Economic Learning in Agent World

Class project for CS760 – Machine Learning
12/22/1999

Mark Rich
richm@cs.wisc.edu

Introduction

Baum believes that Holland's original classifier systems fall victim to the "tragedy of the commons" due to their credit assignment mechanisms [1]. Since it is possible for more than one agent to be receiving credit from the world at any given time, agents will pop up to receive credit even if they perform no useful service to the world. Baum solves this problem by assigning ownership to the world to one agent at a time. Access to the world is now a commodity which can be bought and sold by competing agents, making credit and blame assignment trivial. By looking on the meta-learning level, economic agents with very simple bidding rules can work in cooperation to solve abstract complex problems.

Baum proposes a system called Hayek to implement this economic approach to learning. Hayek is a collection of agents competing for possesion of the world in an auction. Agents in Hayek consist of condition, action pairs. If the condition holds in the current world, the agent will make a bid for their action. The agent with the highest bid would then take control of the world, perform their action, and reap the benefits (or woes) or their actions. Baum has implemented this simulation on the Blocks World, a classic planning environment in AI. His results show the success of this modular economic model to find complex and abstract plans with only minimal knowledge about the topology of the world.

This report is a work in progress: there are countless issues left to explore as discussed in the future work section. First, I implemented Baum's Hayek simulation for the Blocks World. I could then study the behavior in action rather than in abstention through a paper. Second, I translated Hayek into our Agent World simulation, a world with rewards and penalties as in Blocks World, but at a much higher dimension of complexity. The competition of agents inside a Player for control seemed like a very intuitive way to evolve winning strategies based purely on reinforcements from the world. This implementation is now working and preliminary results are available, however more questions seem to be raised than answered.

Blocks World implementation

In order to gain familiarity with the mechanics and specifics of Baum's Hayek simulation, I first chose to implement the Blocks World Hayek as mentioned in his paper. The modularization of Hayek merged well with an Object-Oriented Java implementation. Following is a list of class specifications I wrote:

- **Action:**

Actions in the Blocks World consisted of two parts. The *move* specified what movement to commit, either drop or grab a block. The *stack* specified on which stack of blocks this move would take place. Both these data members were stored as Strings. If the stack specified happened to be "*x" (the equivalent of an existentially quantified variable) a random stack was chosen for this action and recorded for in the *recentStack* variable.

- **Condition:**
  Conditions dealt with equality between locations in Blocks World. Two String arrays, *u* and *v*, stored these coordinates to be used in determining the validity of this condition. Another String, *op*, stored the operator, either = or !=, also to be used in the validity check. There were two special cases of locations, "E" and "H", denoting Empty and Hand. Sufficient restrictions were placed on *u* and *v* such that it could recognize these two cases. As in Actions, "*x" and "*y" were used to denote existentially quantified variables in the locations.

- **Agent:**
  Agents consisted of an integer *bid*, a Vector of *conditions*, and an Action *act*. Agents were also able to accumulate *wealth*, also stored as an integer. Agents were granted the ability to buy the world, change the world with their action, receive money, set their bid, and determine their ability to place a bid. Agents were created with a random *act* and *conditions*, and also given the ability to mutate and return another agent with slightly altered conditions.

- **World:**
  The Blocks World consisted of four integer arrays called *stacks* which held the blocks. *Stacks*[0] denoted the goal stack. The goal of the agent was to reproduce the goal stack on *stacks*[1] using blocks provided in *stacks*[1…3]. The World can decide whether actions and conditions are legal or not and whether the goal has been reached or not. To alter the world, agents can use the methods for grabBlock and dropBlock. The reward for performing actions is determined while the action is performed and can be retrieved by an outside agent.

- **BlockHayek:**
  This ultimate class held a world, an agent pool of possible agents, and the current active agent. Agents would determine their ability to bid, and the highest bidder would seize control of the world from the current active agent. This new active agent would perform their action and collect the reward, then wait to be paid by the next bidding agent.

This implementation, found attached as an appendix, provided invaluable insight as to the details of Agent – World – Action interaction. However, I soon ran into some implementation problems to which I could find no ready solution in the paper. I was confused about the randomness with which existential variables are chosen and how to implement this. I was concerned with my goal states in the Blocks World being impossible to solve. I was worried that I would quickly run out of agents and my Hayek

simulation would never recover. I emailed Eric Baum and asked for his advice on these issues and he was able to point me in the correct direction.

