Lab X

The Ant Trail Problem

The Ant Trail Game

- Consider a game having the following rules:
  - An ant moves on a NxN-cell grid
  - N random cells are initially filled with ‘food’ (their value is +1); all other cells are empty and have value = 0
  - It is possible to control the motion of the ant in four directions: up, down, left, right *(choose between the world reference frame – the grid – and the ant’s reference frame – its viewpoint)*
  - Initially the ant can occupy any empty cell, unless its position is specified otherwise. The cell value is set to -1 *(see the next slide)*
  - When the ant moves, it earns a score equal to the value of the cell into which it moves.

The Ant Trail Game

- When the ant occupies a cell, the cell value is decremented by 1
- If the ant exits the grid, the games ends: N+2 points are subtracted from its score. Equivalently, we can consider the cells immediately outside the grid to have a negative value of -N-2, besides being ‘poisonous’ for the ant that dies if it moves there
- The aim of the game is to maximize the score earned by the ant in 2N moves starting from a given cell
- Suppose the ant has a ‘field of view’ corresponding to a square neighborhood with side size = 2m+1 (‘radius’=m) centered in the cell that the ant currently occupies. Therefore, the ant view can be described using the cells in its neighborhood as attributes and their corresponding content as attribute values

The Ant Trail Game

Write a function `total_score=ant_rec(N,m,filename,seed)` which allows one to 'move' the ant interactively, recording in `filename` all views (described by the ant's neighborhood) encountered during the game, along with the decision taken correspondingly, e.g., (m=1, 3x3 neighborhood) *`0,0,1,0,-1,0,0,0,-1, right.`*
The random seed can be used to play different games using the same board initialization.

- *Use the data gathered in 1, 5, and 10 games as training sets to generate a "player program" using 1) a decision tree, 2) a multi-layer perceptron, and 3) a GP program. Set N=10 and m=1,2*
- Compare the results obtained in the 6 (x3 methods) different tests (each test is identified by a [`m`, `number_of_training_games`] pair), playing 5 test games (use always the same grid layout to allow for a fair comparison) moving the ant according to the decisions taken by the program

The Ant Trail Game

This means, given the two training data sets collected as described:

1. Train:
   - A decision tree that leads to one of four decisions (move up, down, right, left)
   - A multilayer perceptron with 9 (3x3 neighborhood) or 25 (5x5 neighborhood) inputs and one nominal output (classification problem, the output are 4 values, one for each direction)
   - A Genetic Programming tree whose terminals are the 9 or 25 values of the cells in the ant's neighborhood and whose output OUT can be interpreted as:

up / ahead	-inf < OUT <= - K	[K is an arbitrary positive value]
down / backwards	-K < OUT <= 0	
right	0 < OUT <= K	
left	K < OUT <= inf	

Use the four arithmetic operations (with protected division) + ERCs as the function set

The Ant Trail Game

This means, given the training data collected as described:

2. Create a set of 5 boards initialized with N food pieces and set a starting position for the ant
3. For each board, for 2xN moves, if the ant is ‘alive’:
    - Set the ant’s score to 0
    - Collect the values of the cells in a 3x3 or 5x5 neighborhood of the ant’s cell
    - Use them as the input to the trained model
    - Move the ant in the direction ‘decided’ by the model
    - Increase the score by the value of the new ant’s cell and decrease the cell score by 1

The Ant Trail Game

Remarks

- If the ant is surrounded by cells having all the same value (in particular, null values) there is no preferential direction to move to: however, the program will always associate the same move to that state. So, in an empty region the ant tends to drift towards one of the sides. When it reaches it, it is surrounded by negative values (behind it and in front of it). If the tree has been built correctly, the ant should be able to change direction. However, exploration is not efficient.
- *Possible solution*: if there is no other information around the ant but the possible negative value of the cell visited in the previous step, generate a random move in any direction but the one from which the ant has come.

The Ant Trail Game

Remarks about the representation

- For each state, there are three other equivalent states (obtained by rotating the configuration and the decision by 90 degrees for 3 times): it is possible to increase the training set size exploiting those symmetries or to ‘normalize’ the representation such that equivalent states have one and only one representation (e.g., suppose the ant is always oriented in the North-South direction for the representation, then change the decision by rotating it by the same quantity as the actual orientation of the ant with respect to the North-South direction).
- Using a reference system oriented in the direction of the ant’s motion (the ant always watches ahead of itself and ‘annotates’ what it sees from its viewpoint) the problem with symmetries is still there but the representation is more natural if one imagines to observe the world as the ant would see it.

The Ant Trail Game

More remarks

- *Information is incomplete* (the ant cannot see what is beyond the neighborhood): if the most relevant information is outside the neighborhood, playing the optimal move (based on full knowledge of the grid) when playing the 'training games' might generate moves that are inconsistent with the ant's *local* view.

- In general the optimal solution includes moves that keep the ant mainly close to the middle of the grid (it is easier to reach any other cell from the center of the grid). Because of this, the data gathered playing optimal games will not favor the development of «wall following» or «wall avoiding» behaviors, causing the generation of behaviors that tend to induce the ant to exit the grid.

The Ant Trail Game

Possible implementation

- Write the data collection `ant_rec` function
- Write a function `show_board` that displays the board status
- Write a function `ant_train` that trains a classifier using the data collected (using sklearn)
- Write a function `ant_move` that plays a move given a board configuration and the model trained by sklearn
- Anything else needed?
- You may consider writing a class named `ant` and/or a class named `board`.