University of Nebraska - Lincoln
DigitalCommons@University of Nebraska - Lincoln

CSE Technical reports | Computer Science and Engineering, Department of

4-1-2004

An Analysis of Mcmc Sampling Methods for Estimating Weighted Sums in Winnow

Qingping Tao
University of Nebraska - Lincoln

Stephen Scott
University of Nebraska - Lincoln, sscott2@unl.edu

Follow this and additional works at: http://digitalcommons.unl.edu/csetechreports

Part of the Computer Sciences Commons

Tao, Qingping and Scott, Stephen, "An Analysis of Mcmc Sampling Methods for Estimating Weighted Sums in Winnow" (2004). *CSE Technical reports*. Paper 90.
http://digitalcommons.unl.edu/csetechreports/90

This Article is brought to you for free and open access by the Computer Science and Engineering, Department of at DigitalCommons@University of Nebraska - Lincoln. It has been accepted for inclusion in CSE Technical reports by an authorized administrator of DigitalCommons@University of Nebraska - Lincoln.

In Proceedings of Artificial Neural Networks in Engineering (ANNIE 2003), pages 15-20, St. Louis, MO, November 2003.

AN ANALYSIS OF MCMC SAMPLING METHODS FOR ESTIMATING WEIGHTED SUMS IN WINNOW

QINGPING TAO AND STEPHEN D. SCOTT
Department of Computer Science & Engineering
University of Nebraska
Lincoln, Nebraska

ABSTRACT
Chawla et al. introduced a way to use the Markov chain Monte Carlo method to estimate weighted sums in multiplicative weight update algorithms when the number of inputs is exponential. But their algorithm still required extensive simulation of the Markov chain in order to get accurate estimates of the weighted sums. We propose an optimized version of Chawla et al.'s algorithm, which produces exactly the same classifications while often using fewer Markov chain simulations. We also apply two other sampling techniques and empirically compare them with Chawla et al.'s Metropolis sampler to determine how effective each is in drawing good samples in terms of accuracy of weighted sum estimates and total time. We perform our analyses in the context of learning DNF formulas using littlestone's Winnow algorithm.

INTRODUCTION

In this paper, we consider as our application area the problem of learning general DNF formulas, which has been heavily studied in a learning-theoretic framework (e.g. Bshouty et al. (1999); Khardon et al. (2001)). Let $f = P_1 \vee P_2 \vee \ldots \vee P_K$ be the target function, where $P_i = c_{i1} \wedge c_{i2} \wedge \ldots \wedge c_{in}$ is a term and c_{ij} is a constraint on the value of attribute j. If attribute j takes on values from $\{1, \ldots, k_j\}$, then $c_{ij} = \ell \in \{1, \ldots, k_j\}$ means that for an example $\mathbf{x}$ to satisfy constraint c_{ij}, $x_j = \ell$. If $c_{ij} = 0$, then x_j can be any value from $\{1, \ldots, k_j\}$ and still satisfy the constraint. If $x_j = 0$, then this attribute value is unspecified and only satisfies when $c_{ij} = 0$. In other words, $\mathbf{x}$ satisfies P_i iff for all j, either $x_j = c_{ij}$ or $c_{ij} = 0$. The set of terms available for f and the instance space are both $\Omega = \prod_{j=0}^{n-1}\{0, \ldots, k_j\}$. If example $\mathbf{x}$ has $n_{\mathbf{x}}$ specified values, then there are exactly $2^{n_{\mathbf{x}}}$ terms satisfied by it.

