

Tensor Network Models for Structured Data

Guillaume Rabusseau

Assistant Professor at DIRO, UdeM
CIFAR Canada Chair in AI at Mila

November 14, 2020

NeurIPS Workshop on Quantum Tensor Networks in Machine Learning

Introduction

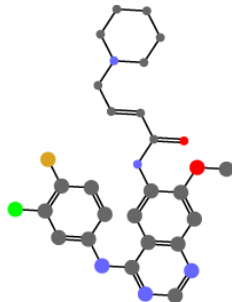
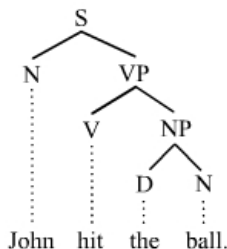
- Tensor network (TN) methods provide efficient ways to deal with very high dimensional data

Introduction

- Tensor network (TN) methods provide efficient ways to deal with very high dimensional data
- [Stoudenmire and Schwab, 2016], [Novikov et al., 2016]:
 - ▶ TN can be used to parameterize linear models in exponentially large spaces
 - ▶ Optimization techniques from quantum physics can be used to optimize these models

Introduction

- Tensor network (TN) methods provide efficient ways to deal with very high dimensional data
- [Stoudenmire and Schwab, 2016], [Novikov et al., 2016]:
 - ▶ TN can be used to parameterize linear models in exponentially large spaces
 - ▶ Optimization techniques from quantum physics can be used to optimize these models
- In this talk we focus on TN methods for **structured data**:



Introduction

Roadmap of the talk:

- Show how **uniform TN** can be used to represent functions over sequences (and trees)

Introduction

Roadmap of the talk:

- Show how **uniform TN** can be used to represent functions over sequences (and trees)
- Present connections with classical models:

uniform MPS \equiv Weighted Automata / HMMs \equiv linear 2nd order RNNs

Introduction

Roadmap of the talk:

- Show how **uniform TN** can be used to represent functions over sequences (and trees)
- Present connections with classical models:

uniform MPS \equiv Weighted Automata / HMMs \equiv linear 2nd order RNNs

- Present the spectral learning algorithm ([Bailly et al., 2009],[Hsu et al., 2009]) through the lens of tensor networks.

Introduction

Roadmap of the talk:

- Show how **uniform TN** can be used to represent functions over sequences (and trees)
- Present connections with classical models:

uniform MPS \equiv Weighted Automata / HMMs \equiv linear 2nd order RNNs

- Present the spectral learning algorithm ([Bailly et al., 2009],[Hsu et al., 2009]) through the lens of tensor networks.
- Show how the spectral algorithm provides a way to convert MPS models to uniform MPS models.

Introduction

Roadmap of the talk:

- Show how **uniform TN** can be used to represent functions over sequences (and trees)
- Present connections with classical models:

uniform MPS \equiv Weighted Automata / HMMs \equiv linear 2nd order RNNs

- Present the spectral learning algorithm ([Bailly et al., 2009],[Hsu et al., 2009]) through the lens of tensor networks.
- Show how the spectral algorithm provides a way to convert MPS models to uniform MPS models.
- Most of the talk is based on our paper


Li, Tianyu, Doina Precup, and Guillaume Rabusseau.

Connecting Weighted Automata, Tensor Networks and RNNs through Spectral Learning.


arXiv preprint arXiv:2010.10029 (2020).

Tensor Networks

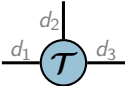
Degree of a node \equiv order of tensor



$\mathbf{v} \in \mathbb{R}^d$



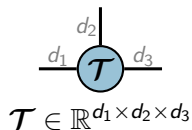
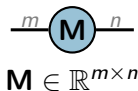
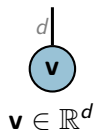
$\mathbf{M} \in \mathbb{R}^{m \times n}$



$\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$

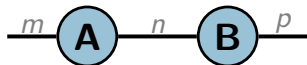
Tensor Networks

Degree of a node \equiv order of tensor



Edge \equiv contraction

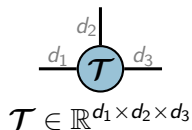
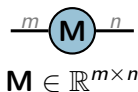
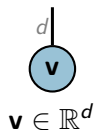
Matrix product:



$$(\mathbf{AB})_{i_1, i_2} = \sum_{k=1}^n \mathbf{A}_{i_1 k} \mathbf{B}_{k i_2}$$

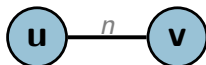
Tensor Networks

Degree of a node \equiv order of tensor



Edge \equiv contraction


Inner product:




$$\mathbf{u}^T \mathbf{v} = \sum_{k=1}^n \mathbf{u}_k \mathbf{v}_k$$

Tensor Networks

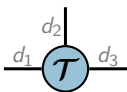
Degree of a node \equiv order of tensor



$\mathbf{v} \in \mathbb{R}^d$



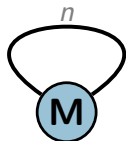
$\mathbf{M} \in \mathbb{R}^{m \times n}$



$\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$

Edge \equiv contraction

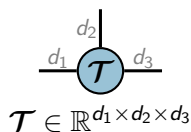
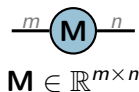
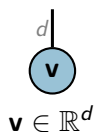
Trace of an $n \times n$ matrix:



$$\text{Tr}(\mathbf{M}) = \sum_{i=1}^n \mathbf{M}_{ii}$$

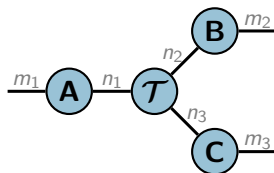
Tensor Networks

Degree of a node \equiv order of tensor



Edge \equiv contraction

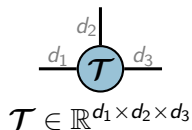
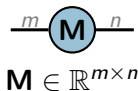
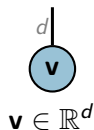
Tensor times matrices:



$$(\mathcal{T} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C})_{i_1, i_2, i_3} = \sum_{k_1=1}^{n_1} \sum_{k_2=1}^{n_2} \sum_{k_3=1}^{n_3} \mathcal{T}_{k_1 k_2 k_3} \mathbf{A}_{i_1 k_1} \mathbf{B}_{i_2 k_2} \mathbf{C}_{i_3 k_3}$$

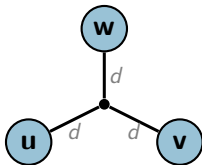
Tensor Networks

Degree of a node \equiv order of tensor



Edge \equiv contraction

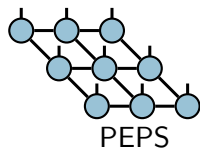
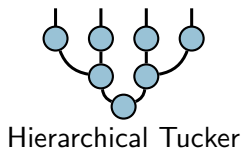
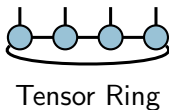
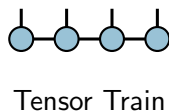
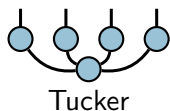
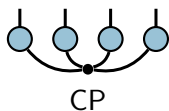
Hyperedge \equiv contraction between more than 2 indices:



$$\sum_{i=1}^n \mathbf{u}_i \mathbf{v}_i \mathbf{w}_i$$

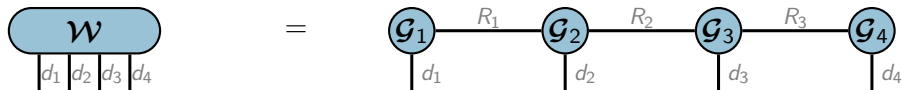
Tensor Decomposition Techniques

- Tensors can quickly get huge
- Simple solution: decompose a tensor into product of small factors
- Tensor decomposition \equiv compressed representation of tensors
- Lots of different ways to decompose a tensor:



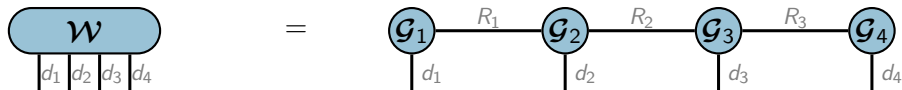
Tensor Train (TT) / Matrix Product States (MPS)

- TT [Oseledets, 2011] / MPS [Fannes et al., 1992] decomposition:



Tensor Train (TT) / Matrix Product States (MPS)

- TT [Oseledets, 2011] / MPS [Fannes et al., 1992] decomposition:

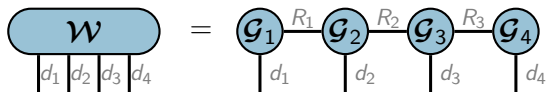


- If the ranks are all the same ($R_1 = R_2 = \dots = R$), can represent a vector of size 2^n with $\mathcal{O}(nR^2)$ parameters!

