The Fortuin-Kastelyn Representation

Consider a classical Ising Model of N spins:

$$H = -\sum_{i,j} J_{ij}\sigma_i\sigma_j$$

where $\sigma_i = \pm 1$, and J is a strictly upper triangular matrix. From here on, we will assume that the interactions are ferromagnetic ($J_{ij} \geq 0$).

The partition function is:

$$\begin{aligned} Z &= \sum_\sigma \left[\exp\left(\beta \sum_{i,j} J_{ij}\sigma_i\sigma_j\right)\right] \\ &= \sum_\sigma \left[\prod_{i,j} \exp(\beta J_{ij}\sigma_i\sigma_j)\right] \end{aligned}$$

We note that each factor in the product can only take one of two forms:

$$\begin{aligned} \exp\left(\beta J_{ij}\sigma_i\sigma_j\right) = &\left\{ \begin{array}{ll} \exp(\beta J_{ij}) & \text{if } \sigma_i = \sigma_j \\ \exp(-\beta J_{ij}) & \text{otherwise} \end{array} \right\} \\ &= \exp(\beta J_{ij})\delta_{\sigma_i\sigma_j} + \exp(-\beta J_{ij})\delta_{\sigma_i,-\sigma_j} \\ &= \exp(\beta J_{ij})\left[\delta_{\sigma_i\sigma_j} + \exp(-2\beta J_{ij})\delta_{\sigma_i,-\sigma_j}\right] \\ &= \exp(\beta J_{ij})\left[\delta_{\sigma_i\sigma_j} + \exp(-2\beta J_{ij})(1-\delta_{\sigma_i\sigma_j})\right] \\ &= \exp(\beta J_{ij})\left[\exp(-2\beta J_{ij}) + (1-\exp(-2\beta J_{ij}))\delta_{\sigma_i\sigma_j}\right] \end{aligned}$$

Let $p_{ij} = 1 - \exp(-2\beta J_{ij})$, the partition function is then:

$$Z = \sum_\sigma \left[\prod_{i,j} \left\{\exp(\beta J_{ij})\left[(1-p_{ij}) + p_{ij}\delta_{\sigma_i\sigma_j}\right]\right\}\right]$$

Next we introduct binary "bond" variables $b_{ij} = 0, 1$ which indicate whether the spins are aligned or not:

$$\begin{aligned} Z &= \sum_\sigma \prod_{i,j} \sum_{b_{ij}} \left\{\exp(\beta J_{ij})\left[(1-p_{ij})\delta_{b_{ij},0} + p_{ij}\delta_{\sigma_i\sigma_j}\delta_{b_{ij},1}\right]\right\} \\ &= \sum_\sigma \prod_{i,j} \sum_{b_{ij}} W(\sigma_i, \sigma_j, b_{ij}) \end{aligned}$$

It is clear that performing the sum over b_{ij} recovers the same expression as above. While local Monte-Carlo updates like Metropolis or Gibbs/heat-bath sample from $p(\sigma)$, Swendsen-Wang/Wolff sample the joint distribution $p(\sigma, \{b_{ij}\})$.

The Swendsen-Wang Algorithm

The Swendsen-Wang algorithm is a cluster update algorithm based on the Fortuin-Kastelyn representation of the partition function. The update procedure is:

1. For all pairs of spins (σ_i, σ_j), set $b_{ij} = 1$ with probability $P_{ij} = p_{ij}\delta_{\sigma_i\sigma_j}$. This is usually done by first checking whether the spins are aligned, and then setting $b_{ij} = 1$ with probability $p_{ij} = 1 - \exp(-2\beta J_{ij})$.
2. Inspect $\{b_{ij}\}$ to identify clusters of spins
3. Flip clusters independently with probability $\frac{1}{2}$.

In the case of nearest-neighbor interactions, each Monte-Carlo step takes $O(N)$ time, where N is the number of spins. However, in the case of long-range interactions, the first part would end up taking $O(N^2)$ time.

