Supervised Learning with Tensor Networks

Richard Fambon

EISTI Pau

April 9, 2020

Original paper by E. M. Stoudenmire and David J. Schwab

Outline

Introduction

Why would we want to use Tensor Networks ?

- Training the model scales only linearly in the training set size
- It makes the training adaptive
- Additional type of regularization that could have interesting consequences for generalization

Here we will only use the MPS (Matrix Product State) tensor network :

Figure 1: MPS decomposition, also known as a tensor train. (Lines represent tensor indices and connecting two lines implies summation

Input Data Encoding

Tensor Networks in physics

Tensor networks in physics are typically used in a context where combining N independent systems corresponds to taking a tensor product of a vector describing each system

To apply similar tensor networks to machine learning, we choose a feature map of the form :

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

j = 1,2,...,N
s_j run from 1 to d
d is the local dimension (hyper-parameter) defining the classification model

Input Data Encoding

Here we will only consider the case of homogeneous inputs with the same local map applied to each x_j

Therefor $\Phi(x)$ can be viewed as a vector in a d^N-dimensional space or as an order-N tensor :

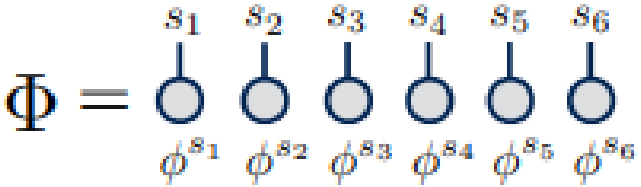

Figure 2: Input data is mapped to a normalized order N tensor with a rank-1 product structure

Classification Model

To classify data with pre-assigned hidden labels :

- "One-versus-all" strategy
- Optimizing a set of functions indexed by a label l
- Classifying an input x by choosing the label l for which $|f^l(x)|$ is largest.

Using :

$$f^\ell(\mathbf{x}) = W^\ell \cdot \Phi(\mathbf{x})$$

Classification Model

We can view W^l as an order $N+1$ tensor where l is a tensor index and $f^l(x)$ is a function mapping inputs to the space of labels :

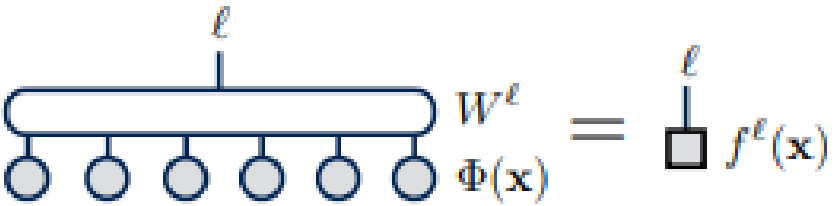

Figure 3: The overlap of the weight tensor W^l with a specific input vector (x) defines the decision function $f^l(x)$

Classification Model

Therefor we can represent W^l as an MPS :

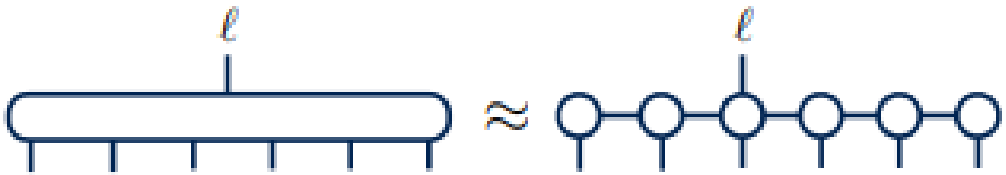

Figure 4: Approximation of the weight tensor W^l by a matrix product state. The label index l is placed arbitrarily on one of the N tensors but can be moved to other locations

W^l has the form :

$$W^{\ell}_{s_1 s_2 \cdots s_N} = \sum_{\{\alpha\}} A^{\alpha_1}_{s_1} A^{\alpha_1 \alpha_2}_{s_2} \cdots A^{\ell;\alpha_j \alpha_{j+1}}_{s_j} \cdots A^{\alpha_{N-1}}_{s_N}$$

"Sweeping" Optimization Algorithm

We want to optimize the quadratic cost :

$$C = \frac{1}{2} \sum_{n=1}^{N_T} \sum_{\ell} (f^{\ell}(\mathbf{x}_n) - y_n^{\ell})^2$$

n runs over the N_T training inputs
y_n^l is the vector of desired outputs for input n