One useful method for solving this problem is using multiplicative weight update (MWU) algorithms such as Winnow (Littlestone, 1988). The learning of Winnow proceeds in *trials*. At trial t, Winnow receives an input vector $\mathbf{x}'_t$ and makes a prediction $\hat{y}_t = 1$ if $W_t = \mathbf{w}_t \cdot \mathbf{x}'_t \geq \theta$ and 0 otherwise, where $\mathbf{w}_t$ is its weight vector and θ is the threshold. Then Winnow updates its weight vector as follows: $w_{t+1,i} = w_{t,i}\alpha^{x'_{t,i}(y_t - \hat{y}_t)}$ for some learning rate $\alpha > 1$, where y_t is the true label. We call it a *promotion* if $w_{t+1,i} > w_{t,i}$, and *demotion* if $w_{t+1,i} < w_{t,i}$. Winnow can be used to learn DNF formulas by using all possible terms as its inputs: e.g. if $k_i = 2$ for all i, there will be $N = |\Omega| = 3^n$ possible terms. Input x'_i to Winnow is 1 if $\mathbf{x}$ satisfies term i and 0 otherwise.

The advantage of using Winnow is that the number of mistakes that Winnow will make on any sequence of examples is at most $O(K \log N) = O(Kn)$, where K is the number of terms in the target disjunction (Littlestone, 1988). But the disadvantage is that brute-force computation of $W_t = \mathbf{w}_t \cdot \mathbf{x}'_t$ is takes time $\Omega(N)$, which is exponential in n. Thus another approach is needed. One possibility is to use kernels, as illustrated by Khardon et al. (2001) for the Perceptron algorithm (i.e. using additive weight updates). However, while they showed that it is possible to efficiently compute the weighted sum for Perceptron when learning DNF, in the worst case, their kernel-based algorithm makes $2^{\Omega(n)}$ prediction

University of Nebraska-Lincoln, Computer Science and Engineering
Technical Report TR-UNL-CSE-2004-0007, published 04/01/2004

mistakes. They also argued that unless P = #P, it is impossible to efficiently exactly simulate Winnow for learning DNF. Thus we look to Chawla et al. (2003), who use MCMC methods to *estimate* W_t for Winnow *with high probability*, as opposed to Khardon et al.'s hardness result that says no *deterministic* simulation of Winnow is possible for DNF.

Although Chawla et al.'s preliminary empirical results are much stronger than what their theoretical results imply, they still required extensive simulation of the Markov chain in order to get accurate estimates of the weighted sums. This significantly slowed their algorithm. Here we propose an optimized version of Chawla et al.'s algorithm, which often uses less computation time without any loss in classification accuracy. We also empirically compare two MCMC techniques (Gibbs and Metropolized Gibbs) to Chawla et al.'s Metropolis sampler in terms of accuracy of weighted sum estimates.

MARKOV CHAIN MONTE CARLO METHODS

Markov chain Monte Carlo methods have been widely used to solve problems in e.g. statistical physics and Bayesian statistical inference. They can provide approximations to quantities by performing statistical sampling experiments. Generally, to apply MCMC methods, a Markov chain M with state space Ω and stationary distribution π is designed to be *ergodic*, that is, the probability distribution over Ω converges asymptotically to π regardless of the initial state. Then M is repeatedly simulated for T steps and generates S samples almost according to π. These S samples are then used to estimate the quantity of interest. Usually we discard states met in the first T_0 steps and assume there would be a rapid convergence to π during these steps. This T_0-step procedure is called *burn-in*. After burn-in, M would be simulated for another S steps and each additional step would generate a new sample. The most commonly used MCMC method is the Metropolis sampler (Metropolis et al., 1953). In each step, the Metropolis sampler repeatedly considers randomly generated changes to one of the components of the current state P and accepts the new state Q with probability $\min\{1, \frac{\pi(Q)}{\pi(P)}\}$.

CHAWLA ET AL.'S MCMC SOLUTION FOR WINNOW

We now describe Chawla et al.'s solution for estimating $W_t(\alpha) = \sum_{P\in\Omega_t} w_{t,P}(\alpha)$, where $w_{t,P}(\alpha) = \alpha^{\varpi_t(P)}$ is term P's weight of training Winnow with learning rate α, and $\varpi_t(P) = u_t(P) - v_t(P)$, where $u_t(P)$ is the total number of promotions of term P at time t and $v_t(P)$ is the total number of demotions.

