Modeling sequences with quantum states: predicting generalization quality of generative models

Tai-Danae Bradley
CUNY Graduate Center
New York, NY
`tbradley@gradcenter.cuny.edu`

E. Miles Stoudenmire
Center for Computational Quantum Physics
Flatiron Institute, New York, NY 10010 USA
`mstoudenmire@flatironinstitute.org`

John Terilla
Tunnel
New York, NY
`john@tunnel.tech`

Abstract

Classical probability distributions on sets of sequences can be modeled using a mathematical formalism identical to that used for quantum states. Here, we do so with a quantum state that is pure and entangled. One advantage of this approach is that the reduced densities describing subsystems also carry information about the complementary subsystem. This is in contrast to the classical marginal distributions on a subsystem for which information about the complementary system is lost. A training algorithm based on the density matrix renormalization group (DMRG) procedure uses the extra information contained in the reduced densities and organizes it into a tensor network model. An understanding of the extra information contained in the reduced densities allow us to examine the mechanics and results of this DMRG algorithm and to study the generalization error of the resulting model. As an illustration, we work with the even-parity dataset and produce an estimate for the generalization error as a function of the fraction of the dataset used in training.

1 Introduction

In this paper, we present a deterministic algorithm for unsupervised generative modeling on strings using tensor networks. The algorithm is deterministic with a fixed number of steps and the resulting model has a perfect sampling algorithm that allows efficient sampling from marginal distributions, or sampling conditioned on a substring. The algorithm is inspired by the density matrix renormalization group (DMRG) procedure [1, 2, 3]. This approach, at its heart, involves only simple linear algebra which allows us to give a detailed "under the hood" look at the algorithm in action. Our analysis illustrates how to interpret and make predictions about the trained model. We work through the algorithm with an exemplar dataset to produce a prediction for the generalization error as a function of the fraction used in training which well approximates the generalization error observed in experiments.

The machine learning problem of interest is to learn a probability distribution on a set of sequences from a finite training set of samples. For us, an important technical and conceptual first step is to pass from *Finite Sets* to *Functions on Finite Sets*. Functions on sets have more structure than sets themselves and we find that the extra structure is meaningful. Furthermore, well-understood concepts and techniques in quantum physics give us powerful tools to exploit this extra structure without incurring significant algorithmic costs [4]. We emphasize that it is not necessary that the datasets being modeled have any inherently quantum properties or interpretation. The inductive bias of the model can be understood as a kind of low-rank factorization hypothesis—a point we expand upon in this work.

Reduced density operators play a central role in our model. In a happy coincidence, they play the central role in both the model's theoretical inspiration and the training algorithm. There is structure in reduced densities that inspire us to model classical probability distributions using a quantum model. The training algorithm amounts to successively matching reduced densities, a process which leads inevitably to a tensor network model, which may be thought of as a sequence of compatible autoencoders. We refer readers unfamiliar with tensor diagram notation to references such as [5, 6, 7].

This paper builds on investigations of tensor networks as models for machine learning tasks. Tensor networks have been demonstrated to give good results for supervised learning and regression tasks [8, 3, 9, 10, 11, 12, 13, 14]. They have also been applied successfully to unsupervised, generative modeling [15, 16, 17, 18]. The work we do proposes and studies an alternative algorithm for optimizing MPS for generative modeling.

2 The model and the training algorithm

Let $\pi : S \to \mathbb{R}$ be a probability distribution on a finite set S. In this work, S is a set of sequences of fixed length N. We will briefly explain three things: first, what it means to model π as a quantum state; second, the basic inputs and outputs of our training algorithm and how we assess performance; and third, the key steps in the training algorithm.

