

DATA MINING W4240

HOMEWORK 2 QUESTIONS

February 7, 2012

Professor: Frank Wood

Preliminary Instructions

1. Download the skeleton code for the assignment at
<http://www.stat.columbia.edu/~fwood/w4240/Homework/index.html>
2. Unzip the downloaded material in an appropriate folder, something like w4240/hw2/
3. Open MATLAB and navigate to the folder containing the downloaded material

For this homework the code is quite a bit more complex. Graphical data structures are difficult to deal with unless you use programming techniques such as recursion and an object oriented paradigm. You will not have to program these things yourself, but you will be much more likely to complete the project correctly if you understand the concepts. Recursion is simply when a function calls itself. When writing general functions to operate on graphs the concept is useful if you want to do an operation, such as message passing for every node in the graph.

Object oriented programming is a paradigm where you write code to define generic objects and then “instantiate” instances of those objects to perform a task. Generic object definitions are called classes. For instance, you might write code to define a variable node and then create a graphical model by instantiating the appropriate number of them. Objects usually have fields and methods associated with them. Fields are properties, such as if a node has been observed or not or the messages a node has received. Methods are functions associated with the object which can act on the properties of the object and can take outside arguments. For example, it would be useful to have a method which can act on the internal properties of the object and return the outgoing message to a particular factor node. The node will have as properties the most recent messages it has received and can then calculate the correct message to return.

I expect this HW to be quite a difficult task due to the programming component. Your grades will take into account the difficulty of the task. Work first on part I and then concentrate on part II.

1. (65 points)

The first part of the homework consists of implementing key parts of the sum-product algorithm for inference in a discrete graphical model. Only one type of object is used in this code base, namely the **node** object. The node class file does not need to be edited, but you should look at it to understand its properties and methods. The code is at `explicit_algorithm/@node/node.m`.

Every variable is binary and so when passing messages the first element of the column vector should be proportional to the probability of TRUE and the second number proportional to the probability of FALSE. The network you are implementing and the appropriate probability tables can be found in Figure 1. The files you are responsible for filling out are :

1. `explicit_algorithm/main_alarm_network.m`
2. `explicit_algorithm/get_marginal_distribution.m`
3. `explicit_algorithm/get_message_fn aj.m`
4. `explicit_algorithm/get_message_fn am.m`
5. `explicit_algorithm/get_message_fn b.m`
6. `explicit_algorithm/get_message_fn bea.m`
7. `explicit_algorithm/get_message_fn e.m`
8. `explicit_algorithm/get_message_fn vn.m`

The graphical structure is set up in `main_alarm_network.m`. At the bottom of that file you can see an example of inference in the model with no values set. **Please edit this file to calculate the probability there was a burglary given that John called.** Finally, since the model is so small, ensure that you are calculating the correct conditional distributions by explicitly calculating the probability of every configuration of the variables and then summing out the appropriate variables.

2. (35 points) The second part of the homework consists of implementing the key parts of the message passing algorithm in a more scalable object oriented setting. The graph is set up for the large ALARM network in Figure 2 . The ALARM network is a network for monitoring patients in intensive care ¹. The graph is set up for you in the m-file

