Blackjack-Playing Agents in an Advanced AI Course

Jeffrey L. Popyack
Dept. of Computer Science
Drexel University
Philadelphia, PA 19104-2875
(215) 895-1846
JPopyack@CS.Drexel.edu

ABSTRACT

Blackjack is a multi-player card game in which each player makes a sequence of decisions based on a partially-observed game state that evolves under uncertainty. In its casino variant, blackjack is typically played as a set of separate contests, each involving a player and the dealer, whose decisions are determined according to a published, fixed policy. We have developed and used a sequence of assignments for an advanced artificial intelligence course in which students determine an optimal strategy for blackjack, modeled as a Markov Decision Process. The course culminates in a blackjack tournament in which agents employ policies students have determined through their analyses.

Categories and Subject Descriptors

G.3 [**Probability and Statistics**]: Markov processes, Probabilistic algorithms, Stochastic processes I.2 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search– *control theory, dynamic programming,* Distributed Artificial Intelligence – *intelligent agents*

General Terms

Algorithms, Performance, Design, Experimentation, Theory

Keywords

Blackjack, Artificial Intelligence, Markov Chain, Markov Decision Process, Tournament.

1. INTRODUCTION:

1.1 Background - The Course:

Drexel University employs the quarter system, which involves twelve ten-week terms of academic study leading to a bachelor's degree. Requirements for the BS and BA programs in computer science include 8 computer science electives, including at least two *tracks* of advanced study in core computing areas. Track requirements are fulfilled by completing 3 courses designated as track courses, some of which ("track foundation courses") may be required of all students pursuing the track. The Artificial Intelligence track has been a popular choice, with one track foundation course, CS 380 (Artificial Intelligence) with prerequisites in mathematical foundations and data structures. Regularly offered track electives include advanced artificial intelligence, robotics, and evolutionary computing. Topics covered include agents, production systems and search, reasoning and knowledge representation. CS 481 (Advanced Artificial Intelligence) is the most mathematically intensive track offering, which includes as prerequisites CS 380 and one of the probability courses required for the BS or BA program. Topics covered in this course generally include machine learning, decision-making under uncertainty, filtering, prediction, Markov decision processes, hidden Markov models, reinforcement learning and neural networks. Material on Markov processes, modeling, Markov decision processes and solution methods comprises approximately 4 weeks of the course. Both courses use [1] as a text, with CS 481 using [2] for the material on reinforcement learning.

1.1.1 Sample Syllabus

Wee k	*Topic*	*Reading*
1	Introduction and Overview. Probabilistic Foundations, Bayes Rule, Inference.	Chap. 13-14 **
2-3	Reinforcement Learning. Evaluative Feedback. Reward Systems. Markov Chains.	Chap. 1 Sect. 2.1-2.2 Chap 3
4-6	Dynamic Programming. Markov Decision Processes. Monte Carlo Methods. ***Midterm Exam***	Chap. 4-5
7	Bayesian Networks. Hidden Markov Models. Filtering, Smoothing, Prediction.	Chap. 15 **
8-10	Learning. Forms of Learning, Statistical Learning. Neural Networks. Course wrap-up.	Chap. 18-20 ** Chap. 10

*All reading is from [2], except material marked **, from [1].*

For further information, see course website [3].

1.2 Background - Blackjack:

In our simple variant of the game of Blackjack ("Twenty-one", "Vingt-et-un"), the *dealer* deals two cards, face-up, to the *player*, and two cards to the dealer (one face-up, one face-down). The cards are dealt from a *shoe* containing multiple standard bridge decks that have been shuffled together. The player is permitted to request repeated *hits* from the dealer (meaning a card is dealt from the top of the deck), which increases the player's total. The total is computed as the sum of face values of the cards, with face cards counting as 10 points, and the Ace counting as either 1 or 11 points, at the player's discretion. The objective is for the player to get as close to 21 points as possible without surpassing 21, in which case the player "busts". The player elects to *stand* when convinced a higher score is not possible without busting. In this simple version, we ignore such game variations as "pushes", "splits", "doubling down", "surrenders", "insurance", "five card Charlie", etc. that are prevalent in casinos [4].

