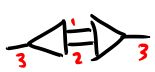
Unsupervised Generative Modeling Using Matrix Product State

Zhao-Yu Han,[1] Jun Wang,[1] Heng Fan,[2,4] Lei Wang,[2,4,*] and Pan Zhang[3,†]

(I) Matrix product states (Tensor Train) & canonical form

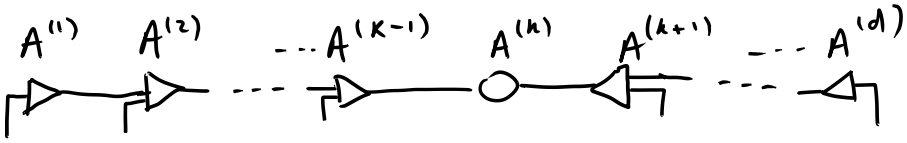
MPS: $T_{m_1 \times \cdots \times m_d} = A^{(1)}_{m_1} R\, A^{(2)}_{m_2} R \cdots R\, A^{(d)}_{m_d}$

Orthogonal Core is a core  s.t.  $= -$ $(= I)$

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

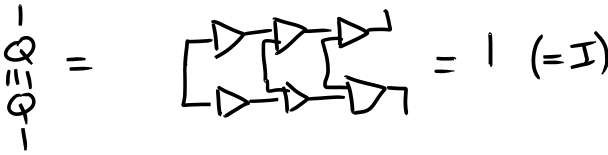
$\Leftrightarrow \quad T_{(3)} T_{(3)}^{T} = I$

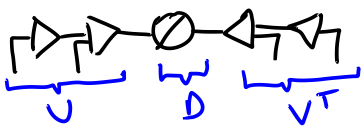
We say an MPS is in canonical form wrt the k th core if:

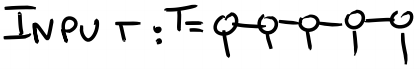


$A^{(1)} \cdots A^{(k-1)}$ are left-orthonormal

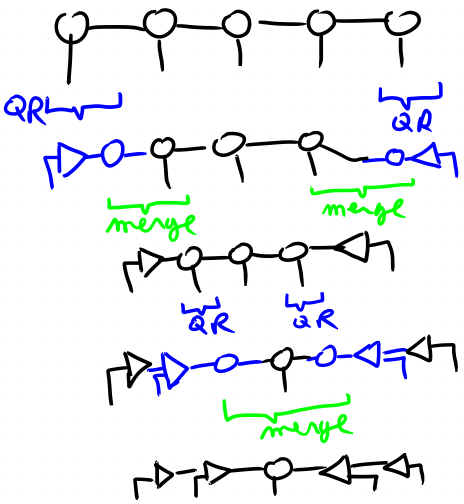
$A^{(k+1)} \cdots A^{(d)}$ are right-orthonormal.

-   $= I$ $(= I)$

- SVD in TT format: 

ALGORITHM FOR CAN. FORM: INPUT: $T =$  , k

OUTPUT: T in can. form wrt the k^{th} core



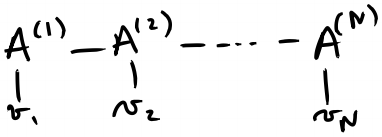
II Generative Modelling with MPS

Distribution over fixed length binary sequences : $\{0,1\}^N$

set of all length N binary sequences (2^N)

For $v = v_1 v_2 \cdots v_N$, $v \in \{0,1\}^N$, $v_i \in \{0,1\}$

$$p(v) = \frac{\psi(v)^2}{Z}$$

where $\psi(v) =$  $\in \mathbb{R}$

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

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

we use $\overset{1}{v_i} = \begin{cases} \binom{1}{0} \text{ if } v_i = 0 \\ \binom{0}{1} \text{ if } v_i = 1 \end{cases}$ and $\overset{1}{\underset{1}{v_i}} = \begin{cases} \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \text{ if } v_i = 0 \\ \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \text{ if } v_i = 1 \end{cases}$

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

$$Z = \sum_x \psi(v)^2 = \sum_{x_1} \sum_{x_2} \cdots \sum_{x_N} \psi(x_1 \cdots x_N)^2$$

$$= \sum_{x_1} \sum_{x_2} \cdots \sum_{x_N} \left(A^{(1)} - A^{(2)} - \cdots - A^{(N)} \text{ with } x_1, x_2, \ldots, x_N \right)^2$$

Sum of an exponential number of terms

$$= \sum_{x_1} \sum_{x_2} \cdots \sum_{x_N} \left(A^{(1)} - A^{(2)} - \cdots - A^{(N)} \text{ with } x_1, x_2, \ldots, x_N \right) \otimes \left(A^{(1)} - A^{(2)} - \cdots - A^{(N)} \right)$$

$$= \left(\sum_{x_1} A^{(1)} \right) - \left(\sum_{x_2} A^{(2)} \right) - \cdots - \left(\sum_{x_N} A^{(N)} \right) \otimes \left(A^{(1)} - A^{(2)} - \cdots - A^{(N)} \right)$$

