Algorithm 19.1 Computing the gradient in a network with table-CPDs

Procedure Compute-Gradient (
  $\mathcal{G}$, // Bayesian network structure over $X_1, \ldots, X_n$
  $\boldsymbol{\theta}$, // Set of parameters for $\mathcal{G}$
  $\mathcal{D}$ // Partially observed data set
)

// Initialize data structures
for each $i = 1, \ldots, n$
  for each $x_i, \boldsymbol{u}_i \in Val(X_i, \mathrm{Pa}^{\mathcal{G}}_{X_i})$
    $\bar{M}[x_i, \boldsymbol{u}_i] \leftarrow 0$
  // Collect probabilities from all instances
for each $m = 1 \ldots M$
  Run clique tree calibration on $\langle \mathcal{G}, \boldsymbol{\theta} \rangle$ using evidence $\boldsymbol{o}[m]$
  for each $i = 1, \ldots, n$
    for each $x_i, \boldsymbol{u}_i \in Val(X_i, \mathrm{Pa}^{\mathcal{G}}_{X_i})$
      $\bar{M}[x_i, \boldsymbol{u}_i] \leftarrow \bar{M}[x_i, \boldsymbol{u}_i] + P(x_i, \boldsymbol{u}_i \mid \boldsymbol{o}[m])$
  // Compute components of the gradient vector
for each $i = 1, \ldots, n$
  for each $x_i, \boldsymbol{u}_i \in Val(X_i, \mathrm{Pa}^{\mathcal{G}}_{X_i})$
    $\delta_{x_i|\boldsymbol{u}_i} \leftarrow \frac{1}{\theta_{x_i|\boldsymbol{u}_i}} \bar{M}[x_i, \boldsymbol{u}_i]$
return $\{\delta_{x_i,|\boldsymbol{u}_i} : \forall i = 1, \ldots, n, \forall (x_i, \boldsymbol{u}_i) \in Val(X_i, \mathrm{Pa}^{\mathcal{G}}_{X_i})\}$

19.2.1.3 Gradient Ascent Algorithm

We now generalize these ideas to case of an arbitrary network. For now we focus on the case of table-CPDs. In this case, the gradient is given by theorem 19.2. To compute the gradient for the CPD $P(X \mid \boldsymbol{U})$, we need to compute the joint probability of x and $\boldsymbol{u}$ relative to our current parameter setting $\boldsymbol{\theta}$ and each observed instance $\boldsymbol{o}[m]$. In other words, we need to compute the joint distribution $P(X[m], \boldsymbol{U}[m] \mid \boldsymbol{o}[m], \boldsymbol{\theta})$ for each m. We can do this by running an inference procedure for each data case. Importantly, **we can do all of the required inference for each data case using one clique tree calibration, since the family preservation property guarantees that X and its parents $\boldsymbol{U}$ will be together in some clique in the tree.** Procedure Compute-Gradient, shown in algorithm 19.1, performs these computations.

Once we have a procedure for computing the gradient, it seems that we can simply plug it into a standard package for gradient ascent and optimize the parameters. As we have illustrated, however, there is one issue that we need to deal with. It is not hard to confirm that all components of the gradient vector are nonnegative. This is natural, since increasing each of the parameters will lead to higher likelihood. Thus, a step in the gradient direction will *increase* all the parameters. Remember, however, that we want to ensure that our parameters describe a legal probability distribution. That is, the parameters for each conditional probability are nonnegative and sum to one.