Let $\Omega_t \subset \Omega$ be the set of 2^{n_t} terms that are satisfied by example $\mathbf{x}_t$ ($n_t \leq n$ is the number of non-zero values in $\mathbf{x}_t$). For each term $P = (p_1, \ldots, p_{n_t}) \in \Omega_t$, $p_i = 1$ if constraint $c_{p_i} > 0$ and otherwise $p_i = 0$. Then a set of Markov chains $\mathcal{M}_t$ is built on the state space Ω_t. Each chain $M_t(\alpha') \in \mathcal{M}_t$ has a specific learning rate α' and a stationary distribution $\pi_{\alpha',t}(P) = \alpha'^{\varpi_t(P)}/W_t(\alpha')$.

Chawla et al. then define $f_{i,t}(P) = \frac{w_{t,P}(\alpha_{i-1,t})}{w_{t,P}(\alpha_{i,t})}$, where $\alpha_{i,t} = (1 + \frac{1}{m_t})^{i-1}$ for $1 \leq i < r_t$, $\alpha_{r_t,t} = \alpha$, r_t is the smallest integer such that $(1 + \frac{1}{m_t})^{r_t-1} \geq \alpha$, and $m_t = u_t(P_e) + v_t(P_e)$ where $P_e = (0, 0, \ldots, 0)$. Because $\mathrm{E}[f_{i,t}(P)] = \frac{W_t(\alpha_{i-1,t})}{W_t(\alpha_{i,t})}$, $\frac{W_t(\alpha_{i-1,t})}{W_t(\alpha_{i,t})}$ can be estimated by computing the sample mean of $f_{i,t}(P)$, which allows $W_t(\alpha)$ to be computed since $W_t(\alpha) = \left(\frac{W_t(\alpha_{r,t})}{W_t(\alpha_{r-1,t})}\right) \cdots \left(\frac{W_t(\alpha_{2,t})}{W_t(\alpha_{1,t})}\right) W_t(\alpha_{1,t})$ and $W_t(\alpha_{1,t}) = W(1) = |\Omega_t| = 2^{n_t}$. Therefore, for each value $\alpha_{2,t}, \ldots, \alpha_{r,t}$, S_t samples are drawn through $M_t(\alpha_{i,t})$. Let $X_{i,t}$ be the sample mean of $f_{i,t}(P)$ and $|\mathcal{M}_t| = r_t - 1$, then the estimate of $W_t(\alpha)$ is $\hat{W}_t(\alpha) = 2^{n_t} \prod_{i=2}^{r_t} 1/X_{i,t}$.

OUR OPTIMIZED MCMC SOLUTION

In Chawla et al.'s MCMC solution, $r_t - 1$ Markov chains need to be simulated. Here we give an optimized solution that is based on the idea that to exactly simulate Winnow, we only need to know what Winnow's prediction is going to be (i.e. on what side of the threshold θ that W will fall on), not what the weighted sum exactly is. So it is possible that we could stop computing our estimate after a subset of the chains in $\mathcal{M}_t$ have been run.

Let $\wp_t^{max} = \max_{P\in\Omega_t}\{u_t(P) - v_t(P)\}$, $\wp_t^{min} = \min_{P\in\Omega_t}\{u_t(P) - v_t(P)\}$, i.e. the maximum and minimum number of net promotions, and $\Psi_t = \{2, \cdots, r_t\}$. We define the following two constraints:

$$\mathcal{C} \prod_{i\in\Psi'} \frac{W_t(\alpha_{i,t})}{W_t(\alpha_{i-1,t})} \left(\frac{\alpha_{i,t}}{\alpha_{i-1,t}}\right)^{-\wp_t^{min}} > \theta, \tag{1}$$

$$\mathcal{D} \prod_{i\in\Psi'} \frac{W_t(\alpha_{i,t})}{W_t(\alpha_{i-1,t})} \left(\frac{\alpha_{i,t}}{\alpha_{i-1,t}}\right)^{-\wp_t^{max}} < \theta, \tag{2}$$

where $\mathcal{C} = 2^{n_t} \prod_{i=2}^{r_t} \left(\frac{\alpha_{i,t}}{\alpha_{i-1,t}}\right)^{\wp_t^{min}}$ and $\mathcal{D} = 2^{n_t} \prod_{i=2}^{r_t} \left(\frac{\alpha_{i,t}}{\alpha_{i-1,t}}\right)^{\wp_t^{max}}$. Then we can prove the following theorem.

