Homework Module:

Evolving Neural Networks for a Flatland Agent

Purpose: Learn to implement an evolving neural network to use as a controller for an agent that moves about in a simple, virtual world.

1 Assignment

You will use your evolutionary algorithm (EA) to evolve the weights for a small artificial neural network (ANN) whose fitness will depend upon the performance of an agent controlled by the ANN. This agent will do a *minimally cognitive* task similar to those used in the field of artificial life (ALife) to illustrate the emergence of rudimentary intelligence. Specifically, the agent will move around in a 2-dimensional grid and try to eat as much "food" as possible while avoiding "poison".

2 Flatland: The Testing Environment

The ANNs that you evolve will control an agent that moves about in Flatland (Figure 1), a simple N x N grid environment in which cells contain food, poison or nothing at all. No cell can contain more than one item. The grid is toroidal, meaning that it has no edges nor corners and thus looks like a torus or doughnut. So when the agent moves off the grid to the far right (left), it comes back onto the grid on the far left (right). Similarly, when it moves off the top (bottom) of the grid, it comes back on the bottom (top).

On each timestep, the agent can stay put or move one cell forward, to the its left, or to its right. The agent has a *front end* from which the directions forward, left and right are based. Before moving left or right, the agent is assumed to turn to the left or right, thus changing its heading direction. So on a left (right) move, the agent actually does 2 things: 1) turns left (right) and 2) moves forward. You can incorporate these into a single operation that is triggered by the left (right) motor output from your neural network.

When the agent moves into a cell, it automatically eats whatever the cell contains. The agent's goal is to eat as much food as possible, while avoiding the poison.

The Food-Poison Distribution (FPD) for Flatland is a pair of probabilities (f,p), where f is the probability of filling an empty cell with food, while p is the probability of filling one of the remaining unfilled cells with poison. Then, a **scenario** in Flatland is a grid with food and poison placed in it stochastically, based on a user-chosen FPD. Hence, one FPD can be the basis for many different scenarios.

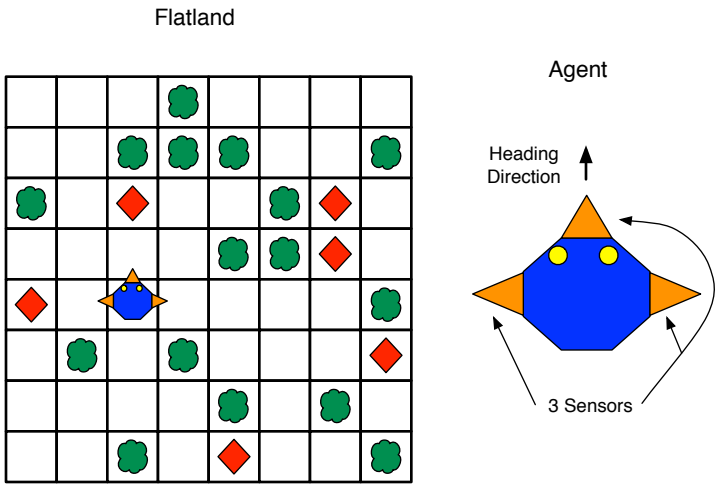

Figure 1: Flatland is an N x N grid, with each cell being open or containing one item of food (green blob) or poison (red diamond). The agent has sensors to detect food and poison in the cell immediately in front, to the left, and to the right of itself. No cell contains more than one item, and the agent must eat any item contained within a cell that it visits. Flatland is a torroidal grid: it wraps around from top to bottom and left to right.

For the purposes of this homework assignment:

- Flatland is an 8 x 8 grid.
- During fitness testing, each agent will be presented with 5 unique scenarios, all based on the same FPD.
- Each agent gets a total of 50 timesteps per scenario.
- In each generation, all agents are exposed to THE SAME group of 5 unique scenarios. However, the scenarios may (or may not) change between generations (as discussed below).
- For each scenario, each agent should begin at the same grid cell (of your choice).
- Use an FPD of (.5 .5) for all scenarios: fill approximately 50% of the cells with food, and then fill approximately 50% of the remaining cells with poison.

3 The Artificial Neural Network

This task does not require a complex network such as a CTRNN. Each neuron can simply integrate all weighted inputs and send the sum through an activation function.

Figure 2 depicts the 2 input layers and output layer of the ANN. There are separate inputs for sensed food and poison, a design choice that normally makes things easier (though you are free to experiment with a more compact design). The output values of the 3 motor neurons should then be used by your code to determine the agent's movement. You may want to include a firing threshold such that if none of the motor neurons exceed it, then the agent does not move on that timestep.

This is only the skeleton of the ANN; you will need to determine some of the important details, including:

1. The number of (hidden) layers, if any, between the sensory inputs and the motor outputs.
2. The number of neurons in any hidden layers.
3. The activation functions used in the neurons of each layer, including thresholds for functions such as sigmoids, hyperbolic tangents, steps and ramps.
4. Whether or not a **bias** neuron is needed as input to some or all of the non-input layers.

You can assume that whenever a layer (A) sends connections to a layer (B) that ALL neurons in A get connected to ALL neurons in B.

The weights on all connections must be determined by your evolutionary algorithm (EA). Other parameters can also be evolved, if you so desire, but that is not a requirement.

Be aware that many of the decisions made during ANN design can be highly interdependent. For example, your choice of activation function could affect not only your choice of threshold, but also topological choices, such as the need for and number of hidden nodes. Also, the legal range of weights that you choose can strongly influence the activation functions and thresholds, so you may need to do some simple calculations to determine whether your choices will a) allow neurons to fire, but b) not fire all the time.