I also had some questions about Baum's method of credit checking. It appeared as though whenever an agent fell below 0 wealth, it was removed from the population and the simulation was backtracked to the previous step to see if this would make a difference. This implied to me that we would need to keep track of every possible action ever taken, since there could be cases where agent A is able to survive above 0 based on payment from B, and B survives based on payment from C. However, if C goes bankrupt, then B could go bankrupt in the previous step, and eventually causing A to go bankrupt. Chains such as these could be enormous and the time to recalculate these moves does not seem to be mentioned in the paper. I have yet to resolve this question.

Agent World implementation

Using the knowledge I gained from my Blocks World implementation, I now had a great interface to use for an Agent World translation of the Hayek simulation. Only minor retooling was necessary to take my classes directly into a working Player class. A list of the modifications follows:

- Actions now only have *move*, which was a sensor direction between 1 and 36, or *s.

- Used sensors instead of locations of blocks for conditions. *u* and *v* could now be one of 36 sensors. A, M, V, W and N for Animal, Mineral, Vegetable, Wall and Nothing, became the atoms possible for comparison. Finally, *s replaced *x and *y in the locations.

- Agents, rather than the world, now perform validity check using Sensor readings. The randomization of the selection of *s was ignored to speed the implementation process towards generating results.

- The collection of current agents in memory were split into Novice and Veteran pools to simplify code. Valid veteran agents were found first, and then if a valid novice was found, veteran agents would be forgotten, since by the rules to set an agent's bid, a novice will always defeat a veteran.

- The credit check for agents was ignored. Baum discusses the implications of this, with speculative bubbles and inflation, but again to speed the implementation to get results, this was ignored. It is still an open question how one would "backtrack" in the constantly changing Agent world where time stops for no player.

- The reward scheme in agent world was slightly modified by adding 0.5 to every reward. This allowed the agents to earn money when their action did not cause a negative reward instead of only when their actions did cause a positive reward.

Preliminary results

My hypothesis for this research was that the Hayek simulation would evolve some comprehensible rules for dealing with the circumstances of Agent world. Some rules I hoped to see were simple, such as

```
Bid = 3.201     (*s) = (V) ==> (*s)
```

This rule essentially says if you see a vegetable, go towards it. Other rules could be more specific, like

```
Bid = 0.601     (5) != (W) ^ (10) != (W) ==> (7)
```

Here we have a rule that determines it is safe to proceed in direction 7, since there are no walls detected around it. We can this that this agent has a low bid, therefore allowing it to be superceded by other agents when new circumstances arise, such as an opposing agent being in direction 7. I had some doubts as to Hayek's ability, given the failure of my neural network and nearest neighbor implementations to achieve any degree of success.

Approximately half of the times I ran this simulation, the number of agents in the agent pool quickly descended to 0. This prevented the agent from ever learning, since new agents added in were rarely able to bid, and when they did, there was no one to replace it with a better action if it's action proved harmful to the agent. The Hayek player was deemed brain dead in these circumstances; brain dead simulations were discontinued as soon as they were discovered. Possible causes and patches for this problem can be found in the future work section.

For two of the successful simulations, where a population was able to pass the initial rent hurdle, the results exceeded my expectations. When left alone in a world with only vegetables and minerals, Hayek seems to do very well, averaging about 255 points over 23 games of 5000 steps each. The following set of agents was able to achieve 496 points in the 16th game of the second run.

The best scoring collection of agents found in run 2.

```
Bid = 2.501     (5) != (0) ^ (3) != (A) ==> (13)
Bid = 2.401     (N) != (*s) ==> (21)
Bid = 1.901     (2) = (13) ==> (7)
Bid = 1.901     (V) != (4) ^ (18) != (W) ==> (17)
Bid = 1.501     (M) != (V) ==> (*s)
Bid = 1.301     (6) != (1) ==> (9)
Bid = 1.301     (31) != (*s) ==> (21)
Bid = 1.201     (10) != (19) ==> (*s)
Bid = 1.201     (24) != (V) ==> (6)
Bid = 1.101     (15) != (*s) ==> (33)
Bid = 1.001     (14) != (M) ==> (28)
Bid = 1.001     (N) != (2) ==> (21)
Bid = 0.901     (V) != (4) ==> (17)
Bid = 0.801     (2) = (3) ==> (7)
Bid = 0.701     (0) != (*s) ==> (32)
Bid = 0.601     (35) != (V) ==> (6)
```

```
Bid = 0.401     (N) != (30) ==> (6)
Bid = 0.201     (10) != (A) ==> (*s)
Bid = 0.201     (M) != (V) ^ (W) = (25) ==> (*s)
Bid = 0.101     (N) != (*s) ==> (21)
Bid = 0.10      (14) != (M) ==> (28)
Bid = 0.00      (8) = (5) ==> (11)
```

