# Learning MC algorithms

Lee Hwee Kuan

March 4, 2018

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

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

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 Hamilitonian

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

$$E = \frac{1}{L^2}\left( \| \Sigma * \left( \Sigma \cdot S^l \right) \| + \| \Sigma * \left( S^u \cdot \Sigma \right) \| \right) \tag{2}$$

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 $\Sigma$ element wise $*$, then sum element wise gives the hamiltonian. $\| \cdot \|$ represents summing over elements in a matrix.

$$S^u = \begin{pmatrix} 0 & -1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & -1 & \cdots & 0 & 0 \\ \vdots & & & & & \\ 0 & 0 & 0 & \cdots & -1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & -1 \\ -1 & 0 & 0 & \cdots & 0 & 0 \end{pmatrix} \tag{3}$$

$$S^l = \begin{pmatrix} 0 & 0 & 0 & \cdots & 0 & -1 \\ -1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 0 & \cdots & 0 & 0 \\ \vdots & & & & & \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & -1 & 0 \end{pmatrix} \tag{4}$$

### 1.1 Tensorflow batch handling

The implementation in tensorflow handles data in batch. We rewrite Eq.(2) in terms of batch and matrix indexing, Eq.(2) becomes,

$$\begin{aligned} E^b &= \frac{1}{L^2}\left(\sum_{i,j,k}\Sigma^b_{ij}\Sigma^b_{ik}S^l_{kj} + \sum_{i,j,k}\Sigma^b_{ij}S^u_{jk}\Sigma^b_{ki}\right) \\ &= \frac{1}{L^2}\sum_{i,j,k}\Sigma^b_{ij}\left(\Sigma^b_{ik}S^l_{kj} + S^u_{jk}\Sigma^b_{ki}\right) \end{aligned} \tag{5}$$

$b$ is the index for individual data in the batch. $\Sigma^b$ is the $b$th spin configuration.

## 2 RevNet architecture

RevNet is originally proposed by A.N.Gomez *et al* (arXiv:1707.04585v1). It is a network that is reversible, that is the network has a dual network which flips the input and output exactly. The network takes two inputs and generate two outputs, its dual takes in the outputs generated by the original network and generates the inputs of the original network. The ising spin magnitude is $\pm 1$ and therefore the RevNet has to be modified a little to satisfy this constraint. Fig. 1 shows the network architecture applicable for the Ising model. Equations for RevNet are,

$$\begin{aligned} \sigma'_2 &= \frac{1}{2}[\sigma_2 + F(\sigma_1)] \\ \sigma'_1 &= \frac{1}{2}[\sigma_1 + G(\sigma'_2)] \\ \sigma_1 &= 2\sigma'_1 - F(\sigma'_2) \\ \sigma_2 &= 2\sigma'_2 - G(\sigma_1) \end{aligned} \tag{6}$$

$F$ and $G$ can be any network or transformation. In our implementation we use the hyperbolic tangent activation function, the reason being than hyperbolic tangent function is bounded by $\pm 1$.

## 3 Using RevNet to recover detailed balance

Detailed balance can be recovered by replacing MLP with RevNet. RevNet takes in two spin configurations, $\sigma_1, \sigma_2$ and output two configurations $\sigma'_1, \sigma'_2$.

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 goest like this

1. Given configurations $\sigma_1, \sigma_2$
2. Pass $\sigma_1, \sigma_2$ through RevNet in the forward or backward direction with the same probability to generate $\sigma_1', \sigma_2'$
3. Accept the new configurations $\sigma_1', \sigma_2'$ with the probability

$$A(\sigma_1, \sigma_2 \to \sigma_1', \sigma_2') = \min(1, \exp(-\beta[(E_1' - E_1) + (E_2' - E_2)])) \tag{7}$$

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

### 3.1 Proof of detailed balance condition

Proof of detailed balance condition of the above algorithm is as follows. Assuming configurations $\sigma_1, \sigma_2$ are sampled independently from the equilibrium distribution then the probability of sampling is

$$\begin{aligned} p(\sigma_1, \sigma_2) &= \exp(-\beta E(\sigma_1)) \exp(-\beta E(\sigma_2))/\mathcal{Z}^2 \\ &= \exp(-\beta(E_1 + E_2))/\mathcal{Z}^2 \end{aligned} \tag{8}$$

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.

$$T[(\sigma_1, \sigma_2) \to (\sigma_1', \sigma_2')] = T[(\sigma_1', \sigma_2') \to (\sigma_1, \sigma_2)] = 0.5 \tag{9}$$

Then the transition matrix becomes the acceptance matrix

$$\frac{A[(\sigma_1, \sigma_2) \to (\sigma_1', \sigma_2')]}{A[(\sigma_1', \sigma_2') \to (\sigma_1, \sigma_2)]} = \frac{\exp(-\beta(E_1' + E_2')}{\exp(-\beta(E_1 + E_2)} = \frac{p(\sigma_1', \sigma_2')}{p(\sigma_1, \sigma_2)} \tag{10}$$

## 4 Training the RevNet - the loss function

The loss function is define on a batch of spin configurations,

$$Loss = \sum_b [(E_1^b + E_2^b) - (E'^b_1 + E'^b_2)]^2 + \frac{\lambda}{2L^2} \left[ \|\Sigma'^b_1 * \Sigma'^b_1\| + \|\Sigma'^b_2 * \Sigma'^b_2\| \right] \text{ with } \lambda < 0 \tag{11}$$

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 $\pm 1$. $L^2$ is the number of lattice sites. It is use to normalize the loss to make it lattice size invariant.