

## Assignment 4

Answer problems 1 and 3. Problems 2 and 4 are extra credit. This assignment is due Wednesday, March 2 at the beginning of lecture. Collaboration is allowed (according to the rules specified in the handout). If you work with a group, be sure to clearly acknowledge the other members of your study group on the first page.

Read 6.1–6.7 in Kleinberg-Tardos.

1. **Card counting.** Bob has a standard deck of 52 playing cards, of which 26 are red and 26 are black. Bob turns over the cards one at a time. Alice starts with 100 dollars. Before each card is revealed, Alice can wager any whole dollar amount (between zero and his current bankroll) on the color of the next card. She gets even odds regardless of the current composition of the deck. What's the most Alice can *guarantee* to end up with?

For example, Alice can guarantee 200 dollars by betting 0 dollars on the first 51 cards, and 100 dollars on the last card (whose color she can deduce by observing the results from the previous 51 cards).

Once you have formulated the appropriate dynamic program, you will need to write a computer program to compute the exact value. You may use any programming language you like. (For reference, my solution is around 20 lines of Java and takes about 20 seconds.) You may collaborate with your study group to write the program jointly. Be sure to turn in a copy of your code.

- † 2. **More card counting.** Alice doesn't want to memorize the complicated dynamic programming strategy from problem 1. Devise a *simple* strategy for Alice that obtains a reasonable guaranteed winnings. Here's an example of a simple strategy (admittedly with an extremely poor guarantee): bet an  $\frac{|r-b|}{r+b}$  fraction of your bankroll, rounded to the nearest dollar, on the suit with the most remaining cards. This is an open-ended problem, and there is no "right" answer as far as I know.

3. **Viterbi path.** Given a hidden Markov model and a sequence of observed events, design an algorithm to find a Viterbi path. The problem (which we define below) was originally developed for error-correcting codes, and is widely used today in communication systems ranging from the Mars Pathfinder to 802.11 wireless LANs. It is now also a standard technique in speech recognition, handwriting analysis, and bioinformatics.

For the purposes of this problem, a *hidden Markov model* is a directed graph  $G = (V, E)$ , where each edge  $(u, v)$  is labeled with a character  $c_{uv}$  from a finite alphabet and a probability  $0 < p_{uv} \leq 1$ . The sum of the edge probabilities leaving each node equals one. The *sequence of observed events* is a string  $s$  over the same alphabet. A directed path is *consistent* with the observed events if its sequence of edge labels equals  $s$ . A *Viterbi path* is a consistent path that maximizes the product of its edge probabilities. Your algorithm should run in  $O(m\ell)$  time and  $O(m + n\ell)$  space, where  $n$  is the number of nodes,  $m$  is the number of edges, and  $\ell$  is the length of the string  $s$ . You may assume that you are working over the binary alphabet (a's and b's).

- † 4. **Viterbi path in linear space.** Your friend Andrew is working for a certain NJ research lab in designing one of those annoying automated phone operator systems with voice recognition. He intends to use a hidden Markov model to solve the voice recognition problem. Unfortunately, there are so many nodes in the graph that when he tries to implement it, he runs out of memory. Please help him design an algorithm to compute a Viterbi path that uses  $O(m\ell \log \ell)$  time, but only  $O(m + \ell)$  space. For credit, your algorithm must output a Viterbi path, not just its probability.