Homework 1: Building a Poker Agent (100 pts)

Due: Monday Jan 23 at 11:59 pm

Instructions

- Submit a pdf of your work to gradescope, making sure to mark all pages relevant to each outlined section on gradescope (including relevant code)
- Append all written code to the end of your pdf
- In addition, submit all code files to the corresponding assignment on Canvas
- You have 3 late days, with a 10 point grade reduction per day. No submissions will be accepted after the third late day.

Overview

The purpose of this assignment is to get you to begin to think about strategies for mechanizing and evaluating decision making in computational agents. You will look at a particular task environment, and should identify important aspects of the task and environment that inform the design of agents to perform the task in the environment. You will also consider a few possible agent designs, and experiment with your designs to develop a quasi-quantitative sense of how well they work in a simplified version of the environment.

Background

Before starting this assignment, you should read and be familiar with Chapter 2 of R & N (Russell and Norvig, 3rd edition).

Description

For reasons that baffle much of the population, televising poker tournaments, such as the World Series of Poker, has mysteriously become a staple of sports network programming. It seems unlikely that athletic grace and prowess are the main attractions of this "sport," so perhaps there is something about the nature of the competitive task, and the complexity of the environment in which the task is conducted, that rivets the attention of the adoring public.

In part to see if you can discover whether this is the case, this problem set consists of questions about the World Series of Poker and about the design of artificial agents to make decisions in this context. The version of poker played is known as "Texas Hold 'em." A useful tutorial for the game can be found at http://www.thehendonmob.com/guide/texas-holdem, and you should be familiar with how the game works before attempting this assignment. A quick google search for "how to play texas holdem" turns up many resources.

Just to be clear about terminology in this assignment, a **hand** is assumed to involve the dealing of cards to the players, followed by rounds of betting and revelation of other cards until one of the players wins the money that has been bet. A fresh deck of cards is used for each hand, and the dealer has no possibility of cheating to benefit any player.

We will assume that a full **game** consists of a table of players who repeatedly play hands until one of them has won all of the money from the others.

Games take place in the context of a **tournament.** In the larger tournament, winners from different tables then play in games against each other; for the purposes of this assignment, assume that each game brings together a group of players that are unfamiliar with each other. This continues until there is one tournament winner. Assume that winnings are not carried over between games, every player in every game in every round of the tournament starts with the same amount of money.

Task 1: Characterizing the Problem (20 pts)

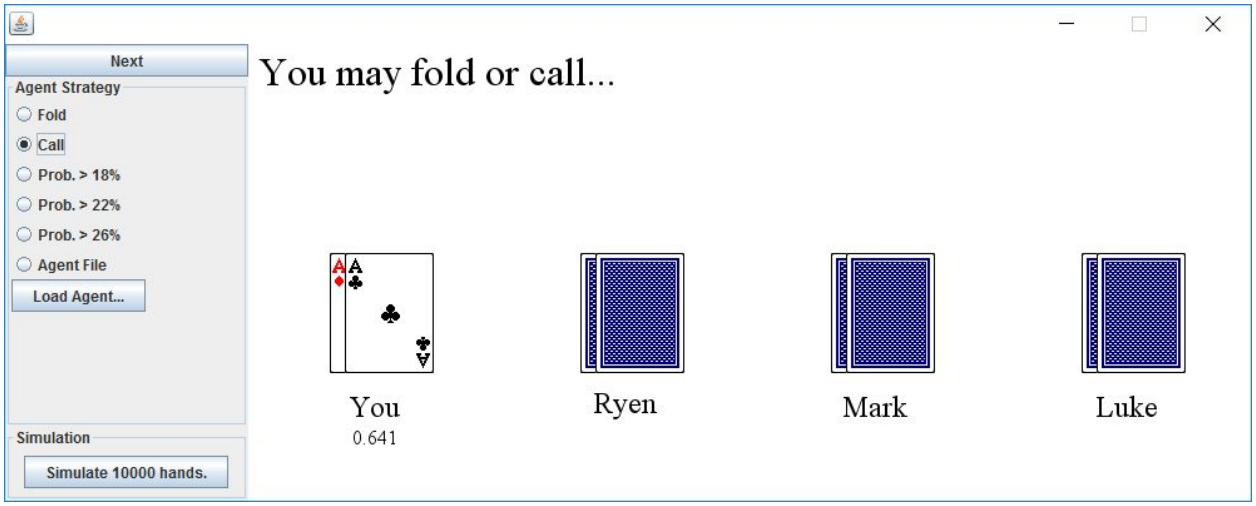
A. Specify a **game** of poker in terms of each element of the PEAS description. Assume that the agent just plays poker, and does not need to move between tables, socialize, order drinks, etc. Explain in 1 or 2 sentences each.
B. Characterize a poker **game** on each of the following axes: Fully vs. Partially Observable, Single vs. Multi Agent, Stochastic vs. Deterministic, Episodic vs. Sequential, Static vs. Dynamic, Discrete vs. Continuous, Known vs. Unknown. Justify each answer with one or two sentences.

