ABSTRACT

There are many applications that one can build by applying Reinforcement Learning (RL). We have chosen to develop a system, popularly called “Village Game” that will teach a computer agent on how to play this educational game to maximize the reward which in this case the total bank balance for the agent.

In developing countries there are serious problems in various areas mainly due to lack of education. We think providing practical education in a game environment can promote social equality and can have a profound effect especially when done at a young age. This game when developed completely can teach the children the responsibilities of a family and how various decisions can affect their livelihood.

The solution of this problem uses Reinforcement Learning using Q-Learning technique in unknown environments where the rewards are time delayed. The game also raises awareness of the importance of decision-making, based on objective data and the assessment of the consequences of various short, medium and long-term action alternatives.

In this problem the agent doesn’t know how the real world works. The agent will have to act in the real world, learn how it works from experiencing it and trying to take actions that yield high rewards. This problem combines our dual objective of an area that we want to understand the various aspects of Q-Learning technique and at the same time create a positive social impact.

Our intention is to implement the game using Pygame, a popular Python library for coding video games. And then implement the AI solution in Python. We intend to publish the details of our implementation, testing and analysis through repeated gameplay.

HISTORY OF THE GAME

This game is created by Stephan Schmitt-Degenhardt [1]. The original version was officially released in July 1995 in Fiji, Vanuatu, Tuvalu and Western Samoa under the name "Navunavici". For international distribution, the game is named Village Game and copyright belongs to ILO - International Labor Organization. The game has been introduced in several countries by DED - Deutsche Entwicklungs Dienst, with financial support from CEFE International / GTZ - German Agency for Technical Cooperation. Many countries including South Africa, Nepal, Sri Lanka and Brazil have adopted the game to promote development using the local culture.

OBJECTIVE

The main objective of the game is to find the optimal policy/optimal value for the agent to maximize the reward. In the process the participants are made aware of the importance of planning, use the available resources, opportunities and conditions of the environment. The consequences of decisions made by players in the game will result in changes to the bank balance. can be assessed at the end of the first round/year and subsequent rounds/years.

HOW IS THE GAME PLAYED

In the board game version, each player anywhere from 2 to 6 rolls a dice to draw the number of "squares" that will advance and determine the "square" to stop in to make a decision and perform some action.

The "houses" of the game dictate that the player make choices about planting, harvesting, storing, and selling crops commonly grown by small farmers..

The game is played in rounds, and a round is completed when the entire length of the board is covered, which corresponds to one agricultural year. During the round, players have opportunities to earn money - by selling crops, livestock and extractive activities, but also have to pay family and other unforeseen expenses.

Each player starts the first round with a starting capital of cash, livestock and labor days represented as ($400, 0, 100). This acts as an input variable for actions, e.g., the player needs 25 Labor Days to plant corn. At the end of each iteration, the financial situation of each player is verified: - How much money, crops and farming and how many working days were used. And wIth the help of an advisor, the quality of the decisions made regarding the use of available opportunities and resources are then evaluated.

EXPLANATION

We assume that all states are Markov states which means that any state depend solely on the state that came before it.

In a Markov Decision Process (MDP), there is a set of Markov states, a set of possible actions for every state, the probabilities of reaching next state from previous one as well as rewards for each transition. The solution of the problem is to find a policy $\pi(s)$ which specifies an action for every single state in order to maximize rewards. To solve the problem it is necessary to determine the optimum value of the policy $V_\pi(s)$ for each state (the value corresponds to the expected utility received by following policy π from state s). Since we are interested in optimum policy values, and we do not want to have probabilities and rewards, we will be using Q-learning technique with function approximation to estimate Q_{opt} for each state (the maximum value). Our intention is to use this technique as a baseline for the solution.

The game can also be developed using Deep Reinforcement Learning which uses Neural Networks to estimate Q_{opt}. The last technique corresponds, in our point of view, to a state-of-the-art solution for our problem.

SIMULATION AND GAME ENGINE

In order to make it feasible to apply AI to solve the game it is necessary to adapt the board game to a digital platform producing a game for one player.

It is possible to perform such adaptation using a platform called Machinations[2]. With this platform we designed, balanced and simulated the desired game systems as well as quickly prototyped and edited parameters until hitting an ideal scenario.

INPUTS AND OUTPUTS

In our environment, the game is played by 1 player over a period of 2 years. Each iteration is a 1 week period. The actions taken can happen stochastically.

We are considering states as given in Table 1.