It is standard for casinos to employ multiple decks shuffled together for blackjack games. Furthermore, it is also standard for the dealer to set a "shuffle point" somewhere in the middle of the deck after shuffling, so that when the deck is depleted to the shuffle point, it is reshuffled. The main purpose of this apparently is to minimize the effects of *card counting* by a skilled player, who by carefully noting what cards have yet to be played will gain an advantage when deciding whether to hit or stand. Card counting is typically easiest and most effective when playing with a single deck and the number of cards remaining is relatively low.

2. ASSIGNMENTS

Students typically have 4-5 assignments in this course, which involve theory, programming, experimentation and number-crunching. A thread of assignments deals with modeling the game of blackjack as a Markov Decision Process (MDP) and finding an optimal solution for hitting or standing depending on the player's current state and the state of the partially-observed dealer state. The students' solutions are used in an end-of-term competition.

2.1 Assignment 1a: Markov Modeling

The first assignment involving blackjack follows at least one preliminary assignment involving review and application of important concepts from probability theory. For this assignment, students model a game of blackjack as a Markov chain. Stochastic processes usually have not been seen previously by the students, and also require some introduction.

We assume the dealer uses multiple decks, with the number of decks unknown or unspecified – so that precise calculation of **prob**($i|k$) i.e., the probability of drawing a card valued i on the next hit, given that k cards with value i have already been drawn, is not possible. Thus, we assume for instance that the probability of drawing an ace is always 1/13, and the probability of being dealt two aces initially is $(1/13)^2 = 1/169$. This is in accordance with the Markov assumption that the probability of making a transition from state m to state n does not depend on how the system reached state m.

The possible *states* a Blackjack player's hand may attain are as follows: the numeric values `4` through `21`, the values `S12` through `S20` (signifying "*soft* 12" through "*soft* 20"), and *bust*. (A *soft n* indicates that the hand contains an ace with remaining cards totaling *n-11*. For instance, `S20` indicates an ace plus cards totaling 9 - if the remaining cards total 19, the state is `20`, not `S20`). We also add a *start* state, which indicates cards have not been dealt yet.

For this exercise, students determine a probability transition matrix **P** for the 29 states described above: p_{ij} is the probability that the state changes from state i to state j on the next transaction. Note that for most states, a transaction is when a *hit* (a single card) is requested. The only exceptions are as follows:

- **prob**(*start*, *j*) is the probability that the system enters state *j* after the *initial two cards* are dealt to the player.
- No hits are allowed when the system reaches the states *bust* or `21` - instead, the system returns to *start*, i.e., **prob**(*bust*, *start*) = **prob**(`21`, *start*) = 1.

The transition matrix $\mathbf{P}\varepsilon\mathbf{R}^{n\times n}$ is a *stochastic matrix*, i.e., all entries are nonnegative, and the elements in each row sum to 1, since when the system makes a transition from any state, it must always end up in one of the states after the transition.

Students are required to justify their calculations, rather than simply giving the solution matrix.

2.2 Assignment 1b: Sampling

An interesting property of a Markov chain is its set of stationary probabilities $\{\pi_i\}$, where π_i represents the long-term probability that the system is in state i when observed randomly, $1 \le i \le n$. These values may be obtained analytically for a given transition matrix **P** according to the matrix equation $\pi\mathbf{P} = \pi$. Since **P** is a stochastic matrix, this system has rank at most *n-1*. Removing one (redundant) equation from the system $(\mathbf{P}^T\text{-I})\pi = 0$ and replacing it with the equation $\underline{1}^T\pi = 1$ (i.e., the π_i sum to 1) allows the system a unique solution if **P** is well-behaved (e.g., no degenerate subcycles), which is the case for **P** in the blackjack example.

