosts are static and move randomly.
   - **test_samemaps_keyboard.arff**: test instances generated using the same agent and maps.
   - **test_othermaps_keyboard.arff**: test instances generated by using the same agent on 5 different maps.

Each training instance will have information of three types:

- **Current state**: Attributes about the current state (turn) of the game (position of the Pac-Man, number of ghosts, distance to the nearest food ball, etc.). They will be the input attributes of the learning algorithms. **It is the student's job to select the attributes that are considered for the classification and prediction tasks that are proposed**.
- **Performed action**: The action taken by the agent after evaluating the current state. It will be the output attribute of the classification algorithm (what we want to classify in phase 2). The action "STOP" is highly recommended to be removed.
- **Information of the next turn**: Attributes that will only be known after the next turn has been played. They will be output attributes of the prediction algorithms (what you want to predict in phase 3).

3 Phase 2: Classification

Once you have collected instances by playing with different agents on different maps, we are going to build a classification model that allows Pac-Man to move towards the ghosts and eat them.

1. The aim of the classification model is that Pac-Man can decide, given a state of the game, the action to perform: North, South, East or West.

2. Data pre-processing: the files generated in Phase 1 have all the considered attributes. You should test how selecting different set of attributes affects each dataset. You should perform an in-depth analysis (similar to the one you carried out in Tutorial 2) to remove, filter, modify or create new attributes from the existing ones. You may also consider to filter some instances you consider not relevant for the learning process.

3. You can not use attributes related to the future (as the score in the next tick) as input for the classifier, since they are not available at the moment of deciding which action to perform. We recommend you to save the file with this attribute deleted to facilitate experimentation.

4. Evaluation: you should try different classifiers, using the datasets generated in Phase 1. You should talk about the quality of each set, and compare them through tables/plots. Consider using the algorithms we have used in the Tutorials, as well as others you find interesting. In order to apply some algorithms, it may be necessary to apply various transformations (normalization, discretization, balancing, etc.), which must be well documented in memory for each algorithm.

5. To evaluate the models, it will be necessary to use two sets of tests different from the training ones. One with the same maps that were used in the training and another set of tests that only has different maps (not trained).

It is necessary to describe and analyze all the results obtained from the experimentation, describing each one verbatim and comparing them in **tables and/or graphs**. For comparisons, the files used for training and testing, and the algorithms can be varied (it will be necessary to make comparisons with at least one classification algorithm that has not been seen in previous tutorials).

4 Phase 3: Building an automatic agent

In this phase, you will connect Pac-Man with Weka, so the generated models tell the agent which action it should execute given the current state of the game. **It will be the student's task to select the model that he/she considers appropriate and the way in which it is going to be used to build this automatic behavior**.

To do so, you should use the *wekaI.py* file that can be downloaded from Aula Global. The following steps indicate how to use this file correctly:

1. Make sure you have a compatible Java compiler and interpreter installed, otherwise it will be necessary to install it:
   sudo apt install default-jdk

2. Open a terminal and install the *java-bridge* and *python-weka-wrapper* packages by executing the following commands:

   pip3 install javabridge
   pip3 install python-weka-wrapper3

3. Import the class *Weka* located in *wekaI.py* inside *bustersAgents.py* by writing the following line at the top of the file:

   from wekaI import Weka

4. Launch the java virtual machine in the *init* method from *BusterAgent* in *bustersAgents.py* by writing the following lines:

   self.weka = Weka()
   self.weka.start_jvm()

5. Call the *predict* method from *Weka* class each time you want to classify/predict using any of the previous generated models. As instance, suppose that our model is a J48 tree that given a state, classifies the action Pac-Man should take. Assuming we have saved this model in the *Weka Explorer* under the name *j48.model*, we would call the *predict* function as follows:

   x = [1.51299,14.4,1.74,'None',74.55,0,7.59,0,0]
   a = self.weka.predict("./j48.model", x, "./training_set.arff"))

   where x is an array with the instance we want to classify (and that will vary depending on the information selected by the student), *training_set.arff* is the training set used to generate the J48 model, and a contains the output action that classifies our J48 model for that instance.

Once you have build the agent, compare the results of this agent with the ones obtained by running the agent you programmed in Tutorial 1 and the keyboard agent.

5 Phase 4: Prediction

This phase aims to explore Weka's prediction algorithms. The experimentation procedure and its documentation in the memory will be equivalent to that of phase 2.

Once the information has been obtained from a series of games played by the different selected agents, a prediction model must be built. To do this, it will be necessary to follow the next steps:

1. The aim of the regression model is that Pac-Man can predict, given a state of the game and the action it is performing, the score in the next tick.

2. Data pre-processing: equivalent to the process described for this task in Phase 2.

3. Evaluation: equivalent to the evaluation performed in Phase 2, but now you have to use different regression algorithms (at least 2) to predict the score in the next tick.

4. Compare the quality of each algorithm in an equivalent way to that performed in phase 2 (two types of test files, etc.).

Please, describe each experiment, represent the results, and correctly analyze them (similar to phase 2).

6 Questions

Answer the following questions:

1. What is the difference between learning these models with instances coming from a human-controlled agent and an automatic one?

2. If you wanted to transform the regression task into classification, what would you have to do? What do you think could be the practical application of predicting the score?

3. What are the advantages of predicting the score over classifying the action? Justify your answer.

4. Do you think that some improvement in the ranking could be achieved by incorporating an attribute that would indicate whether the score at the current time has dropped?

7 Files to Submit

All the lab assignments **must** be done in groups of 2 people. You must submit a .zip file containing the required material through Aula Global before the following deadline: **Tuesday, March 28th at 8:00**. The name of the zip file must contain the last 6 digits of both student's NIA, `i.e., assignment1-123456-234567.zip`

The zip file must contain the following files:

1. A **PDF** document with:

   - Cover page with the names and NIAs of both students.
   - Phase 1: Explanation of the feature extraction function that has been programmed, as well as the mechanism to include future data in the same instance.
   - Phase 2: Explanation of the experimentation as explained in the statement. It must include the justification of the selected algorithms, of the selected attributes and of any treatment on the data that has been carried out. It will conclude with an analysis of the results produced by the chosen algorithms and justification for the choice of the final model.
   - Phase 3: Explanation of which model has been selected and why to build the automatic agent, and a brief description of how it works..
   - Phase 4: Same as phase 2, but with each regression model.
   - The answers to each of the questions that are formulated in the section 6.
   - It should not contain screenshots of code or screenshots with text results of the Weka interface. These results should be shown appropriately in tables whenever possible.
   - Conclusions.
     - Technical conclusions about the task that has been carried out.
     - More general insights such as: what can the obtained model be useful for, if during the practice you have come up with other domains in which machine learning can be applied, etc.
     - Description of the problems encountered when doing this practice.
     - Personal comments. Opinion about the practice. Difficulties encountered, criticisms, etc.

2. The files generated during this tutorial: datasets employed and the generated classification and regression models.

3. The code of your automatic agent.

Please, **be very careful and respect the submission rules**. The clarity of the report, the justification of the answers to the proposed questions, as well as the conclusions provided will be valued. The weight of this practice on the final grade for the course is 1.5 points.