CS 542 - Final Project
Gautam Bhat, Kaviarasan Selvam, Veena Dali

Documentation

Important: This file describes how to run the code for our project, which is our attempt at an NN-based poker hand-learning and game-playing bot, codenamed “AlphaPo”.
It is important to note that, this project does not implement all rules of poker yet, such as betting. For now, our approach is rather boolean, which means AlphaPo decides whether to continue playing or to fold, based on the cards it has received, rather than betting and raising certain amounts of money (which is currently beyond the scope of our project). Moreover, one more aspect that is beyond the scope of our project for now is strength among hands, which basically means deciding who’s hand is stronger, given both have the same category of hands - if both players have a two pair, any two pair, it’s a draw. This is because of lack of data in this aspect. Producing enough data to implement this functionality was not viable given the timeline of our project, and can be easily implemented in the future if the data is provided. We shall refer to this version minus the betting and intra-hand strength aspects as “experimental” Texas Hold’em. The essence is for the bot to

1. be able to recognize the hand it has received (High Card, Two Pair, Royal Flush…etc)
2. be able to understand the value or strength of the cards it has been dealt, despite not having all cards known. I.e., It should be able to decide whether or not the first two (and subsequent) cards it has been dealt are any good. This is how the bot shall decide its next move.

Keeping these points in mind, the following text will describe how to process the dataset, train the neural net, and play experimental poker with AlphaPo.

Part 1: Process Data

1. Run ‘generateData.m’
    a. Location: generateData/generateData.m

   b. Increase and improve dataset to improve classification - each of these outputs a text file with data in the same format as the UCI dataset
2. Run ‘sampleCARDS.m’
   a. Converts data into binary and combines into one array
   b. Exports final data into a file called *binary_poker.txt*
      - This is the file that will be used to train our neural network

Part 2: Train Neural Network

1. Run ‘matlab_nndriver.m’

Part 3: Use NN to play experimental Texas Hold’em Poker

1. Run ‘pokerplayer1.m’
2. The current code accepts dealt cards as input prompts (part of code where cards are dealt by program is commented out). It is important to note down the format in which cards are represented:
   - Each card is numbered uniquely from 1-52.
   - 1 denotes Ace.
   - 2-10 are as usual 2-10
   - 11, 12, and 13 denote Jack, Queen, and King, respectively.
   - 1-13 are Hearts, 14-26 are Spades, 27-39 are Hearts, and 40-52 are Clubs.
   - Therefore, for example, 14 is the Ace of Spades.
   - One more example, 38 is the Queen of Diamonds.
   - Final example and then we move on, 7 is the 7 of Hearts.
3. The program will prompt in the following order (assuming neither AlphaPo, or the opponent, folds) :
   i. First card dealt to AlphaPo
   ii. Second card dealt to Alpha Po
   iii. The opponent’s decision in this round (1 for ‘Continue Playing’, 0 for ‘Fold’)
   iv. The first common card

v. The second common card
vi. The third common card
vii. The opponent's decision in the second round
viii. The fourth common card
ix. The opponent's decision in the third round
x. The fifth common card
xi. The opponent's decision in the fourth round
xii. The opponent's first card
xiii. The opponent's second card

Remember to input the correct card number (1-52) and decision (1 or 0 only) !

Note: Being implemented in matlab, minus GPU-running capabilities, the bot takes about a minute to make its decisions in the first two rounds.

For further questions or queries about the project, contact : gautam@bu.edu