The stationary probabilities may be estimated empirically through *sampling*, or simulating the process: counting the number of visits to each state, and determining the proportion of time spent in each state. In a Markov chain with costs or rewards, the long-run expected cost is easily determined from these. One assignment asks the students to compute analytically the stationary probabilities for a small transition matrix and also determine the probabilities experimentally via sampling. The stationary probabilities for the blackjack transition matrix thus far determined are not especially interesting because the matrix represents a policy in which the player repeatedly chooses to *hit* regardless of the state, which inevitably leads to a *bust*.

2.3 Assignment 2: The Dealer's Transition Matrix

The first assignment modeled blackjack as a Markov chain, completing a probability transition matrix for the 29 states identified, which presumed the player accepted a hit regardless of the state. Here we consider the dealer's options. A standard policy in casinos is for the dealer to play according to a fixed published policy. A reason for this is removal of the need for the house to rely on an especially skilled dealer, who would nevertheless likely attract fewer customers than a dealer less skilled. Because a simply stated policy exists ("dealer hits on 16/stands on 17") that is quite effective for the house, this policy is fairly widespread in casinos. This is a reasonable policy for players also, however is not a winning strategy against the dealer, since a player who busts always loses, even if the dealer busts. Note however that a card-counting player can still gain an advantage, and there may be other fixed policies that are less simply stated but are superior to this policy.

Presuming that the dealer hits on 16/stands on 17, the only possible terminal states are **`17`**, **`S17`**, **`18`**, **`S18`**, ... **`21`**, and **`BUST`**. We construct ***P****(DEALER) by m*odifying ***P****(HIT)* so that each of these states is an *absorbing state*, that is, $p_{jj} =1$ for each terminal state j, rather than taking another hit or returning to *start*.

Assuming the dealer's state transitions are described by ***P****(DEALER)*, we can then determine the probabilities that the dealer will finish in each of the terminal states. A useful property of transition matrices is that the elements of $\boldsymbol{P}^k$ (***P*** raised to the k-th power) are the k-step transition probabilities, i.e., $p_{ij}^{k}(DEALER)$ is the probability of making a transition from state i to state j in exactly k steps. Because the terminal states are absorbing states in ***P****(DEALER)*, raising ***P*** to an arbitrarily high power allows us to determine for each state i, the probability that the dealer finishes in each of the terminal states, given that they start in state i. For each of the non-terminal ("transient") states, the probability should be 0.

Students are encouraged to make use of *Matlab* [5] or other suitable software for performing the matrix arithmetic.

2.4 Assignment 3: The Player's Policy

Knowing both the probabilities of a Blackjack dealer eventually reaching each possible terminating state from each possible starting state, and also knowing the dealer's fixed policy of hitting on 16/standing on 17 is sufficient to determine an optimal policy for the player. Armed with this information, students are asked to model a game of blackjack as an MDP with state given by a tuple (*player's state*, *dealer's state*). There should be additional states `WIN`, `LOSE`, `DRAW`. The player's state is one of the possible states examined previously. The dealer's state is simply the value of the card showing. When the player accepts a *hit*, the player's state changes, but the dealer's state does not change. When the player *stands*, the player's state does not change, but the probability of winning can be determined based on the dealer's state and the probability that the dealer will either *bust* or reach a state between 17 and the player's state. The costs and rewards of the `WIN`, `LOSE`, `DRAW` states will determine the optimal decision (*hit* or *stand*) for each state.

Students are required to determine an optimal policy and communicate it via a file with a specific format, which will be used as data for the Blackjack Tournament simulator. The file contains the student's name, userid and a nickname of their own choosing, with remaining lines specifying "hit" or "stand" selections for each possible state.