What do they mean? I have no clue why it works, but it does. My intuitions about the interpretability of Hayek agents seems to be in error. The fifth rule, with bid 1.501, looks to be a universal random mover. Since its condition will always be true, if nothing else is happening in the world to take precedence, this agent will make a random move. Also the fourth rule, with bid 1.901, seems close to a "search for vegetables" rule. If there's no Wall in direction 18, and there are no vegetables in direction 4, this agent proposes to move in direction 17.

When stuck with other players, Alice and Bob from Homework 4, Hayek's performance plummets. Over 23 games, Alice averages 178 points and Bob brings in 228 points, but Hayek drops to –165. Further exploration is necessary to determine the cause of this change. A preliminary guess would be that Hayek is no longer the only agent affecting the location of minerals in the world, therefore has less control over the location of minerals. Comparison graphs of these test runs can be found in the appendix.

The standard statistical methods for analyzing the performance of algorithms does not seem to apply in agent world. I would like to have some way to perform t-test comparisons, however I doubt their validity. T-tests require that you test different algorithms on the same data and do a side-by-side comparison, however, each agent is presented with a different view of the world, even in the same games.

Future Work

These results from my limited implementation of Hayek in Agent World present contradictions in behavior, however, I believe an economic model of intelligence is a promising solution to reinforcement learning. The following list details the directions of possible future work for refining this implementation.

- I would like to verify and complete my implementation of Blocks World. This could provide more insight into how changes in parameters such as rent, number of agents and probabilities of atom choices affect Hayek's learning ability.

- A statistical comparison is necessary to determine the relative success and failure of the Hayek implementation.

- I have made a number of simplifications from Baum's original work. In dealing with "*s", Baum uses the same direction found in the conditions as the *move* variable for the action. I have broken this consistency, but would like to examine the benefits and costs of this choice. I would also like to examine the differences between selecting a random choice for "*s" versus always picking the first one found.

- The rules generated appear to work well in avoiding walls and picking up vegetables, but their comprehensibility due to the complex connections is limited. A greater understanding of when rules are valid could increase the ability of the user to tweak the performance of Hayek.

- Does seeding with prior knowledge help? An analogy can be made to research in KBANN, where initial knowledge is coded into a neural network. Hayek could be seeded with intuitive rules and bid assignments, then evolved into a trained agent.

- An investigation of the effects of rent would be helpful in solving the "brain-dead" problem previously observed. Should rent be charged when no agent is active? How much rent is enough to purge the population of deadbeat agents?

- The hypothesis space of rules is currently limited by our representation. There is no way to generate an abstract rule such as "If you see an agent in direction x, run in the opposite direction." Creating bins for sensor readings could provide a way to make general inferences and rules. Also many of the rules currently being generated do not use the specifics of the sensor readings except as a comparison. A rule like "If sensor 7 equals sensor 14, move in direction 23" could be fulfilled by any one of our sensor objects. This limits Hayek's ability to find the proper bid for this agent, since at one time 7 and 14 could both be walls, and another time both could be vegetables.

- A further investigation of the necessity and applicability of a credit check to this world could greatly improve Hayek's chance of learning. In Baum's original paper, simulations without the credit check could only succeed in the simplest of blocks worlds with a maximum of five blocks. This seems to be a case where theoretical economics is favored over "real world" economics with bubbles and crashes in the market.

In general, I see many opportunities for exploration on this project. The progress I have made thus far leads me to believe Baum's auction approach to credit assignment is an improvement over the bucket-brigade. I believe the largest problem to solve is the applicability of Hayek in worlds where credit checking is nearly impossible, and this is the direction I would like to continue pursuing in this research.

References

[1] Baum, Eric, A Model of Intelligence as an Economy of Agents, *Machine Learning* 35: 155-185, (1999)