Training Tensor Trains, a brief travail (and comparison with HMMs)

15 December 2020 IFT6269 Group 22: Js Palucc and Hoov {jacob.hoover, jonathan.palucci}@mail.mcgill.ca

Overview

There is a theoretical equivalence between a certain type of tensor network called a matrix product state (MPS) or tensor train (TT), with positive values, and HMMs.

Other similar tensor networks called Born machines are not exactly equivalent but are closely related. We implemented these tensor networks, and tested their capacity to fit categorical data, comparing them to each other and to an HMM model in practice.

What we did

- We based our project on Glasser et al. (2019). They provided code for training these tensor networks. Limitations:
  - code was written in numpy
  - explicit computation of model gradients,
  - training by a custom implementation of a batched gradient descent algorithm.
- For more flexibility in we rewrote the models from scratch
  - rewritten in pytorch using `einsum`
  - autograd for differentiation (simpler and much more flexible!)[A]
  - flexibility to choose optimization algorithms
  - option for homogeneous as well as non-homogeneous models

Contraction algorithm

General algorithm

The learning algorithms used minimize the negative log-likelihood:

$$-\sum_{i=1}^{N} \log \frac{T_{x_i}}{Z_T} = -\log \frac{\hat{p}(x_{1:n})}{Z(A)}.$$

Tensor networks make use of a diagrammatic calculus where collapsing an index corresponds to tensor contraction.

- Contract network to get unnormalized probability value $\hat{p}(x_{1:n})$:
  - Contract left boundary vector, α, with the first tensor core A_1
  - Contract tensor cores with each other from left to right, and terminate by contracting with right boundary vector ω (see Figure 1a)
  - For a Born machine, take the modulus squared to get a positive real number.
- Contract network to get normalizing constant, $Z(A)$.
  - For a positive MPS, we contract the network in the same way but at each tensor core, we sum over all possible values of the input
  - For a Born machine, we take two copies of the network and stick them together (Figure 1b)

Then, the normalized probability is

$$p(x_{1:n}) = \frac{\hat{p}(x_{1:n})}{Z(A)},$$

where $\hat{p}(x_{1:n}) = |f(x_{1:n})|^2$ for Born machines.

Um, what is a Tensor Network?

Tensor T: n-order array (array with n different indices).

