

Learning MC algorithms

Lee Hwee Kuan

February 25, 2018

Consider a system of 2D Ising spins $\sigma = (s_1, s_2, \dots, s_n)$ with $s_i = \pm 1$ on a square lattice of size $L \times L$ sites. Hamiltonian is

$$E(\sigma) = - \sum_{\langle i, j \rangle} s_i s_j \quad (1)$$

where $\langle i, j \rangle$ represents summation over nearest neighbors on the square lattice. Each edge on this graph is sum over once. The objective is to make a RevNet that learns how to do MCMC with detailed balance condition satisfied.

1 Implementation in tensorflow and matrix representation of Hamiltonian

Tensorflow works with linear algebra and hence we need to express our equations in matrix formulation. Let Σ be the 2D spin configurations in the form of a matrix, elements in the matrix corresponds to the site in the square lattice.

$$E = \frac{1}{L^2} (\|\Sigma * (\Sigma \cdot S_l)\| + \|\Sigma * (S_u \cdot \Sigma)\|) \quad (2)$$

where S_l and S_u are matrices that rotate the columns and rows matrix elements such that when the resultant matrix multiply by the original spin matrix Σ element wise $*$, then sum element wise gives the hamiltonian. $\|\cdot\|$ represents summing over elements in a matrix.

$$S_u = \begin{pmatrix} 0 & -1 & 0 & \dots & 0 & 0 \\ 0 & 0 & -1 & \dots & 0 & 0 \\ \vdots & & & & & \\ 0 & 0 & 0 & \dots & -1 & 0 \\ 0 & 0 & 0 & \dots & 0 & -1 \\ -1 & 0 & 0 & \dots & 0 & 0 \end{pmatrix} \quad (3)$$

$$S_l = \begin{pmatrix} 0 & 0 & 0 & \cdots & 0 & -1 \\ -1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 0 & \cdots & 0 & 0 \\ \vdots & & & & & \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & -1 & 0 \end{pmatrix} \quad (4)$$

2 Using RevNet to recover detailed balance

Detailed balance can be recovered by replacing MLP with RevNet. RevNet takes in two spin configurations, σ_1, σ_2 and output two configurations σ'_1, σ'_2 . In RevNet, there are two ends of the network, feed data into one end and the output will be on the other end and vice versa. For clarity, we define the directions “forward” and “backward” but since RevNet is symmetric, forward and backward definition is arbitrarily fixed. Algorithm goes like this

1. Given configurations σ_1, σ_2
2. Pass σ_1, σ_2 through RevNet in the forward or backward direction with the same probability to generate σ'_1, σ'_2
3. Accept the new configurations σ'_1, σ'_2 with the probability

$$A(\sigma_1, \sigma_2 \rightarrow \sigma'_1, \sigma'_2) = \min(1, \exp(-\beta[(E'_1 - E_1) + (E'_2 - E_2)])) \quad (5)$$

When the RevNet learns the transitions with $(E_1 + E_2) \approx (E'_1 + E'_2)$, this algorithm becomes efficient.

2.1 Proof of detailed balance condition

Proof of detailed balance condition of the above algorithm is as follows. Assuming configurations σ_1, σ_2 are sampled independently from the equilibrium distribution then the probability of sampling is

$$\begin{aligned} p(\sigma_1, \sigma_2) &= \exp(-\beta E(\sigma_1)) \exp(-\beta E(\sigma_2)) / \mathcal{Z}^2 \\ &= \exp(-\beta(E_1 + E_2)) / \mathcal{Z}^2 \end{aligned} \quad (6)$$

Make use of the reversibility of RevNet, the forward and backward trial moves probabilities between two sets of configurations linked by RevNet are the same, i.e.

$$T[(\sigma_1, \sigma_2) \rightarrow (\sigma'_1, \sigma'_2)] = T[(\sigma'_1, \sigma'_2) \rightarrow (\sigma_1, \sigma_2)] = 0.5 \quad (7)$$

Then the transition matrix becomes the acceptance matrix

$$\frac{A[(\sigma_1, \sigma_2) \rightarrow (\sigma'_1, \sigma'_2)]}{A[(\sigma'_1, \sigma'_2) \rightarrow (\sigma_1, \sigma_2)]} = \frac{\exp(-\beta(E'_1 + E'_2))}{\exp(-\beta(E_1 + E_2))} = \frac{p(\sigma'_1, \sigma'_2)}{p(\sigma_1, \sigma_2)} \quad (8)$$

3 Loss function

The loss function is define as,

$$Loss = [(E_1 + E_2) - (E'_1 - E'_2)]^2 + \frac{\lambda}{2L^2} [\|\Sigma'_1 * \Sigma'_1\| + \|\Sigma'_2 * \Sigma'_2\|] \text{ with } \lambda < 0 \quad (9)$$

Since the RevNet does not output $s_i = \pm 1$, the second term in the above equation penalize outputs that gives $\sigma' < 1$. Set activation function to be *tanh* to bound spins to ± 1 . L^2 is the number of lattice sites. It is use to normalize the loss to make it lattice size invariant.