Simplified Poker Variant and Simulator (Tasks 2-4)

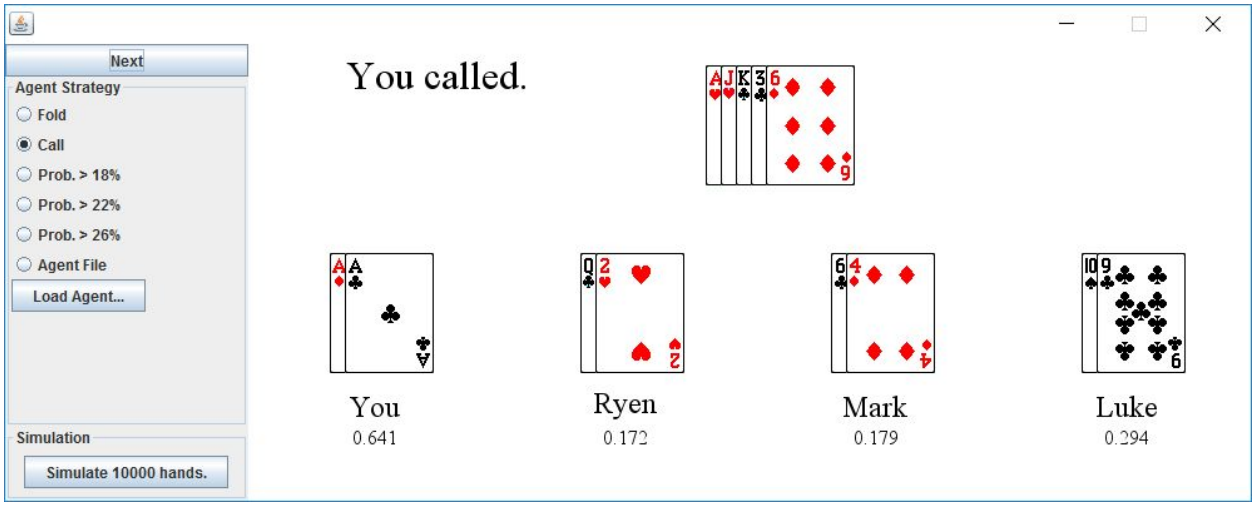
For tasks 2-4, we will consider a simplified version of Texas Hold'em that might make sense in a virtual poker game. There is only one round of betting that occurs right after the hole cards are dealt. An agent's only job in this environment is to determine whether it's a good idea to fold (stop playing) or call (remain in the game). The *only* available sensor information is which hole cards the agent has and the *only* action the agent can take is to choose whether to fold or call. You (or your agent) are set to play against three opponents, named for your favorite GSIs!

We are providing you with a simulator for this version of poker. To run the simulator in Mac or Linux, open a terminal/command prompt in the same directory as your "poker.jar" file. Type "java -jar poker.jar" to run the simulator. On some systems, like Windows, you may also be able to just double click the "poker.jar" file. Note that you will need Java 1.6 or newer installed to run the simulator. This should be true of all the lab computers.

