4F13 Machine Learning: Coursework #1: Message Passing in Factor Graphs

Zoubin Ghahramani & Carl Edward Rasmussen

Due: 4pm Thursday Feb 4th, 2010 to Rachel Fogg, room BNO-37

In this assignment you will implement inference in a family of binary factor graphs. Below are some hints about the data structures which you can use for your implementation; this is not necessarily the most efficient implementation possible, but sufficient for our purposes. We assume that all variables, $x_1, \ldots x_N$, are binary, taking on values 0 and 1. In order to simplify the problem we will also assume that each factor connects to not more than two variables.

Most of the questions in the assignment will concern one particular graph, described below. However, the code which you are asked to write should be applicable in general to singly connected graphs with binary units and at most two variables connected to each factor. The implementation should be written in Matlab (or Octave). Hand in compact but commented printouts of the code and results of running it on the questions below. Your solution should not exceed 5 pages.

The number of variables in the graph is $N = 6$ and the number of factors is $K = 5$. The topology of the graph is given by the binary graph matrix G of size N by K, each element indicating which variables are connected to which factors. The graph matrix is:

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

There is a 2×2 table representing each of the K factors; these are collected in the 3 dimensional array F with the entries which result in running the following Matlab script:

```
F = ones(K,2,2);
F(1,2,2) = 5; F(2,1,2) = 0.5; F(3,1,1) = 0; F(4,1,1) = 2;
```

Here, `F(j,p,q)` is the value of factor `j` when the variable with the smaller numerical index is set to `p` − 1 and the variable with the larger numerical index is set to `q` − 1.

a) 10% : Draw the factor graph for G. Is it singly connected?

b) 50% : Write a function `computeMarginals` which implements factor graph propagation to compute the marginal probabilities of all variables in the graph, updating messages from variables to factors and from factors to variables. The function should be called in the following way:

```
B = computeMarginals(G, F);
```

where G is the graph matrix and F is the factor array, and B is the returned $N \times 2$ matrix of marginal probabilities (e.g. `B(3,1)`=$P(x_3 = 0)$ and `B(3,2)`=$P(x_3 = 1)$). Inside the function, you will need data structures to represent the messages from variables to factors, `VF`, and from factors to variables, `FV`, with the following sizes

```
VF = ones(N, K, 2); FV = ones(K, N, 2);
```

and you will also need binary matrices `Vsent` and `Fsent` of sizes

```
Vsent = zeros(N, K); Fsent = zeros(K, N);
```

to remember which messages have been sent (e.g. `Vsent(2,5)` means variable 2 has sent a message to factor 5). Run your function on `G` and `F` and report the answer `B`.

c) 20% : If you are interested in only knowing the marginal probability of one of the variables, describe how you would modify your algorithm to be more efficient by avoiding sending unnecessary messages (you do not have to implement this modification).

d) 20% : Write a function `bruteForce` which enumerates all 2^N configurations of the variables and returns the marginals

```
B = bruteForce(G, F);
```

Report `B`, compare to the result of `computeMarginals`, and discuss.

e) 0% (optional) : Consider the problem of computing the marginals when conditioning on observed values of one or more of the variables. Describe a way you could do this by adding additional factors to the original graph and using the function `computeMarginals`.

f) 0% (optional) : Use your idea from the question above to compute $p(x_2|x_1 = 0, x_4 = 1)$ and report your result.