**Algorithm 1:** Risk stratification algorithm

**Input:**

$X$: A matrix of size $n_{samples} \times n_{features}$ containing the binary representation of the samples in the given training data set.

$\vec{y}$: A vector of size $n_{samples}$ containing the binary labels of the training samples.

$\vec{n}$: A vector holding the number of input features to each first-layer module.

**Output:**

*funcF*: A set of vectors each holding the I/O function of a first-layer module.

*Risk*: A vector holding a probability between 0 to 1 for each input configuration to the Output module.

*funcO*: A vector holding the I/O function of the Output module.

$n_{states} \leftarrow \{2^{\vec{n}[1]}, 2^{\vec{n}[2]}, 2^{\vec{n}[3]}\}$ ▷ In the simple case, there are 3 first-layer modules.

$\boldsymbol{T_0}, \boldsymbol{T_1} \leftarrow \mathbf{L}(X, \vec{y})$;

**for** $\alpha \in \{0,1,2\}$ **do**

  $(I,J,K) \leftarrow \sigma_\alpha(1,2,3)$;

  $weight \leftarrow [0,...,0]_{n_{states}[I] \times [n_{states}[I]-1]/2}$;

  **for** $j \in \{1,...,n_{states}[J]\}$ **do**

    **for** $k \in \{1,...,n_{states}[K]\}$ **do**

      $idx \leftarrow 1$ ▷ In the following loops, inputs to two of the first-layer modules are fixed and the weights for the other module are determined.

      **for** $i \in \{1,...,n_{states}[I]-1\}$ **do**

        **for** $i' \in \{i+1,...,n_{states}[I]\}$ **do**

          $weight[idx] \leftarrow weight[idx] + \boldsymbol{T_0}[\sigma_\alpha(i,j,k)] * \boldsymbol{T_1}[\sigma_\alpha(i',j,k)] + \boldsymbol{T_0}[\sigma_\alpha(i',j,k)] * \boldsymbol{T_1}[\sigma_\alpha(i,j,k)]$;

          $edges[idx] \leftarrow (i,i')$;

          $idx \leftarrow idx + 1$

        **end**

      **end**

    **end**

  **end**

  $funcF[\alpha+1] \leftarrow solveMaxCut(edges, weight, n_{states}[\alpha+1])$ ▷ The function of each first-layer module is identified by solving a max-cut problem.

**end**

$counter0 \leftarrow [0,...,0]_8$ ▷ Counts "0" labels for each input of the Output module.

$counter1 \leftarrow [0,...,0]_8$ ▷ Counts "1" labels for each input of the Output module.

**for** $i \in \{1,...,n_{states}[1]\}$ **do**

  **for** $j \in \{1,...,n_{states}[2]\}$ **do**

    **for** $k \in \{1,...,n_{states}[3]\}$ **do**

      **if** $\boldsymbol{T_0}(i,j,k) \neq 0$ *or* $\boldsymbol{T_1}(i,j,k) \neq 0$ **then**

        $idxOB \leftarrow 1 + funcF[1][i] * 2^0 + funcF[2][j] * 2^1 + funcF[3][k] * 2^2$;

        $counter0[idxOB] \leftarrow counter0[idxOB] + \boldsymbol{T_0}(i,j,k)$;

        $counter1[idxOB] \leftarrow counter1[idxOB] + \boldsymbol{T_1}(i,j,k)$;

      **end**

    **end**

  **end**

**end**

$Risk \leftarrow [0,...,0]_8$;

**for** $i \in \{1,...,8\}$ **do**

  $Risk[i] \leftarrow \frac{counter1[i]}{counter0[i]+counter1[i]}$ ▷ Risk estimation.

  **if** $counter0[i] > counter1[i]$ **then**

    $funcO[i] \leftarrow 0$;

  **else**

    $funcO[i] \leftarrow 1$ ▷ Majority voting scheme for the Output module function identification.

  **end**

**end**

**return** *funcF*, *Risk*, *funcO*;