Theorem 1 *If $\exists\Psi' \subseteq \Psi_t$ that satisfies constraint (1) , then $W_t(\alpha) > \theta$; If $\exists\Psi' \subseteq \Psi_t$ that satisfies constraint (2), then $W_t(\alpha) < \theta$.*

Proof Sketch: Because $\alpha_{i,t} > \alpha_{i-1,t} > 0$ and $\varpi_t(P) - \wp_t^{min} \geq 0$ for all $P \in \Omega_t$, $\sum_{P\in\Omega_t} \alpha_{i,t}^{\varpi_t(P)-\wp_t^{min}} \geq \sum_{P\in\Omega'_t} \alpha_{i-1,t}^{\varpi_t(P)-\wp_t^{min}}$. So $\frac{W_t(\alpha_{i,t})}{W_t(\alpha_{i-1,t})} \left(\frac{\alpha_{i,t}}{\alpha_{i-1,t}}\right)^{-\wp_t^{min}} \geq 1$. Then $W_t(\alpha) = \mathcal{C} \prod_{i=2}^{r_t} \frac{W_t(\alpha_{i,t})}{W_t(\alpha_{i-1,t})} \left(\frac{\alpha_{i,t}}{\alpha_{i-1,t}}\right)^{-\wp_t^{min}} \geq \mathcal{C} \prod_{i\in\Psi'} \frac{W_t(\alpha_{i,t})}{W_t(\alpha_{i-1,t})} \left(\frac{\alpha_{i,t}}{\alpha_{i-1,t}}\right)^{-\wp_t^{min}} > \theta$. Similarly we can prove the second statement. □

Theorem 1 tells us that it would not always be necessary to run all $r_t - 1$ Markov chains if we were only interested in Winnow's predictions. Instead, we can sometimes limit our simulations to a subset of Markov chains. So what we want is to find such a subset with the smallest size.

Let $\Gamma_1(\Psi_t)$ be the set of all Ψ' that satisfy constraint (1), and $\Gamma_0(\Psi_t)$ be the set of all Ψ' that satisfy constraint (2). We define $\Psi_1^{min} \in \Gamma_1(\Psi_t)$ as a *minimum 1-prediction set* if $|\Psi_1^{min}| \leq |\Psi'|$ for all $\Psi' \in \Gamma_1(\Psi_t)$, and $\Psi_0^{min} \in \Gamma_0(\Psi_t)$ as a *minimum 0-prediction set* if $|\Psi_0^{min}| \leq |\Psi'|$ for all $\Psi' \in \Gamma_0(\Psi_t)$. Then we give Theorem 2.

Theorem 2 *If Ψ_1^{min} exists, $\{r_t, r_t - 1, \cdots, r_t - |\Psi_1^{min}| + 1\}$ is a minimum 1-prediction set, and if Ψ_0^{min} exists, $\{2, 3, \cdots, |\Psi_0^{min}| + 1\}$ is a minimum 0-prediction set.*

Proof Sketch: Using Cauchy's inequality, we can prove that $\frac{W_t(\alpha_{i+1,t})}{W_t(\alpha_{i,t})} \geq \frac{W_t(\alpha_{i,t})}{W_t(\alpha_{i-1,t})}$ for $\forall i, 2 \leq i \leq r_t - 1$. Then let $\Psi' = \{r_t, r_t - 1, \cdots, r_t - |\Psi_1^{min}| + 1\}$,

$$\mathcal{C} \prod_{i\in\Psi'} \frac{W_t(\alpha_{i,t})}{W_t(\alpha_{i-1,t})} \left(\frac{\alpha_{i,t}}{\alpha_{i-1,t}}\right)^{-\wp_t^{min}} \geq \mathcal{C} \prod_{i\in\Psi_1^{min}} \frac{W_t(\alpha_{i,t})}{W_t(\alpha_{i-1,t})} \left(\frac{\alpha_{i,t}}{\alpha_{i-1,t}}\right)^{-\wp_t^{min}} > \theta.$$

