Supervised Learning With Quantum-Inspired Tensor Networks

E. Miles Stoudenmire$^{1,2}$ and David J. Schwab$^3$

$^1$Perimeter Institute for Theoretical Physics, Waterloo, Ontario, N2L 2Y5, Canada
$^2$Department of Physics and Astronomy, University of California, Irvine, CA 92697-4375 USA
$^3$Dept. of Physics, Northwestern University, Evanston, IL (Dated: May 20, 2016)

Tensor networks are efficient representations of high-dimensional tensors which have been very successful for physics and mathematics applications. We demonstrate how algorithms for optimizing such networks can be adapted to supervised learning tasks by using matrix product states (tensor trains) to parameterize models for classifying images. For the MNIST data set we obtain less than 1% test set classification error. We discuss how the tensor network form imparts additional structure to the learned model and suggest a possible generative interpretation.

I. INTRODUCTION

The connection between machine learning and statistical physics has long been appreciated \[1\]–\[9\], but deeper relationships continue to be uncovered. For example, techniques used to pre-train neural networks \[8\] have more recently been interpreted in terms of the renormalization group \[10\]. In the other direction there has been a sharp increase in applications of machine learning to chemistry, material science, and condensed matter physics \[11\]–\[18\], which are sources of highly-structured data and could be a good testing ground for machine learning techniques.

A recent trend in both physics and machine learning is an appreciation for the power of tensor methods. In machine learning, tensor decompositions can be used to solve non-convex optimization tasks \[19\]–\[20\] and make progress on many other important problems \[21\]–\[23\], while in physics, great strides have been made in manipulating large vectors arising in quantum mechanics by decomposing them as tensor networks \[24\]–\[25\]. The most successful types of tensor networks avoid the curse of dimensionality by incorporating only low-order tensors, yet accurately reproduce very high-order tensors through a particular geometry of tensor contractions \[26\].

Another context where very large vectors arise is in non-linear kernel learning, where input vectors \(x\) are mapped into a higher dimensional space via a feature map \(\Phi(x)\) before being classified by a decision function

\[
f(x) = W \cdot \Phi(x) .
\]

The feature vector \(\Phi(x)\) and weight vector \(W\) can be exponentially large or even infinite. One approach to deal with such vectors is the well-known kernel trick, which only requires scalar products of feature vectors \[27\].

In what follows we propose a rather different approach. For certain learning tasks and a specific class of feature map \(\Phi\), we find the optimal weight vector \(W\) can be approximated as a tensor network, that is, as a contracted sequence of low-order tensors. Representing \(W\) as a tensor network presents opportunities to extract information hidden within the trained model and to exploit the structure of \(W\) in developing optimization algorithms.

One of the best understood types of tensor networks is the matrix product state \[25\]–\[28\], also known as the tensor train decomposition \[29\]. Matrix product states (MPS) have been very useful for studying quantum systems, and have recently been proposed for machine learning applications such as learning features of images \[22\] and compressing the weight layers of neural networks \[23\]. Though MPS are best suited for describing one-dimensional systems, they are powerful enough to be applied to higher-dimensional systems as well.

There has been intense research into generalizations of MPS better suited for higher dimensions and critical systems \[30\]–\[32\]. Though our proposed approach could generalize to these other types of tensor networks, as a proof of principle we will only consider the MPS decomposition in what follows. The MPS decomposition approximates an order-\(N\) tensor by a contracted chain of \(N\) lower-order tensors shown in Fig. 1 (Throughout we will use tensor diagram notation; for a brief review see Appendix A).

Representing the weights \(W\) of Eq. 1 as an MPS allows us to efficiently optimize these weights and adaptively change their number by varying \(W\) locally a few tensors at a time, in close analogy to the density matrix renormalization group algorithm used in physics \[25\]–\[33\]. Similar alternating least squares methods for tensor trains have also been explored in applied mathematics \[34\].

This paper is organized as follows: we propose our general approach then describe an algorithm for optimizing the weight vector \(W\) in MPS form. We test our approach, both on the MNIST handwritten digit set and on two-dimensional toy data to better understand the role of the local feature-space dimension \(d\). Finally, we discuss the class of functions realized by our proposed models as well as a possible generative interpretation.

FIG. 1. The matrix product state (MPS) decomposition, also known as a tensor train. Lines represent tensor indices and connecting two lines implies summation. For an introduction to this graphical tensor notation see Appendix A.
II. ENCODING INPUT DATA

The most successful use of tensor networks in physics so far has been in quantum mechanics, where combining $N$ independent systems corresponds to taking the tensor product of their individual state vectors. With the goal of applying similar tensor networks to machine learning, we choose a feature map of the form

$$\Phi^{s_1 s_2 \cdots s_N}(x) = \phi^{s_1}(x_1) \otimes \phi^{s_2}(x_2) \otimes \cdots \phi^{s_N}(x_N).$$ (2)

The tensor $\Phi^{s_1 s_2 \cdots s_N}$ is the tensor product of the same local feature map $\phi^{s_j}(x_j)$ applied to each input $x_j$, where the indices $s_j$ run from 1 to $d$; the value $d$ is known as the local dimension. Thus each $x_j$ is mapped to a $d$-dimensional vector, which we require to have unit norm; this implies each $\Phi(x)$ also has unit norm.

The full feature map $\Phi(x)$ can be viewed as a vector in a $d^N$-dimensional space or as an order-$N$ tensor. The tensor diagram for $\Phi(x)$ is shown in Fig. 2. This type of tensor is said to be rank-1 since it is manifestly the product of $N$ order-1 tensors. In physics terms, $\Phi(x)$ has the same structure as a product state or unentangled wavefunction.

