

Unsupervised Generative Modeling Using Matrix Product State

Zhao-Yu Han,¹ Jun Wang,¹ Heng Fan,^{2,4} Lei Wang,^{2,4,*} and Pan Zhang^{3,†}

① Matrix product states (Tensor Train) & canonical form

MPS:
$$\begin{matrix} \text{---} \\ \diagup \text{---} \\ \text{---} \\ \diagdown \text{---} \\ \text{---} \end{matrix} = A^{(1)} \text{---} R \text{---} A^{(2)} \text{---} R \text{---} \dots \text{---} R \text{---} A^{(d)}$$

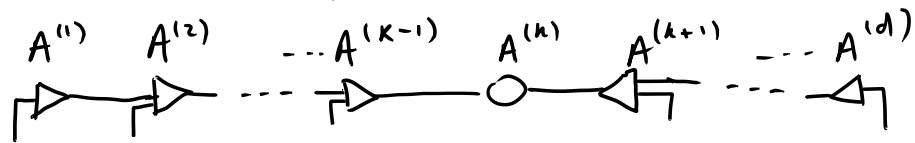
$m_1 \quad m_2 \quad m_d$

Orthogonal Core is a core $\begin{matrix} \text{---} \\ \diagup \text{---} \\ \text{---} \\ \diagdown \text{---} \\ \text{---} \end{matrix}$ s.t. $\begin{matrix} \text{---} \\ \diagup \text{---} \\ \text{---} \\ \diagdown \text{---} \\ \text{---} \end{matrix} = \text{---} (=I)$

$d_1 \times d_2 \times d_3 \quad d_3 \times d_3$

$\Leftrightarrow T_{(3)} T_{(3)}^T = I$

We say an MPS is in canonical form w.r.t the k th core if:



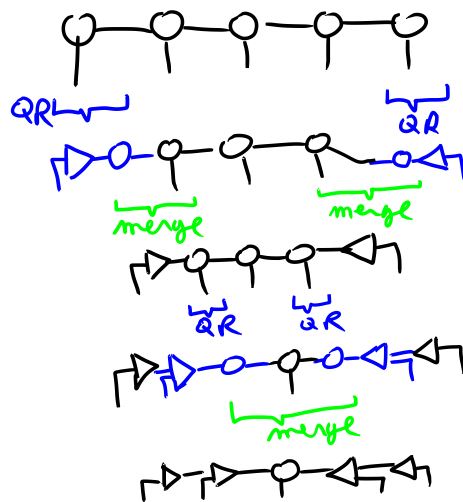
$A^{(1)} \dots A^{(k-1)}$ are left-orthonormal
 $A^{(k+1)} \dots A^{(d)}$ are right-orthonormal.

• $\begin{matrix} \text{---} \\ \diagup \text{---} \\ \text{---} \\ \diagdown \text{---} \\ \text{---} \end{matrix} = \begin{matrix} \text{---} \\ \diagup \text{---} \\ \text{---} \\ \diagdown \text{---} \\ \text{---} \end{matrix}$ $\begin{matrix} \text{---} \\ \diagup \text{---} \\ \text{---} \\ \diagdown \text{---} \\ \text{---} \end{matrix} = I (=I)$

• SVD: $\begin{matrix} \text{---} \\ \diagup \text{---} \\ \text{---} \\ \diagdown \text{---} \\ \text{---} \end{matrix} = \begin{matrix} \text{---} \\ \diagup \text{---} \\ \text{---} \\ \diagdown \text{---} \\ \text{---} \end{matrix} \begin{matrix} \text{---} \\ \diagup \text{---} \\ \text{---} \\ \diagdown \text{---} \\ \text{---} \end{matrix} \begin{matrix} \text{---} \\ \diagup \text{---} \\ \text{---} \\ \diagdown \text{---} \\ \text{---} \end{matrix}$

$U \quad D \quad V^T$

ALGORITHM FOR CAN. FORM: INPUT: $T = \text{---}$, k
 OUTPUT: T in can. form w.r.t the k th core



II Generative Modelling with MPS

Distribution over fixed length binary sequences: $\{0, 1\}^N$
 set of all length N binary sequences (2^N)

For $v = v_1, v_2, \dots, v_N$
 $v_i \in \{0, 1\}$

$$h(v) = \frac{\Psi(v)^2}{Z}$$

where $\Psi(v) = \begin{matrix} A^{(1)} & - & A^{(2)} & \dots & - & A^{(N)} \\ | & & | & & & | \\ v_1 & & v_2 & & & v_N \end{matrix} \in \mathbb{R}$

and $Z = \sum_{v \in \{0, 1\}^N} \Psi(v)^2$ (partition function)

