CSCI 4156 – Reinforcement learning

Assignment 3

Evolutionary Strategy Agent in Backgammon Disengaged Bearoff Game

Noof Al Shehhi

B00832852

**Abstract**

The goal for our task this time is to use the Evolutionary strategy (ES-agent) and modify multi-layer perceptron. Therefore, by applying ES-agent the model should improve and make proper decisions while focusing on the bear off stage of the game. In this report, I will be going through two main points which are how the approach and parametrization are done for this task and then discuss the overall performance of the model after applying the ES agent and a simple comparison with my previous TD model.

**Approach**

**Parameterizing:**

**Credit assignment scheme:**

My model was inspired by both Chellapilla and Fogal (1999) and Pollack and Blair (1998). The idea of checking the fitness of the population and updating it was inspired by Chellapilla and Fogal technique. Also, my model has a specific population size with a size of 50 instances from Multiple layer perceptron. However, my fitness method is based on picking the best player over a certain number of games (100 games in my model) and then I picked the top 10 of my population, then the first individual will be my parent for the new population. For Pollack and Blair (1998) I have been inspired by three main ideas in addition to the way I used them while creating the offspring. First how he mutated the child by adding noise to the child. In my model, I have added a normal random between -1 and 1 to represent mutation, but Pollack used Gaussian distribution to add noise. Secondly, update the weight of the parent after comparing the offspring child if it is greater than our best fit then we update like what Pollack did as well. Lastly, I used the same equation that was used in Pollack and Blair's (1998) paper to update my parent's weight, summing 95% of my parent with 5% of my child's weight.

champion = 0.95∗champion + 0.05∗challenger (from Pollack and Blair (1998)).

**Architecture:**

**Multi layered MLP:**

I did not use specific architecture rather than a multi-layer MLP with three hidden layers with the size of 50, output layer, three weights and bias that are random (-0.5, 0.5). The preprocessing function in my MLP is to normalize our input (data). Then the predict function to sum the weight, apply sigmoid function and return our final layer or output layer amount.

**Hyperparameters:**

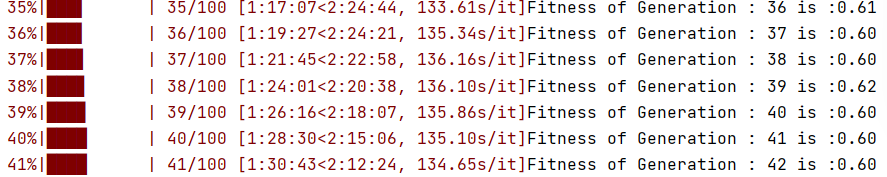
My main hyperparameter setting was my population size which is 50. Moreover, having two hidden layers of the size 50 and neural network with the input size 28 and output with one value outcome. Secondly, setting my random values between -1 and 1 make a huge impact of the variation of the child. And I kept the rate of the parent portion to be 95% and child portion to be 5% same as Pollack and Blair. I checked the best fittest in 100 games which significantly impact the model performance, so if we increase the number the model becomes better but takes time to process. I also have added 2% probability to control the mutation. To sum up, my model starts with a population size 50 and pick the top 10 of each generation. Then I create 40 children from the top 10 during mutation and append it later to create the new generation to end up with 50 again.

**Discussion**:

**Performance:**

For clear address on the improvement of my model performance after training I tried to print my initial population before training, and it was giving 0.514 and after training, my model has increased which means it is learning and getting better and the highest point It achieved was 0.62(Figure1&2). My accuracy was varying between 60 and 63.4 sometimes for the model and the wining rate was 61% (Figure1&2). Therefore, Applying ES-agent or using neuro-evolutionary gave me good results and improved the agent, but when I applied TD-agent I got a better performance for the backgammon game in assignment 2.





**Figure 1&2: screenshot of the performance improvement from my model.**

My ROC curve is very competitive and giving a high value and good AUC = 0.98 which indicates that the model can handle the trade-off and have a constant improve in generations (Figure 3)

A screen shot of a graph

Description automatically generated **A graph of a graph with numbers and lines

Description automatically generated**

**Figure3: screenshot of model ROC curve Figure4: Plot wining rate for 80 number of generations.**

In Figure#4 I used the same submitted model but just trained over 80 generations to show how the model is improving over number of generations and increase winning rate that increases from 0.49 up to 0.62.

**Conclusion:**

I found that ES-agent could vary in its performance during its reliance on randomness during mutation. However, it could be improved and give a really good results of the agent with multiple iteration and good hyperparameters that significantly impact the improvement of the model.

**References:**

Chellapilla and Fogal (1999) Evolving neural networks to play checkers without relying.

on expert knowledge. IEEE Transactions on Neural Networks. 10(6): 1382{1391}

Pollack and Blair (1998) Coevolution in the successful learning of backgammon strategy. Ma-chine Learning 32: 225{240}