Outline

- 1 Tensor Network Models for Sequences
- 2 A Spectral Learning Algorithm for Uniform MPS Models
- 3 Spectral Learning \equiv MPS to Uniform MPS Conversion Algorithm
- 4 Beyond Sequences: Tensor Network Models for Trees
- 5 Conclusion

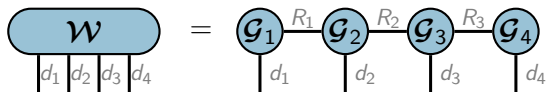
Tensor Train / Matrix Product States



- We can parameterize linear classification models with MPS:

$$f(\mathbf{x}) = \text{sign}(\langle \mathbf{w}, \mathbf{x} \rangle) = \text{sign} \left(\begin{array}{c} \mathcal{G}_1 \quad R \quad \mathcal{G}_2 \quad R \quad \mathcal{G}_3 \quad R \quad \mathcal{G}_4 \\ \diagdown \quad \diagup \quad \diagdown \quad \diagup \\ \mathbf{x} \end{array} \right)$$

Tensor Train / Matrix Product States



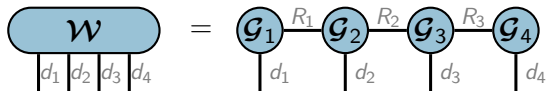
- We can parameterize linear classification models with MPS:

$$f(\boldsymbol{x}) = \text{sign}(\langle \boldsymbol{W}, \boldsymbol{x} \rangle) = \text{sign} \left(\begin{array}{c} \mathcal{G}_1 \text{---} R \text{---} \mathcal{G}_2 \text{---} R \text{---} \mathcal{G}_3 \text{---} R \text{---} \mathcal{G}_4 \\ \diagdown \quad \diagup \quad \diagdown \quad \diagup \\ \boldsymbol{x} \end{array} \right)$$

- We can also model probability distributions with MPS [Han et al., 2018]:

$$\mathbb{P}(\boldsymbol{x}) = \begin{array}{c} \mathcal{G}_1 \text{---} R \text{---} \mathcal{G}_2 \text{---} R \text{---} \mathcal{G}_3 \text{---} R \text{---} \mathcal{G}_4 \\ \diagdown \quad \diagup \quad \diagdown \quad \diagup \\ \boldsymbol{x} \end{array}$$

Tensor Train / Matrix Product States



- We can parameterize linear classification models with MPS:

$$f(\boldsymbol{x}) = \text{sign}(\langle \boldsymbol{W}, \boldsymbol{x} \rangle) = \text{sign} \left(\begin{array}{c} \mathcal{G}_1 \text{---} R \text{---} \mathcal{G}_2 \text{---} R \text{---} \mathcal{G}_3 \text{---} R \text{---} \mathcal{G}_4 \\ \diagdown \quad \diagup \quad \diagdown \quad \diagup \\ \boldsymbol{x} \end{array} \right)$$

- We can also model probability distributions with MPS [Han et al., 2018]:

$$\mathbb{P}(\boldsymbol{x}) = \begin{array}{c} \mathcal{G}_1 \text{---} R \text{---} \mathcal{G}_2 \text{---} R \text{---} \mathcal{G}_3 \text{---} R \text{---} \mathcal{G}_4 \\ \diagdown \quad \diagup \quad \diagdown \quad \diagup \\ \boldsymbol{x} \end{array} \quad \text{or} \quad \mathbb{P}(\boldsymbol{x}) = \left(\begin{array}{c} \mathcal{G}_1 \text{---} R \text{---} \mathcal{G}_2 \text{---} R \text{---} \mathcal{G}_3 \text{---} R \text{---} \mathcal{G}_4 \\ \diagdown \quad \diagup \quad \diagdown \quad \diagup \\ \boldsymbol{x} \end{array} \right)^2$$

MPS for sequence modeling

- We can also use MPS to model functions and distributions over **fixed length** sequences:

$$\mathbb{P}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4) = \begin{array}{cccc} \mathcal{G}_1 & \mathcal{G}_2 & \mathcal{G}_3 & \mathcal{G}_4 \\ | & | & | & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \mathbf{x}_4 \end{array} \text{ or } \mathbb{P}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4) = \left(\begin{array}{cccc} \mathcal{G}_1 & \mathcal{G}_2 & \mathcal{G}_3 & \mathcal{G}_4 \\ | & | & | & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \mathbf{x}_4 \end{array} \right)^2$$

MPS for sequence modeling

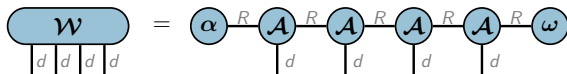
- We can also use MPS to model functions and distributions over **fixed length** sequences:

$$\mathbb{P}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4) = \begin{array}{cccc} \mathcal{G}_1 & \mathcal{G}_2 & \mathcal{G}_3 & \mathcal{G}_4 \\ | & | & | & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \mathbf{x}_4 \end{array} \text{ or } \mathbb{P}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4) = \left(\begin{array}{cccc} \mathcal{G}_1 & \mathcal{G}_2 & \mathcal{G}_3 & \mathcal{G}_4 \\ | & | & | & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \mathbf{x}_4 \end{array} \right)^2$$

↪ How to model distributions over variable length sequences?

Uniform MPS

- **uniform MPS** (uMPS) decomposition \equiv MPS with same core at each site:



Uniform MPS

- **uniform MPS** (uMPS) decomposition \equiv MPS with same core at each site:

$$\mathcal{W} = \alpha \text{---}^R \mathcal{A} \text{---}^R \mathcal{A} \text{---}^R \mathcal{A} \text{---}^R \mathcal{A} \text{---}^R \mathcal{A} \text{---}^R \omega$$

- With uMPS, we can model functions and distributions over **variable length** sequences:

$$\mathbb{P}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4) = \alpha \text{---}^R \mathcal{A} \text{---}^R \mathcal{A} \text{---}^R \mathcal{A} \text{---}^R \mathcal{A} \text{---}^R \omega, \mathbb{P}(\mathbf{x}_1, \mathbf{x}_2) = \alpha \text{---}^R \mathcal{A} \text{---}^R \mathcal{A} \text{---}^R \omega,$$

$$\mathbb{P}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6) = \alpha \text{---}^R \mathcal{A} \text{---}^R \mathcal{A} \text{---}^R \mathcal{A} \text{---}^R \mathcal{A} \text{---}^R \mathcal{A} \text{---}^R \mathcal{A} \text{---}^R \omega, \dots$$

Connections between uMPS and other models

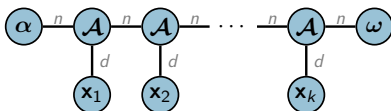
- A uMPS is given by a tuple $(\alpha \in \mathbb{R}^n, \mathcal{A} \in \mathbb{R}^{n \times d \times n}, \omega \in \mathbb{R}^n)$ and maps any sequence of vectors $\mathbf{x}_1, \dots, \mathbf{x}_k \in \mathbb{R}^d$ to a scalar:

$$f(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k) =$$

The diagram illustrates the tensor network for the function $f(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k)$. It consists of a horizontal chain of tensors: a left boundary tensor α , followed by k tensors labeled \mathcal{A} , and a right boundary tensor ω . Each \mathcal{A} tensor has a vertical leg connected to an input vector \mathbf{x}_i . The horizontal legs between tensors are labeled with the index n , and the vertical legs are labeled with the index d .