Notation: for $v_i \in \{0, 1\}$

we use $\begin{matrix} | \\ v_i \end{matrix} = \begin{cases} \begin{pmatrix} 1 \\ 0 \end{pmatrix} & \text{if } v_i = 0 \\ \begin{pmatrix} 0 \\ 1 \end{pmatrix} & \text{if } v_i = 1 \end{cases}$ and $\begin{matrix} | \\ v_i \end{matrix} = \begin{cases} \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} & \text{if } v_i = 0 \\ \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} & \text{if } v_i = 1 \end{cases}$

• In energy-based models (e.g. Restricted Boltzmann Machines) computing Z is often intractable, here it is easy:

$$Z = \sum_x \Psi(v)^2 = \sum_{x_1} \sum_{x_2} \dots \sum_{x_N} \Psi(x_1, \dots, x_N)^2$$

$$= \sum_{x_1} \sum_{x_2} \dots \sum_{x_N} \left(\begin{matrix} A^{(1)} & - & A^{(2)} & \dots & - & A^{(N)} \\ | & & | & & & | \\ x_1 & & x_2 & & & x_N \end{matrix} \right)^2$$

Sum of an exponential number of terms

$$= \sum_{x_1} \sum_{x_2} \dots \sum_{x_N} \begin{matrix} A^{(1)} & - & A^{(2)} & \dots & - & A^{(N)} \\ | & & | & & & | \\ x_1 & & x_2 & & & x_N \\ | & & | & & & | \\ A^{(1)} & - & A^{(2)} & \dots & - & A^{(N)} \end{matrix}$$

$$= \sum_{x_1} \begin{matrix} A^{(1)} & - & A^{(2)} & \dots & - & A^{(N)} \\ | & & | & & & | \\ x_1 & & \sum_{x_2} & & & \sum_{x_N} \\ | & & | & & & | \\ A^{(1)} & - & A^{(2)} & \dots & - & A^{(N)} \end{matrix}$$

Observe $\sum_{v_i \in \{0, 1\}} \begin{matrix} | \\ v_i \end{matrix} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = I$

something we can compute efficiently (inner product between 2 Tensor Train)

$$= \begin{matrix} A^{(1)} & - & A^{(2)} & \dots & - & A^{(N)} \\ | & & | & & & | \\ A^{(1)} & - & A^{(2)} & \dots & - & A^{(N)} \end{matrix}$$

If Ψ is in canonical form, it is even easier!

$$Z = \frac{\Psi}{\langle \cdot | \cdot \rangle} = \left[\begin{array}{c} A^{(1)} \quad A^{(2)} \\ \vdots \\ A^{(k)} \quad A^{(k+1)} \\ \vdots \\ A^{(N)} \end{array} \right] = \left[\begin{array}{c} A^{(k)} \\ \vdots \\ A^{(k)} \end{array} \right]$$

III Training

Given a sample $S \in \{0, 1\}^N$, we want to maximize the likelihood of the data under the MPS model.

Negative log-likelihood: $\mathcal{L} = -\frac{1}{|S|} \sum_{v \in S} \log h(v)$

where $h(v) = \Psi(v)^2 / Z$

- 2 training algorithms:
- SGD \rightarrow need $\partial \mathcal{L} / \partial A^{(k)}$
 - DMRG-like algorithm: \rightarrow need $\partial \mathcal{L} / \partial A^{(k, k+1)}$

$$\Psi = \left[\begin{array}{c} A^{(1)} \\ \vdots \\ A^{(i)} - A^{(i+1)} \\ \vdots \\ A^{(N)} \end{array} \right]$$

Repeat until convergence:

for each pair of cores $A^{(i)}, A^{(i+1)}$

Merge: $\left[\begin{array}{c} A^{(i)} \\ \vdots \\ A^{(i, i+1)} \\ \vdots \\ A^{(i+1)} \end{array} \right] = \left[\begin{array}{c} A^{(i)} \\ \vdots \\ A^{(i+1)} \end{array} \right]$

Gradient step: $A^{(i, i+1)} \leftarrow A^{(i, i+1)} - \eta \nabla_{A^{(i, i+1)}} \mathcal{L}$

Split: $\left[\begin{array}{c} A^{(i, i+1)} \\ \vdots \\ A^{(i)} \\ \vdots \\ A^{(i+1)} \end{array} \right] \approx \left[\begin{array}{c} A^{(i)} \\ \vdots \\ A^{(i+1)} \end{array} \right]$ (truncated SVD)

We need to compute gradients:

Proposition: let $v \in \{0, 1\}^N$ and let $\ell(v) = -\log h(v)$.

$$\frac{\partial \ell(v)}{\partial A^{(k)}} = -2 \frac{\Psi(v)'}{\Psi(v)} = -2 \frac{\sum_x \Psi(x) \Psi(x)'}{Z}$$

where $\Psi(v)' = \frac{\partial \Psi(v)}{\partial A^{(k)}}$ and $Z = \sum_x \Psi(x)$