For a concrete example of this type of feature map, consider inputs which are grayscale images with $N$ pixels, where each pixel value ranges from 0.0 for white to 1.0 for black. If the grayscale pixel value of the $j$th pixel is $x_j \in [0, 1]$, a simple choice for the local feature map $\phi^{s_j}(x_j)$ is

$$\phi^{s_j}(x_j) = \left[ \cos \left( \frac{\pi}{2} x_j \right), \sin \left( \frac{\pi}{2} x_j \right) \right]$$ (3)

and is illustrated in Fig. 3. The full image is represented as a tensor product of these local vectors. From a physics perspective, $\phi^{s_j}$ is the normalized wavefunction of a single qubit where the “up” state corresponds to a white pixel, the “down” state to a black pixel, and superposition corresponds to a gray pixel.

While our choice of feature map $\Phi(x)$ was originally motivated from a physics perspective, in machine learning terms, the feature map Eq. (2) defines a kernel which is the product of $N$ local kernels, one for each component $x_j$ of the input data. Kernels of this type have been discussed previously [35], p. 193] and have been argued to be useful for data where no relationship is assumed between different components of the input vector prior to learning [36].

III. MULTIPLE LABEL CLASSIFICATION

In what follows we are interested in multi-class learning, for which we choose a “one-versus-all” strategy, which we take to mean generalizing the decision function Eq. (4) to a set of functions indexed by a label $\ell$

$$f^\ell(x) = W^\ell \cdot \Phi(x)$$ (4)

and classifying an input $x$ by choosing the label $\ell$ for which $|f^\ell(x)|$ is largest.

Since we apply the same feature map $\Phi$ to all input data, the only quantity that depends on the label $\ell$ is the weight vector $W^\ell$. Though one can view $W^\ell$ as a collection of vectors labeled by $\ell$, we will prefer to view $W^\ell$ as an order $N+1$ tensor where $\ell$ is a tensor index and $f^\ell(x)$ is a function mapping inputs to the space of labels. The tensor diagram for evaluating $f^\ell(x)$ for a particular input is depicted in Fig. 4.

IV. MPS APPROXIMATION

Because the weight tensor $W^\ell_{s_1 s_2 \cdots s_N}$ has $N_L \cdot d^N$ components, where $N_L$ is the number of labels, we need a way to regularize and optimize this tensor efficiently. The strategy we will use is to represent this high-order tensor as a tensor network, that is, as the contracted product of lower-order tensors.

A tensor network approximates the exponentially large set of components of a high-order tensor in terms of a much smaller set of parameters whose number grows
only polynomially in the size of the input space. Various
tensor network approximations impose different assump-
tions, or implicit priors, about the pattern of correlation
of the local indices when viewing the original tensor
as a distribution. For example, a MERA network can
explicitly model power-law decaying correlations while
as a distribution. For example, a MERA network can
approximating tensors with a one-dimensional pattern of
correlations, they can also be a powerful approach for
decomposing tensors with two-dimensional correlations
and highly efficient. Although MPS are best suited for
manipulating and optimizing them are well understood
in physics, here we propose a similar algorithm which
“sweeps” back and forth along an MPS, iteratively min-
imizing the cost function defining the classification task.

The cost function for which we found our best test
results (presented in Section VI) is the quadratic cost
\[ C = \frac{1}{2} \sum_{n=1}^{N_T} \sum_{\ell} (f^\ell(x_n) - \delta_{L_n})^2 \] 

where \( n \) runs over the \( N_T \) training inputs and \( L_n \) is the
known correct label for training input \( n \). The factor of
1/2 is for convenience.

Our strategy for reducing this cost function will be to
vary only two neighboring MPS tensors at a time within
the approximation Eq. (5). We could conceivably just
vary one at a time but as will become clear, varying two
tensors leads to a straightforward method for adaptively
changing the MPS bond dimension.

Say we want to improve the tensors at sites \( j \) and \( j + 1 \)
which share the \( \ell \)th bond. Assume we have moved the
label index \( \ell \) to the \( j \)th MPS tensor. First we combine the
MPS tensors \( A_{\ell j}^{\alpha} \) and \( A_{\ell j+1}^{\alpha} \) into a single “bond tensor
\( B_{\ell j}^{\alpha_1 \alpha_{j+1}} \) by contracting over the index \( \alpha_j \) as shown in
Fig. 3(a). Next we compute the derivative of the cost function
with respect to the bond tensor \( B^\ell \) in order to update
it using a gradient descent step. Because the rest of the
MPS tensors are kept fixed, let us show that to compute
the gradient it suffices to feed, or project, each input
\( x_n \) through the fixed “wings” of the MPS as shown on
the left-hand side of Fig. 3(b). Doing so produces the
projected, four-index version of the input \( \Phi_n \) shown on
the right-hand of Fig. 3(b). The current decision function
can be efficiently computed from this projected input \( \Phi_n \)
and the current bond tensor \( B^\ell \) as
\[ f^\ell(x_n) = \sum_{\alpha_{j-1} \alpha_{j+1}} \sum_{s_j \alpha_{j+1}} B_{\ell j}^{\alpha_{j-1} \alpha_{j+1}} (\Phi_n)_{s_j}^{\alpha_j} (\Phi_n)_{s_{j+1}}^{\alpha_{j+1}} \] 

or as illustrated in Fig. 3(c). Thus the leading-order up-
date to the tensor \( B^\ell \) can be computed as
\[ \Delta B^\ell \overset{\text{def}}{=} \frac{\partial C}{\partial B^\ell} \] 

\[ = \sum_{n=1}^{N_T} \sum_{\ell} (\delta_{L_n} - f^\ell(x_n)) \frac{\partial f^\ell(x_n)}{\partial B^\ell} \] 

\[ = \sum_{n=1}^{N_T} \sum_{\ell} (\delta_{L_n} - f^\ell(x_n)) \Phi_n \] 