Connections between uMPS and other models

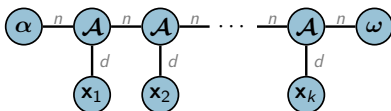
- A uMPS is given by a tuple $(\alpha \in \mathbb{R}^n, \mathcal{A} \in \mathbb{R}^{n \times d \times n}, \omega \in \mathbb{R}^n)$ and maps any sequence of vectors $\mathbf{x}_1, \dots, \mathbf{x}_k \in \mathbb{R}^d$ to a scalar:

$$f(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k) =$$


- If the inputs are one-hot encoding, uMPS \equiv Weighted Automata (generalization of HMMs)
- Linear second order RNNs \equiv uMPS

Connections between uMPS and other models

- A uMPS is given by a tuple $(\alpha \in \mathbb{R}^n, \mathcal{A} \in \mathbb{R}^{n \times d \times n}, \omega \in \mathbb{R}^n)$ and maps any sequence of vectors $\mathbf{x}_1, \dots, \mathbf{x}_k \in \mathbb{R}^d$ to a scalar:

$$f(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k) =$$


- If the inputs are one-hot encoding, uMPS \equiv Weighted Automata (generalization of HMMs)
- Linear second order RNNs \equiv uMPS
- For a thorough discussion of connections between uMPS, stochastic processes and automata, see

Srinivasan, S., Adhikary, S., Miller, J., Rabusseau, G. and Boots, B.

Quantum Tensor Networks, Stochastic Processes, and Weighted Automata

arXiv preprint arXiv:2010.10653 (2020).

\hookrightarrow Similar to the analysis of [Glassner et al., 2019] for uniform TN

Outline

- 1 Tensor Network Models for Sequences
- 2 A Spectral Learning Algorithm for Uniform MPS Models**
- 3 Spectral Learning \equiv MPS to Uniform MPS Conversion Algorithm
- 4 Beyond Sequences: Tensor Network Models for Trees
- 5 Conclusion

Hankel matrix

- We consider the case where inputs are sequences of discrete symbols:
 - ▶ Σ a finite alphabet of size d (e.g. $\{a, b\}$)
 - ▶ Σ^* strings on Σ (e.g. $abba$)
 - ▶ A uMPS computes a function $f : \Sigma^* \rightarrow \mathbb{R}$:

$$f(\sigma_1 \cdots \sigma_k) = \begin{array}{c} \alpha \text{---}^n \text{---} \mathcal{A} \text{---}^n \text{---} \mathcal{A} \text{---}^n \text{---} \cdots \text{---}^n \text{---} \mathcal{A} \text{---}^n \text{---} \omega \\ \quad \quad \quad \downarrow^d \quad \quad \downarrow^d \quad \quad \quad \quad \quad \quad \downarrow^d \\ \quad \quad \quad \sigma_1 \quad \quad \sigma_2 \quad \quad \quad \quad \quad \quad \quad \quad \quad \sigma_k \end{array}$$

Hankel matrix

- We consider the case where inputs are sequences of discrete symbols:
 - ▶ Σ a finite alphabet of size d (e.g. $\{a, b\}$)
 - ▶ Σ^* strings on Σ (e.g. $abba$)
 - ▶ A uMPS computes a function $f : \Sigma^* \rightarrow \mathbb{R}$:

$$f(\sigma_1 \cdots \sigma_k) = \begin{array}{c} \alpha \text{---} n \text{---} \mathcal{A} \text{---} n \text{---} \mathcal{A} \text{---} n \text{---} \cdots \text{---} n \text{---} \mathcal{A} \text{---} n \text{---} \omega \\ \quad \quad \quad \downarrow \quad \quad \downarrow \quad \quad \quad \quad \quad \downarrow \\ \quad \quad \quad \sigma_1 \quad \sigma_2 \quad \quad \quad \quad \quad \sigma_k \end{array}$$

- $\mathbf{H}_f \in \mathbb{R}^{\Sigma^* \times \Sigma^*}$: **Hankel matrix** of $f : \Sigma^* \rightarrow \mathbb{R}$

- ▶ *Definition*: prefix p , suffix $s \Rightarrow (\mathbf{H}_f)_{p,s} = f(ps)$

$$\begin{array}{l} a \\ b \\ aa \\ ab \\ \vdots \end{array} \begin{bmatrix} a & b & aa & ab & \dots \\ f(aa) & f(ab) & \dots & \dots & \dots \\ f(ba) & f(bb) & \dots & \dots & \dots \\ f(aaa) & f(aab) & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

Hankel matrix

- We consider the case where inputs are sequences of discrete symbols:
 - ▶ Σ a finite alphabet of size d (e.g. $\{a, b\}$)
 - ▶ Σ^* strings on Σ (e.g. $abba$)
 - ▶ A uMPS computes a function $f : \Sigma^* \rightarrow \mathbb{R}$:

$$f(\sigma_1 \cdots \sigma_k) = \begin{array}{c} \alpha \text{---} n \text{---} \mathcal{A} \text{---} n \text{---} \mathcal{A} \text{---} n \text{---} \cdots \text{---} n \text{---} \mathcal{A} \text{---} n \text{---} \omega \\ \quad \quad \quad \downarrow \quad \quad \downarrow \quad \quad \quad \quad \quad \downarrow \\ \quad \quad \quad \sigma_1 \quad \sigma_2 \quad \quad \quad \quad \quad \sigma_k \end{array}$$

- $\mathbf{H}_f \in \mathbb{R}^{\Sigma^* \times \Sigma^*}$: **Hankel matrix** of $f : \Sigma^* \rightarrow \mathbb{R}$

- ▶ *Definition*: prefix p , suffix $s \Rightarrow (\mathbf{H}_f)_{p,s} = f(ps)$

$$\begin{array}{c} a \\ b \\ aa \\ ab \\ \vdots \end{array} \begin{bmatrix} \begin{array}{ccccc} a & b & aa & ab & \dots \\ f(aa) & f(ab) & \dots & \dots & \dots \\ f(ba) & f(bb) & \dots & \dots & \dots \\ f(aaa) & f(aab) & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{array} \end{bmatrix}$$

Spectral Learning of uMPS

- $\mathbf{H}_f \in \mathbb{R}^{\Sigma^* \times \Sigma^*}$: **Hankel matrix** of $f : \Sigma^* \rightarrow \mathbb{R}$

Definition: prefix p , suffix $s \Rightarrow (\mathbf{H}_f)_{p,s} = f(ps)$

- Fundamental theorem [Carlyle and Paz, 1971; Fliess 1974]:

$\text{rank}(\mathbf{H}_f) < \infty \iff f$ can be computed by a uMPS

Spectral Learning of uMPS

- $\mathbf{H}_f \in \mathbb{R}^{\Sigma^* \times \Sigma^*}$: **Hankel matrix** of $f : \Sigma^* \rightarrow \mathbb{R}$

Definition: prefix p , suffix $s \Rightarrow (\mathbf{H}_f)_{p,s} = f(ps)$

- Fundamental theorem [Carlyle and Paz, 1971; Fliess 1974]:

$\text{rank}(\mathbf{H}_f) < \infty \iff f$ can be computed by a uMPS

\hookrightarrow Proof is constructive! From a low rank factorization of \mathbf{H}_f we can recover a uMPS computing f ...

