Sampling Trees: Gibbs, CFTP, and Metropolis

Xiaojin Zhu {zhuxj@cs.cmu.edu}

May 12, 2000

1. **Motivation**

   In this project we choose to examine the effectiveness of a few Markov Chain Monte Carlo methods, namely the Gibbs algorithm, Coupling From the Past (CFTP) [1], and the Metropolis-Hasting algorithm. The task is to sample from a probabilistic model on trees. The model is designed to satisfy the monotone condition for CFTP. Meanwhile the model is hard for the traditional Gibbs algorithm. By this project, we wish to answer the following questions:

   (a) How badly will a traditional Gibbs sampler perform under such an adverse model?

   (b) What is the performance of CFTP?

   (c) Will a heuristic proposal distribution dramatically help the Metropolis-Hasting sampler?

2. **The Model**

   We denote by $\mathcal{T}$ the set of binary trees with level L and $2^{L+1}-1$ nodes, in which each node has label 0 or 1. We will define a probablistic model over $\mathcal{T}$. First we define a function $\triangle_l$ with some fixed l on tree node v:

   $$\triangle_l(v) = \begin{cases} \{\text{label of } w \mid w \text{ is a decendant of } v \text{ within } l \text{ edges}\} & \text{if } height(v) \geq l \\ \emptyset & \text{otherwise} \end{cases}$$

   Intuitively $\triangle_l(v)$ summarizes the label of nodes which forms a triangle of height l beneath v. There are four possibilities:

   (a) $\triangle_l(v) = \{0\}$: all nodes within the triangle have label 0.

   (b) $\triangle_l(v) = \{1\}$: all nodes within the triangle have label 1.

   (c) $\triangle_l(v) = \{0, 1\}$: some nodes within the triangle have label 0, while others have label 1.

   (d) $\triangle_l(v) = \emptyset$: v is not high enough in the tree to have such a triangle beneath it.

   We define a model that encourages all-0 and all-1 triangles in a tree. Let $x \in \mathcal{T}$ be a tree, then

   $$p(x) = \frac{1}{Z} e^{\sum_{v \in x} \alpha * (\delta(\triangle_l(v), \{0\}) + \delta(\triangle_l(v), \{1\}))} \tag{1}$$

   $\delta(S_1, S_2) = 1$ if set S_1 equals S_2, otherwise $\delta(S_1, S_2) = 0$. α is the parameter of the model, and should be positive. Z is a normalization factor.

3. **The Gibbs Sampler**

We can generate sample trees from the model p with a Gibbs Sampler [4] [5]. It is easy to show that changing the label of a particular node v while holding all other nodes fixed will affect at most $l+1$ triangles, namely $\triangle_l(v)$, $\triangle_l(parent(v))$, $\triangle_l(parent^2(v)) = \triangle_l(parent(parent(v)))$, …, $\triangle_l(parent^l(v))$. This is good to a Gibbs sampler, because changes are reasonably local. In fact, the *conditional odds* of node v is

$$\begin{aligned}\frac{p(v=1|\neg v)}{p(v=0|\neg v)} &= \frac{p(v=1|\neg v)p(\neg v)}{p(v=0|\neg v)p(\neg v)} \\ &= \frac{p(v=1,\neg v)}{p(v=0,\neg v)} \\ &= e^{\alpha * \sum_{u\in\{v,parent(v),\ldots,parent^l(v)\}}[\delta(\triangle_l(u)|_{v=1},\{1\})-\delta(\triangle_l(u)|_{v=0},\{0\})]}\end{aligned}$$

Since $p(v=1|\neg v)+p(v=0|\neg v)=1$, we have

$$p(v=0|\neg v) = \frac{1}{1+e^{\alpha * \sum_{u\in\{v,parent(v),\ldots,parent^l(v)\}}[\delta(\triangle_l(u)|_{v=1},\{1\})-\delta(\triangle_l(u)|_{v=0},\{0\})]}} \tag{2}$$

$$p(v=1|\neg v) = \frac{e^{\alpha * \sum_{u\in\{v,parent(v),\ldots,parent^l(v)\}}[\delta(\triangle_l(u)|_{v=1},\{1\})-\delta(\triangle_l(u)|_{v=0},\{0\})]}}{1+e^{\alpha * \sum_{u\in\{v,parent(v),\ldots,parent^l(v)\}}[\delta(\triangle_l(u)|_{v=1},\{1\})-\delta(\triangle_l(u)|_{v=0},\{0\})]}} \tag{3}$$

