ASSIGNMENT 3: BLACKJACK

Goal: The goal of this assignment is to find a solution to a real problem using ideas from sequential decision making under uncertainty (Markov Decision Processes).

Scenario: The game of Blackjack. *Terminology*

cards: a standard deck of playing cards is used, i.e., four suits (clubs, diamonds, spades, and hearts) and 13 different cards within each suit (the numbers 2 through 10, jack, queen, king, and ace). In this assignment, we will replace 10, jack, queen and king with a generic 'face card'. We will assume an infinite number of decks available in the pack.

card values: the numbered cards (2 through 9) count as their numerical value. The generic face card (replacing 10, jack, queen, and king) counts as 10, and the ace may count as either 1 or 11 (whichever works in the player's favor).

hand value: the value of a hand is the sum of the values of all cards in the hand. The values of the aces in a hand are such that they produce the highest value that is 21 or under (if possible). A hand where any ace is counted as 11 is called a soft hand. The suits of the cards do not matter in blackjack.

pair: the two card hand where both cards have the same value. (example, two aces, a pair of sixes, and for our assignment, a pair of face cards).

blackjack: is a two-card hand where one card is an ace and the other card is any face card.

busted: the value of the hand has exceeded 21.

Rules of Play

There are some slight variations on the rules and procedure of blackjack. Below is the simplified procedure that we will use for this assignment. We will not be using insurance, surrender or dealer peeking, which are options in a standard casino game.

1. Each player places a bet on the hand.

2. The dealer deals two cards to each player, including himself. The player's cards will be face-up. One of the dealers cards is face-up, but the other is face-down.

3. If a player has blackjack, then that player wins immediately. The player makes a profit of 1.5 times his or her bet.

4. If not, the player must do one of the following:

H - Hit: the player receives one additional card (face up). A player can receive as many cards as he or she wants, but if the value of the player's hand exceeds 21, the player busts and loses the bet on this hand irrespective of dealer's hand.

S - Stand: the player does not want any additional cards

D - Double-down: before the player has received any additional cards, she may double-down. This means that the player doubles her bet on the hand and will receive exactly one additional card. The disadvantage of doubling-down is that the player cannot receive any more cards (beyond the one additional card); the advantage is that the player can use this option to increase the bet when conditions are favorable.

P - sPlit: before the player has received any additional cards, if the original hand is a pair, then the player may split the hand. This means that instead of playing one hand the player will be playing two independent hands. She puts in the bet amount for the second hand. Two additional cards are drawn, one for each hand. The play goes on as if the player was playing two hands instead of one. If the drawn cards result in more pairs she is allowed to resplit. The player is allowed endless resplits in our version of the game. [There is an exception associated with a pair of Aces, see below]

5. Once the player stands, the dealer turns over his face-down card. The dealer then hits or stands according to the following deterministic policy: If the value of the hand is less than 17 (or soft 17), the dealer must hit. Otherwise, the dealer must stand. This means, that the dealer stands if his Cards are (A,2,4) because that makes a soft 17. If the dealer busts, then he loses the bets with the non-busted players.

6. PayOffs (in this order)

(a) If the player had a blackjack she already received 2.5 times her bet thus making a profit of 1.5 times.

(b) If the player busted, she lost her bet.

(c) If the dealer busted, he lost and the dealer pays the player double her bet, i.e., the player makes a profit equal to her bet.

(d) If the value of dealer's hand is greater than player's the player loses her bet.

(e) If the value of player's hand is greater than dealer's the player won and dealer pays double her bet.

(f) If the dealer has blackjack and the player has non-blackjack 21 the dealer wins.

(g) If the value of the two hands is equal, it is a push and the player gets back her bet money. That is, no profit no loss.

7. Other rules and exceptions.

(a) Doubling is allowed after split. That means, after splitting the pair, the player is allowed to double down either or both her hands if she wishes to.

(b) Player can resplit as many times as she desires (whenever allowed).

(c) Splitting Aces. This is an exception to the rule. If the player gets a pair of aces that is a very strong hand. She can split this but she will only get one additional card per split hand, and she will not be allowed to resplit. Moreover, if the card is a face card, it will not be counted as blackjack, and will be treated as a regular 21.

To familiarize yourself with the game, play it online for a few minutes. There are many online applets available, for example, http://www.simcasinogames.com/blackjack/. The rules they follow may be slight variations of those used in the assignment, but you will get the general idea.

Problem Statement: In this assignment your task is to compute the policy for an optimal player. As usual the first step is to carefully think of the state space that you will need. To give you a headstart, we propose that the state space contains the following fields:

1. MinSum: the minimum sum of the player's hand so far.

2. NumAces: the number of Aces in the player's hand so far.

3. dMinSum: the minimum sum of the dealer's hand so far.

4. dNumAces: the number of Aces in the dealer's hand so far.

5. isTwoCards: a Boolean that represents that the player's hand has just two cards so far.

6. isBlackJack: a Boolean that represents that the player got a BlackJack.

7. pair: has value between 0 and 10. Zero indicates that the player doesn't have a pair. Any other value i indicates that the player has two cards of value i.

8. turn: a Boolean to indicate whether it is player's turn or dealer's.