Spectral Learning of uMPS (in a nutshell)

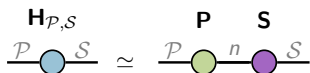
1. Choose a set of prefixes and suffixes, $\mathcal{P}, \mathcal{S} \subset \Sigma^*$.

Spectral Learning of uMPS (in a nutshell)

1. Choose a set of prefixes and suffixes, $\mathcal{P}, \mathcal{S} \subset \Sigma^*$.
2. Estimate the Hankel sub-blocks $\mathbf{h}_{\mathcal{P}} \in \mathbb{R}^{\mathcal{P}}$, $\mathbf{h}_{\mathcal{S}} \in \mathbb{R}^{\mathcal{S}}$, $\mathbf{H}_{\mathcal{P}, \mathcal{S}} \in \mathbb{R}^{\mathcal{P} \times \mathcal{S}}$, $\mathcal{H}_{\mathcal{P}, \Sigma, \mathcal{S}} \in \mathbb{R}^{\mathcal{P} \times \Sigma \times \mathcal{S}}$ defined by
 $(\mathbf{h}_{\mathcal{P}})_u = f(u)$, $(\mathbf{h}_{\mathcal{S}})_v = f(v)$, $(\mathbf{H}_{\mathcal{P}, \mathcal{S}})_{u, v} = f(uv)$ and $(\mathcal{H}_{\mathcal{P}, \Sigma, \mathcal{S}})_{u, \sigma, v} = f(u\sigma v)$

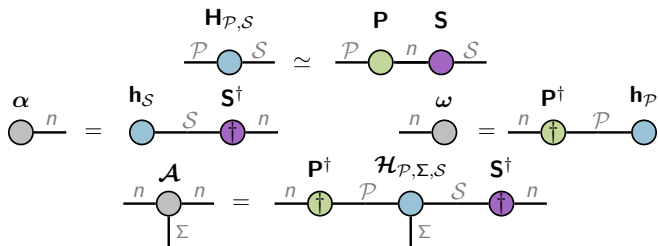
Spectral Learning of uMPS (in a nutshell)

1. Choose a set of prefixes and suffixes, $\mathcal{P}, \mathcal{S} \subset \Sigma^*$.
2. Estimate the Hankel sub-blocks $\mathbf{h}_{\mathcal{P}} \in \mathbb{R}^{\mathcal{P}}$, $\mathbf{h}_{\mathcal{S}} \in \mathbb{R}^{\mathcal{S}}$, $\mathbf{H}_{\mathcal{P},\mathcal{S}} \in \mathbb{R}^{\mathcal{P} \times \mathcal{S}}$, $\mathcal{H}_{\mathcal{P},\Sigma,\mathcal{S}} \in \mathbb{R}^{\mathcal{P} \times \Sigma \times \mathcal{S}}$ defined by $(\mathbf{h}_{\mathcal{P}})_u = f(u)$, $(\mathbf{h}_{\mathcal{S}})_v = f(v)$, $(\mathbf{H}_{\mathcal{P},\mathcal{S}})_{u,v} = f(uv)$ and $(\mathcal{H}_{\mathcal{P},\Sigma,\mathcal{S}})_{u,\sigma,v} = f(u\sigma v)$
3. Recover uMPS parameters $(\alpha, \mathcal{A}, \omega)$:



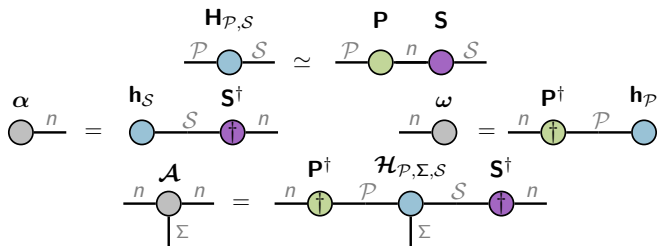
Spectral Learning of uMPS (in a nutshell)

1. Choose a set of prefixes and suffixes, $\mathcal{P}, \mathcal{S} \subset \Sigma^*$.
2. Estimate the Hankel sub-blocks $\mathbf{h}_{\mathcal{P}} \in \mathbb{R}^{\mathcal{P}}$, $\mathbf{h}_{\mathcal{S}} \in \mathbb{R}^{\mathcal{S}}$, $\mathbf{H}_{\mathcal{P},\mathcal{S}} \in \mathbb{R}^{\mathcal{P} \times \mathcal{S}}$, $\mathcal{H}_{\mathcal{P},\Sigma,\mathcal{S}} \in \mathbb{R}^{\mathcal{P} \times \Sigma \times \mathcal{S}}$ defined by $(\mathbf{h}_{\mathcal{P}})_u = f(u)$, $(\mathbf{h}_{\mathcal{S}})_v = f(v)$, $(\mathbf{H}_{\mathcal{P},\mathcal{S}})_{u,v} = f(uv)$ and $(\mathcal{H}_{\mathcal{P},\Sigma,\mathcal{S}})_{u,\sigma,v} = f(u\sigma v)$
3. Recover uMPS parameters $(\alpha, \mathcal{A}, \omega)$:



Spectral Learning of uMPS (in a nutshell)

1. Choose a set of prefixes and suffixes, $\mathcal{P}, \mathcal{S} \subset \Sigma^*$.
2. Estimate the Hankel sub-blocks $\mathbf{h}_{\mathcal{P}} \in \mathbb{R}^{\mathcal{P}}$, $\mathbf{h}_{\mathcal{S}} \in \mathbb{R}^{\mathcal{S}}$, $\mathbf{H}_{\mathcal{P},\mathcal{S}} \in \mathbb{R}^{\mathcal{P} \times \mathcal{S}}$, $\mathcal{H}_{\mathcal{P},\Sigma,\mathcal{S}} \in \mathbb{R}^{\mathcal{P} \times \Sigma \times \mathcal{S}}$ defined by $(\mathbf{h}_{\mathcal{P}})_u = f(u)$, $(\mathbf{h}_{\mathcal{S}})_v = f(v)$, $(\mathbf{H}_{\mathcal{P},\mathcal{S}})_{u,v} = f(uv)$ and $(\mathcal{H}_{\mathcal{P},\Sigma,\mathcal{S}})_{u,\sigma,v} = f(u\sigma v)$
3. Recover uMPS parameters $(\alpha, \mathcal{A}, \omega)$:



→ Efficient and consistent learning algorithms for uMPS/weighted automata [Hsu et al., 2009; Bailly et al. 2009; Balle et al., 2014, ...].

Spectral Learning: when does it work?

Theorem (Exact case)

If the set of prefixes and suffixes $\mathcal{P}, \mathcal{S} \subset \Sigma^$ are such that*

$$\text{rank}(\mathbf{H}_{\mathcal{P}, \mathcal{S}}) = \text{rank}(\mathbf{H}_f) < \infty$$

then the spectral learning algorithm returns a uMPS computing f .

Spectral Learning: when does it work?

Theorem (Exact case)

If the set of prefixes and suffixes $\mathcal{P}, \mathcal{S} \subset \Sigma^*$ are such that

$$\text{rank}(\mathbf{H}_{\mathcal{P}, \mathcal{S}}) = \text{rank}(\mathbf{H}_f) < \infty$$

then the spectral learning algorithm returns a uMPS computing f .

