CS 683: Belief Propagation and VC Dimension

Shaomei Wu, June Andrews
shaomei@gmail.com, astuteAjax@gmail.com

April 25, 2008

1 MWM with Belief Propagation: A Summary

[1] Before going into the new topic of VC dimension, let's summarize our discussion about the belief propagation algorithm with the maximal matching example.

Prove that the solution of belief propagation algorithm converges to the maximum weight matching (MWM) for given complete bipartite graph with n nodes at each side.

Assume that the MWM is unique, and the weight difference between MWM and the second maximal weight match is ϵ, where $\epsilon > 0$. Let w_{max} be the maximal weight for a single edge in the bipartite graph.

The outline of proof:

1. Construct a k levels tree-shape factor graph to represent the given MWM problem.
2. Run belief propagation algorithm over the factor graph and converge to an **optimal solution** Λ^{opt}.
3. Assume that Λ^{opt} is not identical as the **real MWM** Π in the bipartite graph. Without loss of generality, let's further assume that one of the incorrect edges in Λ^{opt} is at level 1 of tree. Then, there must be an **alternative path** P consisting of two segments: one from the level 1 Λ^{opt} **node** all the way down to a leaf variable node, another from the level 1 Π **node** to another leaf variable node. By saying **alternative path**, we mean that the edges of P come from either Λ^{opt} or Π, alternatively.

[1] This part of note is generated with great reference to the original paper by Mohsen Bayati, Devavrat Shah, and Mayank Sharma, *"Max-Product for Maximum Weight Matching: Convergence, Correctness, and LP Duality"*, in IEEE Transactions on Information Theory, VOL.54, NO.3, March 2008.

4. P can be **decomposed** into at least $\frac{k}{n}$ **cycles** in the bipartite graph after adding two extra edges.

5. Since each cycle is a part of P, it is a chain of alternative edges from Λ^{opt} and Π.

   In each cycle, it suffices that the weight of edges from Π is greater than the weight of edges from Λ^{opt} by at least ϵ:

$$\sum_{C_{i,j}\in\Pi, C_{i,j}\in Cycle_l} Weight(C_{i,j}) - \sum_{C_{i,j}\in\Lambda^{opt}, C_{i,j}\in Cycle_l} Weight(C_{i,j}) > \epsilon. \tag{1}$$

6. By replacing edges from Λ^{opt} with edges from Π in each cycle, and removing the two extra edges, we get a new solution for the factor graph as Λ. Because the difference between Λ and Λ^{opt} comes from the substitution of Λ^{opt} edges with Π edges in all cycles (and also the two edges adding to P to form full cycles), assume the number of cycle is m, we then have:

$$Weight(\Lambda) - Weight(\Lambda^{opt}) > m\epsilon - 2w_{max} > \frac{k}{n}\epsilon - 2w_{max}. \tag{2}$$

   As Λ^{opt} is the optimum given by the belief propagation at level k, and the message passed in this factor graph has been designed such that the optimal solution will achieve maximal sum of weights of selected edges, with respect to all the constraints, there must be

$$Weight(\Lambda^{opt}) - Weight(\Lambda) > 0$$

   , together with the inequality (2), we have:

$$0 > Weight(\Lambda) - Weight(\Lambda^{opt}) > \frac{k}{n}\epsilon - 2w_{max}. \tag{3}$$

7. When k goes to infinity, $(\frac{k}{n}\epsilon - 2w_{max})$ will be greater than zero, hence contracdicts to the initial assumption that $\Lambda^{opt} \neq \Pi$.

Now we will show some more details of above proof with an example.

First assume that the original bipartite is given as in figure 1, where the edges in the maximal matching are solid and the other edges are dotted.

Then we construct a 3-level factor graph as in figure 2. This graph can also be called a computational tree.