Note that last expression above is a tensor with the same
index structure as \( B^\ell \) as shown in Fig. 3(d).
Assuming we have computed the gradient, we use it to compute a small update to $B^\ell$, replacing it with $B^\ell + \Delta B^\ell$ as shown in Fig. 7(a). Having obtained our improved $B^\ell$, we must decompose it back into separate MPS tensors to maintain efficiency and apply our algorithm to the next bond. Assume the next bond we want to optimize is the one to the right (bond $j+1$). Then we can compute a singular value decomposition (SVD) of $B^\ell$, treating it as a matrix with a collective row index $(\alpha_{j-1}, s_j)$ and collective column index $(\ell, \alpha_{j+1}, s_{j+1})$ as shown in Fig. 7(b). Computing the SVD this way restores the MPS form, but with the $\ell$ index moved to the tensor on site $j+1$. If the SVD of $B^\ell$ is given by

$$B^\ell_{\alpha_{j-1}\alpha_{j+1}} = \sum_{\alpha_j} U^\ell_{\alpha_j,\alpha_{j-1}} S_{\alpha_j,\alpha_{j+1}} V^\ell_{\alpha_{j+1},\alpha_j} \delta_{Ln} - f^\ell(x_n),$$

then to proceed to the next step we define the new MPS tensor at site $j$ to be $A'_{\alpha_j} = U_{\alpha_j}$, and the new tensor at site $j+1$ to be $A'_{\alpha_{j+1}} = S V^\ell_{\alpha_{j+1},\alpha_j}$, where a matrix multiplication over the suppressed $\alpha$ indices is implied. Crucially at this point, only the $m$ largest singular values in $S$ are kept and the rest are truncated (along with the corresponding columns of $U$ and $V^\ell$) in order to control the computational cost of the algorithm. Such a truncation is guaranteed to produce an optimal approximation of the tensor $B^\ell$; furthermore if all of the MPS tensors to the left and right of $B^\ell$ are formed from (possibly truncated) unitary matrices similar to the definition of $A'_{\alpha_j}$ above, then the optimality of the truncation of $B^\ell$ applies globally to the entire MPS as well. For further background reading on these technical aspects of MPS, see Refs. [25] and [30].

Finally, when proceeding to the next bond, it would be inefficient to fully project each training input over again into the configuration in Fig. 6(b). Instead it is only necessary to advance the projection by one site using the MPS tensor set from a unitary matrix after the SVD as shown in Fig. 7(c). This allows the cost of each local step of the algorithm to remain independent of the size of the input space, making the total algorithm scale only linearly with input space size.

The above algorithm highlights a key advantage of MPS and tensor networks relevant to machine learning applications. Following the SVD of the improved bond tensor $B^\ell$, the dimension of the new MPS bond can be chosen adaptively based on number of large singular values (defined by a threshold chosen in advance). Thus the MPS form of $W^\ell$ can be compressed as much as possible, and by different amounts on each bond, while still ensuring an optimal decision function.

The scaling of the above algorithm is $d^3 m^3 N_L N_T$, where recall $m$ is the MPS bond dimension; $N$ the number of input components; $N_L$ the number of labels; and $N_T$ the number of training inputs. In practice, the cost is dominated by the large number of training inputs $N_T$, so it would be very desirable to reduce this cost. One solution could be to use stochastic gradient descent, but while our experiments at blending this approach with the MPS sweeping algorithm often reached single-digit classification errors, we could not match the accuracy of the full gradient. Mixing stochastic gradient with MPS sweeping thus appears to be non-trivial but we believe it is a promising direction for further research.

VI. MNIST HANDWRITTEN DIGIT TEST

To test the tensor network approach on a realistic task, we used the MNIST data set, which consists of grayscale images of the digits zero through nine [40]. The calcu-
When selecting a model, our main control parameter is the dimension \( d \) of the local indices \( s_1 \) and \( s_2 \). For the case \( d = 2 \), the local feature map is chosen as in Eq. 3. For \( d > 2 \) we generalize \( \phi^{s_1}(x) \) to be a normalized \( d \)-component vector as described in Appendix B.

**A. Regularizing By Local Dimension \( d \)**

To understand how the flexibility of the model grows with increasing \( d \), consider the case where \( P_A \) and \( P_B \) are overlapping distributions. Specifically, we take each to be a multivariate Gaussian centered respectively in the lower-right and upper-left of the unit square, and to have different covariance matrices. In Fig. 9 we show the theoretically optimal decision boundary that best separates \( A \) points (crosses, red region) from \( B \) points (squares, blue region), defined by the condition \( P_A(x_1, x_2) = P_B(x_1, x_2) \). To make a training set, we sample 100 points from each of the two distributions.

Next, we optimize the toy model for our overlapping training set for various \( d \). The decision boundary learned by the \( d = 2 \) model in Fig. 10(a) shows good agreement with the optimal one in Fig. 9. Because the two sets are non-separable and this model is apparently well regularized, some of the training points are necessarily misclassified—these points are colored white in the figure.