Suppose f is computed by a uMPS. By a continuity argument, if we are given noisy estimates

$\hat{\mathbf{H}}_{\mathcal{P}, \mathcal{S}} = \mathbf{H}_{\mathcal{P}, \mathcal{S}} + \boldsymbol{\xi}_{\mathcal{P}, \mathcal{S}}$, $\hat{\mathcal{H}}_{\mathcal{P}, \Sigma, \mathcal{S}} = \mathcal{H}_{\mathcal{P}, \Sigma, \mathcal{S}} + \boldsymbol{\xi}_{\mathcal{P}, \Sigma, \mathcal{S}}, \dots$ we have

$$\lim_{\|\boldsymbol{\xi}_{\mathcal{P}, \mathcal{S}}\| \rightarrow 0, \|\boldsymbol{\xi}_{\mathcal{P}, \Sigma, \mathcal{S}}\| \rightarrow 0} \hat{f} = f$$

where \hat{f} is the estimator returned by the spectral method.

Spectral Learning: when does it work?

Theorem (Exact case)

If the set of prefixes and suffixes $\mathcal{P}, \mathcal{S} \subset \Sigma^*$ are such that

$$\text{rank}(\mathbf{H}_{\mathcal{P}, \mathcal{S}}) = \text{rank}(\mathbf{H}_f) < \infty$$

then the spectral learning algorithm returns a uMPS computing f .

Suppose f is computed by a uMPS. By a continuity argument, if we are given noisy estimates

$$\hat{\mathbf{H}}_{\mathcal{P}, \mathcal{S}} = \mathbf{H}_{\mathcal{P}, \mathcal{S}} + \boldsymbol{\xi}_{\mathcal{P}, \mathcal{S}}, \quad \hat{\mathcal{H}}_{\mathcal{P}, \Sigma, \mathcal{S}} = \mathcal{H}_{\mathcal{P}, \Sigma, \mathcal{S}} + \boldsymbol{\xi}_{\mathcal{P}, \Sigma, \mathcal{S}}, \dots \text{ we have}$$

$$\lim_{\|\boldsymbol{\xi}_{\mathcal{P}, \mathcal{S}}\| \rightarrow 0, \|\boldsymbol{\xi}_{\mathcal{P}, \Sigma, \mathcal{S}}\| \rightarrow 0} \hat{f} = f$$

where \hat{f} is the estimator returned by the spectral method.

↪ When f is a probability distribution, we get an **unbiased and consistent** estimator! [c.f. work of B. Balle]

A closer look at the Hankel matrix of a uMPS

- Let $f : \Sigma^* \rightarrow \mathbb{R}$ be the function computed by a uMPS $(\alpha, \mathcal{A}, \omega)$.
- Define the ℓ th order Hankel tensor $\mathcal{H}^{(\ell)} \in \mathbb{R}^{\Sigma \times \Sigma \times \dots \times \Sigma}$ by

$$\mathcal{H}_{\sigma_1, \sigma_2, \dots, \sigma_\ell}^{(\ell)} = f(\sigma_1 \sigma_2 \dots \sigma_\ell)$$

A closer look at the Hankel matrix of a uMPS

- Let $f : \Sigma^* \rightarrow \mathbb{R}$ be the function computed by a uMPS $(\alpha, \mathcal{A}, \omega)$.
- Define the ℓ th order Hankel tensor $\mathcal{H}^{(\ell)} \in \mathbb{R}^{\Sigma \times \Sigma \times \dots \times \Sigma}$ by

$$\mathcal{H}_{\sigma_1, \sigma_2, \dots, \sigma_\ell}^{(\ell)} = f(\sigma_1 \sigma_2 \dots \sigma_\ell) = \begin{array}{c} \textcircled{\alpha} \text{---}^n \textcircled{\mathcal{A}} \text{---}^n \textcircled{\mathcal{A}} \text{---}^n \dots \text{---}^n \textcircled{\mathcal{A}} \text{---}^n \textcircled{\omega} \\ \qquad \qquad \qquad \downarrow^d \qquad \qquad \downarrow^d \qquad \qquad \qquad \downarrow^d \\ \qquad \qquad \qquad \sigma_1 \qquad \qquad \sigma_2 \qquad \qquad \qquad \sigma_k \end{array} \quad (1)$$

for all $\sigma_1, \dots, \sigma_\ell \in \Sigma$

A closer look at the Hankel matrix of a uMPS

- Let $f : \Sigma^* \rightarrow \mathbb{R}$ be the function computed by a uMPS $(\alpha, \mathcal{A}, \omega)$.
- Define the ℓ th order Hankel tensor $\mathcal{H}^{(\ell)} \in \mathbb{R}^{\Sigma \times \Sigma \times \dots \times \Sigma}$ by

$$\mathcal{H}_{\sigma_1, \sigma_2, \dots, \sigma_\ell}^{(\ell)} = f(\sigma_1 \sigma_2 \dots \sigma_\ell) = \begin{array}{c} \alpha \xrightarrow{n} \mathcal{A} \xrightarrow{n} \mathcal{A} \xrightarrow{n} \dots \xrightarrow{n} \mathcal{A} \xrightarrow{n} \omega \\ \quad \quad \quad \downarrow \quad \quad \quad \downarrow \quad \quad \quad \quad \quad \quad \quad \downarrow \\ \quad \quad \quad \sigma_1 \quad \quad \quad \sigma_2 \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \sigma_k \end{array} \quad (1)$$

for all $\sigma_1, \dots, \sigma_\ell \in \Sigma$

- For each ℓ , the tensor $\mathcal{H}^{(\ell)}$ has low uniform MPS rank:

$$\begin{array}{c} \mathcal{H}^{(\ell)} \\ \downarrow \downarrow \dots \downarrow \\ d \mid d \dots \mid d \end{array} = \begin{array}{c} \alpha \xrightarrow{n} \mathcal{A} \xrightarrow{n} \mathcal{A} \xrightarrow{n} \dots \xrightarrow{n} \mathcal{A} \xrightarrow{n} \omega \\ \quad \quad \quad \downarrow \quad \quad \quad \downarrow \quad \quad \quad \quad \quad \quad \quad \downarrow \\ \quad \quad \quad d \quad \quad \quad d \quad \quad \quad \quad \quad \quad \quad d \end{array} \quad (2)$$

Outline

- 1 Tensor Network Models for Sequences
- 2 A Spectral Learning Algorithm for Uniform MPS Models
- 3 Spectral Learning \equiv MPS to Uniform MPS Conversion Algorithm**
- 4 Beyond Sequences: Tensor Network Models for Trees
- 5 Conclusion

A closer look at the Hankel matrix of a uMPS

- For each ℓ , the tensor $\mathcal{H}^{(\ell)}$ (defined by $\mathcal{H}_{\sigma_1, \sigma_2, \dots, \sigma_\ell}^{(\ell)} = f(\sigma_1 \sigma_2 \cdots \sigma_\ell)$) has low uniform MPS rank:

$$\mathcal{H}^{(\ell)} = \alpha \overset{n}{\text{---}} \underset{d}{\text{A}} \overset{n}{\text{---}} \underset{d}{\text{A}} \overset{n}{\text{---}} \cdots \overset{n}{\text{---}} \underset{d}{\text{A}} \overset{n}{\text{---}} \omega \quad (3)$$