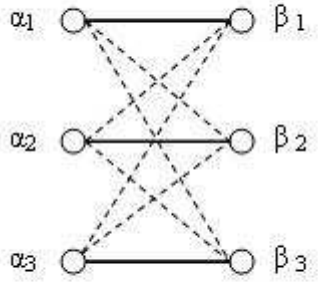

Figure 1: Bipartite graph

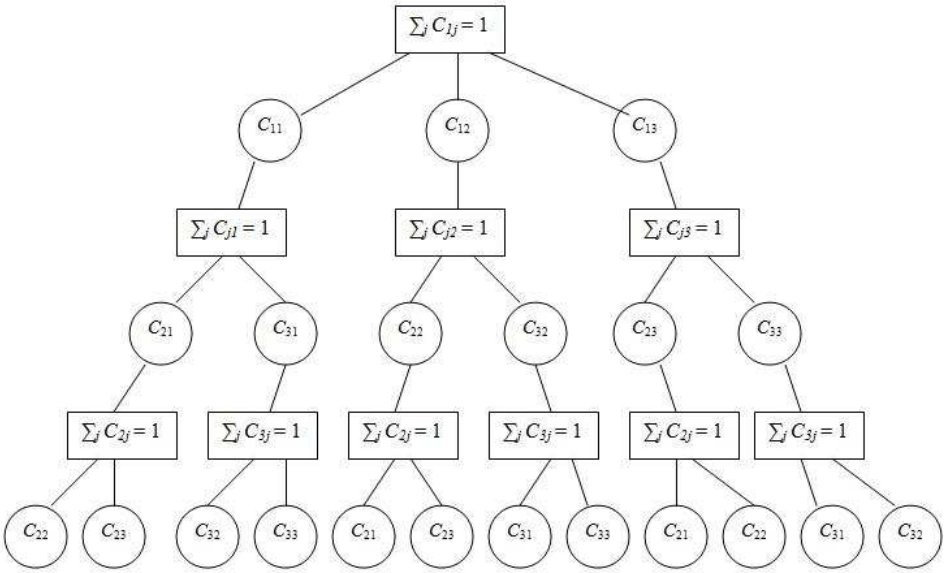

Figure 2: The factor graph (computational tree) for 2×3 bipartite graph

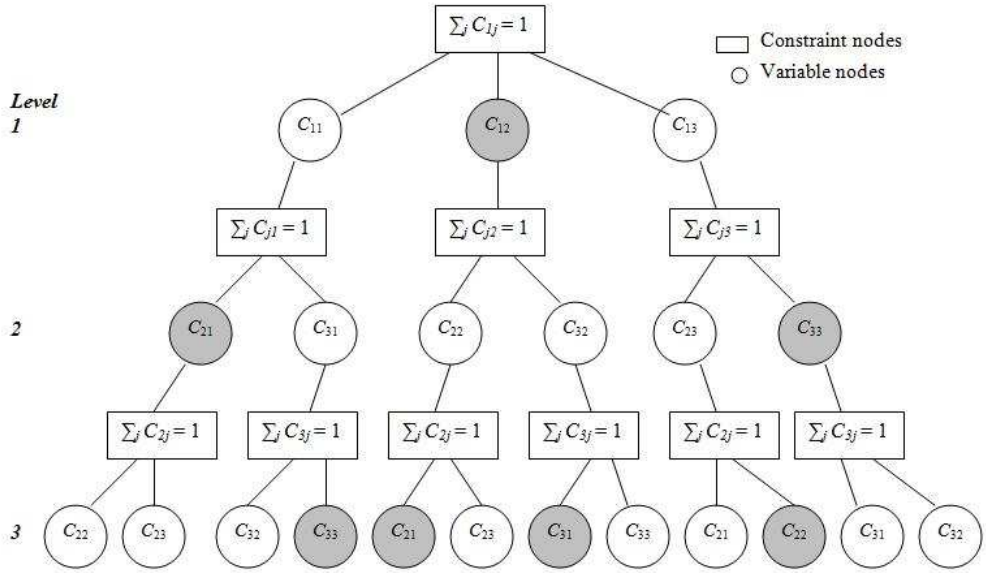

Figure 3: Optimum Λ^{opt} for the computational tree

It's known that by running on a factor graph without loop, the message passing algorithm will converge to a solution, which is an assigment of variables nodes C_{ij}'s. Reflecting in the original bipartite graph, $C_{i,j} = 1$ means the edge (α_i, β_j) is selected. As presented in previous lectures, the messages passed between nodes are designed so that: for each constraint node, there is only one variable node selected among its adjacent variable nodes.

Assume that the optimal solution Λ^{opt} found by belief propagation algorithm is not the MWM in the original bipartite graph. As shown in figure 3, the shadowed nodes belong to Λ^{opt}.

As we know that the MWM $\Pi = \{C_{11}, C_{22}, C_{22}\}$, thus at the first level, node $C_{11} \in \Pi$, while $C_{12} \in \Lambda^{opt}$. Hence, we can find the alternative path $P = \{C_{22}, C_{21}, C_{11}, C_{12}, C_{22}, C_{21}\}$, as marked by the bold lines in figure 4. We can also see clearly from figure 4 that nodes from Π and nodes from Λ^{opt} appear alternatively along P.