$1 \times R \times d \times R$ (here $d=2$ and R is the rank of the MPS)

$$\rightarrow \Psi(v)' = \partial \left(\left[\begin{array}{c} A^{(1)} \\ \vdots \\ A^{(k)} \\ \vdots \\ A^{(N)} \end{array} \right] \right) / \partial A^{(k)} = \left[\begin{array}{c} A^{(1)} \\ \vdots \\ A^{(k-1)} \quad R \quad R \quad A^{(k+1)} \\ \vdots \\ A^{(N)} \end{array} \right]$$

$$\begin{aligned} \rightarrow \sum_x \Psi(x) \Psi(x)' &= \sum_{x_1} \dots \sum_{x_N} \begin{array}{ccccccc} A^{(1)} & \dots & A^{(k-1)} & - & A^{(k)} & - & A^{(k+1)} & \dots & A^{(N)} \\ | & & | & & | & & | & & | \\ x_1 & & x_{k-1} & & x_k & & x_{k+1} & & x_N \\ | & & | & & | & & | & & | \\ A^{(1)} & \dots & A^{(k-1)} & - & A^{(k+1)} & \dots & A^{(N)} \end{array} \\ &= \begin{array}{ccccccc} A^{(1)} & \dots & A^{(k-1)} & - & A^{(k)} & - & A^{(k+1)} & \dots & A^{(N)} \\ | & & | & & | & & | & & | \\ A^{(1)} & \dots & A^{(k-1)} & - & A^{(k+1)} & \dots & A^{(N)} \end{array} \end{aligned}$$

↳ Even easier if MPS is in canonical form w/ k -th core:

$$\sum_x \Psi(x) \Psi(x)' = \begin{bmatrix} A^{(k)} \\ 1 \end{bmatrix}$$

IV Sampling

draw $v \sim \mu(\cdot)$

Sample v_N, v_{N-1}, \dots, v_1 in turn.

Assume can. form. w/ the N -th core. $\Psi(v) = \begin{array}{c} \text{---} \text{---} \text{---} \text{---} \text{---} \\ \uparrow \quad \uparrow \quad \dots \quad \uparrow \quad \uparrow \\ v_1 \quad v_2 \quad \dots \quad v_{N-1} \quad v_N \end{array}$

- Sample v_N from the marginal

$$\mu(v_N) = \sum_{v_1, \dots, v_{N-1}} \mu(v_1, v_2, \dots, v_N)$$

$$\begin{aligned} Z \cdot \mu(v_N) &= \begin{array}{ccccccc} A^{(1)} & - & A^{(2)} & - & \dots & - & A^{(N-1)} & - & A^{(N)} \\ | & & | & & & & | & & | \\ A^{(1)} & - & A^{(2)} & - & \dots & - & A^{(N-1)} & - & \begin{array}{c} v_N \\ A^{(N)} \end{array} \end{array} \end{aligned}$$

$$= \begin{bmatrix} A^{(N)} \\ v_N \\ A^{(N)} \end{bmatrix} \quad (\text{because } \Psi \text{ is in canonical form})$$

- Sample v_{k-1} given v_k, v_{k+1}, \dots, v_N from

$$\mu(v_{k-1} | v_k, \dots, v_N) = \frac{\mu(v_{k-1}, \dots, v_N)}{\mu(v_k, \dots, v_N)}$$

$$\begin{aligned} Z \cdot \mu(v_{k-1}, \dots, v_N) &= Z \sum_{v_1, \dots, v_{k-1}} \mu(v_1, v_2, \dots, v_N) = \begin{array}{ccccccc} A^{(1)} & \dots & A^{(k-1)} & - & A^{(k)} & - & \dots & - & A^{(N)} \\ | & & | & & | & & & & | \\ A^{(1)} & \dots & A^{(k-1)} & - & \begin{array}{c} v_k \\ A^{(k)} \end{array} & - & \dots & - & \begin{array}{c} v_N \\ A^{(N)} \end{array} \end{array} \\ &= \begin{array}{ccccccc} A^{(k)} & - & A^{(k+1)} & - & \dots & - & A^{(N)} \\ | & & | & & & & | \\ \begin{array}{c} v_k \\ A^{(k)} \end{array} & - & \begin{array}{c} v_{k+1} \\ A^{(k+1)} \end{array} & - & \dots & - & \begin{array}{c} v_N \\ A^{(N)} \end{array} \end{array} \end{aligned}$$

Follow-up paper on uniform MPS for language modelling:

Tensor Networks For Language Modeling

Jacob Miller^{1,2} Guillaume Rabusseau^{1,3} John Terilla^{2,4}