A closer look at the Hankel matrix of a uMPS

- For each ℓ , the tensor $\mathcal{H}^{(\ell)}$ (defined by $\mathcal{H}_{\sigma_1, \sigma_2, \dots, \sigma_\ell}^{(\ell)} = f(\sigma_1 \sigma_2 \cdots \sigma_\ell)$) has low uniform MPS rank:

$$\mathcal{H}^{(\ell)} = \alpha \overset{n}{\text{---}} \underset{d}{\text{A}} \overset{n}{\text{---}} \underset{d}{\text{A}} \overset{n}{\text{---}} \cdots \overset{n}{\text{---}} \underset{d}{\text{A}} \overset{n}{\text{---}} \omega \quad (3)$$

- It follows that the Hankel matrix $\mathbf{H}_f \in \mathbb{R}^{\Sigma^* \times \Sigma^*}$ can be decomposed in sub-blocks of low uMPS rank:

$$\mathbf{H}_f = \begin{array}{c} a \\ b \\ aa \\ ab \\ \vdots \end{array} \begin{bmatrix} \begin{array}{cccc} a & b & aa & ab & \dots \end{array} \\ \begin{array}{c} f(aa) & f(ab) & \dots & \dots & \dots \\ f(ba) & f(bb) & \dots & \dots & \dots \\ f(aaa) & f(aab) & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{array} \end{bmatrix}$$

A closer look at the Hankel matrix of a uMPS

- For each ℓ , the tensor $\mathcal{H}^{(\ell)}$ (defined by $\mathcal{H}_{\sigma_1, \sigma_2, \dots, \sigma_\ell}^{(\ell)} = f(\sigma_1 \sigma_2 \dots \sigma_\ell)$) has low uniform MPS rank:

$$\text{Diagram of } \mathcal{H}^{(\ell)} \text{ as a tensor with } \ell \text{ legs of dimension } d \text{ is equal to a matrix product state (MPS) with } \ell \text{ tensors } \mathcal{A} \text{ and boundary tensors } \alpha \text{ and } \omega. \quad (3)$$

- It follows that the Hankel matrix $\mathbf{H}_f \in \mathbb{R}^{\Sigma^* \times \Sigma^*}$ can be decomposed in sub-blocks of low uMPS rank:

$$\mathbf{H}_f = \begin{matrix} & a & b & aa & ab & \dots \\ \begin{matrix} a \\ b \\ aa \\ ab \\ \vdots \\ \vdots \end{matrix} & \begin{bmatrix} f(aa) & f(ab) & \dots & \dots & \dots \\ f(ba) & f(bb) & \dots & \dots & \dots \\ f(aaa) & f(aab) & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \end{matrix} = \begin{matrix} & a & b & aa & ab & ba & bb & aaa & aab & \dots \\ \begin{matrix} a \\ b \\ aa \\ ab \\ ba \\ bb \\ aaa \\ aab \\ \vdots \end{matrix} & \begin{bmatrix} \mathcal{H}_{\Sigma \times \Sigma}^{(2)} & & & \mathcal{H}_{\Sigma \times \Sigma^2}^{(3)} & & & \mathcal{H}_{\Sigma \times \Sigma^3}^{(4)} & \dots \\ & & & & & & & \dots \\ \mathcal{H}_{\Sigma^2 \times \Sigma}^{(3)} & & & \mathcal{H}_{\Sigma^2 \times \Sigma^2}^{(4)} & & & \mathcal{H}_{\Sigma^2 \times \Sigma^3}^{(5)} & \dots \\ & & & & & & & \dots \\ \mathcal{H}_{\Sigma^3 \times \Sigma}^{(4)} & & & \mathcal{H}_{\Sigma^3 \times \Sigma^2}^{(5)} & & & \mathcal{H}_{\Sigma^3 \times \Sigma^3}^{(6)} & \dots \\ & & & & & & & \dots \end{bmatrix} \end{matrix}$$

Back to the spectral learning algorithm

- In the spectral algorithm, we need to estimate $(\mathbf{h}_{\mathcal{P}})_u = f(u)$, $(\mathbf{h}_{\mathcal{S}})_v = f(v)$, $(\mathbf{H}_{\mathcal{P},\mathcal{S}})_{u,v} = f(uv)$ and $(\mathcal{H}_{\mathcal{P},\Sigma,\mathcal{S}})_{u,\sigma,v} = f(u\sigma v)$ for some sets of prefixes and suffixes $\mathcal{P}, \mathcal{S} \subset \Sigma^*$.

Back to the spectral learning algorithm

- In the spectral algorithm, we need to estimate $(\mathbf{h}_{\mathcal{P}})_u = f(u)$, $(\mathbf{h}_{\mathcal{S}})_v = f(v)$, $(\mathbf{H}_{\mathcal{P},\mathcal{S}})_{u,v} = f(uv)$ and $(\mathcal{H}_{\mathcal{P},\Sigma,\mathcal{S}})_{u,\sigma,v} = f(u\sigma v)$ for some sets of prefixes and suffixes $\mathcal{P}, \mathcal{S} \subset \Sigma^*$.
- If we choose $\mathcal{P} = \mathcal{S} = \Sigma^\ell$ we have

$$\mathbf{h}_{\mathcal{P}} = \mathbf{h}_{\mathcal{S}} = \mathcal{H}_{\Sigma^\ell}^{(\ell)}, \quad \mathbf{H}_{\mathcal{P},\mathcal{S}} = \mathcal{H}_{\Sigma^\ell \times \Sigma^\ell}^{(2\ell)} \quad \text{and} \quad (\mathcal{H}_{\mathcal{P},\Sigma,\mathcal{S}}) = \mathcal{H}_{\Sigma^\ell \times \Sigma \times \Sigma^\ell}^{(2\ell+1)}$$

Back to the spectral learning algorithm

- In the spectral algorithm, we need to estimate $(\mathbf{h}_{\mathcal{P}})_u = f(u)$, $(\mathbf{h}_{\mathcal{S}})_v = f(v)$, $(\mathbf{H}_{\mathcal{P},\mathcal{S}})_{u,v} = f(uv)$ and $(\mathcal{H}_{\mathcal{P},\Sigma,\mathcal{S}})_{u,\sigma,v} = f(u\sigma v)$ for some sets of prefixes and suffixes $\mathcal{P}, \mathcal{S} \subset \Sigma^*$.
- If we choose $\mathcal{P} = \mathcal{S} = \Sigma^\ell$ we have

$$\mathbf{h}_{\mathcal{P}} = \mathbf{h}_{\mathcal{S}} = \mathcal{H}_{\Sigma^\ell}^{(\ell)}, \quad \mathbf{H}_{\mathcal{P},\mathcal{S}} = \mathcal{H}_{\Sigma^\ell \times \Sigma^\ell}^{(2\ell)} \quad \text{and} \quad (\mathcal{H}_{\mathcal{P},\Sigma,\mathcal{S}}) = \mathcal{H}_{\Sigma^\ell \times \Sigma \times \Sigma^\ell}^{(2\ell+1)}$$

↪ All the quantities we need to estimate are matricization of low uMPS rank tensors!