The best advice is to begin with a simple ANN and only complexify as needed. Don't start with a 6-layered ANN!

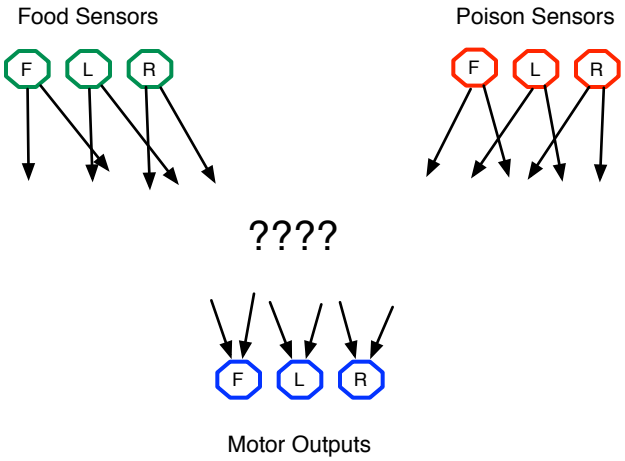

Figure 2: The basic structure of the ANN for a Flatland agent: 6 sensory neurons, 3 each for food or poison in the forward, left or right cell; and 3 motor neurons for the 3 possible directions of movement: forward, left or right.

4 The Evolutionary Algorithm

Your EA need not be very complex. The genotype can be a simple bit vector, while the phenotype can be a list of connection weights. Mutation then involves bit flips, while crossover is the traditional version for bit

vectors. Some textbooks recommend special crossover operators for ANN weight vectors, but these are not necessary for this assignment.

The evolutionary runs that you perform will be of two types:

- **Dynamic** - the set of 5 randomly-generated scenarios changes after each generation. So each new generation of individuals experiences a new set of scenarios.
- **Static** - the same 5 randomly-generated scenarios are used throughout the ENTIRE evolutionary run, i.e., in each and every generation.

All of your evolutionary runs should include some degree of **elitism**: the best individual(s) from generation K are directly copied into the population of generation K+1. Decide for yourself whether more than one such individual needs to be copied.

Other parameters such as population size, number of generations, mutation and crossover rates, adult selection strategies and parent selection mechanisms, etc. will also need to be determined experimentally: try combinations until you find one that gives good results.

4.1 The Fitness Function

This is another detail that you will have to figure out yourself. One of the key issues is the amount of penalty associated with eating poison. There is a wide spectrum, from killing (i.e. giving a fitness of zero to) any agent that ingests any poison, to more *forgiving* approaches that simply detract a small, fixed amount from total fitness for each poison item consumed. Too weak a penalty will not encourage poison-avoidance at all, while too harsh a penalty can produce agents that do nothing (i.e. hardly move at all), for fear of making a fatal mistake.

4.2 Expected Performance

For this task, it can be difficult to evolve an optimal controller, but your EA should be able to find weight vectors that produce reasonably intelligent behavior in an agent: it should be able to eat most of the food and avoid most of the poison in all 5 50-step scenarios (for both static and dynamic runs). Do not expect perfection, but if you are NOT getting a very clear sense that your agent can differentiate between food and poison in MOST **contexts** (i.e. combinations of food and poison inputs to the 3 sensors), then you and your EA have more work to do.

4.3 Visualization

At the end of an evolutionary run, you must be able to visualize the behavior of the best-of-run individual on 5 scenarios. For static runs, these will be the same 5 scenarios that the agent was originally tested on, while for dynamic runs, these will be new, randomly-generated scenarios. This must be a complete 2-d visualization of a grid, not a command-line printout of an array. You must use different colors for food, poison and the agent so that it is very easy to immediately see the structure of a scenario and the behavior of the agent. The delay between visualized timesteps must be adjustable so that we can slow down the run to see exactly what the agent is doing.

You must also visualize the progression of evolution throughout the run by plotting the best-of-generation and average-of-population fitness values (on the y axis) as a function of the generation number (on the x axis). This is the standard **fitness plot** used for EAs.

5 Deliverables

1. A table summarizing the main EA parameters used for the successful runs of your system, along with a brief description of the exploratory process that you used to find them. **2 points**.
2. A complete description of your fitness function in terms of both a mathematical expression and explanatory text. **1 point**
3. A detailed description of the complete ANN design (excluding the actual connection-weight values) that proved most successful in your experiments. This includes the layers of neurons, their activation functions, any thresholds or biases, and the range of acceptable weights that your EA had to work with. You do not need to describe all of the individual weights. Briefly describe the process that you went through to find your design. **4 points**
4. Document the difference in performance of your EA on a static versus a dynamic run. Include fitness plots. Focus on a few of the main differences between the runs and briefly explain why they probably arose. **3 points**
5. A working demonstration of your system evolving a well-functioning ANN during a **static** run followed by a visualization of the best-of-run individual on 5 different scenarios. **5 points**
6. A working demonstration of your system evolving a well-functioning ANN during a **dynamic** run followed by a visualization of the best-of-run individual on 5 different scenarios. **5 points**

5.1 Important Details

- **Your report must not exceed FOUR pages in length. Longer reports can result in a loss of points. Be clear and concise in your writing, but provide each and every piece of information requested above.**
- You may work alone, or in groups of total size 2 or 3, but no larger.
- Each GROUP should write ONE report, but all GROUP MEMBERS must upload that same report to It's Learning under their individual names.
- ALL group members must attend the demonstration session.