

**National University of Singapore  
School of Computing  
CS3243 Introduction to AI**

**Term Project: AI Poker**

---

Issued: Week 4, 2019

Due: Week 13, 2019

---

**Important Instructions**

1. Your project report must be TYPE-WRITTEN using 12-point font. It should NOT be more than 4 pages (inclusive of diagrams and team details described in bullet point 5), excluding references (you may use page 5 onwards for your bibliography *only*).
2. Submit a zipped file of your
  - (a) project report in electronic format (**in PDF format**) and
  - (b) project code in zipped electronic format (see Section 3).
  - (c) You should submit ONE zipped file per team.
3. Each team should have four to five (4-5) members; different group sizes will not be entertained.
4. Indicate clearly your team number (corresponding to the team number on IVLE) and all names and matriculation numbers of team members on page 1 of the project report.
5. Note that the TAs may call any one member of a team to explain the submitted report and demo your program during the marking process. You are not advised to free-ride.

## **1 The Game of Poker**

We first describe how the game of poker (or at least, our variation of the game) is played. Two important notes to remember:

1. You will always be playing a *two player* game, against *only one opponent*. This means you should probably implement an adversarial search strategy (solving a minimax game) with some heuristic to cut off the search.
2. You will always be starting with a pot of £1000 of (fake) dollars. Rest assured that no actual money will change hands during the project.

The following is adapted from various online articles on poker<sup>12</sup>. You can actually play against the current world champion limit Texas Hold'em AI bot here<sup>3</sup>.

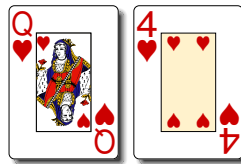
## 1.1 Limit Texas Hold'em Gameplay Overview

A deck of 52 cards is shuffled, and each player receives two private cards that only she can see (these are called the *hole cards*). Next, five cards are placed face down on the table. These are called the *community cards*. At every round, one player acts as the *small blind* and one acts as the *big blind*<sup>4</sup>. The small blind puts down \$10, before the round begins, and the big blind puts \$20. Note that neither player has a choice - they have to bet on these amounts; the idea is to make sure that no player adopts a strategy where they never put down any money unless they believe that they are very likely to win. The game proceeds in *betting rounds*. The first “pre-flop” round begins before any of the five cards is revealed. The small blind begins the betting round. A round of betting continues until

1. One of the players folds OR
2. Both players have put down all of their available money OR
3. Both players have put down an identical bet.

The total amount in the players' bets is referred to as the *pot*. Note that the blind bets are immediately a part of the pot, and the players do not get to keep them (unless they win the round!). Thus, in the first round the small blind (who puts down \$10) must put an additional \$10 for gameplay to continue, else she folds and the blind wins the hand. Matching the other player's bet amount is referred to as *calling*. A player *raises* if she places a bet greater than the opponent's current betting amount (we only allow a raise of \$10 above the opponent's current bet). A player *folds* if she decides to not raise or call, in which case the opponent wins and collects the money in the pot. **note that if a player folds, the winner is not obliged to reveal their hand!**

**Example.** Suppose that Alice's hole cards (the two cards she holds in her hand) are a queen of hearts and a four of hearts. This is a pretty decent hand! Alice (who's the small



blind, and speaks first) decides to try her luck and raise from \$10 to \$30. Bob sees her bet (being the big blind, he has already committed \$20), and puts in an additional \$10. After both players have bet the same amount, gameplay continues.

<sup>1</sup>[https://en.wikipedia.org/wiki/Texas\\_hold\\_%27em](https://en.wikipedia.org/wiki/Texas_hold_%27em)

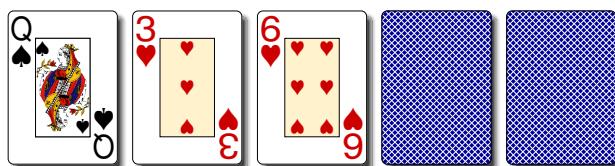
<sup>2</sup><https://www.pokerlistings.com/fixed-limit-texas-holdem-set-up-and-play>

<sup>3</sup><http://poker.srv.ualberta.ca/>

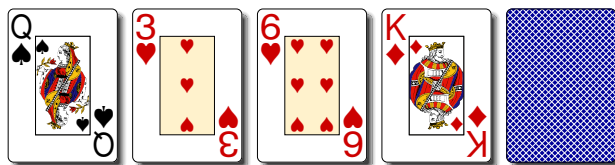
<sup>4</sup>In poker games with more than two players the other players are not called anything, but the big and small blinds are always in play.

