6.874/7.90 Computational Functional Genomics
Spring 2005

Handout 4
April 26

MASSACHVSETTS INSTITVTE OF TECHNOLOGY

Problem Set 4
Due May 3

The second problem in this problem set requires the use of the "Bayes Net Toolbox," a freely-available set of MATLAB functions written by Kevin Murphy. Your first step should be to download and install this toolkit for your copy of MATLAB – there should be a link to the BNT site, and to its (very useful!) tutorial, on the course website.

The problems require perhaps 10-20 total lines of relatively-simple MATLAB code; comments and hints are given in the provided source code. All networks that we learn and work with in this problem set will be composed of discrete-valued nodes.

Problem 1: Quantization

We have provided an almost-complete implementation of the quantization procedure described in the lecture notes on Bayesian Networks. For this problem, we ask you to complete the code for this quantization algorithm (you only need to fill in the method for calculating the *mutual information* between two variables), and then answer several questions about its use and performance.

We will quickly explain the basic structure of the pre-written code, and how to run it: the basic input will be an "expression matrix." This is a $M \times N$ matrix of real numbers, where M is the number of genes and N is the number of experiments. The entry (i, j) of the matrix is the expression value of gene i in experiment j. If `expr` is a particular expression matrix, then (once you've filled in the missing code) you can run the complete quantization algorithm with the command `quantize(expr)` — you should check the comments in this function for an explanation of the various values that are returned.

The quantization algorithm is implemented by converting the $M \times N$ expression matrix into N separate "quantization matrices," one for each experiment. You'll recall from the notes that, at any step in the quantization algorithm all the experiments have the same number of levels — call this number K. Each quantization matrix is a binary matrix of dimension $M \times K$: the element (i, k) is 1 if gene i falls within expression-level k, and 0 otherwise. You should familiarize yourself with how these quantization matrices work, because the function whose code is missing takes two of them as arguments.

The only code that is missing for this problem is the code in `mutual_info(`A_1`, `A_2`)`. Here, A_1 and A_2 represent two different quantization matrices, both of dimension $M \times K$. You can imagine such a quantization as a random variable, with values in $[1, K]$.

Recall that if a random variable X has a set of possible values $V(X)$, and a probability distribution (over those values) $P(X)$, then the entropy associated with that variable is

$$H(X) = -\sum_{x \in V(X)} P(X = x) \cdot \log P(X = x) \tag{1}$$

Furthermore, remember from the notes that the mutual information between two variables X and Y is

$$I(X, Y) = H(X) + H(Y) - H(X, Y) \tag{2}$$

The total mutual information, TMI, is therefore the sum of $I(X, Y)$ over all distinct pairs of X and Y. We have already written the framework for doing all this iterating and testing for you; the only coding *you* need to do for this problem is the implementation of $I(X, Y)$ in `mutual_info()`, which takes its arguments in the form of quantization matrices. As the hints in that file suggest, this should take no more than 7-9 lines of relatively-simple MATLAB code.

For this problem, please perform the following tasks, or answer the following questions, in written form.

a. Please complete the function `mutual_info`, and turn in a print-out of your code.

b. Please use the MATLAB `rand()` function to generate a completely random (uniform) expression matrix of 20 genes and 10 experiments. Run the quantization algorithm on this expression array, and cut and paste the output into your write-up. Please also include the graph of TMI vs. L (as in the lecture notes) — this should be fairly easy to generate, you just need to run a line like `plot(tmi(:, 1), tmi(:, 2), 'bo-')` after running `quantize`.

c. We have also provided some pre-generated data, in the file `ps4_prob1_data.mat`. Load this expression matrix, and run the `quantize` prodcedure on it as well. Include the same two results (output and graph) with your write-up.

d. Examine the code in the functions `quantize()`, and its subsidiary function `reduce_step()`. Notice that we choose the level that *maximizes* $TMI(L)$ in the former function, but we choose to minimize $TMI(L)$ over possible choices for which quantization levels to coalesce in the latter function. Why do we minimize in one and maximize in the other? (Remember that the code is a direct translation of the lecture notes, so you might refer to those if you're having difficulty understanding the details of the MATLAB code). Assume that $TMI(L) - TMI(L-1)$ is a rough approximation to the first derivative of the TMI function with respect to L; what phenomenon does the maximization of that difference in `quantize()` capture?