¹<http://compbio.cs.huji.ac.il/Repository/Datasets/alarm/alarm.htm>

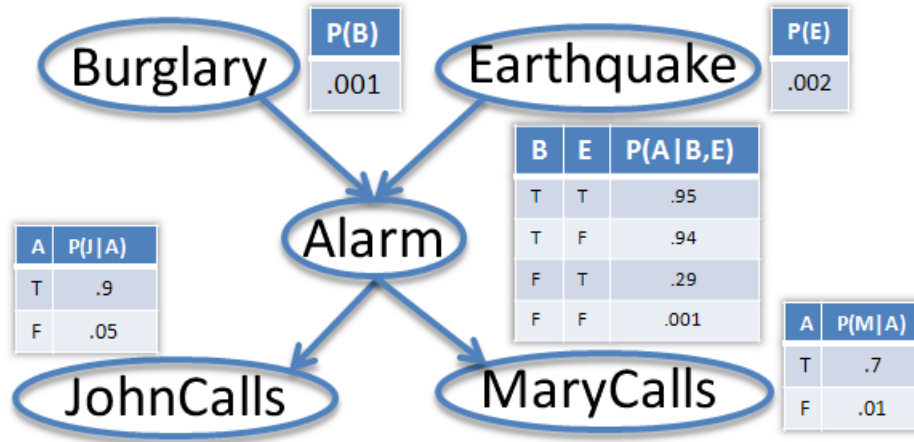


Figure 1: First Bayes Net Example

object_oriented_algorithm/main_big_alarm_net.m. This code base makes extensive use of objects. An object called node is defined in the folder @node and two specific nodes are then refined, namely @factor_node and @variable_node. The refined objects inherit properties and methods from the general @node object unless they are specifically overwritten. You will need to familiarize yourself with all three objects to be able to do the project. There is also one other object called factor for this project. A factor object is an object to hold the factor matrix, basically the probability tables. Finally, you are provided a utility function called multilinear_product.m which you may find useful. You should read the comments at the top of the function which describes its functionality. The last thing you need to know is that if f is the factor matrix inside the factor object `fn_a_b_c` then $f(1,3,2)$ is the factor value of $f(a = 1, b = 3, c = 2)$. Inside each factor node object, the list of neighboring objects is listed in exactly the same way for all factor nodes. That is, for `fn_a_b_c` the neighboring nodes list will be $\{vn_a, vn_b, vn_c\}$.

The first task in this section of the HW will be to understand the code. Then, you will need to implement the method `getMessage(obj, to_unid)` in both the `factor_node` and `variable_node` objects. This will take some creativity and may require you to use a recursive function or make use of the `eval` function in MATLAB. You also need to implement the method `getMarginalDistribution(obj)` in the `variable_node` class definition. Finally, set up the main file to calculate the probability of KinkedTube given SaO2 is set to 1, BP is set to 1, ArtCO2 is set to 1, Press is set to 2, and ExpCO2 is set to 1.

Because the network is actually loopy, we have to perform loopy inference here, but it is not critical that you understand how this is implemented. At the bottom of the file ob-

ject_oriented_algorithm/main_big_alarm_net.m you can see an example of inference in the model after setting some specific values.

3. (20 points) The last problem is only required for students enrolled in **G6240**. You are required to fill out and submit the MATLAB file `object_oriented_algorithm/main_image_denoise.m`. The instruction to fill out this file are included in the file as comments. The task is to work with the Markov random field model discussed for the task of image denoising, but to perform inference using loopy belief propagation. The result of running a loopy BP algorithm will give estimates of the marginal probability distributions for each true pixel value. You can then create a cleaned image by choosing the most likely value for each pixel from the appropriate marginal distribution. In this code you will need to write code to create the graph structure and then use the objects defined in part 2 to do inference. As a quick side note, because of the structure of the graph MATLAB has trouble if you try to clear the objects since they are all interconnected, so don't try to delete the objects once you have created them, just kill the MATLAB session. There may be a way to get around this, but I have not figured one out.

Submitting your HW

You must complete this HW assignment on your own, you are not permitted to work with any one else on the completion of this task. Your grade will reflect your ability to implement a working version of the procedure. Submitted code must run on my machine in less than 3 minutes. Grading will be automated and the submitted files will be run, therefore to submit the HW you will need to follow the following directions exactly.

1. Send an email to Stat.W4240@gmail.com
2. Attach your updated MATLAB files
 - (a) `explicit_algorithm/main_alarm_network.m`
 - (b) `explicit_algorithm/get_marginal_distribution.m`
 - (c) `explicit_algorithm/get_message_fn aj.m`
 - (d) `explicit_algorithm/get_message_fn am.m`
 - (e) `explicit_algorithm/get_message_fn b.m`
 - (f) `explicit_algorithm/get_message_fn bea.m`
 - (g) `explicit_algorithm/get_message_fn e.m`
 - (h) `explicit_algorithm/get_message_fn vn.m`
 - (i) `object_oriented_algorithm/main_big_alarm_net.m`

- (j) `object_oriented_algorithm/@factor_node/factor_node.m`
- (k) `object_oriented_algorithm/@variable_node/variable_node.m`
- (l) (for students enrolled in **G6240**) `object_oriented_algorithm/main_image_denoise.m`

It is imperative that the names be exactly as described here. There should be no folders attached, only raw .m files. You may not attach other MATLAB code files.

3. The subject will be exactly your Columbia UNI followed by a colon followed by hw2. For example, if the professor were submitting this homework the subject would read **fw2196:hw2**
4. If you submit hw more than once, later files will overwrite earlier files.