The \( d = 3 \) decision boundary shown in Fig. 10 begins to show evidence of overfitting. The boundary is more complicated than for \( d = 2 \) and further from the optimal boundary. Finally, for a much larger local dimension \( d = 6 \) there is extreme overfitting. The decision boundary is highly irregular and is more reflective of the specific sampled points than the underlying distribution. Some of the overfitting behavior reveals the structure of the model; at the bottom and top of Fig. 10(c) there are

![FIG. 9. Training points sampled from multivariate Gaussian distributions \( P_A(x_1, x_2) \) [crosses] and \( P_B(x_1, x_2) \) [squares]. The curve separating the red \( A \) region from the blue \( B \) region is the theoretically optimal decision boundary.](image-url)
FIG. 10. Toy models learned from the overlapping data set. The results shown are for local dimension (a) \( d = 2 \), (b) \( d = 3 \), and (c) \( d = 6 \). Background colors show how every spatial point would be classified. Misclassified data points are colored white.

lobes of one color protruding into the other. These likely indicate that the finite local dimension still somewhat regularizes the model; otherwise it would be able to overfit even more drastically by just surrounding each point with a small patch of its correct color.

B. Non-Linear Decision Boundary

To test the ability of our proposed class of models to learn highly non-linear decision boundaries, consider the spiral shaped boundary in Fig. 11(a). Here we take \( P_A \) and \( P_B \) to be non-overlapping with \( P_A \) uniform on the red region and \( P_B \) uniform on the blue region.

In Fig. 11(b) we show the result of training a model with local dimension \( d = 10 \) on 500 sampled points, 250 for each region (crosses for region A, squares for region B). The learned model is able to classify every training point correctly, though with some overfitting apparent near regions with too many or too few sampled points.

VIII. INTERPRETING TENSOR NETWORK MODELS

A natural question is which set of functions of the form \( f^\ell(x) = W^\ell \cdot \Phi(x) \) can be realized when using a tensor-product feature map \( \Phi(x) \) of the form Eq. (2) and a tensor-network decomposition of \( W^\ell \). As we will argue, the possible set of functions is quite general, but taking the tensor network structure into account provides additional insights, such as determining which features the model actually uses to perform classification.

A. Representational Power

To simplify the question of which decision functions can be realized for a tensor-product feature map of the form Eq. (2), let us fix \( \ell \) to a single label and omit it from the notation. We will also consider \( W \) to be a completely general order-\( N \) tensor with no tensor network constraint. Then \( f(x) \) is a function of the form

\[
  f(x) = \sum_{\{s\}} W_{s_1,s_2,\ldots,s_N} \phi^{s_1}(x_1) \otimes \phi^{s_2}(x_2) \otimes \cdots \phi^{s_N}(x_N).
\]  

(13)

If the functions \( \{\phi^s(x)\} \), \( s = 1, 2, \ldots, d \) form a basis for a Hilbert space of functions over \( x \in [0,1] \), then the tensor product basis

\[
  \phi^{s_1}(x_1) \otimes \phi^{s_2}(x_2) \otimes \cdots \phi^{s_N}(x_N)
\]

(14)
forms a basis for a Hilbert space of functions over \( x \in [0,1]^N \). Moreover, if the basis \( \{\phi^s(x)\} \) is complete,
then the tensor product basis is also complete and \( f(x) \) can be any square integrable function.

Next, consider the effect of restricting the local dimension to \( d = 2 \) as in the local feature map of Eq. 3 which was used to classify grayscale images in our MNIST benchmark in Section [IV]. Recall that for this choice of \( \phi(x), \)

\[
\phi(0) = [1, 0] \quad \text{and} \quad \phi(1) = [0, 1].
\]

Thus if \( \tilde{x} \) is a black and white image with pixel values of only \( x_j = \{0, 1\} \), then \( f(\tilde{x}) \) is equal to a single component \( W_{s_1 s_2 \ldots s_N} \) of the weight tensor. Because each of these components is an independent parameter (assuming no further approximation of \( W \), \( f(x) \) is a highly non-linear, in fact arbitrary, function when restricted to these black and white images.

Returning to the case of grayscale images \( x \) with pixels \( x_j \in \{0, 1\} \), \( f(x) \) cannot be an arbitrary function over this larger space of images for finite \( d \). For example, if one considers the \( d = 2 \) feature map Eq. 3, then when considering the dependence of \( f(x) \) on only a single pixel \( x_j \) (all other pixels being held fixed), it has the functional form \( a \cos(\pi/2 x_j) + b \sin(\pi/2 x_j) \) where \( a \) and \( b \) are constants determined by the (fixed) values of the other pixels.

\[ x = [1, 0] \quad \text{and} \quad \phi(1) = [0, 1]. \]

Thus if \( \tilde{x} \) is a black and white image with pixel values of only \( x_j = \{0, 1\} \), then \( f(\tilde{x}) \) is equal to a single component \( W_{s_1 s_2 \ldots s_N} \) of the weight tensor. Because each of these components is an independent parameter (assuming no further approximation of \( W \), \( f(x) \) is a highly non-linear, in fact arbitrary, function when restricted to these black and white images.

Returning to the case of grayscale images \( x \) with pixels \( x_j \in \{0, 1\} \), \( f(x) \) cannot be an arbitrary function over this larger space of images for finite \( d \). For example, if one considers the \( d = 2 \) feature map Eq. 3, then when considering the dependence of \( f(x) \) on only a single pixel \( x_j \) (all other pixels being held fixed), it has the functional form \( a \cos(\pi/2 x_j) + b \sin(\pi/2 x_j) \) where \( a \) and \( b \) are constants determined by the (fixed) values of the other pixels.

\[ x = [1, 0] \quad \text{and} \quad \phi(1) = [0, 1]. \]

Thus if \( \tilde{x} \) is a black and white image with pixel values of only \( x_j = \{0, 1\} \), then \( f(\tilde{x}) \) is equal to a single component \( W_{s_1 s_2 \ldots s_N} \) of the weight tensor. Because each of these components is an independent parameter (assuming no further approximation of \( W \), \( f(x) \) is a highly non-linear, in fact arbitrary, function when restricted to these black and white images.