Therefore a Gibbs sampling algorithm might look like:

(a) initialize the tree with arbitrary labels
(b) for v in node 1, 2, …, 2^L-1 do
(c) generate $r \sim unif(0,1)$, let $v=0$ if $r < p(v=0|\neg v)$; $v=1$ otherwise
(d) goto 3b

This algorithm creates a single long Markov Chain. We can let it "burn in" for a while, and then take out a sample after every certain steps.

Albeit simple, the Gibbs sampler will have great trouble with this model when α is large. When α is large, the probability mass will concentrate on a few trees with all-1 or all-0 triangles in them. Once the sampler discovers an all-1 or all-0 triangle, it will stay at the state for many steps before it will go through some low probability states to find other states with different triangles. Asymptotically this behavior is reasonable and will lead to the correct stationary distribution (i.e. the distribution specified by our model). However in practice the chain length is limited, so the samples will have large variance. Our model is just another version of the "witch's hat" distribution that are bad for Gibbs samplers.

4. **CFTP**

We define a partial order $\preceq$ over trees as follows:

$$\forall x,y \in \mathcal{T}, x \preceq y \text{ iff } (\forall \text{ node } v, x(v)=1 \Rightarrow y(v)=1) \tag{4}$$

i.e. whenever a node in x is 1, the corresponding node in y is 1. Obviously

$$\hat{0} \preceq x \preceq \hat{1}, \forall x \in \mathcal{T}$$

where $\hat{0}$ is the all-0 tree, and $\hat{1}$ the all-1 tree.

We claim the MCMC chain under the updating rule 3c is *monotone*. To prove this, we need to prove

$$\forall x, y \in \mathcal{T}, x \preceq y \Rightarrow \frac{p_x(v=1|\neg v)}{p_x(v=0|\neg v)} \leq \frac{p_y(v=1|\neg v)}{p_y(v=0|\neg v)} \tag{5}$$

which is to say, under the updating rule 3c, if any particular node in x is updated to 1, the corresponding node in y must also be updated to 1 to preserve the partial order. The proof is trivial by noting that these inequalities always hold if $x \preceq y$:

$$\begin{array}{l} \forall v, \forall u \in \{v, parent(v), \ldots, parent^l(v)\} : \\ \delta_x(\triangle_l(u)|_{v=1}, \{1\}) \leq \delta_y(\triangle_l(u)|_{v=1}, \{1\}) \\ \delta_x(\triangle_l(u)|_{v=0}, \{0\}) \geq \delta_y(\triangle_l(u)|_{v=0}, \{0\}) \end{array}$$

The CFTP algorithm is as follows:

(a) $T \leftarrow 1$
(b) repeat
(c) $upper \leftarrow \hat{1}$, $lower \leftarrow \hat{0}$
(d)   for $t = -T$ to -1
(e)   if t equals $-1, -2, -4, -8, -16, \ldots$ then $RandomSeed(f(t, id))$
(f)   for v in node $1, 2, \ldots, 2^L - 1$ do
(g)   generate $r \sim unif(0,1)$
(h)   in tree $upper$, let $v = 0$ if $r < p_{upper}(v=0|\neg v)$; $v = 1$ otherwise
(i)   in tree $lower$, let $v = 0$ if $r < p_{lower}(v=0|\neg v)$; $v = 1$ otherwise
(j) $T \leftarrow 2 * T$
(k) until $upper = lower$
(l) return $upper$

Two aspects need special care: First, on line 4e the random number generator seed is reset by a deterministic function when t reaches the start of a previous pass. This ensures that the same randomness is experienced by the same pass, which is essential to the correctness of CFTP. In our implementation, the seed is fully determined by which pass it is in (i.e. the current value of t), and a unique sample id for each sample we produce (so that we get different randomness for different samples)[1]. Second, on line 4h and 4i the two updates share the same randomness r.