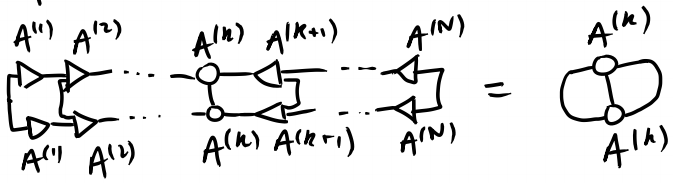
Observe $\sum_{v_i \in \{0,1\}} \overset{1}{\underset{1}{v_i}} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = I$

$$= A^{(1)} - A^{(2)} - \cdots - A^{(N)} \otimes A^{(1)} - A^{(2)} - \cdots - A^{(N)}$$

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

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

$$Z = \sum_{(\dots)} \Psi^2 =$$



(III) Training

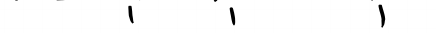
Given a sample $S \subseteq \{0,1\}^N$, we want to maximize the likelihood of the data under the MPS model.

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 → need $\partial \mathcal{L} / \partial A^{(k)}$

- DMRG-like algorithm : → need $\partial \mathcal{L} / \partial A^{(k,k+1)}$

$\Psi =$ 

Repeat until convergence :

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

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

(truncated SVD)

We need to compute gradients :

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

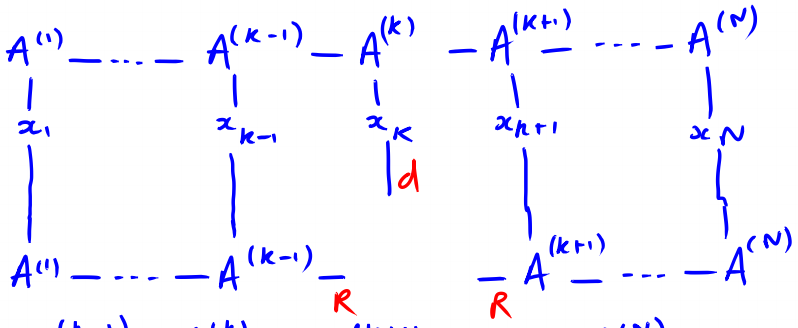
$$\frac{\partial \ell(v)}{\partial A^{(k)}} = -2 \frac{\Psi(v)'}{\Psi(v)} - 2 \frac{\sum_x \Psi(x) \Psi(x)'}{Z}$$

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

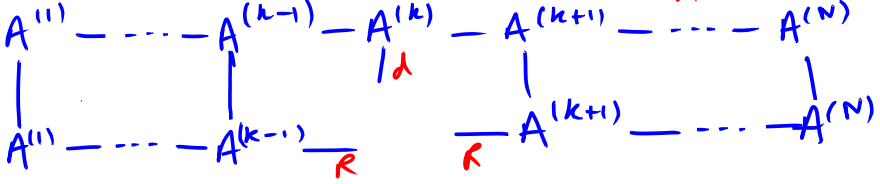
$1 \times R \times d \times R$ (here $d=2$ and R is the rank of the MPS)

$$\rightsquigarrow \Psi(v)' = \partial \left(A^{(1)}_{v_1} \cdots A^{(N)}_{v_N} \right) / \partial A^{(k)} = A^{(1)}_{v_1} \cdots A^{(k-1)}_{v_{k-1}} \underline{R} \; \underset{v_k}{\overset{d}{\underline{R}}} A^{(k+1)}_{v_{k+1}} \cdots A^{(N)}_{v_N}$$

$\rightsquigarrow \sum_x \psi(x)\psi(x)' = \sum_{x_1} \cdots \sum_{x_N}$



$=$



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

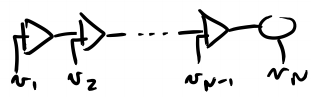
$$\sum_x \psi(x)\psi(x)' = \left[A^{(k)} \right]$$

(IV) Sampling

draw $v \sim h(\cdot)$

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

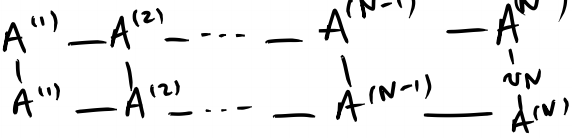
Assume can. form. wrt the N-th core. $\psi(v) =$



- Sample v_N from the marginal

$$h(v_N) = \sum_{v_1 \cdots v_{N-1}} h(v_1 v_2 \cdots v_N)$$

$Z \cdot h(v_N) =$



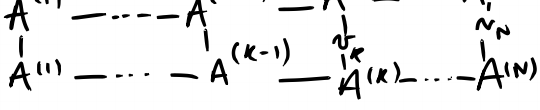
$=$

(because ψ is in canonical form)

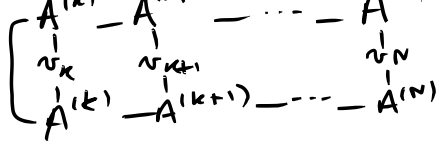
- Sample v_{k-1} given $v_k, v_{k+1}, \ldots, v_N$ from

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

$Z \cdot h(v_k, \ldots, v_N) = Z \sum_{v_1 \cdots v_{k-1}} h(v_1 v_2 \cdots v_N) =$



$=$



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]