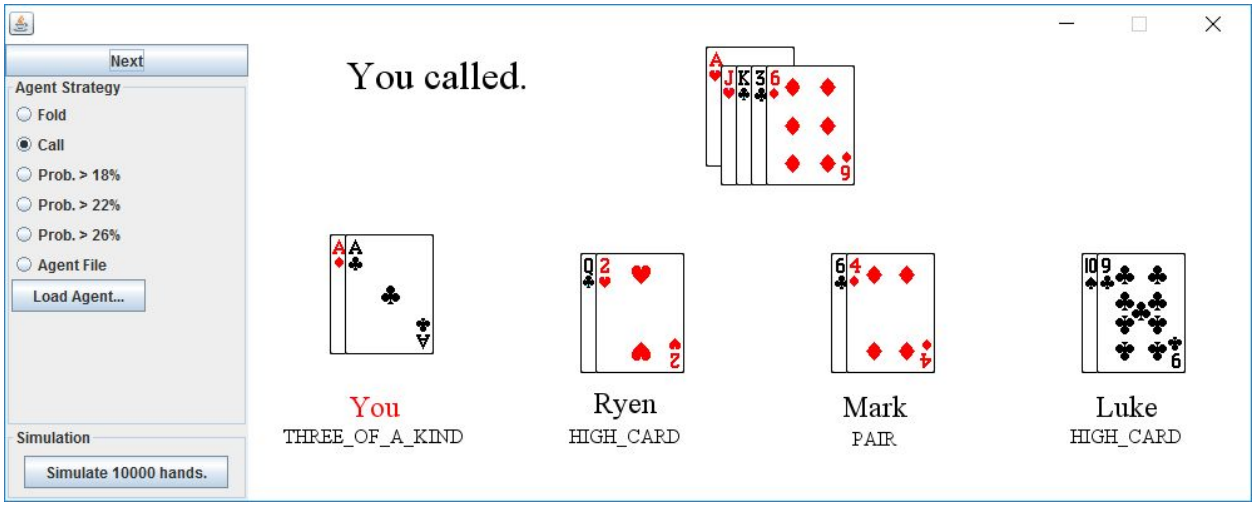
At the start of a hand, cards are dealt to each player but only yours are visible.



In order to get a feel for the system and to try out agents you will implement below, you can select a strategy for your agent to use from the left hand side. When you click the "next" button the agent will use the selected strategy to decide whether to fold or call. Additionally the other players' cards and all 5 communal cards (the flop, turn, and river cards) are now displayed.



You can take a moment to try to determine the winner of the hand for yourself before clicking "next" one more time to reveal the results (the winner's name is highlighted in red and the cards used are shown). Clicking "next" a third time will deal a new hand.



Task 2: Rationality (10 pts)

Before answering the following questions, you should make sure you are familiar with the definition of a rational agent (see section 2.2.1 in the textbook).

A. After observing a particular agent for a bit, we notice it sometimes seems to make the wrong decision. For example, it folded on a [2♣, 4♣] but the communal cards turned out to be [3♣, 9♦, jack ♠, 5♣, 6♣] giving our agent a straight flush and what would have been the win that round! What can we say about the rationality of our agent? Explain your answer.
B. Consider a hypothetical agent that decides whether to fold or call based solely on the value of the largest card in its pair. Is this agent rational? Why or why not?
C. Betting different amounts is an important part of playing poker and can be advantageous for a player. Our agent doesn't have actuators that allow it to do things like go "all-in" on a bluff to scare the other players into folding. With this in mind, could our agent still be considered rational? Explain your answer.

Task 3: Simple Reflex Player (25 pts)

Your goal in this task is to design, implement, and evaluate a simple reflex agent whose job is to decide whether a position should fold or call based solely on the two hole cards.

The behavior of an agent is defined by its agent function and implemented through some kind of agent program. For simple reflex agents, there are two common approaches: **condition-action rules** and **table-lookup**. The two approaches are equivalent – throughout this assignment we will ask you to discuss or design agents in terms of rules, but to convert these to a tabulated format for interfacing with the simulator.