Unlike the Gibbs sampler, the CFTP algorithm is guaranteed to generate exact samples from a given model. However the price is long coalesce time per sample. We predict that with harder and harder models (i.e. increasing α), the average time for CFTP to generate a sample will increase significantly.

[1] If we let different passes share the same seed number, i.e. if we use $RandomSeed(f(id))$ or $RandomSeed(constant)$ instead of $RandomSeed(f(t, id))$, we will introduce bias to the sampling process. Such bias is especially prominent when a chain coalesces fast, and remains noticable when coalescence takes longer time.

5. **The Metropolis-Hasting Sampler with a Heuristic Proposal Distribution**

   Our tree model implicitly utilizes the following function, or *feature*:

$$f(v) = \delta(\triangle_l(v), \{0\}) + \delta(\triangle_l(v), \{1\}) = \begin{cases} 1 & \text{if the triangle of } v \text{ is all-0 or all-1;} \\ 0 & \text{otherwise} \end{cases}$$

   By the definition of the model, it can be expected that 'interesting' samples (i.e. samples with large probabilities) are those in which the feature is active. This is a common heuristic for many maximum entropy models for natual language processing [2] [3]. The Gibbs sampler can't use this heuristic, and it only takes small steps in the state space. In contrast, the Metropolis-Hasting algorithm can use a proposal distribution which is specifically designed for the heuristic and meanwhile is exploratory, to hopefully get better samples.

   We design the proposal distribution to be an interpolation of a heuristic part and an exploratory part:

$$q(y|x) = (1-\lambda) \cdot h(y|x) + \lambda \cdot e(y|x)$$

   where $\lambda \in (0, 1]$ is the interpolation weight. The larger the λ, the more exploratory (but less heuristic) the proposal distribution is. We let the exploratory part be the uniform distribution over the state space $\mathcal{T}$:

$$e(y|x) = 1/|\mathcal{T}|$$

   We let the heuristic part be such that we will propose y that is similar to x, but with a feature enforced. The way we do it is as follows: For any node v high enough to have a triangle beneath it in x, we create two trees by setting the label of the nodes within the triangle beneath v to be all-0 and all-1 respectively, while copying all other labels from x. We do this for all such nodes and denote the set of trees created in this fashion as $\mathcal{H}$. Then we let

$$h(y|x) = \begin{cases} 1/|\mathcal{H}| & \text{if } y \in \mathcal{H} \\ 0 & \text{otherwise} \end{cases}$$

   With the proposal distribution q, we use the standard Metropolis-Hasting sampler to generate tree samples from the model.

6. **Experiment Results**

   (a) **The Ground Truth**

   To simplify the analysis, we choose to run the experiment with the smallest interesting tree size: $L = 2$ and $l = 1$, i.e. trees with 7 nodes and triangles with 3 nodes. It is easy to verify that the size of the state space is 128, and there can be up to 3 all-0 or all-1 triangles in a tree.

   We consider 20 different models, with $\alpha = 0, \ldots, 20$ respectively. For a given α (i.e. a given model), we could compute the true distribution over the 128 states, and compare it with the empirical sample distributions from variaous MCMC samplers. However, to make things even more simpler, we choose to monitor the marginal distribution of these three events:

