Sparse basis

December 18, 2020

Construction of the basis

Starting from a parity check matrix H (sparse) with m rows and n columns represented by a factor graph G with n variable nodes and m check nodes (with $n > m$). The variables nodes can have degree 1 or 2, and the check nodes have arbitrary degree profile $P(X)$.

One performs the leaf removal (LR) algorithm. At each time step t, one picks a variable v_t uniformly at random among the leaves of the graph G_t. The unique node a_t that was attached to v_t is also removed, thus uncovering new leaves in the new factor graph G_{t+1}. Since in our setting variables have at most degree 2, the core of G is empty, therefore the procedure ends at $t = m$, when all the check nodes have been removed. The variables have been splitted in two sets: the dependent ones $v_1, \dots, v_m$, and the remaining ones are independent variables (denoted $w_1, \dots, w_{n-m}$).

This procedure allows to re-write H in upper triangular form by a permutation of the rows and columns of H. For the order of columns one puts first the block of dependent variables in the order of increasing time $v_1, \dots, v_m$. The rows are ordered correspondingly: the t^{th} row corresponds to check node a_t (i.e one puts $a_1, \dots, a_m$ from top to bottom). The right block contains the $n - m$ independent variables (in arbitrary order).

After this permutation H is transformed into the matrix $(T|U)$ with T a $m \times m$ matrix in upper triangular form, and U is a $m \times (n - m)$ matrix. Both T and U are sparse, since H is sparse.

Then one performs Gaussian elimination to transform the matrix $(T|U)$ into $(I|U')$, with I the identity matrix of size m, and $U' = T^{-1}U$ a $m \times (n - m)$ matrix.

The basis is obtained as the columns of the matrix of size $n \times (n - m)$, built with two blocks: on top there is $-U'$, below its the identity I (of size $n - m$). The goal is to show that U' is sparse.

Row operations

During gaussian elimination, one performs additions of the rows $L_1, \dots, L_m$ of the matrix $(T|U)$. By definition the row L_m at the bottom does not have to be modified (it is already in the right form). Then looking at the row above

L_{m-1}: if the unique element at the right of the diagonal term is non-zero then $L_{m-1} \leftarrow L_{m-1} + L_m$. If not then it means that L_{m-1} is already in the right form and no need to change it. Proceeding that way going from bottom row to top row: $i = m \rightarrow 1$. When one looks at the i^{th} row L_i, all the rows below are already in the right form (only the diagonal term is non zero). One has to look for the non-zeros elements of L_i, say they are at position $u > i$, $v > i$, $w > i$, then one needs to do the operation $L_i \leftarrow L_i + L_u + L_v + L_w$.

The operation on each row can be written as $L'_t = L_t + \sum_{s \in B_t} L_s$, with B_t a subset of $\{t+1, \ldots, m\}$. B_t can be interpreted in terms of the paths explored by the leaf removal.

Say that at some arbitrary time $r-1$ the graph G_{r-1} contains leaves q leaves denoted $u_1, \ldots, u_q$. Then the algorithm picks one leaf: $v_r = u_i$ for some $i \in 1 \ldots, q$, and will be able to explore other variables uncovered when removing the chek node associated to u_i. B_r is the set of all the variables that have been visited by a path starting from node v_{r+1}: $B_r = \{v_{r'} : r' > r$ there is a path explored by leaf removal from v_r to $v_{r'}\}$. If at a later time $s > r$ another leaf u_j is explored: $v_s = u_j$, then a new subset B_s will be built starting from v_s. Note that B_r and B_s are disjoints ($B_r \cap B_s = \emptyset$). Also, B_r is a tree, with root v_r: indeed there cannot be loops, since two variables cannot be visited twice by the algorithm.

Effect of row operations on U

We would like to know how many non-zero entries are created in U' when the row operations $L'_t = L_t + \sum_{s \in B_t} L_s$ are performed. The columns of U corresponds to the independent variables $w_1, \ldots, w_{n-m}$ that have not been removed by the LR algorithm. This means that at some time t the degree of these variables has become zero. One can distinguish between two cases: either the independent variable w_i had degree 1 in the original graph G, of it had degree 2.

1. In the first case let t be the time at which the unique check node a_t linked to w_i has been removed. This means that in the i^{th} column of U, corresponding to the independent variable w_i, there is a unique non-zero entry at the t^{th} position. The variable v_t that has been removed at t is such that $v_t \in \partial a_t \setminus w_i$, with ∂a_t the set of vertices linked to a_t. Moreover, v_t belongs to a subset B_{t_0}, with v_{t_0} a variable that was a leaf in the initial graph G. Let $\mathcal{P}_{t_0 \rightarrow t} \subseteq B_{t_0}$ be the path that goes from v_{t_0} to v_t in the tree B_{t_0}. Then one can check that in the i^{th} column of U', there is non-zero entries in all positions corresponding to the variables in $\mathcal{P}_{t_0 \rightarrow t}$: i.e. $U'_{r,i} = 1$ for r such that $v_r \in \mathcal{P}_{t_0 \rightarrow t}$, and zeroes elsewhere.

