(CE7456 Lecture Notes 2) Tree Sampling

Yufei Tao

Today, we discuss *tree sampling*, a fundamental technique useful in many sampling applications.

1 The Problem

Let T be a tree where

- each node u is associated with a positive integer weight $w(u)$;
- for each internal node u, it holds that $\sum_{v \in child(u)} w(v) \leq w(u)$, where $child(u)$ is the set of child nodes of u.

Define

$$\begin{aligned} L &= \text{the set of leaves in } T \\ W &= \sum_{z \in L} w(z). \end{aligned}$$

The goal of tree sampling is to sample the leaves with probabilities proportional to their weights. Specifically, we want to randomly draw an element Z from L such that

$$\mathbf{Pr}[Z = z] = \frac{w(z)}{W} \tag{1}$$

for each leaf $z \in Z$. It is worth pointing out that

$$W \leq w(u_{root})$$

where u_{root} is the root of T.

Our access to T is by way of the following oracle:

Child-Sampling Oracle. Given a node u in T, the oracle draws a random element $X \in child(u) \cup \{\perp\}$ according to the distribution:

$$\mathbf{Pr}[X = x] = \begin{cases} w(x)/w(u) & \text{for each child } x \in child(u) \\ 1 - \frac{1}{w(u)} \sum_{v \in child(u)} w(v) & \text{for } x = \perp \end{cases}$$

In the beginning, we are given only the root of T. The objective is to obtain the desired Z satisfying (1) by calling the oracle as few times as possible.

2 The Algorithm

Consider the algorithm below:

`tree-sample` (T)
1. $u \leftarrow$ root of T
2. **while** u is not a leaf **do**
3. $\quad v \leftarrow$ the outcome of child sampling on u
4. $\quad$ **if** $v = \perp$ **then return** "failed" **else** $u \leftarrow v$
5. **return** u

We say that the algorithm *succeeds* if it manages to return a leaf (i.e., it does not return "failed").

Lemma 1. *The* `tree-sample` *algorithm succeeds with probability* $W/w(u_{root})$*. When the algorithm succeeds, its returns a leaf according to the distribution in* (1)*.*

Proof. Fix an arbitrary leaf $z \in L$. Denote by $u_1, u_2, ..., u_\ell$ ($\ell \geq 1$) the sequence of nodes encountered as we descend from the root of T to z (note: u_1 is the root and u_ℓ is z). The `tree-sample` algorithm returns z if and only if the following event occurs at each $i \in [1, \ell - 1]$

$$\text{the child sampling on } u_i \text{ returns } u_{i+1}.$$

The event of value i occurs with probability $w(u_{i+1})/w(u_i)$. As the $\ell - 1$ events are independent, the probability for `tree-sample` to return z equals

$$\frac{w(u_2)}{w(u_1)} \cdot \frac{w(u_3)}{w(u_2)} \cdot ... \cdot \frac{w(u_\ell)}{w(u_{\ell-1})} = \frac{w(z)}{w(u_{root})}.$$

It thus follows that `tree-sample` succeeds with probability

$$\sum_{z \in L} \frac{w(z)}{w(u_{root})} = \frac{W}{w(u_{root})}. \tag{2}$$

Hence, for each $z \in L$, we get

$$\mathbf{Pr}[z \text{ returned} \mid \text{success}] = \frac{w(z)}{W}$$

as claimed. □

The algorithm may fail returning a sample, in which case we simply repeat it until getting a success. By (2), the number of repeats needed to induce one success is $\frac{w(u_{root})}{W}$ in expectation. The following is a more precise characterization of how many repeats are needed probabilistically.

Corollary 2. *For any integer* $t \geq 1$*, the probability for repeating the* `tree-sample` *algorithm* $t \cdot \frac{w(u_{root})}{W}$ *times without a success is at most* $1/e^t$*.*

Proof. As one run of the algorithm fails with probability $1 - \frac{w(u_{root})}{W}$, the probability of failing $t \cdot \frac{w(u_{root})}{W}$ times consecutively is

$$\left(1 - \frac{w(u_{root})}{W}\right)^{t \cdot \frac{w(u_{root})}{W}} \quad \leq \quad e^{-t}$$

where the derivation used the fact $1 + x \leq e^x$ for all $x \in \mathbb{R}$. □