Returning to the case of grayscale images \( x \) with pixels \( x_j \in \{0, 1\} \), \( f(x) \) cannot be an arbitrary function over this larger space of images for finite \( d \). For example, if one considers the \( d = 2 \) feature map Eq. 3, then when considering the dependence of \( f(x) \) on only a single pixel \( x_j \) (all other pixels being held fixed), it has the functional form \( a \cos(\pi/2 x_j) + b \sin(\pi/2 x_j) \) where \( a \) and \( b \) are constants determined by the (fixed) values of the other pixels.

\[ x = [1, 0] \quad \text{and} \quad \phi(1) = [0, 1]. \]

Thus if \( \tilde{x} \) is a black and white image with pixel values of only \( x_j = \{0, 1\} \), then \( f(\tilde{x}) \) is equal to a single component \( W_{s_1 s_2 \ldots s_N} \) of the weight tensor. Because each of these components is an independent parameter (assuming no further approximation of \( W \), \( f(x) \) is a highly non-linear, in fact arbitrary, function when restricted to these black and white images.

Returning to the case of grayscale images \( x \) with pixels \( x_j \in \{0, 1\} \), \( f(x) \) cannot be an arbitrary function over this larger space of images for finite \( d \). For example, if one considers the \( d = 2 \) feature map Eq. 3, then when considering the dependence of \( f(x) \) on only a single pixel \( x_j \) (all other pixels being held fixed), it has the functional form \( a \cos(\pi/2 x_j) + b \sin(\pi/2 x_j) \) where \( a \) and \( b \) are constants determined by the (fixed) values of the other pixels.

\[ x = [1, 0] \quad \text{and} \quad \phi(1) = [0, 1]. \]

Thus if \( \tilde{x} \) is a black and white image with pixel values of only \( x_j = \{0, 1\} \), then \( f(\tilde{x}) \) is equal to a single component \( W_{s_1 s_2 \ldots s_N} \) of the weight tensor. Because each of these components is an independent parameter (assuming no further approximation of \( W \), \( f(x) \) is a highly non-linear, in fact arbitrary, function when restricted to these black and white images.

Returning to the case of grayscale images \( x \) with pixels \( x_j \in \{0, 1\} \), \( f(x) \) cannot be an arbitrary function over this larger space of images for finite \( d \). For example, if one considers the \( d = 2 \) feature map Eq. 3, then when considering the dependence of \( f(x) \) on only a single pixel \( x_j \) (all other pixels being held fixed), it has the functional form \( a \cos(\pi/2 x_j) + b \sin(\pi/2 x_j) \) where \( a \) and \( b \) are constants determined by the (fixed) values of the other pixels.

\[ x = [1, 0] \quad \text{and} \quad \phi(1) = [0, 1]. \]
tion over the set of inputs \( x \) belonging to class \( \ell \). A major motivation for this interpretation would be that many insights from physics could be applied to machine learned models. For example, tensor networks in the same family as MPS, when viewed as defining a probability distribution, can be used to efficiently perform perfect sampling of the distribution they represent [43].

Let us investigate the properties of \( W^\ell \) and \( \Phi(x) \) required for a consistent interpretation of \(|f^\ell(x)|^2\) as a probability distribution. For \(|f^\ell(x)|^2\) to behave like a probability distribution for a broad class of models, we require for some integration measure \( d\mu(x) \) that the distribution is normalized as

\[
\sum_{\ell} \int_x |f^\ell(x)|^2 d\mu(x) = 1 \tag{18}
\]

no matter what weight vector \( W^\ell \) the model has, as long as the weights are normalized as

\[
\sum_{\ell} \sum_{s_1,s_2,...,s_N} W^\ell_{s_1s_2...s_N} W^{\bar{s}_1\bar{s}_2...\bar{s}_N} = 1. \tag{19}
\]

This condition is automatically satisfied for tensor-product feature maps \( \Phi(x) \) of the form Eq. (2) if the constituent local maps \( \phi^s(x) \) have the property

\[
\int_x \bar{\phi}^s(x) \phi^s(x) d\mu(x) = \delta_{ss'}, \tag{20}
\]

that is, if the components of \( \phi^s \) are orthonormal functions with respect to the measure \( d\mu(x) \). We continue to require the local vectors remain normalized as

\[
\sum_s |\phi^s(x)|^2 = 1 \tag{21}
\]

for all \( x \in [0,1] \).

Unfortunately neither the local feature map Eq. (4) nor its generalizations in Appendix [3] meet the first criterion Eq. (20). A different choice that satisfies both the orthogonality condition Eq. (20) and normalization condition Eq. (21) could be

\[
\phi(x) = [\cos(\pi x), \sin(\pi x)]. \tag{22}
\]

However, this map is not suitable for inputs like grayscale pixels since it is anti-periodic over the interval \( x \in [0,1] \) and would lead to a periodic probability distribution. An example of an orthogonal, normalized map which is not periodic on \( x \in [0,1] \) is

\[
\phi(x) = [e^{i(3\pi/2)x} \cos\left(\frac{\pi}{2} x\right), e^{-i(3\pi/2)x} \sin\left(\frac{\pi}{2} x\right)]. \tag{23}
\]

This local feature map meets the criteria Eqs. (20) and (21) if the integration measure chosen to be \( d\mu(x) = \frac{1}{2} dx \).

