

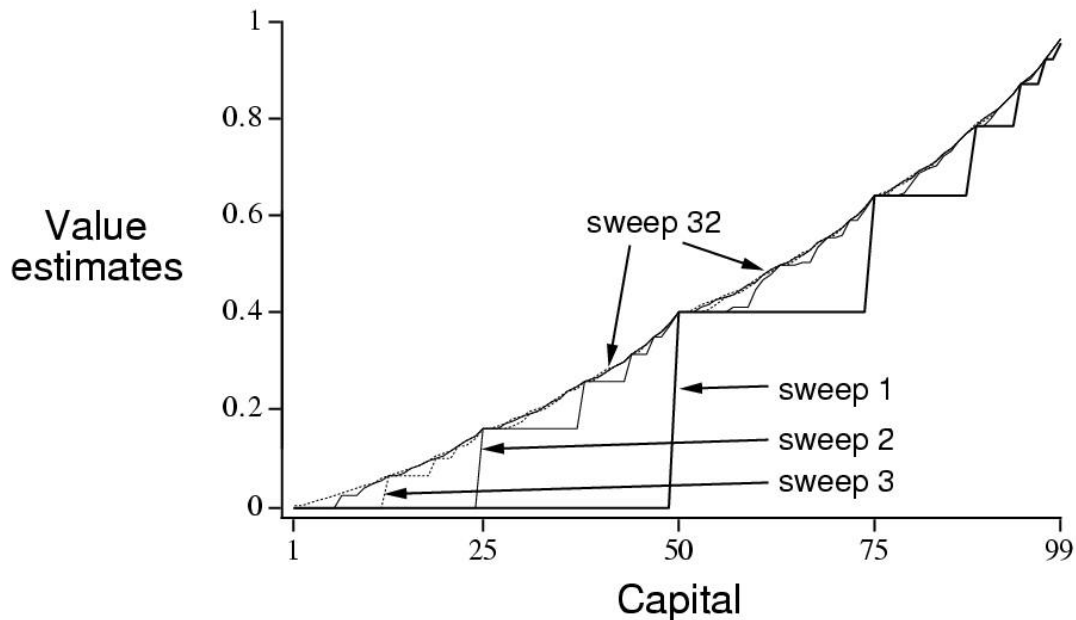
Assignment: Gambler's Ruin

CS 416 Artificial Intelligence
Fall 2004
Due: 5:00 Monday, November 29th

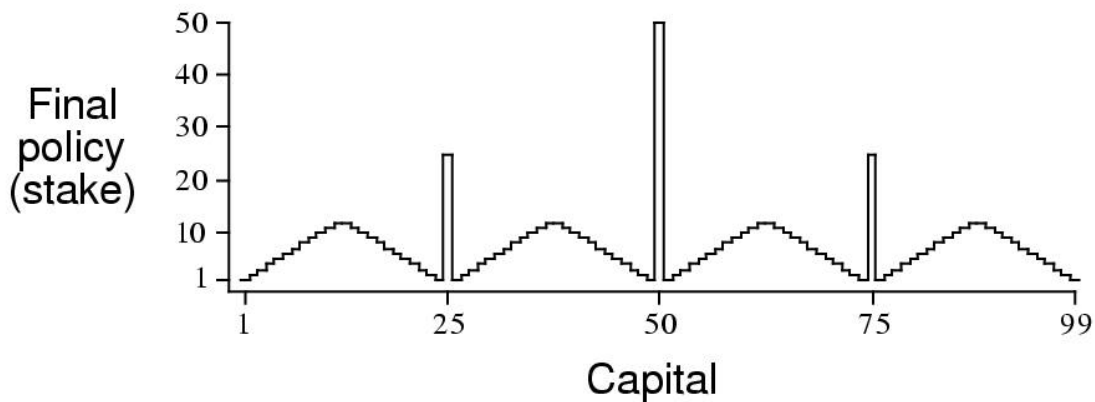
A gambler¹ has the opportunity to make bets on the outcomes of a sequence of coin flips. If the coin comes up heads, then he wins as many dollars as he has staked on that flip, but if it is tails then he loses his stake. The game ends when the gambler wins by reaching his goal of 100 dollars, or loses by running out of money. On each flip, the gambler must decide what portion of his capital to stake, in integer numbers of dollars. This problem can be formulated as a MDP.

The state is the gambler's capital, $s \in \{1, 2, \dots, 99\}$, and the actions are stakes, $a \in \{0, 1, \dots, \min(s, 100 - s)\}$. The reward is **zero** on all transitions except those on which the gambler reaches his goal, when it is **+1**. The state-value function then gives the probability of winning from each state. A policy is a mapping from levels of capital to stakes. The optimal policy maximizes the probability of reaching the goal. Let p denote the probability of the coin coming up heads. If p is known, then the entire problem is known and it can be solved, e.g., by value iteration. Figure 1 shows the change in the value function over successive sweeps of value iteration, and the final policy found, for the case of $p = 0.4$.

Figure 1: The solution to the gamblers problem for $p = 0.4$. The upper graph shows the value function found by successive sweeps of value iteration. The lower graph shows the final policy.



