APMLA: assignment 2 Block II

Caterina De Bacco and Martina Contisciani

Clarifications

- The parameter *alpha* in the code inside the notebook is not needed, keep it as assigned per default (we introduced it to make initialization more flexible).
- The BP equations should be derived for the question (a), i.e. the uniform probability over the matching size (NOT for the question b)).
- You have the freedom to pick the values of the variables σ_{ij}, i.e. some discrete values at your choice. Of course, then the compatibility function changes based on this choice. Depending on what you choose, the Hint of question b) may not be needed.

Exercise 2: random matching problem

Matching is another classical problem of graph theory. It is related to a dimer problem in statistical physics. Given a (unweighted, undirected) graph $G(V,E)$ a matching $M \subseteq E$ is defined as a subset of edges such that if $(ij) \in M$ then no other edge that contains node i or j can be in M. In other words a matching is a subset of edges such that no two edges of the set share a node.

Fill up the *jupyter* notebook uploaded on github with the skeleton of a code to solve this assignment.

(a) Write a probability distribution that is uniform over all matchings on a given graph. *Hint*: consider binary random variables σ_{ij} on the network edges.

(b) Write a probability distribution that that gives a larger weight to larger matchings, where the size of a matching is simply $|M|$. *Hint*: you need to write $|M|$ as a function of $|E|$ and $\sum_{(ij)\in E} \sigma_{ij}$ for $\sigma_{ij} \in M$.

(c) Draw a factor graph corresponding to it for the example of a graph with 6 nodes and edges $E = \{(1,2),(1,3),(2,3),(1,4),(2,5),(3,6)\}$. You can add it to the notebook as a figure or using

(d) Using BP to model marginals of the matching assignment, denote as:

- $\nu^{(ij)\to i}_{\sigma_{ij}}$ the messages from *variable* node (ij) to *function* node i.
- $\hat{\nu}^{i\to(ij)}_{\sigma_{ij}}$ the message from *function* node i to *variable* node (ij).

Note that they are both functions of the state σ_{ij} of variable node (ij).
Hint: here variable nodes (circles) correspond to an edge $e = (i,j) \in E$ and factor nodes (squares) correspond to nodes $i \in V$ in the *original* network $G(V,E)$.
Write BP equations for this model.
Hint: write an explicit derivation for *each of the two possible* values of σ_{ij}. Notice that we are implementing *hard* constraints. This means that for each value of σ_{ij}, there can be only a limited number of *allowed* configurations of $\{\sigma_{ik}\}$, for $k \in \partial i \setminus j$. In principle there should be 2^{k_i-1} of them, but in practice, many are not allowed once σ_{ij} is fixed; we remain with k_i allowed configurations (for one of the two values of σ_{ij}) and only one configuration allowed for the other

value of σ_{ij}. Do not consider the configuration where $(ij) \notin M$ AND $(ik) \notin M$, $\forall k \in \partial i \setminus j$, for a total of $k_i - 1$ allowed configuration to be considered in the BP implementation.

(e) Write the equation for the one-point marginal $P(\sigma_i)$ and the two-point marginal $P(\sigma_i, \sigma_j)$ obtained from BP.

(f) Fill the *Jupyter* notebook that you find on github by answering to the following points.

(i) Implement the BP equations for Erdős-Rényi graphs of $N = 200$ nodes as derived above.

(ii) Check how the behavior depends on the order of update for the "random" initialization, i.e. compare what happens if you update all messages at once or sequentially at random one by one.

(iii) Initialize BP in 3 different ways (recall that they are a function of random variables $\sigma_{ij} \in \{0, 1\}$):

- "random" : each of the two possible values of σ_{ij} is selected uniformly at random in $(0, 1)$;
- "all-negative" : close to the point $(1, 0)$;
- "all-positive" : close to the point $(0, 1)$;

and iterate the equations until convergence for each of these three. Define converge as the time when the $err < \tilde{\epsilon}$ (as defined in tutorial 10).
Plot the behavior of err as a function of iteration time for the 3 different initializations (one plot with 3 curves).
Use the update (parallel or sequential at random) that converges faster.

(iv) For parameters where the update converges, plot the convergence time as a function of the average degree $c = [2, \dots, 7]$. Do this on as large graphs as is feasible with your code.

(v) Assign one color value $\in \{0, 1\}$, 1 if the edge belong to the matching, 0 otherwise, to each edge at convergence, based on the argmax of the suitable marginals. Compute the fraction of violations of the matching requirement you get over $N_{real} = 10$ initializations (choose one initialization type at your choice) of the graph and plot them as a function of c.

(vi) Plot also the fraction of edges in the set M as a function of c and for the 3 different initializations on the same plot.

(vii) Comment on how the behavior depends on the initialization.