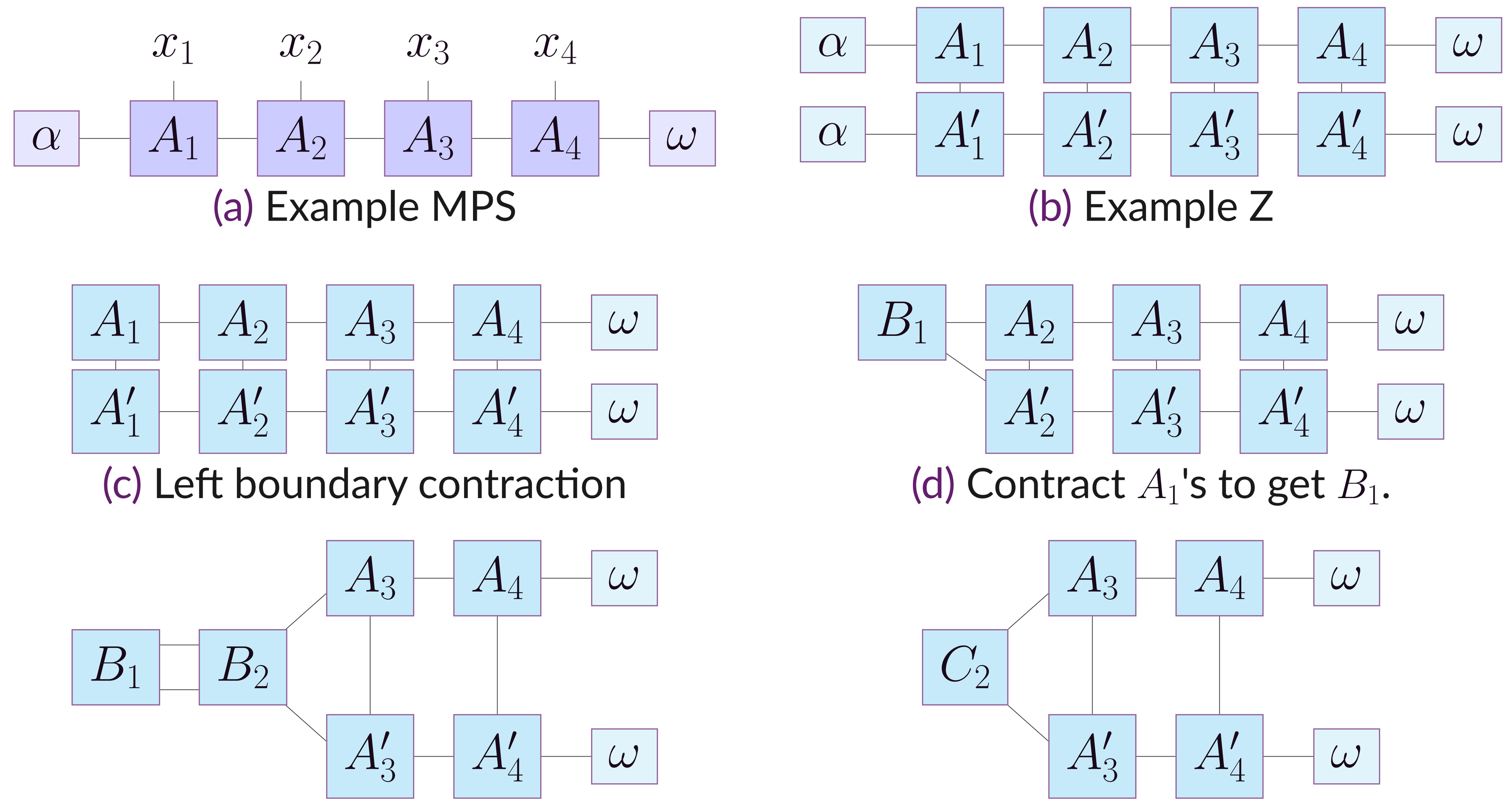
- vector: A matrix: A 3-order tensor: A
- tensor contraction: A B $=$ AB

Representing probability distribution

A probability distribution for discrete $X_1, \ldots, X_N$ over $\{1, \ldots, d\}$ represented as tensor T with d^N entries, $T_{X_1,\ldots,X_N} = P(X_1, \ldots, X_N)$.

T

Tensor network: a factorization of a large (non-negative) tensor into a network of smaller tensors.



(a) Example MPS (b) Example Z

(c) Left boundary contraction (d) Contract A_1's to get B_1.

(e) Contract A_2's to get B_2. (f) Contract B_1 and B_2 to get C_2.

Figure 1: Demonstration of contraction.