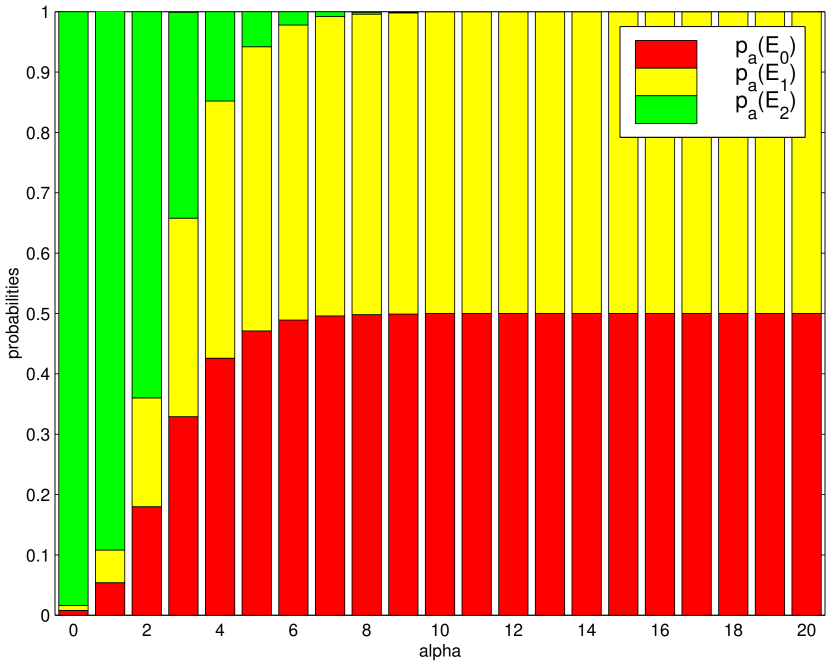

Figure 1: True Marginal Distributions under Different α

- E_0: the tree is all 0.
- E_1: the tree is all 1.
- E_2: none of the above.

To emphasize the fact that the marginal probabilities depend on α, we denote them as $p_\alpha(E_0)$, $p_\alpha(E_1)$ and $p_\alpha(E_2)$. Obviously $p_\alpha(E_0)+p_\alpha(E_1)+p_\alpha(E_2)=1$ for all α. Note E_0 and E_1 both correspond to a single state, while E_2 corresponds to 126 states.

Figure 1 shows the true marginal distributions under different α values. In the figure, each stacked bar is a complete marginal distribution under a specific α. When α is small, the probability mass is spread more or less evenly over the state space. Thus $p_\alpha(E_2)$, which corresponds to many states, dominates. However when α becomes large, the probability mass quickly concentrates on E_0 and E_1, or the all-0 and all-1 state. Also note that $p_\alpha(E_0) \equiv p_\alpha(E_1)$.

(b) **The Gibbs Sampler**

For a tree model with a given α value, we generate 100,000 samples with the Gibbs sampler in section 3. We choose to run a single MCMC chain. Since we have no *a priori* knowledge of how long we should run the chain, we arbitrarily choose the following setting: We let the chain 'burn in' for 1000 iterations, and then take a sample after every 10 iterations. Once we have generated 100,000 samples, we compute the empirical marginal distribution $\tilde{p}_\alpha(E_0)$, $\tilde{p}_\alpha(E_1)$ and $\tilde{p}_\alpha(E_2)$ from the samples. If the Gibbs sampler works well, the empirical marginal distribution should be close to the true marginal distribution of the same α value in Figure 1.

We repeat the above process for α bwtween 0 and 20, and plot the empirical marginal distributions in Figure 2. Compared to those in Figure 1, the empirical

Figure 2: Empirical Marginal Distributions under Different α with the Gibbs Sampler

marginal distributions are a good approximation to the true marginals when $\alpha < 6$, but become inaccurate when α is larger, and completely distorted when $\alpha > 13$.

What is the problem? From Figure 1, we know the model is bimodal when α is large. Once the Gibbs sampler is at one mode, it has to travel through several intermediate low probability states to reach the other mode. But this will happen with very small probability if α is large. Therefore the Gibbs sampler is essentially trapped in one of the modes during our experiment and generates samples with high variance. To see this, we fix $\alpha = 16$ and repeat the above process 20 times, each time starting from a random state and generating 100,000 samples. Then we plot the empirical marginal distribution of these 20 trials in Figure 3. What we see is the MCMC chain quickly converges to one of the two modes (perhaps during burn in), and stays there till the end of a trial.