First, we pass from the finite set S to a vector space V that has S as an orthonormal basis. We use $|s\rangle$ to denote the vector in V corresponding to the element $s \in S$. The vector $|\psi\rangle \in V$ defined by

$$|\psi\rangle := \sum_{s\in S} \sqrt{\pi(s)}|s\rangle \tag{1}$$

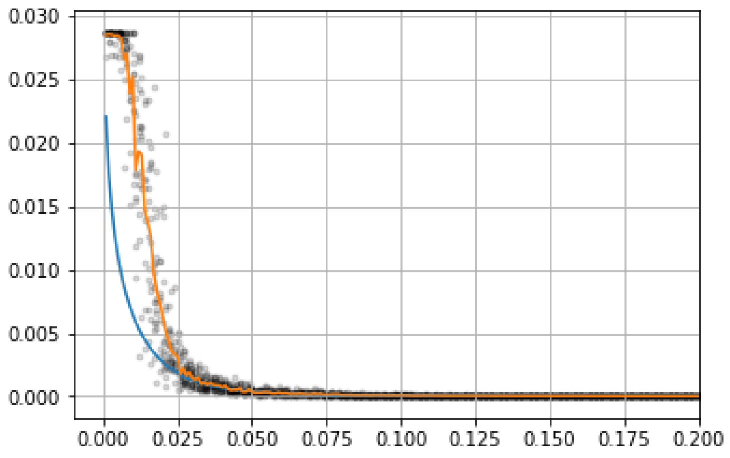
has the property that $\pi(s) = |\langle\psi|s\rangle|^2$ for all $s \in S$. The dimension of V equals the number of elements in the set S, which is exponentially large in the sequence length N. The vector $|\psi\rangle$ defines a density operator on V. Generally speaking, a density operator is called a quantum state and describes a system of interacting "particles," in this case N of them, and determines the expectation values and correlation functions of all possible observables on that system. The density operator defines reduced densities that describe the states of smaller subsystems. In this work, the density operator on V is defined by a probability distribution on S. The reduced densities contain classical information about the marginal probabilities, along with extra information. The extra information concerns how the subsystems interact with environment systems with which they are entangled and is fully encoded in the spectral information of the reduced densities. Harnessing this spectral information is a key part of our training algorithm.

The training algorithm we use begins with a training set $T \subseteq S$ drawn independently from π and produces a vector $|\psi_{\text{MPS}}\rangle$. The vector $|\psi_{\text{MPS}}\rangle$ is produced as a matrix product state (MPS), which is a particular kind of factorization of a vector into a sequence of tensors of order two and three. The algorithm is deterministic and depends only on the training set T and a chosen "bond dimension" for the MPS, which is a small integer hyperparameter. In our numerical experiments, the bond dimension is set to 2. The resulting vector $|\psi_{\text{MPS}}\rangle$ has unit norm and therefore defines a probability distribution on S by setting the probability of $s \in S$ to be $|\langle\psi_{\text{MPS}}|s\rangle|^2$. In order to assess the quality of the trained model, we compute a weighted Bhattacharya distance between the true distribution and the one defined by the trained model. Properties of tensor networks and the special structure of the parity dataset allow us to exactly compute the Bhattacharya distance over the entire data set S. This measure yields a good measure of model performance and in particular, the ability of the model to generalize.

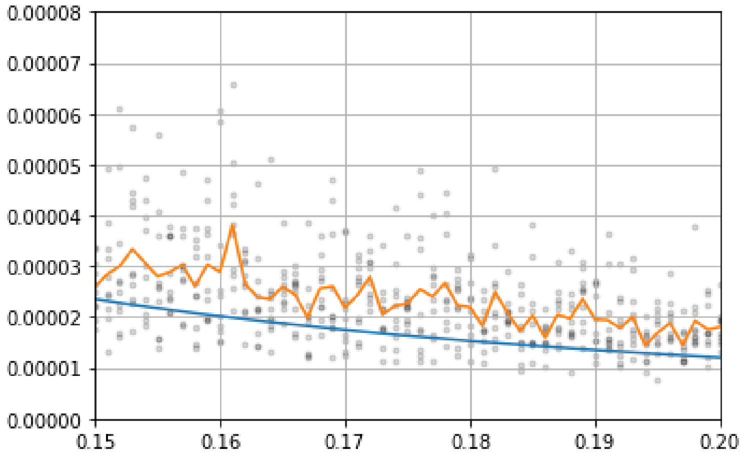
The algorithm itself operates by building the tensors that will comprise the final tensor-factorized state by an inductive procedure in N steps. To understand an inductive step, suppose that we have computed the MPS tensors $T_0, T_1, \ldots, T_k$ up to step $k < N$. This information partially defines $|\psi_{\text{MPS}}\rangle$ and thus allows us to understand important information about a reduced density describing a smaller subsystem. The next tensor T_{k+1} is determined by the interactions between that subsystem and a small extension of the subsystem as witnessed in the training set. We do so in two steps. The first step is exact, and the second truncates the spectral information according to the bond dimension hyperparameter. The second step should be regarded as a kind of regularization: keeping all of the spectral information amounts to memorizing information found in the training set, and truncating this information is akin to ignoring artifacts arising from sampling.