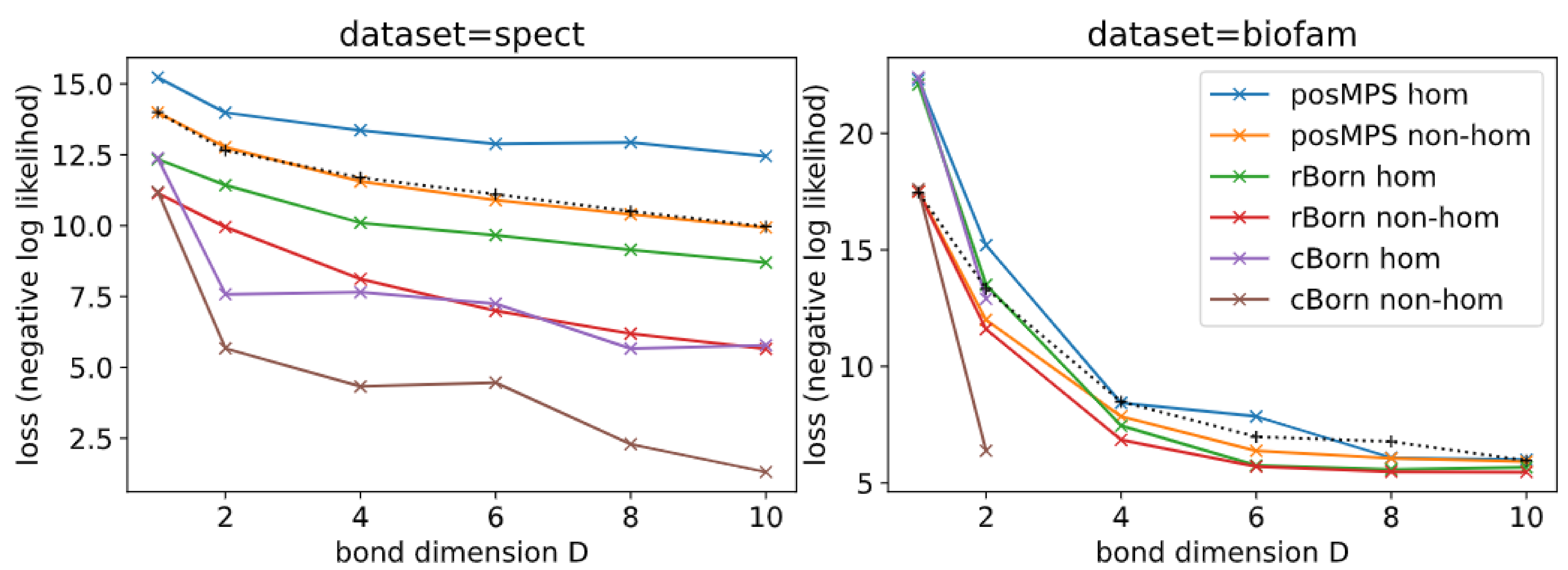


Figure 2: Results on two different datasets. Note that the *biofam* dataset is naturally sequential, while *spect* is not. Results from complex Born models may be unreliable: these models behave unstably when loss is low (particularly at higher bond dimensions), so some results are omitted.

Numerical Stability Trick

- initialize direction unit vector $\tilde{v}_0 \leftarrow \alpha / \|\alpha\|$
- initialize log norm scalar $c_0 \leftarrow \log \|\alpha\|$

`for` i in length of sequence:

- $v_i \leftarrow \exp[c_i] \tilde{v}_i$
- $\tilde{v}_{i+1}^{(\text{temp})} \leftarrow v_i$ contract with $A_{i+1} x_{i+1}$
- $c_{i+1} = c_i + \log \left\| \tilde{v}_{i+1}^{(\text{temp})} \right\|$
- $\tilde{v}_{i+1} \leftarrow \tilde{v}_{i+1}^{(\text{temp})} / \log \left\| \tilde{v}_{i+1}^{(\text{temp})} \right\|$

`endfor`

- return unnormalized probability $\hat{p}(x_{1:n}) = \exp[c_n] + \log(\tilde{v}_n^\top \omega)$

Training and comparison

HMM model for comparison

Modified version of the HMM implemented in HW4 to work on multiple sequences of categorical data. Also a model from package `pomegranate' (Schreiber, 2018) for comparison.

Preliminary Results (Figure 2)

- for the most part, the homogeneous models have worse performance than the non-homogeneous models
- qualitatively we see the same pattern that Glasser et al. report.
  - Compared to Positive MPS, Born models achieve better fit, with complex outperforming real; higher bond dimension = better fit
  - performance of non-homogeneous MPS is identical to homogeneous HMM
  - log-stability improves performance beyond Glasser's results for real Born

Next steps

- test generalization performance
- understand effect of homogeneity (depends on data set)
- understand why complex models are unstable throughout training

Note and References

Glasser, I., Sweke, R., Pancotti, N., Eisert, J., and Cirac, I. (2019). Expressive power of tensor-network factorizations for probabilistic modeling. In *Advances in Neural Information Processing Systems*, pages 1496--1508.

Miller, J., Rabusseau, G., and Terilla, J. (2020). Tensor networks for probabilistic sequence modeling.

Schreiber, J. (2018). pomegranate: Fast and flexible probabilistic modeling in python. *Journal of Machine Learning Research*, 18(164):1--6.

Notes

[A] Also pytorch-nightly v1.8 just came out with autograd support for einsum with complex floats!