A Markov Decision Process may be solved by combining the sampling process with the Policy Iteration algorithm. The algorithm is described as follows:

1. Given controls $a_0...a_{k-1}$, transition matrices $\boldsymbol{P}(a_0)$, ..., $\boldsymbol{P}(a_{k-1})$ and cost vectors $c(a_0)$, ..., $c(a_{k-1})$: the goal is to determine a *policy* (choice of controls for each of the n states) which minimizes the long-term expected undiscounted cost of the system.
2. Choose an initial policy $u = (u_0...u_{n-1})$. For instance, choose $u=a_0$. Another possibility is to choose u_i as the control a which minimizes $c_i(a)$ for each i.
3. Form the matrix $\boldsymbol{P}(u)$ and cost vector $c(u)$ corresponding to this policy.
4. Determine stationary probabilities $\pi = (\pi_0...\pi_{n-1})$ using sampling as in part A.
5. Compute the system cost as $g = \pi_0 c_0(u) + ...+ \pi_{n-1} c_{n-1}(u)$
6. Was u the optimal policy? Define $v_i = \pi_i c_i(u)$, $i=0,1,...n-1$ and determine $min_a\{ sum_j(p_{ij}(a)v_j) + c_i(a) \}$. In other words, figure out which controls provide the lowest values for this step. (Notice that this is equivalent to the second option in step 2, assuming $v_i=0$ initially.) If the choices of controls have not changed for any of the states, the answer is optimal and the algorithm terminates.
7. Otherwise, return to step 3.

For extra credit, students may solve the MDP via other algorithms, e.g., Value Iteration or Policy Iteration, either as a discounted or undiscounted MDP.

3. THE BLACKJACK TOURNAMENT

We have created software that permits a tournament to be played between the dealer and up to four players, using a graphical user interface. The system was developed in Visual Studio using C#. Using the students' submitted policy files as data, along with photographs from the class photo list provided by the university, the software chooses students randomly to participate in a round of blackjack. In a round, the cards will be dealt and displayed, so that the dealer's card and players' cards are visible. For each player in turn, their policy ("hit" or "stand", based on my state and dealer's state) is consulted and acted on, and the display is updated accordingly, continuing until either the player busts or a "stand" directive is reached. When all players have completed play, the dealer's hand is played, each player is determined a winner or loser, winnings are updated accordingly, and a new round may begin.

Figure 1 shows a screen dump from a sample game in progress. The dealer's hand originally showed 4, and the players' states were 19, 11, 8 and 5 respectively. Player 1 chose to stand at 19, while each of the others took one hit and then stood. None busted. The dealer stopped at 18, and so only Player 1 won this hand.

4. CONCLUSIONS

An end-of-term project that builds on core course material provides closure for the course. A blackjack competition in which each player is pitted against an anonymous dealer apparently invites less stress for students than player vs. player competitions (e.g., checkers) that we have used in the introductory AI course for teaching heuristics and minimax. The mathematics proves to be challenging, even for advanced students.

Other methods for determining optimal policies for playing blackjack can be employed in this class, using the infrastructure produced through the earlier assignments (transition matrices, student programs). Likewise, some of the standard variations (splitting, doubling down, etc.) may also be included. We intend to adapt this assignment to use of reinforcement learning in a future course offering.

5. REFERENCES

[1] Russell, Stuart J. and Norvig, Peter. Artificial Intelligence: A Modern Approach, 2nd edition. Prentice-Hall, 2003, ISBN: 0-13-790395-2. Text Website: http://aima.cs.berkeley.edu/ .

[2] Sutton, Richard S. and Barto, Andrew G. Reinforcement Learning: An Introduction.MIT Press, 2002, ISBN: 0-262-19398-1. Text Website: http://www.cs.ualberta.ca/~sutton/book/the-book.html .

[3] CS 481 Winter 2009 class website, http://www.cs.drexel.edu/~jpopyack/Courses/AI/Wi09 .

[4] Thorp, Edward O. Beat the Dealer: A Winning Strategy for the Game of Twenty-one, Vintage Books, 1966. ISBN 0394703103.