So $\{r_t, r_t - 1, \cdots, r_t - |\Psi_1^{min}| + 1\}$ is a minimum 1-prediction set. Similarly we can prove $\{2, 3, \cdots, |\Psi_0^{min}| + 1\}$ is a minimum 0-prediction set. □

According to Theorem 2, if $W_t(\alpha) > \theta$ and we simulate Markov chains in the order of $r_t, r_t - 1, \cdots, 2$, and halt when constraint (1) is satisfied, we need no more computation than any other sequence of Markov chains in $\mathcal{M}_t$. Similarly we use no more chains than any other sequence if $W_t(\alpha) < \theta$ and we simulate them in the order of $2, 3, \cdots, r_t$ and check if constraint (2) is satisfied. Then we get an optimized MCMC solution for Winnow, which chooses one of the two orders ($\{r_t, \cdots, 2\}$ or $\{2, \cdots, r_t\}$) by guessing the most likely prediction y'_t. When we use Winnow to predict, we could just assume y'_t is 1. When we are training Winnow, we can set y'_t as the class label of training example $\mathbf{x}$. But a better way is that at the t-th training iteration let $y'_t = \hat{y}_{t-1}(\mathbf{x})$, where $\hat{y}_{t-1}$ is the prediction of $\mathbf{x}$ at the $(t-1)$-th iteration. The heuristic is that the weighted sum of $\mathbf{x}$ might not change too much after the last time Winnow met $\mathbf{x}$. At the beginning of training, all weights of Winnow are 1. So $W_1(\alpha) = 2^{n_t}$ for all examples. If $2^{n_t} \geq \theta$, $y'_1 = 1$, otherwise 0.

SAMPLING FROM $\pi_{\alpha,t}$

In Chawla et al.'s MCMC solution, the computation time of estimating $W_t(\alpha)$ depends on the number of chains $r_t - 1$, the number of burn-in steps T_0 and the sample size S. Besides our optimized solution, another way to reduce the computation time is choosing relatively small T_0 and S. This could be achieved by using a good sampler that can efficiently draw samples from the desired distribution. Chawla et al. (2003) applied the Metropolis sampler to their MCMC solution. Here we adapt two other MCMC sampling techniques, Gibbs sampler and Metropolized Gibbs sampler.

Gibbs Sampler for Winnow

The Gibbs sampler (Geman & Geman, 1984) has a number of distinct features. In a single step of the Gibbs sampler, each component is replaced with a value picked from its distribution conditional on the current values of all other components. The conditional distributions are constructed on prior knowledge of π. Furthermore, the Gibbs sampler is, by construction, multidimensional. It generates new values for all components and only after that it outputs a sample.

For any state $P \in \Omega_t$, each variable p_i only has two possible values, 0 and 1. If $P_0 = (p_1, \cdots, p_i = 0, \cdots, p_{n_t})$ and $P_1 = (p_1, \cdots, p_i = 1, \cdots, p_{n_t})$, the conditional distribution is $\pi_{\alpha,t}(p_i|P\backslash\{p_i\}) = \frac{\pi_{\alpha,t}(P)}{\pi_{\alpha,t}(P_0)+\pi_{\alpha,t}(P_1)}$. Then we get the conditional distribution of the Gibbs sampler for $M_{DNF,t}(\alpha)$, that is,

$$\pi_{\alpha,t}(0|P\backslash\{p_i\}) = 1/(1+\alpha^{\varpi_t(P_1)-\varpi_t(P_0)}) \quad (3)$$

$$\pi_{\alpha,t}(1|P\backslash\{p_i\}) = 1/(1+\alpha^{\varpi_t(P_0)-\varpi_t(P_1)}). \quad (4)$$