After the pre-flop round, three of the community cards are revealed. This is called a *flop round*. After the flop, players continue with another betting round. The blind begins this betting round, and all subsequent betting rounds. After the flop betting round concludes, another community card is revealed, followed by a third betting round; finally, after the fifth card is revealed, a fourth betting round begins. If the fourth betting round ends with no player folding, both players must *show their hand* (this is called a *showdown*); the player with the better hand wins the pot (see Section 1.3 for an elaboration on hand strength).

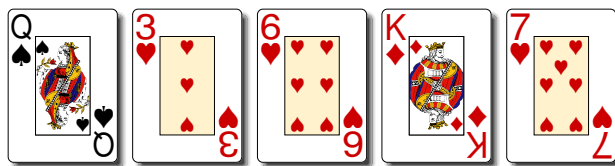
**Example.** Suppose that after the pre-flop round, the following three cards are revealed Alice couldn't be happier! Her queen matches the queen of spades to make a pair,





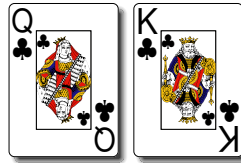
and there are two more heart cards on the table, which means that she has a very good chance of making a flush hand using the two community heart cards and the two heart cards she holds. She chooses to raise again, setting her bet to \$40; Bob calls, evening the bets. This betting round is over, and the next card is revealed. It's a king of



diamonds, which doesn't help her heart flush, nor her pair of queens (if another queen was dealt, she would've held three of a kind, not as good as a flush but still a strong hand). Alice chooses to raise again, setting her bet to \$50; Bob decides to raise this time! His bet is now \$60; Alice chooses to call, raising by another \$10. There is now \$60 + \$60 = \$120 to be won. The final card is revealed:



It's a seven of hearts! With  and  as her hole cards, Alice can make a flush! She raises by another \$10 and Bob calls. Both players have now put down \$70. Since neither of them folded, we have reached a showdown (see Section 1.2), and both players reveal their hand. Bob's hand was A very strong hand, and with the appearance of



the king of diamonds giving Bob *two pairs*, which is not too shabby. However, Alice's flush is a much better hand, and she takes the pot.

## 1.2 The Showdown

If a player makes a bet and the opponent folds in response (i.e. is not willing to match the betting amount), that player is awarded the entire pot, and is not required to show their hand. In the showdown, the value of a player's hand is equal to the *best hand* that they can form with the cards in their hand and the five community cards. Players are not required to use any of the cards in their hand to form the best hand (although the best that a player can hope for in this case is to split the pot). If both players have equally good hands, then the pot is equally split amongst them. Note that all five cards are used to evaluate hands, so ties are extremely unlikely (see Section 1.3).

## 1.3 Poker Hands

Table 1 (adapted from Wikipedia) offers a complete description of all poker hands in increasing order of rank; if the hand involves  $< 5$  cards, the extra cards are used to compare the hands. Cards have a numerical value:  $\heartsuit 2 < \spadesuit 3 < \dots < \clubsuit 10 < \spadesuit J < \heartsuit Q < \heartsuit K < \clubsuit A$ ; their color (also known as *suit*) is unimportant.

## 1.4 Betting

At every round, one player is the *small blind*, and posts a *small blind* equal to \$10; the other player puts up a *big blind* of \$20. The small blind chooses to act first, and the game proceeds in turns from there. It is important to note that a betting round must end with both players betting the same amount! At every round, the person starting the round can choose to either

**Fold:** the round ends, and the opponent receives whatever amount of money is in the pot.

**Check:** the turn goes to the other player.

**Raise:** the player increases their bet by \$10

**See:** if the opponent raises their bet to \$ $x$ , the player simply matches their betting amount to be \$ $x$  as well and the round ends.