Because the training algorithm uses purely linear algebraic techniques, we are able to gain an in-depth understanding of it to predict experimental results given only the fraction of training samples used. To make this theoretical prediction, we develop a detailed understanding of how spectral information encodes subsystem interactions. Analyzing the combinatorics of subsystem interactions allows us to make statistical predictions about the expected interactions and how errors will propagate through the algorithm. Let us pause to share more detail about the theory behind these ideas.

Briefly, at step $k < N$ of the algorithm, each sequence in the training set T is viewed as the concatenation of a sequence of length k, called a prefix, with a sequence of length $N - k$, called a suffix. The combinatorics of prefix-suffix interactions completely determine the entries of the matrix representing the reduced density obtained at step k. This reduced density carries information about the prefix subsystem, which is always kept small by the regularization step mentioned above. We understand the combinatorics in full, and therefore we have explicit expressions for the matrix representing the reduced density and thus for its eigenvectors. These eigenvectors are then used to build the next tensor in the MPS tensor network. As a consequence, we have an explicit expression of any error accrued: it is a function of the eigenvectors, and thus ultimately of the combinatorics of the prefix-suffix interactions.



(a) The experimental average (orange) and theoretical prediction (blue).



(b) A closer look for $0.15 \leq f \leq 0.2$.

Figure 1: The experimental average (orange) and theoretical prediction (blue) of the weighted Bhattacharya distance between the probability distribution learned experimentally and the theoretical prediction for bitstrings of length $N = 16$ and training set fractions of $0 < f \leq 0.2$.

In short, by following the combinatorics of subsystem interactions through the linear algebra, we obtain a transparent look into the algorithm's internal anatomy. We then use this knowledge to find the expected statistics of the combinatorics, which are used build a single expected MPS, given a certain fraction of the data set S used to form a typical training set T.

3 Experimental results

The plots in Figure 1 compare our theoretical estimate based on the expected MPS against numerical experiments based on sampling over many training sets of a certain size. We take the entire dataset S to be the set of all even-parity bitstrings of length $N = 16$. For each fixed fraction $f = 0.001, 0.002, \ldots, 0.200$, we run the algorithm on ten different randomly selected datasets T, each containing $f \cdot |S|$ bitstrings. To measure performance, we compute the average of Bhattacharya distances between the probability distribution determined by $|\psi_{\mathrm{MPS}}\rangle$ and the true distribution (orange curve), and we compare it to our theoretical prediction (blue curve).

4 Conclusion

Models based on tensor networks open interesting directions for machine learning research. Tensor networks can be viewed as a sequence of related linear maps, which by acting together on a very high-dimensional space allows the model to be arbitrarily expressive. The underlying linearity and powerful techniques from linear algebra allow us to pursue a training algorithm where we can look "under the hood" to understand each step and its consequences for the ability of our model to reconstruct a particular data set, the even-parity data set.