Metropolized Gibbs Sampler for Winnow

The Metropolized Gibbs sampler (Liu, 1996) is a modification of the Gibbs sampler, which has been proven to be statistically more efficient than the Gibbs sampler. The sampler draws a new value p'_i with probability $\frac{\pi_{\alpha,t}(p'_i|P\backslash\{p_i\})}{1-\pi_{\alpha,t}(p_i|P\backslash\{p_i\})}$, and accepted with the Metropolis-Hasting acceptance probability $\min\{1, \frac{1-\pi_{\alpha,t}(p_i|P\backslash\{p_i\})}{1-\pi_{\alpha,t}(p'_i|P\backslash\{p_i\})}\}$.

As mentioned before, each component in state P only has two possible values. So the Metropolized Gibbs sampler becomes a Metropolis sampler that repeatedly updates all components in a fixed order. We then build the Metropolized Gibbs sampler for $M_t(\alpha)$ that updates each component p_i in P one by one with a value p'_i randomly drawn from 0 and 1, and accept it with the acceptance probability

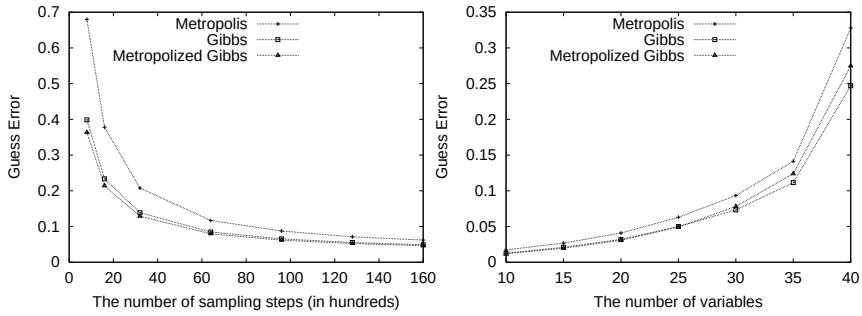
$$\min\{1, \alpha^{\varpi_t(P\backslash\{p_i\}\cup\{p'_i\})-\varpi_t(P)}\}. \quad (5)$$

EXPERIMENTAL RESULTS

In our experiments, we compared three MCMC sampling techniques (Metropolis, Gibbs and Metropolized Gibbs) on two types of data (simulated data used by Chawla et al. (2003) and UCI data sets). We then compared the computation costs of our optimized solution with Chawla et al.'s algorithm in term of the total number of Markov chain simulations. In all experiments, we set the burn-in time $T_0 = n^2$ because it takes about T^2 steps to move to a state T steps away due to the random walk nature of MCMC samplers (Neal, 1995). Our experiments showed that this burn-in time worked well. In each experiment, we also counted each update of a single component of current state as a single sampling step. We used the same number of sampling steps T_s for all three samplers.

Comparisons on Estimating Weighted Sums

To evaluate how well the weight estimation procedures with different samplers guessed the weighted sums, we compared them using the measure *Guess Error* given in Chawla et al. (2003), which is the average error of the estimates ($|\hat{W} - W|/W$). We trained Winnow for 20 rounds on 10 partitions of Voting data while varying the sampling time T_s. Figure 1(a) shows averages over more than 70000 estimates obtained from the experiment. For all T_s, Guess Errors of Gibbs and Metropolized Gibbs are always lower than Metropolis. Although Metropolized Gibbs has a lower Guess Error than Gibbs, the difference between these two samplers is very small, especially when $T_s = 9600, 12800, 16000$. To evaluate the effect of varying n, we measured Guess Error of all three samplers on the simulated data. In Figure 1(b), T_s is fixed at 10000. Gibbs and Metropolized Gibbs are still always better than Metropolis. The Guess Error of Metropolized Gibbs is lower than Gibbs when $n \leq 25$. But when $n > 25$, Gibbs becomes the best sampler. Considering results in Figures 1, we can say that Gibbs sampler is the best choice on average in terms of the accuracy of estimating weighted sums.