Some important notes









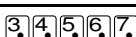

Name	Description	Example
Highcard	Simple value of the card. Lowest: 2 - Highest: Ace	
Pair	Two cards with the same value	
Two pairs	Two times two cards with the same value	
Three of a kind	Three cards with the same value	
Straight	Increasing sequence of 5 cards <sup>5</sup>	
Flush	5 cards of the same suit	
Full house	Combination of three of a kind and a pair	
Four of a kind	Four cards of the same value	
Straight flush	Straight of the same suit	
Royal flush	Straight flush from Ten to Ace	

Table 1: Poker hands

1. If an opponent raises, the player may see and then raise in then turn. For example, Alice and Bob both bet \$30; at her turn, Alice raises to \$40. Bob can see her bet (and put an additional \$10) or see and raise (and put an additional \$20). If this happens, then Alice needs to decide whether she sees, raises or folds.
2. The only exception to the above is in the very first turn of the first betting round, where the bets are asymmetric (big blind \$20 and small blind \$10). The small blind can choose to fold, see (match to \$20) or raise (to \$30)<sup>6</sup>.

## 2 Heuristics for Designing a Poker Player

The goal of this project is to develop a competent AI poker playing agent. The agent's action, denoted by  $a$ , is determined by the current round state (see Section 1.4). Each state of the game consists of four components:

1. The private cards visible to the player.
2. The community cards visible to both player and opponent.
3. The current bet placed by you and
4. The current bet placed by the opponent.

<sup>6</sup>Seeing the bet as a small blind is considered by many professional poker players as a bad strategy, and is called limping. In fact, the AI poker bot that first beat professional players limped often, and was nicknamed *Claudico* (which is limper in latin).

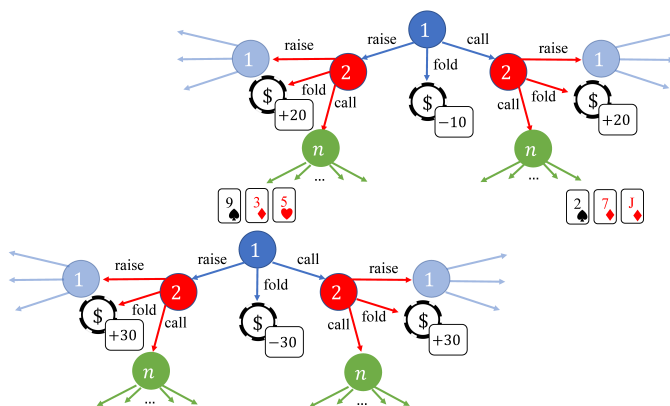


Figure 1: A portion of a poker game tree. The blue nodes are controlled by the small blind; red nodes are controlled by the big blind, and nature controls the green nodes.

Poker is an extensive form, zero-sum game with uncertainty, for which an optimal strategy can be computed using the minimax tree search we discussed in class. It would thus make sense to search down the tree in order to decide optimal play. Unlike the minimax tree search we have seen in class, however, poker has an element of uncertainty, derived from the initial shuffling of the deck. Figure 1 presents a part of the poker game tree, from the perspective of the small blind. At first the small blind gets a set of cards (while they are randomly assigned to the small blind, they are known so there's no uncertainty about them). Next, the small blind needs to decide whether to raise (make their bet \$30), call (make their bet \$20, same as the big blind) or fold and forfeit the hand. Next, the big blind decides what to do, followed by a round of *nature randomly selecting three cards to be revealed to both players*. Note that there can be additional sequences of raises not shown in the tree (we will allow at most four subsequent raises to make things easier). **Your poker agent will most likely need to generate this game tree up to a certain depth.** Note that randomization makes the branching factor of the tree quite sizable.