Problem 2: Parameter Learning and Model Comparison

This problem is an introduction to two of the most basic features of the Bayes Net Toolkit. We will learn how to do *parameter learning* (in the presence of complete data) and *inference* in a simple Bayes Net built with the toolkit. You should begin by downloading and installing the BNT. If you have any problems, the TA will be happy to walk you through the installation procedure.

We have also provided pre-generated data for use in this problem. The data is located in the file `ps4_prob2_data.mat`; that files loads a matrix whose structure is graphically described in Fig. 1.

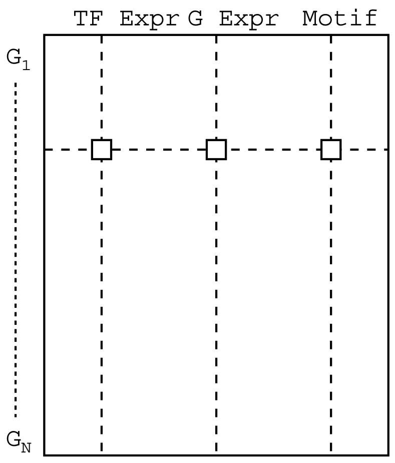

Figure 1: Diagram of Data for Problem 2

The matrix has N rows, corresponding to N different observations. It also has three columns, corresponding to three different variables whose value is given in each observation.

This is a cleaned-up dataset that will allow us to explore a very simple method of transcriptional regulation: that of genes which (we might believe) are controlled by a single common transcription factor. Each row corresponds to such a gene; the `TF Expr` variable gives a value for the expression of the transcription factor that controls the gene, `G Expr` gives the expression value of the gene itself, and `Motif` is a binary indicator variable denoting whether or not the gene *does* or *does not* contain a binding motif for that transcription factor in its upstream regulatory region.

There's no need to worry about quantization in this problem. We've already quantized the expression levels for you; each expression value should be an integer between 1 and 5 inclusive.

We ask you to answer the following questions:

a. Please draw (and turn in) the structure of a Bayes Net on these three nodes that captures the following conditional independence assertion: *`TFExpr` and `Motif` are asserted to be independent, unless the value of `GExpr` is known.* (Your answer should be a simple, directed, acyclic graph on the three variables.)

b. Next, please create the BNT object required to represent this graph, along with finite CPT's for the conditional probability at each node. Load the data, and use the method `learn_params()` from the BNT to learn the parameters of the model. Please include a copy of the code you used to do this in your writeup.

   You can use the function we have provided, `print_cpt()` to print out the parameters of the CPT for each node in the network; use this method to print out all three CPTs, and copy the results into your writeup as well.

   Hint: This shouldn't be too much work – not more than 5-8 lines to set up and learn the network. Most of this can be adapted in a straightforward way from the BNT tutorial that is online.

c. Finally, we give a table of data-points below: each contains a value for `TFExpr` and `GExpr`, but *not* for the `Motif` variable. Use the junction tree inference engine in the BNT (implemented as `jtree_inf_engine`) to evaluate the posterior probability of the "has a Motif" event (i.e., of `Motif` == 2) on this data.

   Do this for each of the data-points given, and include both the code you used to do this inference and the posterior probability of motif occurrence for each point in your writeup.

TFExpr	GExpr	$P($`Motif` $== 2)$
2	3	?
2	4	?
5	5	?
5	1	?

   Hint: Again, while this may sound *complicated, but one of the points of this problem set is that the BNT makes such an inference (in a small network) very easy to do. Come talk to the TA if you need any help.*