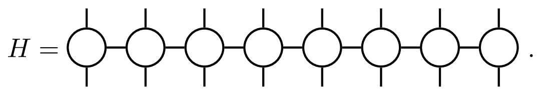
Homework 7

1 MPO and Finite State Automata

Introduction Recall the tensor network representation of a *Matrix Product Operator* (MPO). We are particularly interested in representing an Hamiltonian of an N-body one-dimensional system



For concreteness, let us start by considering the following Hamiltonian

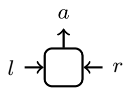
$$H = \sum_{i=1}^{N} X_i, \quad X_i = \underbrace{\mathbb{1} \otimes \cdots \otimes \mathbb{1}}_{(i-1)\text{ times}} X \underbrace{\mathbb{1} \otimes \cdots \otimes \mathbb{1}}_{(N-i)\text{ times}},$$

where X is some local operator on site i. By labeling the local operators with states (or characters) from a finite alphabet $\Sigma = \{\texttt{0}, \texttt{1}\}$, $\mathbb{1} \sim \texttt{0}$, $X \sim \texttt{1}$, we can rewrite the Hamiltonian as a series of strings,

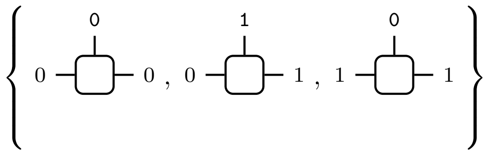
$$H \sim \texttt{10000}\ldots + \texttt{01000}\ldots + \texttt{00100}\ldots + \texttt{00010}\ldots + \ldots$$

We can then view the MPO associated to H as a *Finite State Automaton*, that goes through a string $a_1 \ldots a_N$, $a_i \in \Sigma$, accepting it or rejecting it. The MPO is then a machine that just takes a string, and checks if it matches some kind of pattern. If it matches, it outputs a weight W corresponding to the correct pre-factor for each term in the Hamiltonian. Otherwise, the pattern is rejected. The resulting Hamiltonian is then the sum of all possible combinations of valid inputs.

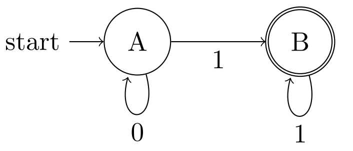
In this finite automata picture, each tensor in the chain takes an input states $l, r \in Q$, on the virtual level, and produces outputs state $a \in \Sigma$ on the physical level. Since neighboring tensors share a virtual leg, the inputs on the shared legs must be equal.



The way a tensor produces its output is called a *rule*. Convince yourself that the rules for the previous Hamiltonian are



Equivalently, we can think of the automaton as a walker having an internal state. At each site it outputs a state a and can change its internal state according to the set of rules. As in the example above, the transitions may not be unique. We can then represent the finite automaton by a transition map



where the highlighted node indicates the input of the automation is "accepting". In this case the weight of the accepting state is set to 1.

The corresponding tensor M of the MPO is straightforward to write, we put the different states into a matrix generated by the column index l and the row index r

$$M = \begin{pmatrix} \mathbb{1} & X \\ 0 & \mathbb{1} \end{pmatrix},$$

in which each entry of the matrix acts on the physical indices of the tensor. Explicitly perform the matrix multiplication to show that the tensor M generates the original Hamiltonian. What are the tensors at the boundary (assuming open boundary conditions)?

Nearest-Neighbor Interaction Using the same character labels, compute the rules, the transition diagram and the MPO tensor for the Hamiltonian

$$H \sim \texttt{11000}\ldots + \texttt{01100}\ldots + \texttt{00110}\ldots + \texttt{00011}\ldots + \ldots$$

Notice how this Hamiltonian couples nearest-neighbors.

Heisenberg Model Generalize the previous result to compute the MPO for the celebrated Heisenberg Hamiltonian for spin-1/2 particles,

$$H = \sum_{i=1}^{N-1} \vec{S}_i \cdot \vec{S}_{i+1}.$$

Next Nearest-Neighbor Interaction Find the MPO representation of the Hamiltonian

$$H = J_1 \sum_i Z_i Z_{i+1} + J_2 \sum_i Z_i Z_{i+2}.$$

Long-Range Interactions Show that the tensor

$$M = \begin{pmatrix} \mathbb{1} & X & 0 \\ 0 & \lambda\mathbb{1} & J\lambda X \\ 0 & 0 & \mathbb{1} \end{pmatrix},$$

gives rise to exponentially decaying interaction

$$H = \sum_{i=1}^{N} \sum_{r>0} J e^{-r/\xi} X_i X_{i+r}.$$

As a closing remark, we can approximate any arbitrary interaction $J(r)$ as a sum of exponentially decaying interactions[1], simply by finding the scalars c_α that minimize the distance $\|J(r) - \sum_\alpha \lambda_\alpha^r\|^2$.

Note Notice how the bond dimension χ corresponds and the number of states $|Q|$ of the finite automaton.

2 MPS Expectation Values*

Construct the MPS representation for the states $|\uparrow\uparrow\ldots\uparrow\uparrow\rangle$, $|\downarrow\downarrow\ldots\downarrow\downarrow\rangle$ and $|\mathrm{GHZ}\rangle$ for 10-50 sites. Create a function to numerically compute overlaps between MPS and calculate the overlaps between the previous states. Now construct the MPO for the Heisenberg model $H = \sum_i \vec{S}_i \cdot \vec{S}_{i+1}$ and compute the expectation values $\langle H \rangle$ for the states above.

Tips If you don't have experience with handling tensors, let me break it down into smaller steps

(a) Use some package[2], to familiarize yourself with contracting tensors.

(b) Define an index order convention for your MPS and MPO tensors, such as `(left,right,up)`.

(c) Construct your MPS/MPO as an array of tensors, one for each site.

(d) When computing your overlap, mind the contraction order!

[1] G.M. Crosswhite, A.C. Doherty, G. Vidal, Phys. Rev. B (2008) arXiv:0804.2504 and F. Fröwis, V. Nebendahl, W. Dür, Phys. Rev. A (2010) arXiv:1003.1047

[2] For Matlab/Octave I recommend `ncon`, available in the supplementary material of arXiv:1402.0939. Multiple options exist for Python, but you may start off with `tensordot` or `einsum`. For Julia, try the `TensorOperations` package.