Much like other extensive form games, exploring the entire game tree will be extremely time consuming; your poker agent will need to employ some kind of heuristic evaluation function estimating the expected quality of each action at the cutoff stage. There are many criteria one might factor in when trying to evaluate a particular node in the decision tree. Some of them would be

1. The expected strength of your hand vs. the opponent's hand (see Table 1).
2. The amount of money in the pot
3. The opponent's behavior (this is probably the most important, and trickiest to model!) - what are your assumptions about the opponent's play strategy? You

can even try to *learn* opponent behavior, since you will be playing several hundreds of hands against your opponent.

One can try and set up a linear evaluation function of the form

$$\phi(s) = \sum_{j=1}^m w_j \phi_j(s)$$

where  $\phi_j(s)$  is the numerical value of some feature of the current state  $s$  (e.g. amount of money in the pot), and  $w_j$  is the weight (or importance) that your agent assigns to this feature. You can hard-code these weights, but it is preferable that you *learn them by training your agent!*

Some poker agents employ *abstraction*: this is a way of controlling the branching factor. The idea is simple: we treat different positions in the search tree in the same way. For example, instead of considering the (several thousand) possible branches of the nature move individually, one can group them together in a clever way into buckets of “worth” (e.g.  $\heartsuit J \spadesuit J \clubsuit 7$  can be very strategically similar to  $\heartsuit Q \heartsuit Q \clubsuit 6$ ). Another abstraction can occur in the betting amounts - one can decide that pots of size \$0–\$20 are strategically equivalent, and that pots of size \$20–\$50 are also equivalent. These are all design heuristics that can be hard-coded, but preferably *learned by training your agent!*

It is possible, in theory, to ‘cheat’ and have your agent try and explore the entire search tree, which will result in it getting stuck in a very long computation. To avoid this, we allot your agent **at most 100ms to perform an action**. After 100ms, your agent will play a default action (folding).

#### Remark

Students sometimes have a tendency to throw everything at an agent in the hopes that something works out (designing evaluation functions that have hundreds of features, training using various convoluted methods etc.). This is usually indicative of an unstructured approach to the problem, which would most likely be difficult to justify (see Table 4: we will place weight on how well you explain your work), or to effectively train. A lean, clever agent is often much more effective than a large agent that tries to do everything and accomplishes nothing.

## 2.1 Training Your Agent

Most AI poker agents are not completely hard-coded; at least some of their key parameters are learned. The most straightforward way to do so is by letting variations of your agents play against each other, and use some form of *reinforcement learning*. For example, let  $c_1, \dots, c_n$  be the game configurations you are testing. One simple way of updating their performance is observing the game’s reward (say from one hand, ten

hands or 100 hands) and updating your agent's weight using an update function of the form

$$Q(c_i) = (1 - \alpha)Q(c_i) + \alpha \left( \sum_{j=1}^n r(c_i, c_j) \right)$$

where  $r(c_i, c_j)$  is some reward function reflecting the relative performance of configuration  $c_i$  against configuration  $c_j$ . You may consider other methods of updating weights.

### 3 Grading Criteria

Note that you can propose your own agent's strategy; you do not necessarily have to implement the utility-based agent specified in Section 2.

All members of the team will be given the same grade/mark<sup>7</sup>. The grading criteria are:

17%	<b>Project report:</b> This 4-page report should document your agent's strategy, experimental results demonstrating its performance, and observations and analysis to discuss why your strategy performs well (or doesn't). State clearly and explicitly the <i>novel</i> and <i>significant</i> contributions and achievements (if any) of your team that, in your opinion, will distinguish your agent's strategy and project work from that of the other teams and the existing literature. See Appendix A for more details.
5%	<b>Performance against the agents designed by other project teams:</b> We will let your agent play against the other teams' agent in a tournament (featuring several hundreds of hands to control for blind luck). Be ready to demo your program when called upon by the TAs. You will not receive any points in this category if your code cannot be compiled on our machine, or if it takes a very long time to run on our machine. We encourage you to test your code before submission (see Section B.2 for details).
2%	<b>Project program:</b> A template is provided. See Appendix B.1 for more details.
1%	<b>Following instructions:</b> The instructions are provided on page 1, and in the Appendix.