Our work also highlights the advantages of working in a probability formalism based on the 2-norm. This is the same formalism used to interpret the wavefunction in quantum mechanics; here we use it as a framework to treat classical data. Density matrices naturally arise as the 2-norm analogue of marginal probability distributions familiar from conventional 1-norm probability. Marginals still appear as the diagonal of the density matrix. Unlike marginals, the density matrices we use hold sufficient information to reconstruct the entire joint distribution. Our training algorithm can be summarized as estimating the density matrix from the training data, then reconstructing the joint distribution step-by-step from these density matrix estimates.

The theoretical predictions we obtained for the generative performance of the model agree well with the experimental results. Note that care is needed to compare these results, since the theoretical approach involves averaging over all possible training sets to produce a single typical weight MPS, whereas the experiments produce a different weight MPS for each training-set sample. In the near future, we look forward to extending our approach to other measures of model performance and behavior, and certainly other data sets as well.

More ambitiously, we hope this work points the way to theoretically sound and robust predictions of machine learning model performance based on empirical summaries of real-world data. If such predictions can be obtained for training algorithms that also produce state-of-the art results, as tensor networks are starting to do, we anticipate this will continue to be an exciting program of research.

References

[1] Ulrich Schollwöck. The density-matrix renormalization group in the age of matrix product states. *Annals of Physics*, 326(1):96–192, 2011.

[2] Steven R. White. Density matrix formulation for quantum renormalization groups. *Physical Review Letters*, 69(19):2863–2866, 1992.

[3] E. Miles Stoudenmire and D. J. Schwab. Supervised learning with quantum-inspired tensor networks. *Advances in Nueral Information Processing Systems (NIPS)*, 29:4799–4807, 2016.

[4] M. M. Wolf D. Perez-Garcia, F. Verstraete and J. I. Cirac. Matrix product state representations. *Quantum Information and Computation*, 7:401–430, 2007.

[5] The tensor network, 2019. `http://tensornetwork.org`.

[6] Glen Evenbly. Tensors.net, 2019. `https://www.tensors.net`.

[7] Román Orús. A practical introduction to tensor networks: Matrix product states and projected entangled pair states. *Annals of Physics*, 349:117–158, 2014.

[8] Alexander Novikov, Mikhail Trofimov, and Ivan Oseledets. Exponential machines. *arxiv:1605.03795*, 05 2016.

[9] E Miles Stoudenmire. Learning relevant features of data with multi-scale tensor networks. *Quantum Science and Technology*, 3(3):034003, 2018.

[10] Ivan Glasser, Nicola Pancotti, and J. Ignacio Cirac. Supervised learning with generalized tensor networks. *arxiv:1806.05964*, 06 2018.

[11] Chu Guo, Zhanming Jie, Wei Lu, and Dario Poletti. Matrix product operators for sequence-to-sequence learning. *Phys. Rev. E*, 98:042114, Oct 2018.

[12] Glen Evenbly. Number-state preserving tensor networks as classifiers for supervised learning. *arxiv:1905.06352*, 2019.

[13] Ding Liu, Shi-Ju Ran, Peter Wittek, Cheng Peng, Raul Blázquez García, Gang Su, and Maciej Lewenstein. Machine learning by unitary tensor network of hierarchical tree structure. *New Journal of Physics*, 21(7):073059, jul 2019.

[14] Stavros Efthymiou, Jack Hidary, and Stefan Leichenauer. TensorNetwork for machine learning. *arXiv:1906.06329*, 2019.

[15] Zhao-Yu Han, Jun Wang, Heng Fan, Lei Wang, and Pan Zhang. Unsupervised generative modeling using matrix product states. *Phys. Rev. X*, 8:031012, Jul 2018.

[16] Zhuan Li and Pan Zhang. Shortcut matrix product states and its applications. *arxiv:1812.05248*, 12 2018.

[17] James Stokes and John Terilla. Probabilistic modeling with matrix product states. *arxiv:1902.06888*, 02 2019.

[18] Song Cheng, Lei Wang, Tao Xiang, and Pan Zhang. Tree tensor networks for generative modeling. *Phys. Rev. B*, 99:155131, Apr 2019.