Grad AI. 15-780 Fall, 2006

Homework 3

- *Homework deadline: 10:30am on Nov. 1*
- *Please print your code and hand it in with the hard copy of your homework. Also send a copy of your code by e-mail to both TAs.*

1. **Bayesian Networks [33 pts]**. This problem involves a theoretical analysis of *admissible* Bayesian networks. Recall from lecture that an admissible Bayesian network must be a Directed Acyclic Graph (DAG).

   (a) **[3 pts]** Draw all admissible Bayesian networks with 3 variables A, B, and C.

   (b) **[5 pts]** Group all possible three node Bayesian networks with variables A, B and C into equivalence classes. Networks in the same equivalence class should convey the same independence assumptions. In other words, if two networks are in the same class then the joint probability distribution implied by the first network is the same as the one implied by the second. For example, for networks with two nodes A and B there are 2 classes: The first contains one network (two nodes and no edges) and the second contains two networks: A $\rightarrow$ B and B $\rightarrow$ A. These two networks are equivalent since $P(A)P(B|A) = P(B)P(A|B) = P(A, B)$.

   (c) **[10 pts]** Derive an upper bound on the number of admissible Bayesian networks with n nodes (please explain your answer). Note that smaller upper bounds will gain more credit (i.e. an upper bound of ∞ will get no points).

   (d) **[15 pts]** Derive a lower bound on the number of admissible Bayesian networks with n nodes (please explain your answer). Note that larger lower bounds will gain more credit (i.e. a lower bound of 0 will get no points). Hint: if you can show that it is always possible to create $f(n)$ admissible networks with n nodes then $f(n)$ is a valid lower bound. Consider how many admissible networks can be constructed after an arbitrary ordering is imposed on the variables. Alternatively you may consider an inductive argument by assuming your bound holds for $n-1$ variables and showing that it continues to hold when you add the n'th.

2. **Maximum Likelihood Estimation [23 pts]**. In class we derived the Maximum Likelihood Estimator (MLE) for the single parameter of a Binomial distribution (e.g. the probability that a coin lands heads after observing the outcome of n independent flips of the coin). In this problem we will derive the MLE for the parameters of a multinomial distribution where the variable of interest, X, can take on k values rather than 2.

   (a) **[5 pts]** Given data describing n independent identically distributed observations of X, $\mathbf{d} = \{d_1, \ldots, d_n\}$, each of which can be one of k values, express the likelihood of the data given $k-1$ parameters for the distribution over X. Let n_i represent the number of times X takes on value i in the data.

(b) **[6 pts]** Find the MLE for one of the $k-1$ parameters, θ_j in terms of the other parameters.

(c) **[12 pts]** At this point you should have $k-1$ equations describing MLE's of different parameters. Show how those equations imply that the MLE for a parameter θ_j representing the probability that X takes on value j is equal to $\frac{n_j}{n}$. You may find the following hint useful for this: In order to remove the k'th parameter from the likelihood in part (a) you had to represent it with an equation, $\theta_k = f()$. At this point you may find it helpful to replace all occurrences of $f()$ with θ_k. After replacing $f()$ with θ_k you can substitute all occurrences of each other parameter in $f()$ with its MLE from part (b). This should allow you to solve for the MLE of θ_k, which can then be used to simplify all of the other equations.

3. **Hidden Markov Models [44 pts]**. Many of you may be familiar with the T9 input paradigm commonly found on cell phones for interpreting a series of key presses on the 9 button keypad as text. Typically, each of the keys 2-9 represents about 3 different letters (Figure 1 provides the exact mapping). When the user inputs a series of key presses, such as 3-6-4, the T9 system provides a list of words from its dictionary that potentially match the sequence (e.g. "dog" and "fog" both match the sequence above). In this problem you will build a next generation text messaging cell phone input system, SmarT9, by training an HMM on an English text corpus. For simplicity, we will ask you to build your system only for 5 letter words and we will provide a small corpus on the course website.

Figure 1: A typical phone keypad.

(a) **[5 pts]** Describe the hidden states and observable outputs of an HMM designed to interpret 5 letter words based on 9-digit keypad interactions as described above.

(b) **[3 pts]** In your model from part (a) describe the probability that a state emits each particular output (e.g. $P(O_t = o_i \mid S_t = s_t)$).

(c) **[10 pts]** Write a program that reads in a list of english words and constructs a single table indicating the probability that state $t+1$ will have each of its possible values given the value of state t (assume that the transition probability is not dependent on the value of t). The value at the i'th row and j'th column of your table should be the probability that $P(S_{t+1} = s_j \mid S_t = s_i)$. Read in the list of words given in the support code and **provide a print out of the table constructed by your program**. You may find the following Matlab facts and functions helpful:

- **textscan()**: this command scans a file and returns cells with its contents, view the Matlab help description for more information.
- Strings in Matlab are treated as vectors of characters. A string S = "dog" can be indexed to access its individual characters as follows, S(1) = 'd', S(2) = 'o', S(3) = 'g'.
- **char2number()**: this function is provided in the support code, it takes as input a character "a-z" (not case sensitive) and returns its position in the alphabet 1-26. If the input is not a character "a-z" the function returns -1.
- **number2char()**: this function is provided in the support code, it is the inverse of char2number(). It returns the character corresponding the letter in the position given as input (1-26). Any other input results in a return value of -1.

(d) **[6 pts]** Write a program that reads in a list of english words and learns the probability that the initial state takes each of its possible values, $P(S_1 = s_i)$. Provide the output of your program when run on the list of words provided with the support code.

(e) **[10 pts]** Write a program using the Hidden Markov Model and parameters learned from the list of words provided with the support code (parts a-d) to predict the probability that state k takes each of its possible values given observations 1 through t with $1 \leq t \leq k \leq 5$ (denoted $P(S_k = s_i \mid O_1, \ldots, O_t)$).

(f) **[10 pts]** Use a modification of the Viterbi algorithm to implement the SmarT9 prototype: write a program that predicts the most likely 5 letters after of observing a sequence of t digits between 2 and 9 ($0 \leq t \leq 5$).