When submitting, Zip your completed code with the project report (in PDF format) for submission. **The file should be named Poker[GROUP NUMBER].zip**; please do not deviate from this naming convention (**or risk losing points for not following instructions**).

<sup>7</sup>Free riders are given zero.



# Appendix

## A Grading Criteria for Project Report

Your project report should follow the formatting instructions in the templates provided on IVLE (we will use the IJCAI 2018 formatting guidelines<sup>8</sup>). Your writeup should not exceed 4 pages. An additional page containing *only* references is allowed. You may also include an appendix containing additional relevant material, which will be read at my discretion (but do not assume that it will be read).

I will place significant emphasis on how you trained your agent (e.g. using self-play as described in Section 2.1). You *must discuss this part of your project*. Since training may take a long time, you might want to consider parallel/distributed learning on multiple cores/machines<sup>9</sup>.

Tables 2–6 in the Rubrics Section (Appendix C) describe the grading rubrics we will use to assess the project writeup. Some important notes:

1. Note the **top 5%** row: this contains the guidelines for getting a near-perfect score for this criterion.
2. Reviewing papers is never an exact science, and always leaves some students with the feeling that their project was graded unfairly. I strongly urge you to consult with your group tutor if you are not sure that your work is up to our standards.

Make sure that you write in clear, correct, academic English. I recommend that you use L<sup>A</sup>T<sub>E</sub>X, rather than MS Word to typeset your work; in any case *the submitted file must be in PDF format*, and your writeup should follow the IJCAI 2018 formatting instructions provided.

## B Project Program

### B.1 Writing Your Code

You should only modify the `declare_action()` function within the project code. **Do not change any other part of the code!**

- (1) The rest of the code should NOT be modified to suit your needs; such requests will NOT be entertained. We will use the original code provided anyway (overwriting yours) and deduct one point from your grade if you do. If your `declare_action()` function does not work as a result, you will be given a day (24 hours) to modify it, **with an additional grade deduction of 1–3 points, depending on the amount of work required to change the code, and at the course staff’s discretion.**

---

<sup>8</sup>IJCAI — the international joint conference on artificial intelligence — is the leading international AI conference

<sup>9</sup>NUS offers several excellent computing clusters; you may also use the NSCC. However, we do not offer any dedicated computing resources for this task.

- (2) You may write local functions that are called by the `declare_action()` function. You may also refer to other parts of the code in your implementation, as long as this does not violate (1).
- (3) You should provide detailed comments in your code to facilitate our understanding (uncommented code will be less likely to be graded well).
- (4) While you are encouraged to train your agent, you should make sure that the training phase does not occur “live” while your agent is picking actions. In other words, the agent we receive should already be implemented after being trained, and should not do any *costly* training during its run. Of course, if you wish to have your agent adapt to an opponent’s strategy that is absolutely fine, but take into account that your agent is given at most 100ms to choose an action, and should thus not take too long to run.

## B.2 Testing Your Agent

For your convenience, we have provided you with an option to test your code before submission. We will put up an online portal to which you can submit your code. The portal will run a simple ‘capture the flag’ protocol. At first, we will have a simple poker agent up, whom you’ll need to beat. If you manage to beat our simple agent, then your team’s agent will be the leader. Subsequent teams can try to beat your agent, and if they do, their agent becomes the leader (holding the flag). You can feel free to resubmit your agent as many times as you like, but please do not spam the server. Teams that submit their code an unreasonable number of times (e.g. several hundreds of submissions overall, or several times in a row for no good reason) may be banned from the website. **The website is not meant for you to train your agent! Please do not run training sessions on the portal! If you do, you will probably be preventing other teams from using the service, and we may ban you from using the portal.**

## C Grading Rubrics for Project Writeup

<b>Writing Quality - General</b>	
0%–20%	The paper fails basic rules of academic writing. English quality is poor, making the paper essentially unreadable
21%–40%	The paper suffers from serious organizational issues and exhibits poor writing level. It is readable after significant effort.
41%–60%	The paper is not an easy read, but it was possible to get through it with some cognitive effort.
61%–80%	Writing quality is okay, but the paper would have benefited from careful proofreading.
80%–100%	The writing quality is good, adhering to academic English guidelines and has a good narrative flow.
<b>Top 5%</b>	The paper is a pleasure to read. It is an example of how academic project papers at SoC (indeed, NUS!) should be written.