The Luijten-Blöte Algorithm

In the Swendsen-Wang we check that $\sigma_i = \sigma_j$ first, and then sample $b_{ij} \sim \text{Bernoulli}(p_{ij})$. Luijten and Blöte proposed doing this in reverse, that is, sample the candidate bonds first, and then activate the candidate bonds based on whether the spins are aligned. This could potentially speed up the first step of the SW procedure, since we'd no longer have to iterate through every pair of spins.

To show that we will indeed need to iterate over significantly fewer bonds, we compute the expected number of candidate bonds for a d-dimensional Ising Model:

$$\begin{aligned}
\sum_{ij} p_{ij} &= \sum_{ij}(1 - \exp(-2\beta J_{ij})) \\
&\sim \sum_i \int_1^{N^{1/d}} r^{d-1}(1 - \exp(-2\beta J(r)))dr \\
&= N \int_1^{N^{1/d}} r^{d-1}(1 - \exp(-2\beta J(r)))dr \\
&\sim 2\beta N \int_1^{N^{1/d}} r^{d-1} J(r)dr
\end{aligned}$$

In the second line we assumed that the interaction strength depended only on the distance between the two sites, and in the third we assumed translational invariance. In the last line we Taylor expanded the exponential, assuming that $J(r)$ is small. The integral converges to a finite constant as $N \to \infty$ provided that $J(r)$ decays faster that r^{-d}. Hence, the expected number of candidate bonds scales as $O(N)$. [**?**, **?**]

The Luijten-Blöte algorithm gives a way to sample the candidate bonds. We begin by enumerating all possible bonds as $m = 1, 2, \dots, N_b$.

$$p_m = 1 - \exp(-2\beta J_m)$$

where J_m is the interaction strength of the mth bond. Define the distribution:

$$q_m^{(0)} = p_m \prod_{l=1}^{m-1}(1 - p_l)$$

which is the probability that the mth bond is chosen after the first $m - 1$ bonds were rejected. After the mth bond is sampled from $q^{(0)}$, we then sample n from

$$q_n^{(m)} = p_n \prod_{l=m+1}^{n-1}(1 - p_l)$$

and repeat until we've considered all possible candidate bonds.

Naively, we'd expect that we'd need to build $N_b \sim N^2$ different distributions $q_n^{(m)}$, each of which is a vector of size at most N_b. This would require $O(N_b^2) = O(N^4)$ memory. However, we can use the fact that $q_n^{(m)}$ can be easily constructed from $q_m^{(0)}$. Consider the cumulative distributions: $C_m^{(0)} = \sum_{l=1}^m q_l^{(0)}$ and $C_n^{(m)} = \sum_{l=m+1}^n q_l^{(m)}$

Observe:

$$\begin{aligned}
C_n^{(0)} - C_m^{(0)} &= \sum_{l=1}^{n} q_l^{(0)} - \sum_{l=1}^{m} q_l^{(0)} = \sum_{l=m+1}^{n} q_l^{(0)} = \sum_{l=m+1}^{n} p_l \prod_{i=1}^{l-1}(1-p_i) \\
&= \sum_{l=m+1}^{n} p_l \prod_{i=m+1}^{l-1}(1-p_i) \prod_{i=1}^{m}(1-p_i) \\
&= \sum_{l=m+1}^{n} q_l^{(m)} \prod_{i=1}^{m}(1-p_i) \\
&= \sum_{l=m+1}^{n} q_l^{(m)} \frac{q_{m+1}^{(0)}}{p_{m+1}} \\
&= C_n^{(m)} \frac{q_{m+1}^{(0)}}{p_{m+1}} \\
\implies C_n^{(m)} &= \frac{p_{m+1}}{q_{m+1}^{(0)}} \left(C_n^{(0)} - C_m^{(0)} \right)
\end{aligned}$$

Note that there's a typo in the indices of the above expression in the Fukui-Todo paper. The above identity allows us to avoid having to store every $q_n^{(m)}$ and it's CDF. We can hence sample from $C_n^{(m)}$ easily by transforming the random variable.