2. In the second case where the independent variable w_i had degree 2 in the original graph G, let $s < t$ be the two time steps at which the two check nodes a_s and a_t linked to w_i have been removed. In the i^{th} column of U there is 2 non-zero entries at positions s and t. The variables v_s (resp. v_t) that have been removed at s (resp. at t) belongs to $\partial a_s \setminus w_i$ (resp. to

$\partial a_t \setminus w_i$). Moreover, one has $v_s \in B_{s_0}$ and $v_t \in B_{t_0}$, with v_{s_0} and v_{t_0} that were two leaves in the inital graph G. It might be that these two leaves coincides: $v_{s_0} = v_{t_0}$. Let $\mathcal{P}_{s_0 \to s} \subseteq B_{s_0}$ (resp. $\mathcal{P}_{t_0 \to t} \subseteq B_{t_0}$) be the path going from v_{s_0} to v_s in the tree B_{s_0} (resp. from v_{t_0} to v_t in the tree B_{t_0}).

(a) if $v_{s_0} \neq v_{t_0}$ then one can check that in the i^{th} column of U' there is non-zero entries in all the positions corresponding to the variables belonging to one of the two paths $\mathcal{P}_{s_0 \to s}$ and $\mathcal{P}_{t_0 \to t}$

(b) if $v_{s_0} = v_{t_0}$ then the beginning of the two paths coincides for some variables $\{v_{s_0} \to \cdots \to v_r\} = \mathcal{P}_{s_0 \to s} \cap \mathcal{P}_{t_0 \to t}$. After some check node a_r the path splits in two parts. The subset $\{w_i\} \cup \mathcal{L}_i$ with $\mathcal{L}_i = (\mathcal{P}_{s_0 \to s} \cup \mathcal{P}_{t_0 \to t}) \setminus (\mathcal{P}_{s_0 \to s} \cap \mathcal{P}_{t_0 \to t})$ forms a loop containing both a_s, a_t and a_r. One can see that the non-zero entries in the i^{th} column of U' coincides with the variables belonging to $\mathcal{L}_i$.

Connection with seaweeds

One can observe that the paths $\mathcal{P}_{s_0 \to s}$, $\mathcal{P}_{t_0 \to t}$ and the loop $\mathcal{L}_i$ look very similar to the seaweeds defined in [1], Appendix A. However, it is not exactly the same, because here the paths are not monotonous in depths. The depth of a variable is computed with a modified leaf removal algorithm described in [1]. Starting from G, depth '1' is assigned to each the vertex of degree 1. Then they are all deleted as well as their associated hyperedge. In the new graph the depth '2' is assigned to the leaves, and so on.

In our case one can check that the path are not monotonous in depth, looking for instance at the 1st case (in which the independent variable w_i had degree 1 in G). The path $\mathcal{P}_{t_0 \to t}$ is going from the variable v_{t_0} of depth 1 to the variable v_t of depth 2 through other variables that have possibly depth greated than '2'.

One hypothesis is that for the version of the LR algorithm one uses (i.e at each time step pick a leaf uniformly at random among all leaves) there is a high probability to visit first all the variables with smallest depth d before visiting variables with higher depth $d+1$. It could be interesting to show that this is true, because then the 1st case (w_i had degree 1 in G) would have low probability to happen. And in the second case (w_i had degree 2 in G) the paths and loops obtained would satisfy the properties needed to be a seaweed with large probability.

In any case, we still can use a slightly modified leaf removal algorithm to build the basis, that follows the additional rule: 'at time t, pick a variable of degree 1 among the ones that have the smallest depth'. In that way I think one is certain that the paths and loops created satistfy the properties needed to be a seaweed (in particular, the 1st case in which w_i had degree 1 in G never happens). The root of the seaweed would be the independant variable w_i itself, since it has the largest depth. Then the two branches that starts from w_i are upward branches, because when following them each new vertex encountered has a smaller depth than the previous one. Then we can use the result of [1] to

say that columns of U' have small (sub-linear) hamming weight because they are seaweeds, so that U' is sparse.

References

[1] M. Mézard, F. Ricci-Tersenghi, and R. Zecchina. Alternative solutions to diluted p-spin models and XORSAT problems. *J. Stat. Phys.*, **111**, 505 (2003).