Boltzmann Generators I

Contents

1 Boltzmann Samplers

Now let us be a little more open minded. If we are looking to generate a random tree with 10 000 nodes, we may be perfectly happy to have a binary tree with 10 001 nodes generated *provided that the probability of any given tree is uniform by size*. We can do a very clever trick to get this.

1.1 A Boltzmann model of a combinatorial class

The Boltzmann sampler draws a random object from the class corresponding to the Boltzmann model, depending on some real parameter x. The idea is to adjust the parameter x so that the Boltzmann sampler draws a structure of size *close to* n, instead of using a procedure that draws samples of size exactly equal to n. The advantage is an enormous savings in computation time; the time per generation being $O(n)$ for a structure of a size from $[n(1-\varepsilon), n(1+\varepsilon)]$. The straightforward idea of selecting x so that the expected size of a drawn object becomes n works well whenever the Boltzmann model looks reasonably normal, like set partitions.

Recall the generating function $A(z)$ is

$$A(z) = \sum_{\alpha \in \mathcal{A}} z^{|\alpha|} = \sum_n A_n z^n.$$

A positive number x strictly smaller than the radius of convergence of $A(z)$ is said to be an admissible value for $\mathcal{A}$. In such a case the generating function converges as a sum. A Boltzmann model at x assigns to each $\alpha \in \mathcal{A}$ a probability

$$\mathbf{P}_x(\alpha) = \frac{x^{|\alpha|}}{A(x)}.$$

First remark that

$$\sum_{\alpha \in \mathcal{A}} \mathbf{P}_x(\alpha) = \sum_{\alpha \in \mathcal{A}} \frac{x^{|\alpha|}}{A(x)} = A(x)/A(x) = 1$$

so these are properly assigned probabilities. Essentially $A(x)$ acts as a normalizing constant. Now that the distribution is spread over *all* objects of $\mathcal{A}$, but on each size the distribution is uniform (i.e., two objects of the same size have the same probability to be chosen).

We can develop a very efficient method for random sampling under this model. Imagine that you want to draw elements of a class *around* the size N. We describe below algorithm now takes as input x and draws objects in a class $\mathcal{A}$ such that the probability of an element of size n is $\mathbf{P}_x(\alpha) = \frac{x^{|\alpha|}}{A(x)}$. Depending on $A(z)$, it may be simple to find the "good" x so that most of the elements are in the range of N that you wish, and you simply *reject* the rest.

1.2 Example: Binary trees

The generating function for Binary trees (the kind with exactly 0 or 2 children at each vertex) is

$$B(z) = \frac{1 - \sqrt{1 - 4z^2}}{2z}.$$

So, a Boltzmann scheme should generate trees of size $n = 2m + 1$ with the following probabilities for $n = 1, 3, 5, 7$:.

n	1	3	5	7
$x = 0.2$	0.95832	0.038332	0.001533	0.00006
$x = 0.49$	0.59950	0.14394	0.03456	0.008297
$x = 0.49999$	0.50316	0.12578	0.031444	0.007860

We see that such a sampler would generate a single vertex 95% of the time if the parameter was set to $x = .2$ and about half of the time when $x = .49999$. Any admissible value must be below $.5$, since that is the value that makes the square root equal to 0. (In general, we are looking for the smallest real value that is a *singularity* of the generating function in the complex analysis sense. If that is not meaningful to you, then look for values that make the denominator equal to 0, or where the square root is equal to 0.)

1.3 Basic Samplers

So, how do we actually make such a sampler? First, if we have a Boltzmann sampler for a class $\mathcal{A}$, let us denote it $\Gamma A(x)$. For each admissible operator, we describe a way to describe a sampler in terms of a sampler of the argument.

What should we expect? A sampler for the atomic class should return a thing of size 1 with probability $x^1/Z(x)$ where $Z(x)$ is the generating function for the atomic class. Well, $Z(x) = x$, hence this probability is 1, that is a Boltzmann sampler for an atomic class *always returns a single atom*. Likewise, a sampler for the neutral class returns an element of size 0 with probability $x^0/E(x)$. Since $E(x) = 1$, this probability is $1/1 = 1$, and a sampler for the neutral class *always returns a single* ϵ.

Boltzmann Sampler: Atom

```
// input: x a positive real number
// output: Z, an atom
gammaZ:= proc(x)
      Return Z
 end proc;
```

Boltzmann Generator Generator: Epsilon

```
// input: x, a positive real number
// ouput: an Epsilon
  genE:= proc(x)
      Return E
  end proc;
```

Now let us consider first combinatorial sum and product.

1.4 Combinatorial sum

Imagine $\mathcal{A} = \mathcal{B} + \mathcal{C}$. So, we assume that we have $\Gamma B(x)$ and $\Gamma C(x)$ known. We want to describe how to make sure that $\alpha \in \mathcal{A}$ is generated with probability $\frac{x^{|\alpha|}}{A(x)}$. We need a way to decide whether or not to draw from $\mathcal{B}$ or $\mathcal{C}$. Let us consider their probabilities. Well the probability that a properly generated α originally came from $\mathcal{B}$ is the sum over those elements:

$$\sum_{\beta \in \mathcal{B}} \frac{x^\beta}{A(x)} = \frac{B(x)}{A(x)}.$$

Now we see how to get a generator!