Table 2: Grading Criteria: Writing Quality

<b>Introduction</b>	
0%–20%	The introduction is not introducing the paper at all, or is nonexistent. There is no main thesis.
21%–40%	The introduction is more confusing than helping, but does contain at least some idea of what the main thesis is.
41%–60%	The introduction is not great but has a clear statement of the main contribution which can be elicited after some effort/wading through irrelevant details.
61%–80%	It is possible to understand the main idea with minimal cognitive effort, but some key points are not highlighted.
80%–100%	The introduction is focused and clear. Has examples/intuitions and is fun to read. Tells the reader what is the method, why it was used, and what are the key results achieved.
<b>Top 5%</b>	The introduction rivals that of good published academic papers. I would have been happy with this being an introduction to one of my papers.

Table 3: Grading Criteria: Introduction

<b>Justification</b>	
0%–20%	The paper contains no justifications for the methods used. The source code offers better reasoning than the paper.
21%–40%	Some parts of the work are justified, but most steps are not discussed/explained.
41%–60%	The team provides reasonable justifications for some steps, but at least one major step is not explained.
61%–80%	The methodology is mostly explained well, but some minor parts are unexplained, or some major parts are poorly explained.
81%–100%	The methodology is explained well overall. All major steps/design choices in the implementation of the agent/learning method are discussed and reasoned about.
<b>Top 5%</b>	The justifications are complete, clear and make a compelling case as to why this is the best agent that the team could come up with. The agent's merits and limitations are discussed, leaving absolutely no doubt in the reader's mind as to the team's working process when implementing their agent.

Table 4: Grading Criteria: Justification

<b>Technical Quality/Creativity</b>	
0%–20%	There is little novelty; the group basically implemented code they downloaded, or created a bare-minimum compiling agent
21%–40%	The team tried out a few things, but ended up with an agent that is fundamentally flawed/very simple.
41%–60%	The agent was implemented in a reasonable manner. It presents some sensible design paradigms, but has at least one major design flaw (for example, it does not try to search through the game tree at all)
61%–80%	The team implemented a solid, fairly competent poker agent that works, but is rather unimaginative/unambitious in its design.
81%–100%	The team implemented a novel, competent poker agent that is able to play well, and has meaningfully implemented an adversarial search algorithm with an evaluation function cutoff/abstraction to reduce the search space.
<b>Top 5%</b>	The students extensively experimented with different techniques and implemented what they believe was the best solution to each part. Their ideas are novel and resulted in significant performance improvement. The team might want to develop a competitive AI poker agent!

Table 5: Grading Criteria: Technical Quality

<b>Training Method</b>	
0%–20%	The agent was not trained at all
21%–40%	The agent was trained, but either minimally so, or in a very flawed/unstructured manner. The group did not seem to understand why they were doing this part.
41%–60%	The agent was trained in a somewhat unconvincing manner. There were some serious methodological issues that the group did not explain/overlooked entirely.
61%–80%	The team ran some experiments that worked somewhat well, but did not consider significant elements of the training process/did not offer sufficient evidence of performance improvement due to training.
81%–100%	The team competently implemented a learning process to train their agent. The agent's performance improvement as a result of the training process is clearly illustrated in figures. The training paradigm is well thought-out and explained in detail.
<b>Top 5%</b>	The training process that the students describe is novel, extensive and resulted in a significant performance boost. The training paradigm is well reasoned (e.g. the students prove that with enough training cycles their agent is guaranteed to be playing a subperfect Nash equilibrium) and is backed up with clear figures illustrating the data/learning process.

Table 6: Grading Criteria: Agent Training