[5] *Matlab* numerical computing environment, http://www.mathworks.com/

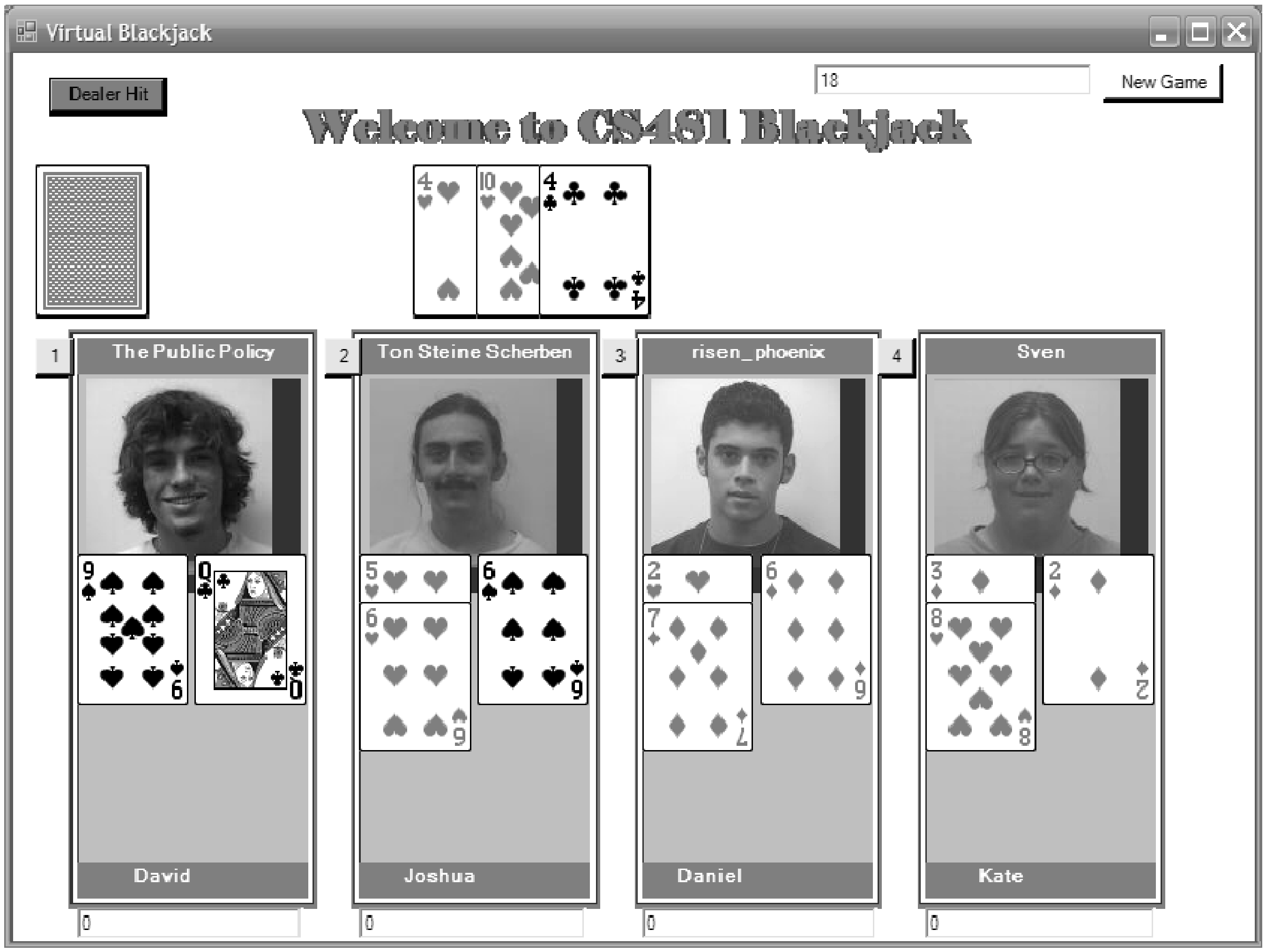

Figure 1. A Game in Progress