Such alternative path P can be decomposed to a cycle C_1 and a subpath P_1 in the original bipartite graph, as shown in figure 5.

Now, for cycle C_1, it is clear that if we pick the edges $\{C_{11}, C_{22}\}$ and instead of $\{C_{12}, C_{21}\}$, we can gain weight for the matching by at least ϵ. Repeat this process to cycle C_2. Eventually, by replacing edges from Λ^{opt} with edges from Π in all the cycles(including the cycle with two extra edges), we can get a new set of edges Λ'. In general, Λ' **may not be a solution** for the computational tree as Λ' may contain the **two extra edges**. Removing those two edges from

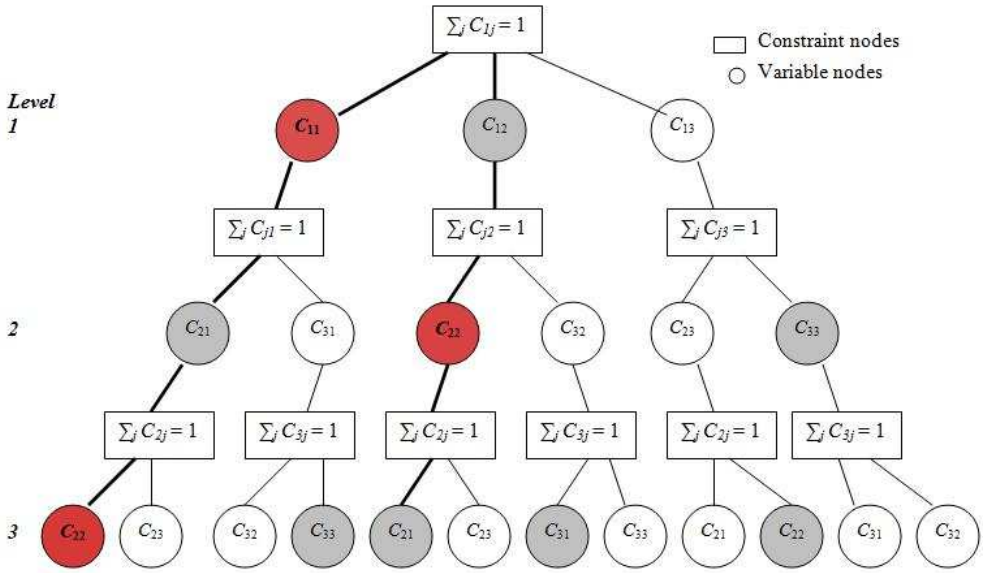

Figure 4: Alternative path in the computational tree

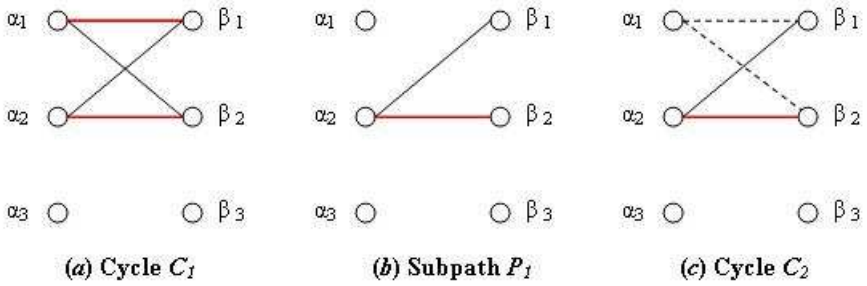

Figure 5: Projection of the alternative path P is decomposed to (a):cycle C_1 of length 4, and (b): path P_1 of length 2. Bying adding two extra edges C_{12}, C_{21}, we can complete P_1 to another cycle C_2.

Λ' if they exist, we will finally get a set of edges Λ which should also satisfy the constraints and thus be **a new solution** for the computational tree. In our particular example, the nodes in Λ are colored in red in figure 4.

Above reasoning gives us the following inequalities:

$$Weight(\Lambda') - Weight(\Lambda^{opt}) > m\epsilon,$$

$$Weight(\Lambda') - Weight(\Lambda) \leq 2w_{max}.$$

Combine them, we will get

$$Weight(\Lambda) - Weight(\Lambda^{opt}) > m\epsilon - 2w_{max} > \frac{k}{n}\epsilon - 2w_{max}.$$