¹ Problem excerpted from *Reinforcement Learning* by Sutton and Barto



Exercise 1: Why does the optimal policy for the gambler's problem have such a curious form? In particular, for capital of 50 it bets it all on one flip, but for capital of 51 it does not. Why is this a good policy?

Exercise 2: Implement value iteration for the gambler's problem and solve it for $p = 0.25$ and $p = 0.55$. In programming, you may find it convenient to introduce two dummy states corresponding to termination with capital of 0 and 100 dollars, giving them values of 0 and 1 respectively. Show your results graphically as in Figure 1. Does γ influence your rate of convergence (monitor the number of iterations before your algorithm terminates)?

Feedback

As with any assignment, we cannot foresee all possible sources of confusion. E-mail cs416@cs.virginia.edu with any questions or comments about the assignment. However, in general such e-mails will *not* be answered directly. Rather, this section of this document will be updated to reflect those questions or comments worth addressing (i.e., those not addressing problems specific to a student's particular block of code, etc.), so check back before e-mailing us to make sure your issue has not already been addressed.

- 🔊 Is there a reward for each state?
 - ➡ The reward for every state except the final state is zero.
- 🔊 How is the program to receive input?
 - ➡ The program should take a single float as a command line argument. That float represents p , the probability of getting heads on a single coin flip.
- 🔊 What should the program output, and in what format?
 - ➡ The output should be to stdout and should contain two lines. The first line should contain 99 floats, separated by spaces, where the first float corresponds to the value estimate for having capital=1, etc. The second line should contain 99 integers, separated by spaces, where the first integer corresponds to the policy (i.e., how much to stake) for having capital=1, etc. Part of your submission includes graphs generated from this output data. You can use the program of your choice (e.g., Excel, MatLab, GIMP) to create these graphs.
- 🔊 How close to the "true" utility should we be before terminating the search?
 - ➡ Continue until the utilities no longer change from time step to time step, or if that is taking too long, choose an epsilon (e.g., 10^{-7}) such that if no change in utility exceeds epsilon the program will halt. You should mention in your README file which option you choose.
- 🔊 What gamma should we use for the graphs we submit?
 - ➡ Use gamma = 0.999999. ($1-10^{-6}$)

- Do we need to keep two arrays for utilities, one for the old values and one for the currently being calculated values, or can we calculate the utilities “in-place”?
 - ↳ Either option is acceptable, but you should mention in your README file which option you choose.
- My optimal policy for $p=0.4$ isn't matching the graph you provided. What might be wrong?
 - ↳ There are several things that might be wrong. However, even if your code is exactly correct, you might be suffering from imprecision in IEEE floating points. Also, you should realize that there are several optimal policies, but the one we provided is the optimal policy that chooses the lowest stake. In order to accommodate for this and the aforementioned imprecision, you should choose the smallest stake that is within 10^{-6} of being optimal. This can be achieved with code similar to:


```

phi=0.000001
max_util=-1
best_stake=-1
for all stakes
  if (calc_util > max_util + phi)
    max_util = calc_util
    best_stake = this_stake
          
```
- Doesn't this imprecision mean that we might not actually be getting the optimal policy?
 - ↳ In theory, yes. However, I've run this code in Mathematica with infinite precision and verified that there are indeed ties, and that the optimal policy with the lowest stake for $p=0.4$ is the one we provided you.

Submission

The source code and a supporting document need to be submitted.

Use the URL: <http://www.cs.virginia.edu/~cs416/submit.html>

to submit your work.

- If you are submitting a Linux/Unix solution
 - The name of the file you submit should be gambler.tgz. As the name suggests, it should be a gzipped tarball. To create a gzipped tarball, type:


```

gtar -cvzf gambler.tgz README.pdf Makefile *.cpp *.h
          
```

 The README document may be a PostScript, PDF, or Word Doc. Because it should include a graph of your results an ascii file will not work
 - The solution needs to be able to compile and run on blue.unix
 - Be sure to include a Makefile so that all I need to type to compile your submission is “make”
 - The name of the executable should be “gambler”
 - The README file should include your answers to the exercises above, including graphs. It should contain your name and any information that may be helpful in the assessment of your submission
- If you are submitting a Microsoft solution:
 - The name of the file you submit should be gambler.zip. As the name suggests, it should be a zip file. You can use WinZip or even regular “zip” on blue.unix (even gtar). If you don't know how to do this, see the Linux/Unix directions above. Be sure to include all files in the zip file that are necessary to compile your solution. Also be sure to include a file named “gambler.doc” (or gambler.{PS|PDF} if you are so inclined) that includes
 - Your answers to the exercises above, including graphs. It should contain your name and any information that may be helpful in the assessment of your submission. It should indicate what version of MS C++ you were using (i.e. v6.0 or .NET).
 - The name of the executable should be gambler.exe

Late Days

Students have five late days to use during the semester. Each late day provides a 24-hour extension.

Honor

Do not use the web to acquire any information related to the algorithmic portion of this assignment. All necessary information can be found in the textbook. You may use the web to look up programming questions, graphing software, tomorrow's weather, etc. Let the instructor or TA know if you have particular questions. Also, do not consult with others in the class about the nature of their solutions or their algorithms. **Unlike the first assignment, you cannot communicate with others in the class about your work.** This is a small assignment and the assistance granted by the TA and the instructor will be carefully allocated to avoid simplifying things too much.