Introduction to Machine Learning
Fall 2014 Learning Agent - Battle Simulator

Objectives

The objective of this assignment is for students to gain practical experience in working with machine-learning methods and intergrading them into software systems using WEKA. The assignment is to be worked on in a group of maximum 3 students. There is both a programming and a report writing part to the assignment.

Description

Your task is to write an intelligent behavior into a computer-controlled agent in a battle-like computer game, specifically the ability of the agent to learn. The game is a somewhat simplified imitation of the battle system used in the (unfortunately now retired) game *Viking of Thule* (http://www.gogogic.com/games/vikingsofthule/) developed by the Icelandic game-development company *Gogogic ehf.*

The game is played on a 5 x 5 grid. Two agents, either human- or computer-controlled, start from fixed positions on the board. The agents can move around on the board, attack each other with various attacking moves, rest, or defend off an attack. The type of moves an agent can do vary based on the agent's physical condition at each time, which is determined by two attributes: *stamina* and *health*. Moving around, attacking, and defending gradually reduces the agent's stamina. If the stamina gets too low it prevents the agent from performing some actions otherwise possible. The agent can, however, decide to rest to replenish its stamina. The health gets reduced if an agent is hit by an opponent attack; if the agent's health points reach zero the game is lost. The game is conducted as follows: Both players simultaneously play a card indicating the action their agent should take. The actions for both players are then carried out, modifying the agents' locations, stamina, and health as prescribed (movement actions are carried out before others). This is repeated until one of the agents is out of health or a preset upper-limit on the number of rounds is reached (in which case the games is scored as a tie).

Programming

The code of the battle game is provided (in Java) and your task is to build-in an intelligent behavior into the agent by extending the *Agent* class. A code for a few simple computer-controlled agents is included. Please study it carefully as it helps you to better understand how to use some of the provided support classes (e.g., *StateAgent* and *StateBattle*). You can add classes as needed, but there should not be any need for modifying the existing classes (with the exception of the main *BattleSim* class where you indicate which agents to us in a battle). You are allowed to change other classes for your testing purposes, e.g. for setting up your experiments in main, or trying out a new card, etc. However, make sure that your agent does not rely on such changes and that it still compiles and runs with the original code base (plus classes you have added).

You have a considerable freedom in your implementation as of which machine-learning techniques to use in building intelligence into the agent. As a part of the assessment of the assignment, your agent will be faced in a battle against other agents. Each match will consists of hundreds of battles and the agent winning more battles wins the match. A match will be conducted as follows: 1) first your assigned opponent will play a number of games against other agents and the games logged (the state of the game and the action performed); 2) the learning method in the *Agent* class is called next with the log from the matches (in WEKA Instances form), giving your agent the opportunity to perform some learning, that is, trying to predict how your opponent reacts in different situations; 3) next the real match begins, consisting of hundreds of games (time permitting), where your agent plays repeatedly against the opponent agent and the result of all battles will be scored. Note that for your testing purposes various parameters, such as number of games to play in training/playing and game step-length, can be changed via command-line arguments. When your program will be assessed you will have relatively little time for choosing the actions (measured in tens of ms.), if you exceed the allocated time a *Rest* move will be played for your agent. You will, however, have more time in the learning call (at least 30 sec.).

Note that the deck of cards to choose actions from can vary from one match to the next (your agent will use the same deck throughout all games of a single match). However, you might be given a different deck in a new duel match against the next opponent. This might include removing or adding cards, some of which you have not encountered before, or changing the attributes of some of the existing ones (e.g., using up more stamina points; you may however assume that the *Rest* and *Defense* cards are available in all decks). This makes it important that your agents can learn.

The available actions in the default deck are:

- *Rest*. The agent stays on the same square without attacking or defending. This move does not cost stamina points; on the contrary the agent's stamina increases by 3 points.

- *MoveUp, MoveDown, MoveLeft, and MoveRight*. These actions move the agent around on the grid one square in the indicated direction. The agent cannot move off the will stay in the same square if it tries so). Actions use up 1 stamina point.

- *LeapLeft and LeapRight*. These actions also move the agent around in the indicated direction, but by two squares. Actions use up 1 stamina point.

- *Defense*. This reduces the effect of the opponent's attack by 2 hit points and costs 2 stamina points.

- *Attack*. Attack the opponent. If the opponent is in range his health will be reduced by 1 point (unless they defend). There are three different attack moves each with a different range: AttackCardinal attacks the four a horizontally and vertically adjacent squares (Up,Down,Left,Right), AttackDiagonal attacks the four diagonal squares, and AttackLong attacks only horizontally but two squares in each direction. All the attack moves in addition attack the square the agent is on (note that both agents can occupy the same square). Attacking actions use up 2 stamina points and have a hit value of 1.

Experiments and Report

You must test your code thoroughly and then run tests where you report on how well your agent performs against the benchmark agents. Both measure how well it predicts the opponents' actions and how well it scores in a match against them. Include table(s) and graphs as needed. In addition to using only the simple agent strategies that are included, you might want to additionally write a somewhat more sophisticated fixed agent and see how your agent does against it (if you do so, please describe the logic of that agent briefly in the report, e.g., is it aggressive or defensive). Also, you might want to try removing or adding a new (powerful) card that the agent has not seen before and see how well it fares in figuring out how to use them (you can extend the card Class for adding cards). Experiment with more than one type of a classifier, possibly resulting in more than one new agent. However, please nominate only one of your agents (the best) to participate in the competition.

Write a brief report (approx. 3-5 pages) describing the inner workings of your agent as well as summarizing your experimental findings. Hand in both the report and all your code.