As a basic consistency check of the above generative interpretation, we performed an experiment on our toy model of Section [VII] using the local feature map

\[
\sum_{\ell} \int_x |f^\ell(x)|^2 d\mu(x) = 1
\]

FIG. 13. Average KL divergence between the learned model and original model used to generate data for a two-dimensional toy system as a function of number of sampled training points \( N_s \). The solid curve is a fit of the form \( \sigma/\sqrt{N_s} \).

Eq. (23). Recall that our toy data can have two possible labels \( A \) and \( B \). To test the generative interpretation, we first generated a single, random “hidden” weight tensor \( W \). From this weight tensor we sampled \( N_s \) data points in a two step process:

1. Sample a label \( \ell = A \) or \( \ell = B \) according to the probabilities \( P_A = \int_x |f^A(x)|^2 = \sum_{s_1,s_2} |W^A_{s_1s_2}|^2 \) and \( P_B = 1 - P_A \).

2. Sample a data point \( x = (x_1, x_2) \) according to the distribution \( p(x|\ell) = |f^\ell(x)|^2/P_\ell \) for the chosen \( \ell \).

For each collection of sampled points we then trained a toy model with weight tensor \( \bar{W} \) using the log-likelihood cost function

\[
C = -\sum_{n=1}^{N_s} \log |f^{L_n}(x_n)|^2 \tag{24}
\]

where recall \( L_n \) is the known correct label for training point \( n \).

We repeated this procedure multiple times for various sample sizes \( N_s \), each time computing the Kullback-Liebler divergence of the learned versus exact distribution

\[
D_{KL} = \sum_{\ell} \int_x p(\ell, x) \log \left( \frac{p(\ell, x)}{\bar{p}(\ell, x)} \right) \tag{25}
\]

where \( p(\ell, x) = |f^\ell(x)|^2 = |W^\ell : \Phi(x)|^2 \) and \( \bar{p}(\ell, x) \) has similar definition in terms of \( \bar{W} \). The resulting average \( D_{KL} \) as a function of number of sampled training points \( N_s \) is shown in Fig. 13 along with a fit of the form \( \sigma/\sqrt{N_s} \) where \( \sigma \) is a fitting parameter. The results indicate that given enough training data, the learning process can eventually recapture the original probabilistic model that generated the data.
IX. DISCUSSION

We have introduced a framework for applying quantum-inspired tensor networks to multi-class supervised learning tasks. While using an MPS ansatz for the model parameters worked well even for the two-dimensional data in our MNIST experiment, other tensor networks such as PEPS, which are explicitly designed for two-dimensional systems, may be more suitable and offer superior performance. Much work remains to determine the best tensor network for a given domain.

There is also much room to improve the gradient descent algorithm, adopting it to incorporate standard tricks such as mini-batches, momentum, or adaptive learning rates. It would be especially interesting to investigate unsupervised techniques for initializing the tensor network. Additionally, while the tensor network parameterization of a model clearly regularizes it in the sense of reducing the number of parameters, it would be helpful to understand the consequences of this regularization for specific learning tasks. It could also be fruitful to include standard regularizations of the parameters of the tensor network, such as weight decay or $L_1$ penalties. We were surprised to find good generalization without using explicit parameter regularization.

There exist tensor network coarse-graining approaches for purely classical systems [44, 45], which could possibly be used instead of our approach. However, mapping the data into an extremely high-dimensional Hilbert space is likely advantageous for producing models sensitive to high-order correlations among features. We believe there is great promise in investigating the power of quantum-inspired tensor networks for many other machine learning tasks.

Acknowledgments

We would like to acknowledge helpful discussions with Juan Carrasquilla, Josh Combes, Glen Evenbly, Bohdan Kulchytskyy, Li Li, Roger Melko, Pankaj Mehta, U. N. Niranjan, Giacomo Torlai, and Steven R. White.

Appendix A: Graphical Notation for Tensor Networks

Though matrix product states (MPS) have a relatively simple structure, more powerful tensor networks, such as PEPS and MERA, have such complex structure that traditional tensor notation becomes unwieldy. For these networks, and even for MPS, it is helpful to use a graphical notation. For a more complete review of this notation and its uses in various tensor networks see Ref. [39].

The basic graphical notation for a tensor is to represent it as a closed shape. Typically this shape is a circle, though other shapes can be used to distinguish types of tensors (there is no standard convention for the choice of shapes). Each index of the tensor is represented by a line emanating from it; an order-N tensor has N such lines. Figure 14 shows examples of diagrams for tensors of order one, two, and three.

To indicate that a certain pair of tensor indices are contracted, they are joined together by a line. For example, Fig. 15(a) shows the contraction of an order-1 tensor with the an order-2 tensor; this is the usual matrix-vector multiplication. Figure 15(b) shows a more general contraction of an order-4 tensor with an order-3 tensor.

Graphical tensor notation offers many advantages over traditional notation. In graphical form, indices do not usually require names or labels since they can be distinguished by their location in the diagram. Operations such as the outer product, tensor trace, and tensor contraction can be expressed without additional notation; for example, the outer product is just the placement of one tensor next to another. For a network of contracted tensors, the order of the final resulting tensor can be read off by simply counting the number of unpaired lines left over. For example, a complicated set of tensor contractions can be recognized as giving a scalar result if no index lines remain unpaired.