(a) Guess Error vs. T_s on Voting **(b) Guess Error vs. n on simulated data.**

Figure 1: Comparison on Guess Error.

Comparisons on Computation Cost

Here we report speedups of our optimized algorithm over Chawla et al.'s solution in terms of the total numbers of Markov chains that are used. Table 3 summarizes average results of 10-fold cross-validation on simulated data with 20 training rounds and five UCI data sets with 100 training rounds. In Table 3, "MCMC" is the number of chains used by Chawla et al.'s solution. "Opt MCMC" is the number of chains used by our algorithm. "Savings" is the percentage of chains our algorithm saved. Results in Table 3 confirms that we do not need to run all chains sometimes. However, for different data sets, our algorithm

has different performance. A possible reason is that weighted sums of Winnow are much less or greater than threshold for some data sets. Thus, there are probably more chances for our algorithm to stop early.

Table 3: Comparisons on the total number of Markov chains (in thousands).

(a) Simulated data

n	MCMC	Opt MCMC	Savings(%)
10	4.6	4.1	11.3
15	11.9	11.2	6.3
20	20.8	18.7	10.0
25	23.5	21.2	9.7
30	40.8	37.5	8.1
35	58.7	53.0	9.7
40	60.9	53.2	12.7

(b) UCI data sets

Data Sets	MCMC	Opt MCMC	Savings(%)
iris	217.7	200.1	8.0
car	14272.7	14032.0	1.7
breast cancer	45176.2	45111.5	0.2
voting	1185.5	1132.0	4.5
auto	13822.8	13751.7	0.5

CONCLUSIONS

We proposed an optimized MCMC solution for estimating weighted sums in Winnow, which often uses less computation time than Chawla et al.'s solution without any loss of classification accuracy. Our experimental results confirmed that our algorithm only needs to use a subset of all Markov chains implied by the original solution. We showed how to get such a subset with the smallest size. We also empirically applied two MCMC sampling techniques: Gibbs and Metropolized Gibbs. They all showed better performance than Chawla et al.'s Metropolis sampler in terms of accuracy of weighted sum estimates.

ACKNOWLEDGMENTS

This work was funded in part by NSF grants CCR-0092761 and EPS-0091900 and a grant from the University of Nebraska Foundation. It was also supported in part by NIH Grant Number RR-P20 RR17675 from the IDeA program of the National Center for Research Resources. This work was completed in part utilizing the Research Computing Facility of the University of Nebraska.

REFERENCES

Bshouty, N. H., Jackson, J., & Tamon, C., 1999, "More efficient PAC-learning of DNF with membership queries under the uniform distribution," *Proc. of the 12th Annual Conf. on Computational Learning Theory*, pp. 286–295.

Chawla, D., Li, L., & Scott, S. D., 2003, "*On approximating weighted sums with exponentially many terms,*" (Technical Report TR03-02-02). Univ. of Nebraska-Lincoln. Early version in COLT'01.

Geman, S. & Geman, D., 1984, "Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 6, pp. 721-741.

Khardon, R., Roth, D., & Servedio, R., 2001, "Efficiency versus convergence of boolean kernels for online learning algorithms," *Advances in Neural Information Processing Systems 14*.

Littlestone, N., 1988, "Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm," *Machine Learning*, vol 2, pp. 285–318.

Liu, J.S., 1996, "Peskun's Theorem and A Modified Discrete-State Gibbs Sampler," *Biometrika*, vol. 83, pp. 681-682.

Metropolis N., Rosenbluth A.W., Rosenbluth M.N., Teller A.H. and Teller E., 1953, "Equations of state calculations by fast computing machines," *J. of Chemical Physics*, vol. 21, pp. 1087-1091.

Neal, R. M., 1995, "*Suppressing random walks in Markov chain Monte Carlo using ordered overrelaxation,*" (Technical Report No. 9508). Dept. of Statistics, University of Toronto.