Table 1: Game State Description

State	Description	Values
Week	The current week of the year	0-104
Planted Corn	Units of planted corn	0-3
Planted Cotton	Units of planted cotton	0-3
Harvested Corn	Units of harvested corn	0-3
Harvested Cotton	Units of harvested cotton	0-3
Unhandled Cotton	Unhandled units of Cotton	0-3
Handled Cotton	Handled units of Cotton	0-3
Stored Corn	Stored units of Corn	0-3
Cow	Units of cow	0-3
Money Value	It is an integer between zero and ten.	0-9
Labor Days Value	It is an integer between zero and ten	0-9

The initial states space has a size of 104 x 4^8 x 10^2 = 681,574,400. During the solution development we can reduce this space in order to optimize calculations.

At each iteration, different actions (11) can be performed:

- Plant 1 unit of corn (in some epochs of the year, with a cost of 25 LD - Labor Days per unit)
- Plant 1 unit of cotton (in some epochs of the year, with a cost of 35 LD per unit)
- Handle 1 unit of cotton (in some epochs of the year, if there are units of handled cotton and with a cost of 10 LD per unit)
- Harvest 1 unit of corn (in some epochs of the year, if there are units of planted corn and with a cost of 5 Labor Days per unit)
- Harvest 1 unit of cotton (in some epochs of the year, if there are units of handled cotton or planted cotton and with a cost of 10 LD per unit)
- Store 1 unit of corn (in some epochs of the year, if there are units of harvested corn and with a cost of 2 LD per unit)
- Sell 1 unit of corn (anytime, if the user has units of harvested corn, with a base price of 50 money value)
- Sell 1 unit of stored corn (some epochs of the year, if the user has units of stored corn, with a base money value of 80)
- Sell 1 unit of cotton (anytime, if the user has units of harvested cotton and with a base money value of 100)
- Buy cow (any time with a base money value of 40)
- Sell cow (anytime, if the user has units of cows and with a base money value of 40)

As the agent plays the game, the game engine provides rewards as money earned. The game ends when there is no money in the bank.

The output is the net balance in bank account at the end of each iteration. The final **output** is money earned over many iterations of the game. The description of the game environment is as given below in Figure 1.

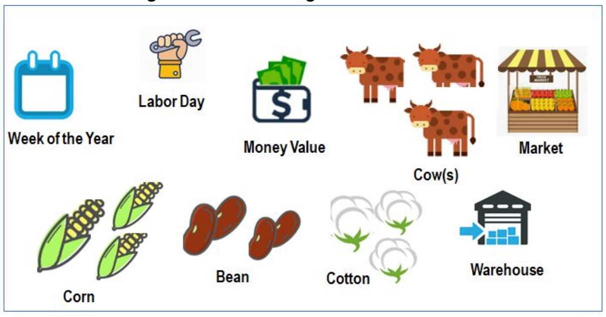

Figure 1: Game Environment

The game ends when there is no money in the bank or we have reached 104 iterations (which equals to 2 years).

BASELINE AND ORACLE

Baseline: This will help us to determine the bottom number on performance. Since the player has to pay 14 units of money each week, and receives 500 units of money in the beginning of the game, after 35 weeks being idle (taking no actions at all) the agent will spend all the initial money and end the game with 0 money value.

Oracle: This will help us to determine the upper limit on performance. For example, if a player has a good knowledge of the game and understands the rules to optimize the performance, then the player can play 104 weeks and make a profit approximately equal to 200 units of money (40 % of profit). We create the simulation using machinations tool and played multiple times to make this amount of money.

In a reduced problem where we have limited the game to 2 years and other constraints, we can observe the gap between baseline and oracle is wide enough where we can make our agent to reach the Oracle solution. When the number of years is relaxed and allow the agent to play for a longer time, then we can really see how optimal is our algorithm that we have implemented.

RELATED WORK

Several research groups have explored the application of reinforcement learning to arcade games and strategy games [3, 4, 5]. [1]One of the state-of-the-art is the paper released by Google DeepMind in 2015[6] which presents the novel concept of a deep Q-network, which combines Q-learning with neural networks and experience replay to decorrelate state and update the action-value function.

REFERENCES

[1] "Village Game - Facilitator's Manual ". Instituto Centro CAPE, Belo Horizonte, MG, Brazil, 1997.Berges, Vincent-Pierre, Pryanka Rao, and Reid Pryzant.

[2] Adams, Ernest, Joris Dormans. *Game Mechanics Advanced Game Design*. New Riders, Berkeley, 2012

[3] "Reinforcement Learning for Atari Breakout". *Stanford University, CS 221 Project Paper.*

[4] Amato, Crhistopher, Guy Shani. "High-level Reinforcement Learning in Strategy Games". *AAMAS '10 Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems – Volume 1* (2010): 75-82.

[5] Hyrkäs, Jarno. "Reinforcement Learning in a turn-based strategy game". Theseus, Finland, 2015.

[6] "Human-level Control through Deep Reinforcement Learning." DeepMind. N.p., n.d. Web. 12 Dec. 2016.