- One-hot encoding :
  If the correct label of x_n is L_n, then $y_n^{l_n} = 1$ and y_n^l for all other labels l

Tensor Improving

For exemple, we want to improve tensors at sites j and $j+1$

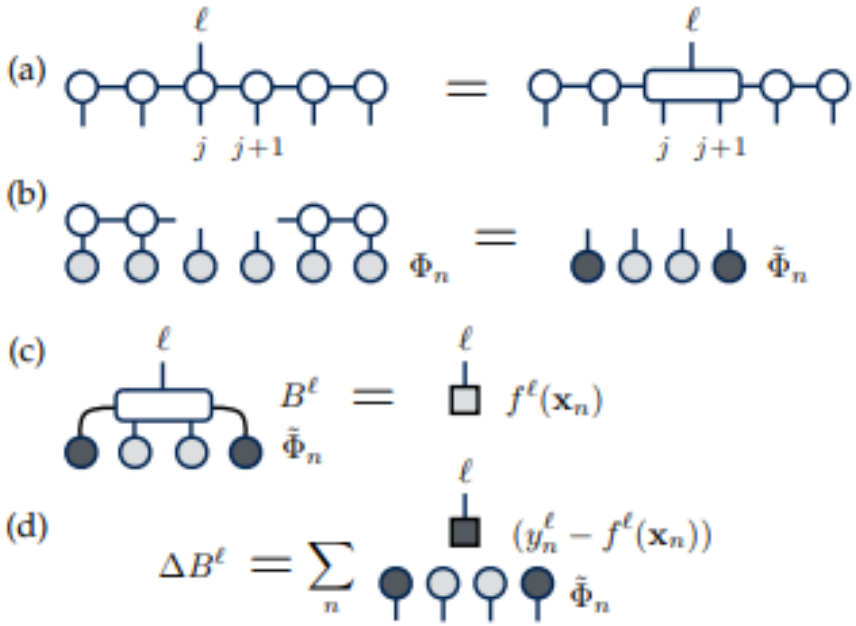

Figure 5: (a) forming the bond tensor;
(b) projecting a training input into the "MPS basis" at bond j;
(c) computing the decision function in terms of a projected input;
(d) the gradient correction to B^l

Tensor Improving

Having obtained our improved B‘ , we must decompose it back into separate MPS tensors to maintain efficiency and apply our algorithm to the next bond using SVD (Singular Value Decomposition):

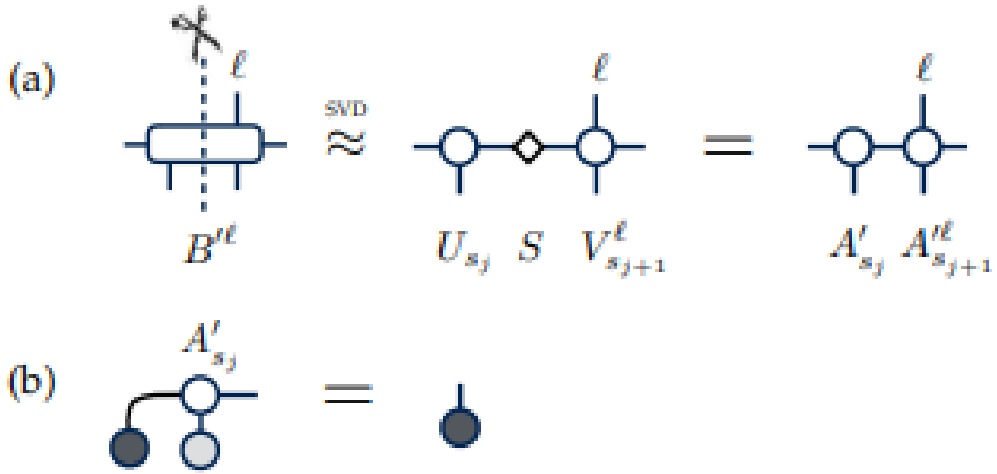

Figure 6: (a) Restoration of MPS form
(b) advancing a projected training input before optimizing the tensors at the next bond. In diagram (a), if the label index l was on the site j tensor before forming B^l , then the operation shown moves the label to site j + 1.