Appendix B: Higher-Dimensional Local Feature Map

As discussed in Section II, our strategy for using tensor networks to classify input data begins by mapping each component $x_j$ of the input data vector $\mathbf{x}$ to a $d$-component vector $\phi^{s_j}(x_j)$, $s_j = 1, 2, \ldots, d$. We always choose $\phi^{s_j}(x_j)$ to be a unit vector in order to apply
physics techniques which typically assume normalized wavefunctions.

For the case of \( d = 2 \) we have used the mapping
\[
\phi^s_j(x_j) = \left[ \cos \left( \frac{\pi}{2} x_j \right), \sin \left( \frac{\pi}{2} x_j \right) \right]. \tag{B1}
\]
A straightforward way to generalize this mapping to larger \( d \) is as follows. Define \( \theta_j = \frac{\pi}{2} x_j \). Because \((\cos^2(\theta_j) + \sin^2(\theta_j)) = 1\), then also
\[
\left( \cos^2(\theta_j) + \sin^2(\theta_j) \right)^{d-1} = 1 . \tag{B2}
\]
Expand the above identity using the binomial coefficients \( \binom{n}{k} = n!/(k!(n-k)!) \).
\[
\left( \cos^2(\theta_j) + \sin^2(\theta_j) \right)^{d-1} = 1 = \sum_{p=0}^{d-1} \binom{d-1}{p} \left( \cos \theta_j \right)^{2(d-1-p)} \left( \sin \theta_j \right)^{2p} . \tag{B3}
\]
This motivates defining \( \phi^s_j(x_j) \) to be
\[
\phi^s_j(x_j) = \sqrt{\left( \frac{d-1}{s_j - 1} \right)} \left( \cos \left( \frac{\pi}{2} x_j \right) \right)^{d-s_j} \left( \sin \left( \frac{\pi}{2} x_j \right) \right)^{s_j - 1} \tag{B4}
\]
where recall that \( s_j \) runs from 1 to \( d \). The above definition reduces to the \( d = 2 \) case Eq. [B1] and guarantees that \( \sum_j |\phi^s_j|^2 = 1 \) for larger \( d \). (These functions are actually a special case of what are known as spin coherent states.)

Using the above mapping \( \phi^s_j(x_j) \) for larger \( d \) allows the product \( W^T \cdot \Phi(x) \) to realize a richer class of functions. One reason is that a larger local dimension allows the weight tensor to have many more components. Also, for larger \( d \) the components of \( \phi^s_j(x_j) \) contain ever higher frequency terms of the form \( \cos \left( \frac{\pi}{2} x_j \right) \), \( \cos \left( \frac{3\pi}{2} x_j \right) \), \ldots, \( \cos \left( \frac{(d-1)\pi}{2} x_j \right) \) and similar terms replacing \( \cos \) with \( \sin \). The net result is that the decision functions realizable for larger \( d \) are composed from a more complete basis of functions and can respond to smaller variations in \( x \).

[1] J. J. Hopfield, “Neural networks and physical systems with emergent collective computational abilities,” PNAS 79, 2554–2558 (1982).
[2] Daniel J. Amit, Hanoch Gutfreund, and H. Sompolinsky, “Spin-glass models of neural networks,” Phys. Rev. A 32, 1007–1018 (1985).
[3] Bernard Derrida, Elizabeth Gardner, and Anne Zippelius, “An exactly solvable asymmetric neural network model,” EPL (Europhysics Letters) 4, 167 (1987).
[4] Daniel J. Amit, Hanoch Gutfreund, and H. Sompolinsky, “Information storage in neural networks with low levels of activity,” Phys. Rev. A 35, 2293–2303 (1987).
[5] HS Seung, Haim Sompolinsky, and N. Tishby, “Statistical mechanics of learning from examples,” Physical Review A 45, 6056 (1992).
[6] Andreas Engel and Christian Van den Broeck, Statistical mechanics of learning (Cambridge University Press, 2001).
[7] Dörthe Malzahn and Manfred Opper, “A statistical physics approach for the analysis of machine learning algorithms on real data,” Journal of Statistical Mechanics: Theory and Experiment 2005, P11001 (2005).
[8] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh, “A fast learning algorithm for deep belief nets,” Neural Computation 18, 1527–1554 (2006).
[9] Marc Mezzard and Andrea Montanari, Information, physics, and computation (Oxford University Press, 2009).
[10] Pankaj Mehta and David Schwab, “An exact mapping between the variational renormalization group and deep learning,” arxiv:1410.3831 (2014).
[11] Christopher C. Fischer, Kevin J. Tibbetts, Dane Morgan, and Gerbrand Ceder, “Predicting crystal structure by merging data mining with quantum mechanics,” Nat. Mater. 5, 641–646 (2006).
[12] Geoffroy Hautier, Christopher C. Fischer, Anubhav Jain, Tim Mueller, and Gerbrand Ceder, “Finding nature’s missing ternary oxide compounds using machine learning and density functional theory,” Chemistry of Materials 22, 3762–3767 (2010). [http://dx.doi.org/10.1021/cm100795d].
[13] Matthias Rupp, Alexandre Tkatchenko, Klaus-Robert Müller, and O. Anatole von Lilienfeld, “Fast and accurate modeling of molecular atomization energies with machine learning,” Phys. Rev. Lett. 108, 058301 (2012).
[14] Yousef Saad, Da Gao, Thanh Ngo, Scotty Bobbitt, James R. Chelikowsky, and Wanda Andreoni, “Data mining for materials: Computational experiments with AB compounds,” Phys. Rev. B 85, 104104 (2012).
[15] John C. Snyder, Matthias Rupp, Katja Hansen, Klaus-Robert Müller, and Kieron Burke, “Finding density functionals with machine learning,” Phys. Rev. Lett. 108, 253002 (2012).
[16] Ghanshyam Pilania, Chenchun Wang, Xun Jiang, Sangutevar Rajasekaran, and Ramamurthy Ramprasad, “Accelerating materials property predictions using machine learning,” Scientific Reports 3, 2810 EP – (2013).
[17] Louis-François Arsenault, Alejandro Lopez-Bezanilla, O. Anatole von Lilienfeld, and Andrew J. Millis, “Machine learning for many-body physics: The case of the Anderson impurity model,” Phys. Rev. B 90, 155136 (2014).
[18] Juan Carrasquilla and Roger G. Melko, “Machine learning phases of matter,” arxiv:1605.01735 (2016).
[19] Animashree Anandkumar, Rong Ge, Daniel Hsu, Sham M. Kakade, and Matus Telgarsky, “Tensor decompositions for learning latent variable models,” Journal of Machine Learning Research 15, 2773–2832 (2014).
[20] Animashree Anandkumar, Rong Ge, Daniel Hsu, and Sham M. Kakade, “A tensor approach to learning mixed membership community models,” J. Mach. Learn. Res. 15, 2239–2312 (2014).
[21] Ánh Huy Phan and Andrzej Cichocki, “Tensor decompositions for feature extraction and classification of high...
dimensional datasets,” Nonlinear theory and its applications, IEICE 1, 37–68 (2010).