Back to the spectral learning algorithm

- In the spectral algorithm, we need to estimate $(\mathbf{h}_{\mathcal{P}})_u = f(u)$, $(\mathbf{h}_{\mathcal{S}})_v = f(v)$, $(\mathbf{H}_{\mathcal{P},\mathcal{S}})_{u,v} = f(uv)$ and $(\mathcal{H}_{\mathcal{P},\Sigma,\mathcal{S}})_{u,\sigma,v} = f(u\sigma v)$ for some sets of prefixes and suffixes $\mathcal{P}, \mathcal{S} \subset \Sigma^*$.
- If we choose $\mathcal{P} = \mathcal{S} = \Sigma^\ell$ we have

$$\mathbf{h}_{\mathcal{P}} = \mathbf{h}_{\mathcal{S}} = \mathcal{H}_{\Sigma^\ell}^{(\ell)}, \quad \mathbf{H}_{\mathcal{P},\mathcal{S}} = \mathcal{H}_{\Sigma^\ell \times \Sigma^\ell}^{(2\ell)} \quad \text{and} \quad (\mathcal{H}_{\mathcal{P},\Sigma,\mathcal{S}}) = \mathcal{H}_{\Sigma^\ell \times \Sigma \times \Sigma^\ell}^{(2\ell+1)}$$

- ↪ All the quantities we need to estimate are matricization of low uMPS rank tensors!
- This leads to an efficient learning algorithm:
 - ▶ Estimate $\mathcal{H}^{(\ell)}$, $\mathcal{H}^{(2\ell)}$, $\mathcal{H}^{(2\ell+1)}$ directly in the MPS/TT format
 - ▶ Use the spectral algorithm to convert the MPS decomposition into a **uniform** MPS model.

Spectral Learning \equiv Conversion from MPS to uMPS

- Let $f : \Sigma^* \rightarrow \mathbb{R}$ be a function for which we have access to an MPS decomposition of the Hankel tensors $\mathcal{H}^{(\ell)}, \mathcal{H}^{(2\ell)}, \mathcal{H}^{(2\ell+1)}$.
 $\rightarrow f$ can be a probability distribution, or the wave function of a quantum system.
- Spectral learning algorithm \equiv **efficient** way to recover a uMPS computing f from the 3 Hankel tensors

Spectral Learning \equiv Conversion from MPS to uMPS

- Let $f : \Sigma^* \rightarrow \mathbb{R}$ be a function for which we have access to an MPS decomposition of the Hankel tensors $\mathcal{H}^{(\ell)}, \mathcal{H}^{(2\ell)}, \mathcal{H}^{(2\ell+1)}$.
 - $\rightarrow f$ can be a probability distribution, or the wave function of a quantum system.
 - Spectral learning algorithm \equiv **efficient** way to recover a uMPS computing f from the 3 Hankel tensors
- \hookrightarrow From $\mathcal{H}^{(\ell)}, \mathcal{H}^{(2\ell)}, \mathcal{H}^{(2\ell+1)}$, we can compute the value of f on **sequences of arbitrary length!**

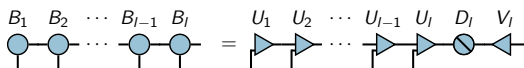
Input: $\mathcal{H}^{(\ell)} = \begin{array}{c} A_1 \quad A_2 \quad \cdots \quad A_{l-1} \quad A_l \\ \circ \quad \circ \quad \cdots \quad \circ \quad \circ \\ | \quad | \quad \cdots \quad | \quad | \end{array}$, $\mathcal{H}^{(2\ell)} = \begin{array}{c} B_1 \quad B_2 \quad \cdots \quad B_{2l-1} \quad B_{2l} \\ \circ \quad \circ \quad \cdots \quad \circ \quad \circ \\ | \quad | \quad \cdots \quad | \quad | \end{array}$, $\mathcal{H}^{(2\ell+1)} = \begin{array}{c} C_1 \quad C_2 \quad \cdots \quad C_{2l} \quad C_{2l+1} \\ \circ \quad \circ \quad \cdots \quad \circ \quad \circ \\ | \quad | \quad \cdots \quad | \quad | \end{array}$

Output: uMPS $(\alpha, \mathcal{A}, \omega)$ computing f

Input: $\mathcal{H}^{(\ell)} = \begin{array}{c} A_1 \quad A_2 \quad \cdots \quad A_{l-1} \quad A_l \\ \circ \text{---} \circ \text{---} \cdots \text{---} \circ \text{---} \circ \\ | \quad | \quad \quad \quad | \quad | \end{array}$, $\mathcal{H}^{(2\ell)} = \begin{array}{c} B_1 \quad B_2 \quad \cdots \quad B_{2l-1} \quad B_{2l} \\ \circ \text{---} \circ \text{---} \cdots \text{---} \circ \text{---} \circ \\ | \quad | \quad \quad \quad | \quad | \end{array}$, $\mathcal{H}^{(2\ell+1)} = \begin{array}{c} C_1 \quad C_2 \quad \cdots \quad C_{2l} \quad C_{2l+1} \\ \circ \text{---} \circ \text{---} \cdots \text{---} \circ \text{---} \circ \\ | \quad | \quad \quad \quad | \quad | \end{array}$

Output: uMPS $(\alpha, \mathcal{A}, \omega)$ computing f

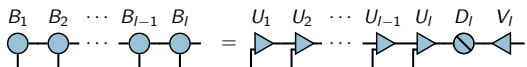
1. Left-orthonormalisation of B_1, \dots, B_ℓ (first half of $\mathcal{H}^{(2\ell)}$)



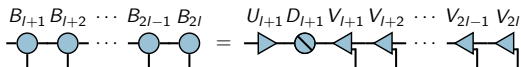
Input: $\mathcal{H}^{(\ell)} = \begin{array}{c} A_1 \quad A_2 \quad \cdots \quad A_{l-1} \quad A_l \\ \circ \text{---} \circ \text{---} \cdots \text{---} \circ \text{---} \circ \\ | \quad | \quad \quad \quad | \quad | \end{array}$, $\mathcal{H}^{(2\ell)} = \begin{array}{c} B_1 \quad B_2 \quad \cdots \quad B_{2l-1} \quad B_{2l} \\ \circ \text{---} \circ \text{---} \cdots \text{---} \circ \text{---} \circ \\ | \quad | \quad \quad \quad | \quad | \end{array}$, $\mathcal{H}^{(2\ell+1)} = \begin{array}{c} C_1 \quad C_2 \quad \cdots \quad C_{2l} \quad C_{2l+1} \\ \circ \text{---} \circ \text{---} \cdots \text{---} \circ \text{---} \circ \\ | \quad | \quad \quad \quad | \quad | \end{array}$

Output: uMPS $(\alpha, \mathcal{A}, \omega)$ computing f

1. Left-orthonormalisation of B_1, \dots, B_ℓ (first half of $\mathcal{H}^{(2\ell)}$)



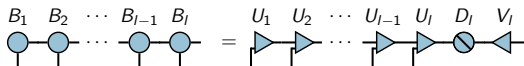
2. Right-orthonormalisation of $B_{\ell+1}, \dots, B_{2\ell}$ (second half of $\mathcal{H}^{(2\ell)}$)



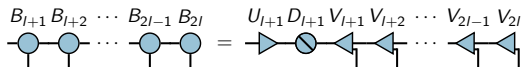
Input: $\mathcal{H}^{(\ell)} = \begin{array}{c} A_1 \quad A_2 \quad \dots \quad A_{l-1} \quad A_l \\ \circ \quad \circ \quad \dots \quad \circ \quad \circ \\ | \quad | \quad \dots \quad | \quad | \end{array}, \mathcal{H}^{(2\ell)} = \begin{array}{c} B_1 \quad B_2 \quad \dots \quad B_{2l-1} \quad B_{2l} \\ \circ \quad \circ \quad \dots \quad \circ \quad \circ \\ | \quad | \quad \dots \quad | \quad | \end{array}, \mathcal{H}^{(2\ell+1)} = \begin{array}{c} C_1 \quad C_2 \quad \dots \quad C_{2l} \quad C_{2l+1} \\ \circ \quad \circ \quad \dots \quad \circ \quad \circ \\ | \quad | \quad \dots \quad | \quad | \end{array}$