We begin by considering possible agent designs. Remember, we are only considering the pair of hole cards to decide whether to fold or call. Your agent may use any combination of the following functions (you don't need to use them all, of course):

`pair()` - (boolean) true if the hole cards are a pair
`same_suit()` - (boolean) true if the hole cards have the same suit
`distance_between()` - (integer) returns the distance between the cards (useful for detecting possible straights)
`high_card_value()` - (integer) returns the value of the high card (Jack=11, Queen=12, King=13, Ace=14)

So an agent that only calls on pairs would look like:

```
if(pair()) then
   return call
else
   return fold
endif
```

Call this `Agent1`.

A. Using the above function, write condition-action rules as pseudo-code for the following simple reflex agent designs:
   i. `Agent2`: An agent who calls on pairs or if the cards are the same suit and folds otherwise.
   ii. `Agent3`: An agent who calls if it has a jack or higher card in its hand (this would include aces) and folds otherwise.
   iii. `Agent4`: An agent of your own design you feel would perform well. (Also describe in English the rationale behind your design.)

Now, we would like some way to evaluate whether these agent designs are any good. The provided simulator is able to interface with table-lookup agents where the percept entry in the table is the pair of hole cards. Thus, we need to convert our condition-action rule based agents into table-lookup agents.

For each of agents 1-4, you need to generate a text file in the following format:

```
c2,c3,fold
c2,c4,fold
c2,c5,call
…
…
…
s14,s11,fold
s14,s12,call
S14,s13,call
```

For example, the first line above specifies our agent will fold if its hole cards are the two and three of clubs, while the last line specifies our agent will call if its hole cards are the ace and king of spades. We've provided an example file for an agent who always folds in case this is helpful to you (exampleAgent.txt).

When generating your own agent files, use the letters `c`, `d`, `h`, `s` for the suits clubs, diamonds, hearts and spades. When specifying the value for a face card (`j`, `q`, `k`, `a`) you may feel free to use the numerical equivalent (`11`, `12`, `13`, `14`) which is probably easier when programmatically generating the files. (For simplicity we are ignoring the possibility of ace low, i.e. Ace cannot take the value 1 in our case).

B. Give a brief description of the process you used for converting the poker agents defined by condition-action rules to a tabulated format. Feel free to use pseudocode in your description, though this is not required.

Your files should have 52*51=2652 lines, though it is also interesting to consider how many of these may be redundant. Obviously, [c1,c2] is the same hand as [c2,c1] but we also must consider more subtle cases such as the fact [c1,c2] is equivalent to [h1,h2] but better than [c1,h2] since the latter is less useful in working toward a flush (or straight flush).

C. Determine the number of functionally distinct pairs of hole cards. That is, count the number of ways in which two hole cards can be dealt from a 52 card deck such that no two ways are equally favorable for the player. Justify your answer.

To reiterate, in order to interface with the simulator you should include all 2652 lines in your file. Obviously, it isn't a good idea to do this by hand. You can use any programming language to generate the files. Your code will not be graded directly, but you should still submit it as a reference at the end of your pdf and ALSO on canvas (it may help with partial credit if something else is wrong).

When you have generated your files and are ready to test, just place the files in the same directory as the simulator. Open the simulator as described above and click the "Load Agent…" button. Type the name of the desired agent file, including the extension, and click ok. If everything works alright, the name of the agent will appear as the currently selected radio button. As described above, you can now click next repeatedly to step through a number of games and confirm your agent is behaving as intended. Once you're confident your agent is working correctly, click the "Simulate 10000 hands" button to get your results.

Make absolutely sure your agents are behaving as intended and check the simulation results for reasonableness.

If we think in terms of a classification task, what our agent is really trying to do is classify a pair of hole cards as a win (in which case it calls) or a loss (in which case it folds). From this point of view, we can consider the four numbers reported by the 10000 hand simulation as follows:

- False Positive – Called, eventually lost.
- False Negative – Folded, but would have won.
- True Positive – Called, eventually won.
- True Negative – Folded, and would have lost anyway.