[22] J.A. Bengua, H.N. Phien, and H.D. Tuan, “Optimal feature extraction and classification of tensors via matrix product state decomposition,” in 2015 IEEE Intl. Congress on Big Data (BigData Congress) (2015) pp. 669–672.

[23] Alexander Novikov, Dmitry Podoprikhin, Anton Osokin, and Dmitry Vetrov, “Tensorizing neural networks,” arxiv:1509.06569 (2015).

[24] Jacob C. Bridgeman and Christopher T. Chubb, “Handwaving and interpretive dance: An introductory course on tensor networks,” arxiv:1603.03039 (2016).

[25] U. Schollwöck, “The density-matrix renormalization group in the age of matrix product states,” Annals of Physics 326, 96–192 (2011).

[26] G. Evenbly and G. Vidal, “Tensor network states and geometry,” Journal of Statistical Physics 145, 891–918 (2011).

[27] K. R. Muller, S. Mika, G. Ratsch, K. Tsuda, and B. Scholkopf, “An introduction to kernel-based learning algorithms,” IEEE Transactions on Neural Networks 12, 181–201 (2001).

[28] Stellan Östlund and Stefan Rommer, “Thermodynamic limit of density matrix renormalization,” Phys. Rev. Lett. 75, 3537–3540 (1995).

[29] I. Oseledets, “Tensor-train decomposition,” SIAM Journal on Scientific Computing 33, 2295–2317 (2011).

[30] F. Verstraete and J. I. Cirac, “Density matrix renormalization group and periodic boundary conditions: A quantum information perspective,” Phys. Rev. Lett. 93, 227205 (2004).

[31] Andrzej Cichocki, “Tensor networks for big data analytics and large-scale optimization problems,” arxiv:1407.3124 (2014).

[32] Efi Efrati, Zhe Wang, Amy Kolan, and Leo P Kadanoff, “Real-space renormalization in statistical mechanics,” Reviews of Modern Physics 86, 647 (2014).

[33] Steven R. White, “Density matrix formulation for quantum renormalization groups,” Phys. Rev. Lett. 69, 2863–2866 (1992).

[34] Sebastian Holtz, Thorsten Rohwedder, and Reinhold Schneider, “The alternating linear scheme for tensor optimization in the tensor train format,” SIAM Journal on Scientific Computing 34, A683–A713 (2012).

[35] Vladimir Vapnik, The Nature of Statistical Learning Theory (Springer-Verlag New York, 2000).

[36] W. Waegeman, T. Pahikkala, A. Airola, T. Salakoski, M. Stock, and B. De Baets, “A kernel-based framework for learning graded relations from data,” Fuzzy Systems, IEEE Transactions on 20, 1090–1101 (2012).

[37] F. Verstraete, D. Porras, and J. I. Cirac, “Density matrix renormalization group and periodic boundary conditions: A quantum information perspective,” Phys. Rev. Lett. 93, 227205 (2004).

[38] G. Vidal, “Entanglement renormalization,” Phys. Rev. Lett. 99, 220405 (2007).

[39] G. Evenbly and G. Vidal, “Entanglement renormalization,” Phys. Rev. Lett. 99, 220405 (2007).

[40] E. M. Stoudenmire and Steven R. White, “Studying two-dimensional systems with the density matrix renormalization group,” Annual Review of Condensed Matter Physics 3, 111–128 (2012).

[41] Andrzej Cichocki, “Tensor networks for big data analytics and large-scale optimization problems,” arxiv:1407.3124 (2014).

[42] Christopher J.C. Burges Yann LeCun, Corinna Cortes, “MNIST handwritten digit database,” http://yann.lecun.com/exdb/mnist/.

[43] Tensor Library (version 2.0.7) http://itensor.org.

[44] Michael Nielsen, Neural Networks and Deep Learning (Determination Press, 2015).

[45] Andrew J. Ferris and Guifre Vidal, “Perfect sampling with unitary tensor networks,” Phys. Rev. B 85, 165146 (2012).