Tensor Improving

The current decision function can be efficiently computed from this projected input $\Phi_n$and the current bond tensor B^l as :

$$f^{\ell}(\mathbf{x}_n) = \sum_{\alpha_{j-1}\alpha_{j+1}} \sum_{s_j s_{j+1}} B^{\alpha_{j-1}\ell\alpha_{j+1}}_{s_j s_{j+1}} (\tilde{\Phi}_n)^{s_j s_{j+1}}_{\alpha_{j-1}\ell\alpha_{j+1}}$$

And the gradient update to the tensor B^l can be computed as :

$$\Delta B^{\ell} = -\frac{\partial C}{\partial B^{\ell}} = \sum_{n=1}^{N_T} (y_n^{\ell} - f^{\ell}(\mathbf{x}_n))\tilde{\Phi}_n$$

And the SVD of B^l is given by :

$$B^{\alpha_{j-1}\ell\alpha_{j+1}}_{s_j s_{j+1}} = \sum_{\alpha'_j \alpha_j} U^{\alpha_{j-1}}_{s_j \alpha'_j} S^{\alpha'_j}{}_{\alpha_j} V^{\alpha_j \ell \alpha_{j+1}}_{s_{j+1}}$$

Results and Discussion

The scaling of the above algorithm is $d^3 m^3 N N_L N_T$ with :
m the recall which is the typical MPS bond dimension
N the number of components of input vectors x
N_L the number of labels
N_T the size of the training data set

The algorithm scales linearly in the training set size, when typical kernel-trick methods typically scale at least as N_T^2

Results and Discussion

When using the algorithm on a slightly modified MNIST Handwritten Digit Test, it quickly converges after 5 or less "sweeps"
The test error rates also decreases rapidly with the maximum MPS bond dimension m :

- $m = 10 => 5\%$
- $m = 20 => 2\%$
- $m = 120 => 0.97\%$

Important note : MPS bond dimensions in physics applications can reach many hundreds or even thousands

Results and Discussion

The algorithm also induces an implicit feature selection during the SVD and we can contract the MPS at will without losing too much accuracy:

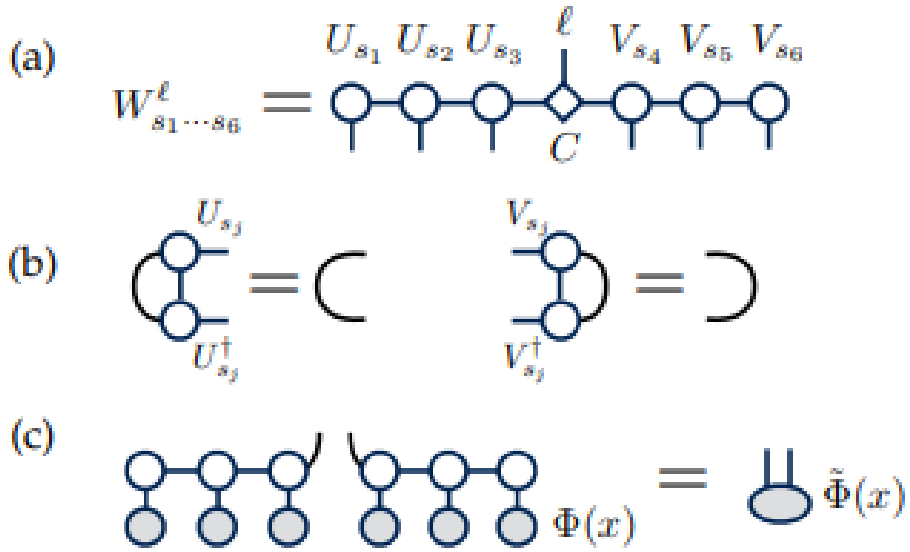

Figure 7: (a) Decomposition of W^l as an MPS with a central tensor and orthogonal site tensors.
(b) Orthogonality conditions for U and V type site tensors.
(c) Transformation defining a reduced feature map $\Phi(x)$

Results and Discussion

There is still room for improvement :

- Try other tensor networks (PEPS, MERA, ...)
- Mini-batches, momentum or adaptative learning rates
- Include standard regularizations of parameters (weight decay or L_1 penalties)

End of the presentation

Thank you for your attention !

Any questions ?

Paper link : http://papers.nips.cc/paper/6211-supervised-learning-with-tensor-networks.pdf