How to sample from a finite discrete distribution

Given a probability mass function

$$p : \{1, \ldots, M\} \mapsto [0, 1]$$

In order to draw samples according to this distribution, we first need to compute the cumulative distribution F, which is just the cumulative sum of $\{p_i\}$ (this takes $O(M)$ time and space, but we only need to compute it once).

To sample from p, we draw $u \sim U[0, 1]$, and then find m such that

$$F_{m-1} \leq u \leq F_m$$

Naively searching through F for such an m would take linear time. However, we can accelerate this process by using a binary search, giving $O(\log M)$ time per sample!

We may use this procedure directly to draw a sample from $q_n^{(m)}$ using its CDF $C_n^{(m)}$ when $m = 0$. In order to sample from the distributions where $m \neq 0$, we draw $u \sim U[0, 1]$ and find n such that

$$\begin{aligned}
C_{n-1}^{(m)} \leq \quad & u & \leq C_n^{(m)} \\
\frac{p_{m+1}}{q_{m+1}^{(0)}} \left(C_{n-1}^{(0)} - C_m^{(0)} \right) \leq \quad & u & \leq \frac{p_{m+1}}{q_{m+1}^{(0)}} \left(C_n^{(0)} - C_m^{(0)} \right) \\
C_{n-1}^{(0)} \leq C_m^{(0)} - & \frac{q_{m+1}^{(0)}}{p_{m+1}} u & \leq C_n^{(0)}
\end{aligned}$$

So we just need to sample $u \sim U[0,1]$, apply the above transformation, and then use the transformed random variable as the input of the binary search.

The Luijten-Blöte algorithm will perform one search per candidate bond. Hence, the expected run-time of one MC step will be $O(N \log N)$ in the case where $J(r)$ decays faster than r^{-d}.

Fukui-Todo algorithm

We begin by extending the Fortuin-Kastelyn representation of the partition function. Replacing the binary bond variables $b_{ij} = 0, 1$ with $k_{ij} = 0, 1, 2, \ldots$ such that $P(b_{ij} = 0|\sigma_i = \sigma_j) = P(k_{ij} = 0|\sigma_i = \sigma_j)$ and $P(b_{ij} = 1|\sigma_i = \sigma_j) = P(k_{ij} > 0|\sigma_i = \sigma_j)$. Demanding that k_{ij} be a Poisson random variable satisfies all the desired properties.

Recall the Poisson distribution with mean λ:

$$f(k; \lambda) = \frac{e^{-\lambda}}{k!}\lambda^k$$

Considering all bonds $k_{ij} > 0$ as “activated”, gives:

$$P(k_{ij} > 0|\sigma_i = \sigma_j) = \sum_{k_{ij}=1}^{\infty} f(k_{ij}; \lambda) = 1 - \exp(-\lambda)$$

Hence we define $\lambda_{ij} = 2\beta J_{ij}$. This yields the extended Fortuin-Kastelyn representation of the partition function:

$$Z = \sum_{\sigma} \prod_{i,j} \sum_{k_{ij}=0}^{\infty} \frac{\exp(-2\beta J_{ij})}{k_{ij}!} (2\beta J_{ij})^{k_{ij}} \left(\delta_{k_{ij},0} + (1 - \delta_{k_{ij},0})\delta_{\sigma_i,\sigma_j}\right)$$

As an aside, this representation allows us to derive new estimators for certain quantities, such as the energy and specific heat:

$$E = \sum_l J_l - \frac{1}{\beta} \left\langle \sum_l k_l \right\rangle_{MC}$$

$$C = \frac{1}{N} \left[\left\langle \left(\sum_l k_l \right)^2 \right\rangle_{MC} - \left\langle \sum_l k_l \right\rangle_{MC}^2 - \left\langle \sum_l k_l \right\rangle_{MC} \right]$$

An interesting feature of these estimators is that one does not need to iterate over all bonds, just the ones that are active at each MC step, which reduces the runtime from $O(N_b)$ to $O(N)$ for $J(r)$ decaying faster than r^{-d}. Note that in the expectations above, $\sum_l k_l \neq k_t$ as we've coupled k_{ij} with the spin variables.

