Advanced Probabilistic Machine Learning and Applications

Caterina De Bacco and Martina Contisciani

1 Tutorial 10: Belief Propagation

Exercise 1: implementing BP for graph coloring

In tutorial 9 we saw the coloring problem of graph theory.
Given a (unweighted, undirected) graph $G(V, E)$ a coloring $M \subseteq E$ is an assignment of labels, called colors, to the vertices of a graph such that no two adjacent vertices share the same color.

We saw the probability distribution is:

$$P(\mathbf{s}) = \frac{1}{Z} \prod_{(ij)\in \tilde{F}} \psi_{ij}(s_i, s_j) = \frac{1}{Z} \prod_{(ij)\in E} \mathrm{e}^{-\beta\, \mathbb{I}(s_i = s_j)} \quad ,$$

where we used the soft constraint $\psi_{ij}(s_i, s_j) = \mathrm{e}^{-\beta\, \mathbb{I}(s_i=s_j)}$, and then we let $\beta \to \infty$.
This resulted in BP updates:

$$\chi_{s_j}^{j\to(ij)} = \frac{1}{\tilde{Z}^{j\to i}} \prod_{k\in\partial j\setminus i} \left[1 - \left(1-\mathrm{e}^{-\beta}\right) \chi_{s_j}^{k\to(kj)}\right] \quad , \tag{1}$$

where we can interpret:

- $1 - \left(1-\mathrm{e}^{-\beta}\right) \chi_{s_j}^{k\to(kj)}$ as the probability that neighbor k is fine with j taking color s_j.
- $\prod_{k\in\partial j\setminus i} \left[1 - \left(1-\mathrm{e}^{-\beta}\right) \chi_{s_j}^{k\to(kj)}\right]$ as the probability that all the neighbors are fine that j taking color s_j (if i was excluded and (ij) erased).

Notice that we only distinguish whether a variable takes the same color of its neighbor or not, but we do not distinguish what color among the different ones.

The one-point and two-point marginals are:

$$\chi_{s_j} = \frac{1}{Z^{(i)}} \prod_{k\in\partial j} \left[1 - \left(1-\mathrm{e}^{-\beta}\right) \chi_{s_j}^{k\to(kj)}\right] \tag{2}$$

$$\chi_{s_i, s_j} = \frac{1}{Z^{(ij)}} \psi_{ij}(s_i, s_j) \chi_{s_j}^{j\to(ij)}\, \chi_{s_i}^{i\to(ij)} \tag{3}$$

which are valid at convergence.

In this tutorial we will code the belief propagation for Erdős-Rényi graphs coloring for $\beta \to \infty$.

(a) Initialize BP close to be uniform fixed point, i.e. $1/q + \epsilon_s^{j\to i}$ and iterate the equations until convergence. Define converge as the time when the

$$err < \tilde{\epsilon}$$

where $err = \frac{1}{2q|E|} \sum_{(ij)\in E} \sum_s |\left(\chi_s^{i\to(ij)}(t+1) - \chi_s^{i\to(ij)}(t)\right)|$, with suitably chosen small $\tilde{\epsilon}$.

(b) Check how the behavior depends on the order of update, i.e. compare what happens if you update all messages at once or randomly one by one.

(c) For parameters where the update converges, plot the convergence time as a function of the average degree c. Do this on as large graphs as is feasible with your code.

(d) Assign one color to each node at convergence, based on the argmax of the one-point marginals. Count how many violations you get over $N_{real} = 5$ random initializations of the graph and plot them as a function of c. Repeat and plot the number of violations as a function of the number of colors q.

(e) Check how the behavior depends on the initialization. What if initial messages are random? What if they are all points towards the first color?