**UNDERGRADUATE PROJECT PROPOSAL: GROUP 3:**

Mohammed Qasim Khan 50085658

Ogan Batu Aktolun 500861276

Aidan Uchimaru 500883656

Ivan Golovine 500813431

**Problem**

Teach an agent how to play blackjack through reinforcement learning. Each game will consist of a 20 preset rounds where every game is winnable, the dealer’s and agent’s card draw sequence is known to us (but unknown to the agent), and the optimal win card sequence(s) is hardcoded so we can test against a defined “win” sequence. It should also generate a good sample set and we will aim to have a Win vs Predicted Win Ratio of 30-40% (not sure if too high or too low)

We are not going to implement betting as traditional blackjack has a player place a defined bet that doesn’t increase or decrease as the round is playing out so it doesn’t affect the theoretical “value function” of the round.

**Solution**

*Defining Constraints and Actions*

We will have only 2 possible deterministic actions: hit(s) or stand(s). The game will end when either the agent or the dealer busts (goes above hand value 21) so our end state is also defined.

*Methods Used*

We are going to start off using a Monte Carlo simulation based on an arbitrary policy (e.g. if hand value <14 then hit else stand) over 50,000 rounds to develop rough versions of optimal sequences to win cards and assign values based on them.

After the initial simulation, we are going to seek greedy policy improvement via Q-Learning (Temporal Difference Learning) to determine best-value cards and the optimal hand value to stand on. This part will be done counting cards against the dealer as well, to provide better probabilities and therefore values.

*Value and Reward*

Our value is going to be defined by either the closest you can get to 21 without busting based on our current hand value or staying below 21 if the dealer is likely to bust. Eg: We have a hand value of 13 and we are evaluating whether we should hit. Our agent will count against cards shown by the dealer and determine the value using probability of what cards remain to be drawn and what it’s “intuition” (derived from greedy policy improvement) tells it to do.

Our rewards will be +1,0,-1 defined over hand value ranges and our ƛ will be determined by number of future drawn cards in the sequence removed

**Tasks/Milestones**

*Write the game code*

* Write the blackjack code with the specified characteristics
* Implement the agent which will be used during the run
* Implement statistic tracking

*Create policies*

* Code actions: *hit* and *stand*
* Code policy to follow determined actions with their respective probabilities and rewards

*Generate episodes*

* Test policy over a set number of rounds

*Implement algorithm to update Q values based on episodes*

* Q values will change as episodes are run and the agent will learn to play better over time
* Seek patterns in hand values and sequences. Improve policy again based on findings

*Testing*

* Test agent/player against preset rounds, compare recorded win sequences vs optimal win sequences
* Tweak and improve policy based on results, repeat any process if needed