Generating a Poisson random variable for each bond gives us an alternative, but equivalent, method of activating bonds between parallel spins. However, generating a sample from a Poisson distribution takes $O(\lambda)$ time, so it is of course not practical to generate a Poisson variable for each candidate bond separately. Instead, we will use the properties of the Poisson distribution to combine all of these bond variables into one, reducing the number of Poisson random samples to just one per MC step.

Consider the joint probability of N_b bond variables k_l where l is the bond index:

$$\prod_{l=1}^{N_b} f(k_l; \lambda_l) = f(k_t; \lambda_t)(k_t!) \prod_{l=1}^{N_b} \frac{1}{k_l!} \left(\frac{\lambda_l}{\lambda_t} \right)^{k_l}$$

where $\lambda_t = \sum_l \lambda_l = 2\beta \sum_l J_l$ and $k_t = \sum_l k_l$. The right-hand side of the above expression is the probability of sampling the Poisson random variable k_t with mean λ_t times the probability of sampling $\{k_l\}_l$ from a multinomial distribution with k_t trials and category probabilities given by $\{\lambda_l/\lambda_t\}_l$.

The first part of each Monte Carlo step is now:

a. Initialize all bonds $k_{ij} = 0$
b. Draw $k_t \sim \text{Poisson}(\lambda_t)$
c. Repeat k_t times: Sample a bond $l = (i, j)$ from $p_l = \lambda_l/\lambda_t = J_l/\sum_l J_l$. If $\sigma_i = \sigma_j$, increment k_{ij} by one.

After this, proceed just like the regular Swendsen-Wang procedure: identify all clusters, then flip them independently with probability $\frac{1}{2}$. The time complexity of step b is $O(\lambda_t) \sim O(\beta \sum_{ij} J)$ which for Ising Models with $J(r)$ decaying faster than r^{-d} is $O(\beta N)$. The time complexity of step c is dependent on how we choose to sample from the distribution p_l. Using the fastest method discussed so far (binary search), we'll need $O(\log N_b) \sim O(\log N^2) \sim O(\log N)$ time. This is hardly an improvement over the Luijten-Blöte algorithm. However, there exist even *faster* methods to sample from a discrete probability distribution. In fact, a method exists which allows us to do so in constant time!

Walker's Alias Method

Given a probability distribution with M events, the main idea behind the Alias Method is to define M bins with height equal to 1, and then assign each bin a cut-off probability $c_i \in [0, 1]$, along with an "alias" $a_i \in \{1, 2, \ldots, M\}$. To sample from such a structure, we generate 2 uniform random variables:

$$i \sim U\{1, 2, \ldots, M\} \qquad u \sim U[0, 1]$$

If $u < c_i$, we return i, otherwise we return a_i. It is obvious that such a sampling procedure would take constant time.

> Aside: depending on how fast the Pseudo-Random Number Generator is, it may be beneficial to generate i and u from a single random number:
>
> $$x \sim U[0, 1]$$
>
> $$\implies i = \lfloor Mx \rfloor + 1 \qquad u = Mx - i + 1$$

Now, we will discuss how to build such an alias table (i.e. both the cut-off and alias arrays). First, we note that the alias table is not unique (this can be seen by parameter counting). Walker's original paper [**?**] gave an initialization algorithm which took $O(M^2)$ time, though this could have easily been reduced to $O(M \log M)$ had he used a better intermediate data structure (say, a heap).

However, Vose [**?**] gave a simple method to construct the alias table in $O(M)$ time.

...

Note: oftentimes the distribution is only initialized once, but is used to generate many samples, so the runtime of the initialization step may not be too important.

> Aside: If we wanted to further optimize the sampling step, we must construct an alias table which minimize the frequency of indexing the alias array. Doing this, however, is NP-Hard [**?**, **?**].