(c) **CFTP**

Since coalescence generally takes long time, we don't have the luxury to generate as many samples as in the previous section. Instead, we generate only 100 samples for each model. We plot the empirical marginal distribution in Figure 4. Obviously the samples are much better than those obtained by the Gibbs sampler, even when α is large. We performed a χ^2 test at the level of significance $\alpha_0 = 0.10$ on the hypothesise that the samples come from their corresponding models, and accepted all the hypothesise except the one with $\alpha = 5$. Therefore it seems CFTP technique really helped to generate true samples from the models.

The price CFTP has to pay to generate true samples is long running time. The CFTP algorithm in section 4 doesn't give the exact number of iterations in generating a sample, but rather an upper bound rounded to the nearest power of

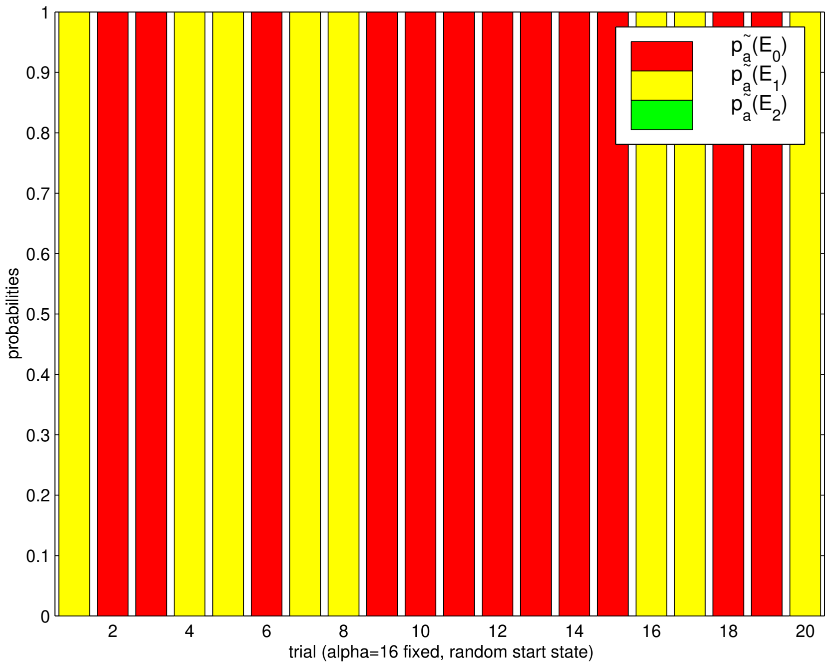

Figure 3: Empirical Marginal Distributions of 20 Trials with $\alpha = 16$ and the Gibbs Sampler

Figure 4: Empirical Marginal Distributions under Different α with CFTP

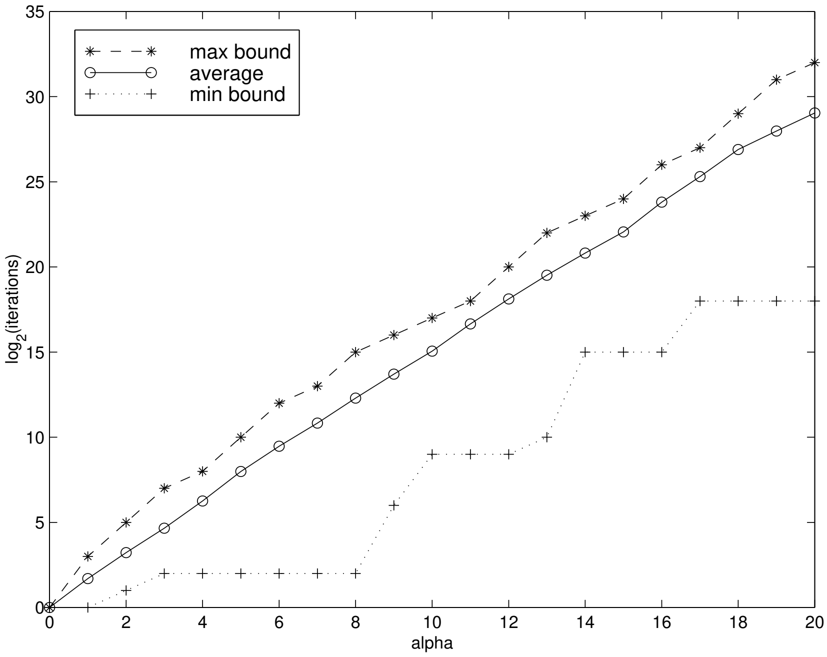

Figure 5: Maximal, Average, and Minimal Bounds on the Number of Iterations for CFTP to Generate One Sample