Boltzmann Sampler: Combinatorial Sum $\mathcal{A} = \mathcal{B} + \mathcal{C}$

```
// input: x a real number
// output: a uniformly generated element of A
gammaA:= proc(x)
        let  u = rnd(0,1):
        if u< B(x)/A(x) then Return gammaB(x) else Return gammaC(x);
    od:
  end proc;
```

probability of an element β of $\mathcal{B}$ being chosen is thus $\frac{B(x)}{A(x)}\frac{x^{|\beta|}}{B(x)} = \frac{x^{|\beta|}}{A(x)}$. With these choices we have a Boltzmann generator for the sum.

1.5 Product

Imagine $\mathcal{A} = \mathcal{B} \times \mathcal{C}$. Again, we assume that we have $\Gamma B(x)$ and $\Gamma C(x)$ known. We want to describe how to make sure that $\alpha \in \mathcal{A}$ is generated with probability $\frac{x^{|\alpha|}}{A(x)}$. Assume $\alpha = (\beta, \gamma)$. Then

$$\frac{x^\alpha}{A(x)} = \frac{x^{|\beta|+|\gamma|}}{A(x)} = \frac{x^{|\beta|+|\gamma|}}{B(x)C(x)} = \frac{x^{|\beta|}}{B(x)}\frac{x^{|\gamma|}}{C(x)}.$$

Nice. This means we can generate the two components independently.

Boltzmann Sampler: Product $\mathcal{A} = \mathcal{B} \times \mathcal{C}$

```
// input: x, a real number
// output: a uniformly generated element of A
gammaA:= proc(x)
      (gammaB(x) , gammaC(x));
    od:
  end proc;
```

1.6 Binary Trees

Let us go back to our good friend the binary tree. Not only do we know the coefficient sequence exactly, but we know the generating function explicitly, so we can easily evaluate it at different values of x.

Recall

$$B(x) = \frac{1 - \sqrt{1 - 4x^2}}{2x}.$$

So a generator for a binary tree looks like:

Boltzmann Sampler: Binary Tree

```
// input: x a real number
// output: a uniformly generated binary tree
gammaB:= proc(x)
       let  u = rnd(0,1)
       if u< x/B(x) then Return gammaZ(x)
       else Return (gammaB(x) , gammaB(x))
    od
  end proc

gammaZ:= proc(x) Return Z end proc
```

1.7 Sequence

To generate a sequence $\mathcal{A} = \textsc{Seq}(\mathcal{B})$ we could always combine a sum and a product since

$$\mathcal{A} = \textsc{Seq}(\mathcal{B}) \equiv \mathcal{A} = \mathcal{E} + \mathcal{B} \times \mathcal{A}$$

We can be more direct. Instead, to generate an $\mathcal{A}$ element we first determine the number of components, say k, and then we make k independent calls tp $\Gamma\mathcal{B}(x)$. But, with which probability should we generate an element with k components?

We draw it with a geometric distribution with parameter $\lambda = B(x)$. Recall that the probability variable X is of geometric rate λ if

$$\mathbb{P}(X = k) = (1 - \lambda)\lambda^k$$

So, for example, if $\lambda = \frac{1}{3}$, the probability of $k = 5$ is $(2/3)^5 = 0.131687\ldots$. Let $\alpha = (\beta_1, \beta_2, \ldots, \beta_k)$. Then

$$\mathbb{P}_x(\alpha) = (1 - B(x))B(x)^k \frac{x^{|\beta_1|}}{B(x)} \frac{x^{|\beta_2|}}{B(x)} \cdots \frac{x^{|\beta_k|}}{B(x)} = (1 - B(x))B(x)^k \frac{x^{|\beta_1|+\cdots+|\beta_k|}}{B(x)^k} = \frac{x^{|\alpha|}}{A(x)}$$

which is the correct probability. Thus we have the following procedure:

Boltzmann Sampler: Sequence $\mathcal{A} = \textsc{Seq}(\mathcal{B})$

```
// input: x, a real number
// output: an element of A
gammaA:= proc(x)
     draw k according to Geometric(B(x))
     return k-tuple [gammaB(x) , ..., gammaB(x)];
   od:
 end proc;
```

1.8 Questions

Let us consider some relevant questions:

1. How do you evaluate the generating function?
2. How fast is this?
3. How do you pick x?
4. How do we implement a geometric generator?

One can state this firmly under the assumption that the evaluations at x (which are real numbers) of the generating functions intervening in the decomposition of $\mathcal{A}$ are known exactly. This assumption is known as the oracle assumption (one imagines that an oracle provides the values for us). In practice, one evaluates the generating functions with a fixed precision, say N digits (typically $N = 20$) and in the unlikely case one needs more digits during the generation of an object, one computes a few more digits of the generating functions (adaptative procedures).

Theorem. *Let $\mathcal{A}$ be a decomposable class in terms of the constructions $\{+, \times\}$ and the basic classes $\{1, \mathcal{Z}\}$, and let $\Gamma A(x)$ be the Boltzmann sampler obtained from the sampling rules. Then, under the oracle assumption, the generation of an object $\alpha \in \mathcal{A}$ by $\Gamma A(x)$ takes time $O(|\alpha|)$.*

2 References

- A. Nijenhuis and H. S. Wilf. Combinatorial Algorithms (second edition), Academic Press, 1979
- Ph. Flajolet, P. Zimmerman, Paul; and B. Van Cutsem. A calculus for the random generation of labelled combinatorial structures. Theoret. Comput. Sci. 132 (1994), no. 1-2, 135.
- Ph. Duchon, Philippe; Ph. Flajolet; G. Louchard; and G. Schaeffer, Gilles. Boltzmann samplers for the random generation of combinatorial structures. (English summary) Combin. Probab. Comput. 13 (2004), no. 4-5, 577625.