

PACMAN and reinforcement learning

Machine Learning
2010 Fall Term

October 14, 2010

1 PACMAN and reinforcement learning

An area where reinforcement learning is useful is computer games. In many games there are game characters that interact with the user and it is measure of the quality of the game the closer of its behaviour to a realistic one or if the skills of the character can adapt to the skills of the human player.

Usually it is difficult to create an algorithm able to make a decision in any possible circumstances a character would face, so to allow it to learn its behaviour by means of interaction with real players could be a good way to solve the problem.

Is in this task where reinforcement learning is useful. A character will be in a set of different states that can be enumerated and that can be described by the current state of the character and the state of the player. The character has a set of actions that can perform and has to be able to decide which is the best. The character can receive a positive or negative reward depending of the outcome of a sequence of decisions. This reward will depend on the game and the goals of the character. If for instance the goal is to defeat the player it will obtain a positive reward when this happens and a negative reward on the opposite case.

2 Modeling PACMAN with reinforcement learning

For this example we are going to model only the behaviour of a *ghost* of the PACMAN game using reinforcement learning. The quality of the behaviour of the character will depend on the level of detail of the model.

We could, for example, use the model obtained using reinforcement learning as a decision procedure for obtaining high level decisions and use a more specific decision mechanism to obtain the fine grain movements or to model the problem in high detail so the model learned gives directly primitive movements for the character to execute.

2.1 A simple model for the ghost

To model the problem using reinforcement learning wue are going to assume that is a decision markov process that has a set of possible states an a set of actions.

To obtain a simple decision model we are going to reduce the problem to a set of states that model in a general way the state of the ghost. For instance, we are going to model the distance from the ghost to PACMAN by discretizing its values to only four: distance of more than 10 positions, more than 5 positions, less that 5 positions and over PACMAN. The state of the problem also includes the state of PACMAN, it can be vulnerable or invulnerable, for the ghost these are different circumstances. This way the set of states for the decision process of the ghost will be the product set of:

$$\{> 10, > 5, \leq 5, \text{PACMAN}\} \times \{\text{Vulnerable, Invulnerable}\}$$

There are eight states for the ghost. Of these state there are two that are goal states $\{\text{PACMAN, Vulnerable}\}$ and $\{\text{PACMAN, Invulnerable}\}$, these state will give us the reward.

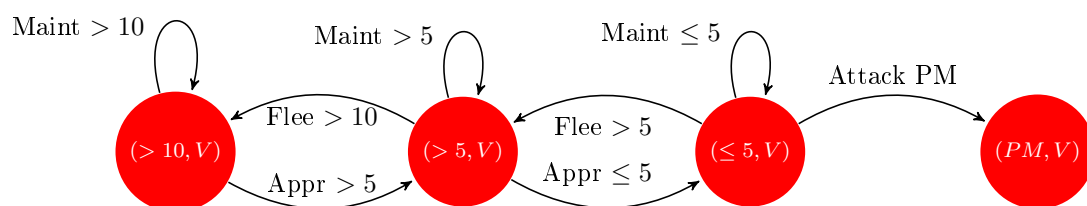
We need also the actions that allow to change from one state to another. We have assumed that we are only to use the model as a high level decision procedure and the fine grained action will be the task of another decision procedure, so the actions should be high level actions that

give us an idea about what to do but not how to do it. For example, we could use the actions flee, approach or maintain distance:

$$\text{Actions} \left\{ \begin{array}{ll} \text{Approach} > 5 & \text{Approach} \leq 5 \\ \text{Attack PACMAN} & \\ \text{Flee} > 10 & \text{Flee} > 5 \\ \text{Maintain} > 10 & \text{Maintain} > 5 \quad \text{Maintain} \leq 5 \end{array} \right.$$

We can decide the connectivity among state, but we can assume that the states when PACMAN is vulnerable are not connected with the states where PACMAN is invulnerable because we can not change this in the state with our actions, only our position. In fact we will have two decision mechanisms that will give us actions under these two circumstances.

This could be the set of states when PACMAN is vulnerable (the same automata will do for the invulnerable case):



We could add more connections among states, so we could for instance decide to flee > 10 from the state $(\leq 5, V)$. It should be evaluated in practice if to have more connections is an advantage or not. The states (PM, V) and (PM, I) are absorbing states, so once we are on them the sequence of movement is ended.

For the rewards it is obvious that we should have a positive reward when we arrive to the state (PM, V) and a negative one when we arrive to state (PM, I) .

Previous to the use of the model we have to decide how we are going to train it. Now it is time to think about the characteristics of the problem. Actually we have two models because the two automata are not connected, so we need a separate set of sequences for each model.

Other circumstance to take in account is how are the training sequences that can be obtained from the game. During a game the state of PACMAN changes from vulnerable to invulnerable without our control. The only states where we receive a reward are when we catch PACMAN when we are caught. This means that of we have a sequence of actions an does not end on a goal state we will no obtain information from that sequence. This can waste a lot of partial sequences that could be used in some way.

To use that sequences we can improve the representation of the problem including mode rewards in the other states. It is clear that when PACMAN is vulnerable the best action is approach it and to flee when it is invulnerable. This means that we can put a increasing reward in the states of the automata that are near to a vulnerable PACMAN and an increasing reward to the states that are far from a invulnerable PACMAN. The value of these rewards should be adjusted experimentally.

This not only will fix the problem of using partial sequences of actions but also will give to the ghost the commonsense behaviour of attacking PACMAN when vulnerable and flee PACMAN when invulnerable.

This model will be a simple model that can allow as to train the behaviour of the ghost. To apply the model we only have to access the trained automata, pick the current state and choose the action that results in the larger reward. How to execute the action will be solved in other way, for example computing the shortest path that moves from the current position to a position that is in the state that we want.

2.2 Detailed model for the ghost

We can also use reinforcement learning to obtain primitive actions for the problem. In this case the state should be modeled by the coordinates of the ghost, the coordinates of PACMAN and the state of PACMAN. We already know that the model will have to parts depending on the state of PACMAN, so we are only going to talk about the model when PACMAN is vulnerable.

In this case the states of our automata will be the product set of the possible coordinates of the ghost and PACMAN. Obviously the number of states is larger than before, but as we want a decision procedure at primitive level we need to use the maximum detail. Depending on the size of the game the model will have some thousand states, but for the computational complexity of the learning algorithm this is feasible.

In the definition of the model the actions will be the possible movements of the ghost, we only have four: up, down, left, right:

The automata can be represented as a matrix that combines the coordinates of the ghost and PACMAN. The absorbing states will be those combinations where the coordinates are the same, those states will end the sequence of movements. The reward can be a function of the distance among coordinates solving also the problem of incomplete sequences and obtaining the desired behaviour for the ghost.

This model after training will answer with an action for any possible state of the ghost. With enough training the paths obtained following its policy would converge to the shortest path to approach or flee PACMAN that a shortest path algorithm would obtain.

The shortcomings would be the size of the model (although it is easy to access because states can be indexed) and the training time, larger than with the previous model.