Take a few minutes and convince yourself that the aforementioned representation is adequate to model the different relevant states of the game. If you think it is not, feel free to change it to suit your convenience. Design a state transition function, and reward model to encode the dynamics of the play. Solve the game to compute the best play in each state. The best play is defined as the action (hit/stand/double/split) that maximizes the expected return. Make sure you double or split only in the states it is allowed. Assume that the player bets $1.

Program for a BlackJack(p) game. Assume that the probability of getting a face card is p (an input to the program) and the probability of getting all other cards, 2-9 and Ace, is uniformly (1-p)/9. Note that p = 4/13 captures the standard Blackjack game.

After you solve the problem, the solution to BlackJack(4/13) should look very close to this. In the first column, representing your hand, a single integer represents the sum of the two cards, and indicates that

they are not a pair and that neither is an ace. *For the output of this assignment, you need to return only the first action that you will take*. Thus your output need not distinguish between "D" and "DS".

Output format:

The output format is very close to the policy in the link. The first value in a row is your hand – it goes from 5 to 19. Then we have special cases, first when one of the cards is an ace, A2 to A9 and finally pairs, 22 to 1010. After your hand there is a tab '\t' and then there are 10 values for different open cards of the dealer (2 to 10 then ace). The letter will indicate your first action if you get this hand. Here is a sample output:

5 H H H H H H H H H H

6 H H H H H H H H H H

7 H H H H H H H H H H

8 H H H H H H H H H H

9 H D D D D H H H H H

10 D D D D D D D D H H

11 D D D D D D D D H H

12 H H S S S H H H H H

13 S S S S S H H H H H

14 S S S S S H H H H H

15 S S S S S H H H H H

16 S S S S S H H H H H

17 S S S S S S S S S S

18 S S S S S S S S S S

19 S S S S S S S S S S

A2 H H H H D H H H H H

A3 H H H D D H H H H H

A4 H H H D D H H H H H

A5 H H D D D H H H H H

A6 H D D D D H H H H H

A7 S D D D D S S H H H

A8 S S S S S S S S S S

A9 S S S S S S S S S S

22 P P P P P P H H H H

33 P P P P P P H H H H

44 H H H P P H H H H H

55 D D D D D D D D H H

66 P P P P P H H H H H

77 P P P P P P H H H H

88 P P P P P P P P H H

99 P P P P P S P P S S

1010 S S S S S S S S S S

AA P P P P P P P P P H

As an example, the table above suggests that if you get a pair of 5s and the dealer got a 9 then you should double.

Use a tab as field separator between the "hand" field and the action fields. Use a space as field separator between the actions. Do not give a header or footer row, or any other extraneous output. Make sure you have the correct number of rows and colums. Note that there is no endline (‘\n’) after the last row.

Hint

There are many ways to solve the problem. One way is to model as a Markov Decision Process. This gets tricky when you try to split a hand. An easier way is to write a set of recursive equations and use value

iteration to solve them. Both ideas work. If you are confused after 3-4 days of modeling please send us an email explaining your ideas.

Code

Your code should compile, if necessary, and run on attu.cs.washington.edu. Please supply a compile.sh script if compiling is necessary. Also supply a run.sh script. We will run your code as follows:

./compile.sh

./run.sh 0.307

./run.sh 0.4

The parameter in front of the ./run.sh will be the value p that represents the probability of the face card. When we call run.sh in the above format, it needs to save the policy to a file "Policy.txt" in the present working directory.

What is being provided?

We provide an output checker, formatcheck.py, that tests whether your output is in the required format or not. To help students unfamiliar with shell scripting, we also provide a sample run.sh script that takes the input parameter from the command line and passes it to your own program (assuming your program is called BlackJackPlayer). Modify this file to replace "BlackJackPlayer" with the name of your executable. If your code is in an interpreted language (like python), you can replace this with "python YourPythonScript.py".

What to submit?

1. Submit a zipped folder (not tar.gz) containing your code and writeup. Make sure that when we run "unzip yourfile.zip" on attu the following files are produced in the present working directory:
   compile.sh (if your code needs to be compiled)
   run.sh
   Writeup.pdf

   You will be penalized for any submissions that do not conform to this requirement. We will run your code on a few sample parameters and verify the accuracy of your policy against a gold standard.

2. Your writeup should describe how you modeled the dynamics and rewards of problem. Include a discussion on how you handled double down and split.

Evaluation Criteria

1. [6 points] Your writeup.
2. [5 points] Soundness of your code.
3. [1.5 points] Discretionary points based on the writeup.

What is allowed? What is not?

1. You may work in teams of two or by yourself. We do not expect a different quality of assignment for 2 people teams. At the same time, please spare us the details in case your team cannot function smoothly. Our recommendation: this assignment may be a little hard for students who are learning this material for the first time; work in teams if you can find a workable partner.
2. You can use any programming language as long as it runs on the attu environment.
3. You can use any generic MDP code, however you may not use any code written specifically for Blackjack. For example, here is some code in java: http://code.google.com/p/aima-java/.
4. You must not discuss this assignment with anyone outside the class. Make sure you mention the names in case you discuss with anyone in the class outside your team.
5. Your code will be automatically evaluated. There will be negative penalty if your output is not automatically parsable and the TA has to go back and forth with your team to get your code/output working. Make sure it satisfies the format checker before you submit.