Output: uMPS $(\alpha, \mathcal{A}, \omega)$ computing f

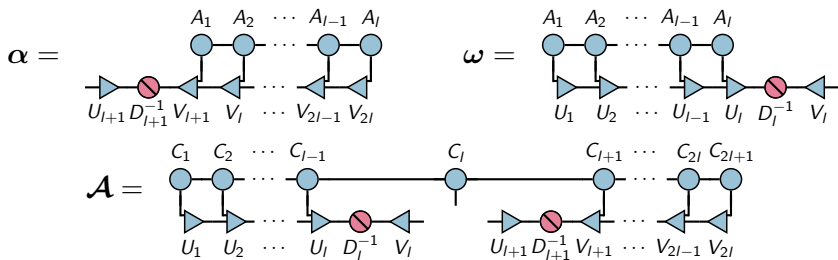
1. Left-orthonormalisation of B_1, \dots, B_ℓ (first half of $\mathcal{H}^{(2\ell)}$)



2. Right-orthonormalisation of $B_{\ell+1}, \dots, B_{2\ell}$ (second half of $\mathcal{H}^{(2\ell)}$)



3. Computation of the uMPS parameters:



Theoretical guarantees for MPS to uMPS conversion

Theorem (Exact case)

If ℓ is such that $\text{rank}(\mathbf{H}_{\Sigma^\ell \times \Sigma^\ell}^{2\ell}) = \text{rank}(\mathbf{H}_f) < \infty$, then the MPS to uMPS conversion algorithm returns a uMPS computing $f : \Sigma^ \rightarrow \mathbb{R}$.*

Theoretical guarantees for MPS to uMPS conversion

Theorem (Exact case)

If ℓ is such that $\text{rank}(\mathbf{H}_{\Sigma^\ell \times \Sigma^\ell}^{2\ell}) = \text{rank}(\mathbf{H}_f) < \infty$, then the MPS to uMPS conversion algorithm returns a uMPS computing $f : \Sigma^ \rightarrow \mathbb{R}$.*

- Similarly to the spectral learning algorithm, the guarantees can be extended to the approximate setting.
- Works also with the Born rule ($|f(\sigma_1\sigma_2 \cdots \sigma_k)|^2 = \mathbb{P}(\sigma_1\sigma_2 \cdots \sigma_k)$).

Theoretical guarantees for MPS to uMPS conversion

Theorem (Exact case)

If ℓ is such that $\text{rank}(\mathbf{H}_{\Sigma^\ell \times \Sigma^\ell}^{2\ell}) = \text{rank}(\mathbf{H}_f) < \infty$, then the MPS to uMPS conversion algorithm returns a uMPS computing $f : \Sigma^* \rightarrow \mathbb{R}$.

- Similarly to the spectral learning algorithm, the guarantees can be extended to the approximate setting.
 - Works also with the Born rule ($|f(\sigma_1\sigma_2 \cdots \sigma_k)|^2 = \mathbb{P}(\sigma_1\sigma_2 \cdots \sigma_k)$).
- ⇒ **Provable algorithm to convert MPS to uMPS!**
- ▶ Any equivalent algorithm in quantum physics literature?
 - ▶ Any use in quantum physics?

Outline

- 1 Tensor Network Models for Sequences
- 2 A Spectral Learning Algorithm for Uniform MPS Models
- 3 Spectral Learning \equiv MPS to Uniform MPS Conversion Algorithm
- 4 Beyond Sequences: Tensor Network Models for Trees
- 5 Conclusion

Beyond Strings: Weighted Tree Automata

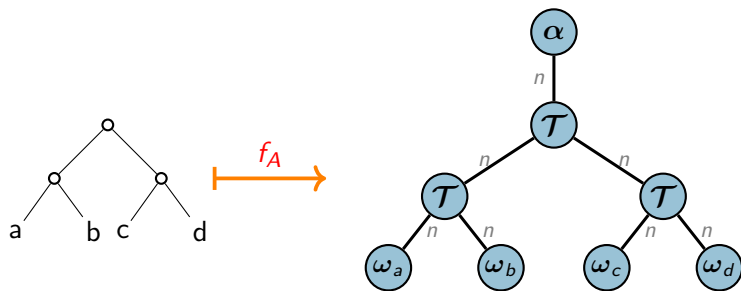
- A **weighted tree automaton** (WTA) is a tuple $A = \langle \alpha, \mathcal{T}, \{\omega_\sigma\}_{\sigma \in \Sigma} \rangle$

$\alpha \in \mathbb{R}^n$: vector of **initial weights**

$\mathcal{T} \in \mathbb{R}^{n \times n \times n}$: tensor of **transition weights**

$\omega_\sigma \in \mathbb{R}^n$: vector of **final weights** associated with $\sigma \in \Sigma$

- A WTA computes a function $f_A : \mathfrak{T}_\Sigma \rightarrow \mathbb{R}$.



Learning Weighted Tree Automaton

- WTA and other models:
 - ▶ Probabilistic WTA \equiv Probabilistic Context Free Grammars
 - ▶ WTA \equiv Linear Recursive Tensor Neural Network (Socher et al., 2010)
 - ▶ Similarly to sequences, WTA \equiv Uniform Tree Tensor Networks

Learning Weighted Tree Automaton

- WTA and other models:
 - ▶ Probabilistic WTA \equiv Probabilistic Context Free Grammars
 - ▶ WTA \equiv Linear Recursive Tensor Neural Network (Socher et al., 2010)
 - ▶ Similarly to sequences, WTA \equiv Uniform Tree Tensor Networks
- The spectral learning algorithm works also for trees (Bailly, Habrard and Denis, 2010)

Learning Weighted Tree Automaton

- WTA and other models:
 - ▶ Probabilistic WTA \equiv Probabilistic Context Free Grammars
 - ▶ WTA \equiv Linear Recursive Tensor Neural Network (Socher et al., 2010)
 - ▶ Similarly to sequences, WTA \equiv Uniform Tree Tensor Networks
- The spectral learning algorithm works also for trees (Bailly, Habrard and Denis, 2010)
- ↪ Spectral Learning \equiv Efficient way to convert tree TN to uniform tree TN!

Outline

- 1 Tensor Network Models for Sequences
- 2 A Spectral Learning Algorithm for Uniform MPS Models
- 3 Spectral Learning \equiv MPS to Uniform MPS Conversion Algorithm
- 4 Beyond Sequences: Tensor Network Models for Trees
- 5 Conclusion

Discussion

- Lots of connections between quantum TN, probabilistic models, formal languages, machine learning, ...

Discussion

- Lots of connections between quantum TN, probabilistic models, formal languages, machine learning, ...
- Connections between MPS and weighted automata:
 - ▶ Provable learning algorithm from formal languages / probabilistic models viewed through the lens of TN
 - ▶ Learning \equiv MPS to uMPS conversion algorithm. (Any use in physics?)
 - ▶ Connections can also be leveraged for versatile sampling techniques for uMPS models, see

Jacob Miller, Guillaume Rabusseau, and John Terilla. *Tensor Networks for Language Modeling*. arXiv preprint arXiv:2003.01039 (2020).

Discussion

- Lots of connections between quantum TN, probabilistic models, formal languages, machine learning, ...
- Connections between MPS and weighted automata:
 - ▶ Provable learning algorithm from formal languages / probabilistic models viewed through the lens of TN
 - ▶ Learning \equiv MPS to uMPS conversion algorithm. (Any use in physics?)
 - ▶ Connections can also be leveraged for versatile sampling techniques for uMPS models, see
Jacob Miller, Guillaume Rabusseau, and John Terilla. *Tensor Networks for Language Modeling*.
arXiv preprint arXiv:2003.01039 (2020).
- Next steps:
 - ▶ Similar connections and algorithms can be derived for models on trees
 - ▶ What about graphs? (e.g. potential connections between TN and GNN)

Thank you! Questions?