2. Since there is one bound for each sample and we generated 100 samples from each model, we can use the largest bound, average of bounds, and the smallest bound of these 100 samples as rough indicators of running time for each model. Figure 5 shows these bounds versus α, the 'difficulty' of the model. Note the Y-axis is in *log* scale, thus the time to generate one sample grows *exponentially* with the model parameter α. In the model where $\alpha = 20$, it takes an average of 2^{29} iterations to generate a sample, which will run for hours on a PC!

Now that we empirically know the number of iterations needed for CFTP to generate a sample, the chain we used in the Gibbs sampler appears to be too short. Indeed, we repeated the Gibbs sampler experiment with $\alpha = 16$ again, but instead of taking a sample after every 10 iterations, we take a sample after the maximal bound iterations, which is 2^{26} in this case. We generated 100 Gibbs samples with this long chain. Unlike the extreme behavior in Figure 3, exactly 50 samples we get are all-0 tree and the other 50 are all-1 tree. This confirms that once a MCMC chain is run *long enough*, it will generate the exact samples. Nevertheless in practice CFTP is always preferred if applicable because it eliminates the need to guess how long is enough.

(d) **The Metropolis-Hasting Sampler**

We run the Metropolis-Hasting sampler under the same setting as the original Gibbs sampler. That is, for each model we generate 100,000 samples by running a single chain. The first 1000 iterations in the chain is used as burn in. Then after every 10 iterations a sample is taken out. We then compute the empirical marginal distribution. Note we have an additional parameter λ which is the balance between being heuristic and exploitory.

Figure 6: Empirical Marginal Distributions with $\lambda = 1$, Metropolis-Hasting Sampler

Figure 6 shows the empirical marginal distributions for models with $\alpha = 0, \ldots, 20$ and $\lambda = 1$, i.e. the most exploratory proposal distribution. It has no heuristic at all, yet the result is very close to the true distribution. When we gradually add more heuristic to the proposal distribution (i.e. decrease λ), we get essentially the same result. Figure 7 shows the case where $\lambda = 0.1$, which is similar to Figure 6.

It should not be concluded that the Metropolis-Hasting sampler is in general better than CFTP. Our model has only 128 states, which is easily covered by the proposal distribution even when λ is small. In the MCMC chain each state is tried several times. Therefore it is not surprising that the Metropolis-Hasting sampler discovered both modes. In practical problems the state space is usually much larger and the performance of Metropolis-Hasting sampler may be different. Likewise, it should not be concluded that the heuristic doesn't help. This model is simply too small to judge the heuristic.

References

[1] J. Propp, D. Wilson. *Exact Sampling with Coupled Markov Chains and Applications to Statistical Mechanics* ??

[2] A. Berger, V. Della Pietra, S. Della Pietra. *A maximum entropy approach to natural language processing*. Computational Linguistics, 22, No. 1, 1996

[3] S. Della Pietra, V. Della Pietra, J. Lafferty. *Inducing Features of Random Fields*. IEEE PAMI, Vol. 19, No. 4, April 1997.

Figure 7: Empirical Marginal Distributions with $\lambda = 0.1$, Metropolis-Hasting Sampler

[4] A. Gelman, J.Carlin, H. Stern, D. Rubin. *Bayesian Data Analysis*. Chapman & Hall, London, 1995

[5] R. Neal. *Probabilistic Inference Using Markov Chain Monte Carlo Methods*. Tech. Rpt. CRG-TR-93-1, Dept. of Computer Science, University of Toronto, 1993