D. Record for each of agents 1-4 the outcomes in a table with the following format:

Agent	False Positive (FP)	False Negative (FN)	True Positive (TP)	True Negative (TN)
#	# of hands	# of hands	# of hands	# of hands

E.

i. Compute the percentage of true positives out of total positives and the percentage of true negatives out of total negatives for each agent. (TP/(TP+FP) and TN / (TN + FN))
ii. Order the agents by how many total positives each had (TP+FP). Based on this, and the ratios you computed in (i), do you notice any trends among agents 1-3? Explain why this makes sense based on how each agent plays.
iii. Compare your agent (agent 4) to the others in terms of the ratios from (i) and ranking from (ii). Why do you think your agent performed the way it did?
iv. Describe what criteria or additional information you would need to determine which agent has the best performance (i.e. which agent makes a less severe errors than another). Which agent do you think performed the best overall? State your assumptions and explain your reasoning.
v. Suppose you have a fifth agent that chooses to fold or call randomly, with a 50% chance of each. What are the expected values of the tables entries for this agent? (i.e., what is the expected number of false positives, false negatives, etc.?)

Task 4: Utility-Based Player (25 pts)

The program used in the previous task shows not only the hole cards, but also shows a percentage number below each pair of hole cards. This number is the (approximate) probability that the player will win given *only* knowledge of their two hole cards. (Note this means the four numbers will not necessarily add to 1.)

A. Explain a way these probabilities could be estimated.

To make things more concrete, let's consider agents who call if the probability of winning is above a certain threshold. Specifically, let's consider agents with thresholds of 0.18, 0.22 and 0.26. We would like to determine which of these thresholds is most ideal – the one which strikes the best balance between risk/reward.

B. Run the simulation for 10000 hands for each threshold by selecting the radio button for the appropriate threshold strategy. Record the results from all thresholds in a table like the one in problem 3D.

For simplicity assume there are three possible outcomes:

1. The agent folds and has a net loss of 2 units (due to the ante)
2. The agent calls and loses the hand having a net loss of 8 units (2 to ante, 6 to call)

3. The agent calls and wins the hand, getting all the money bet that hand. We assume all of the others call, so the net gain would be 24 (3*2 from antes, 3*6 from calls)

C. Based on your numbers from part b, determine the net winnings (or losings) of each threshold strategy over the 10000 hands. Which threshold (0.18, 0.22, or 0.26) worked the best?

A worthy goal would be to determine the best possible threshold value. One way to do this is to consider the expected net gain for a single hand based on which action the agent selects.

D. What is the expected net gain for one hand if the agent folds?
E. What is the expected net gain for one hand if the agent calls? (Express this in terms of the probability p of winning)
F. Use parts D and E to determine the optimal threshold value. That is, determine for which values of p it is better to call than to fold.

Task 5: Learning (20 pts)

Now, let us return to the full game of Texas Hold'em, as played in the World Series of Poker. Remember that, unlike the training program we've used, each player only sees his or her own hole cards until the hand ends.

We will only consider a single table of players. At the table, there will be many hands of poker played, with each hand proceeding according to the full rules (as linked to in the introduction). Note that players can only leave the table when they either lose all their money, or else take the money from all others at the table, and no new players can join the table during the game.

An important aspect of poker is to spend some time getting a feel for the strategies of your opponents. Suppose you were associating the opponent's pair of hole cards (for simplicity assume these are shown after the hand even if they folded) with his/her decision about whether to stay or fold. The player could use one of the following three methods above to model the opponent: **Table-lookup, Reflex rule, Threshold**

A. For each method, describe in a few sentences how easy/difficult it would be to infer the model. Briefly sketch a systematic method for building these models while the game progresses.
B. Given your answers from part A, which of these methods would you use to model your opponents? Why?
C. How would an agent use these opponent models to its advantage?
D. If an opponent thinks that the agent is trying to model it, what might the opponent do to hinder those efforts?