Learning Relevant Features of Data with Multi-Scale Tensor Networks

Presenters: Tayssir Doghri, Aayushi Kulshrestha, Tapopriya Majumdar

E. Miles Stoudenmire

March 19, 2019

Paper Outline

- Introduction
- Applying decomposition technique to matrices, which would then be extended to high order tensors
- Unsupervised Coarse Graining
- Experiment on classification task
- Mixing prior
- Conclusion

High Level Idea

- Provides a method to compress data originally in high dimensional space
- Preserves data properties at large scale while normalizing over smallest length scales
- Main idea comes from physics
- Eg. Looking at temperature of the entire system to understand state instead of dynamics about each particle
- Significantly reduces the size of feature space

Research Significance

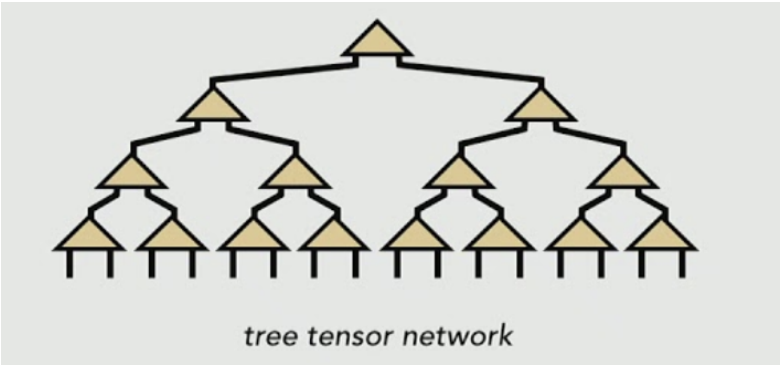
- Computational Efficiency
- Building block for many machine learning tasks
- The algorithm proposed is unsupervised
- Can be applied to very large datasets with a large set of features

Main Contributions

- Uses tensor networks to produce a hierarchical representation of data using low order tensors
- Unsupervised learning based on statistical properties of data
- Only a single topmost layer of tensor needs to be optimized based on task
- Can be used with prior estimates of weights to make learning faster

Key Points

- Compressed space is represented using a layered tree tensor network
- The algorithm scales linearly with both the dimension of the input and training set size
- Uses kernel learning
- The tree tensor network obtained is generalizable to various tasks



Peek into Matrices

- Consider a model $f(x) = W \cdot \Phi(x)$, where $\Phi(x)$ is the kernel space mapping of the training data.
- The optimal weights belong to the span of the training data within feature space.
- Using *Representer Theorem*, W:

$$W = \sum_{j=1}^{N_T} \alpha_j \Phi^T(x_j) \tag{1}$$

- Quadratic or worse dependence on training set size

Dealing with scale

$$W = \sum_{j=1}^{N_T} \alpha_j \Phi^T(x_j) \tag{2}$$

- W resides in the span of the $\{\Phi^T(x_j)\}_{j=1}^{N_T}$
- $$W = \sum_n \beta_n U_n^T \tag{3}$$

  where U_n^T spans the same space as $\Phi^T(x_j)$
- One way to obtain U^T could be by performing SVD on $\{\Phi(x_j)\}$.

$$\Phi_j^s = \sum_{nn'} U_n^s S_{n'}^n (V^T)_j^{n'} \tag{4}$$

- Truncating singular values very close to zero, U^T will give the transformation from entire feature space to the reduced parameter space

Covariance to the Rescue!

- SVD is computationally challenging for large datasets
- Alternative method :

$$\rho_s^{s'} = \frac{1}{N_T} \sum_{j=1}^{N_T} \Phi_j^{s'} (\Phi_s^j)^T = \sum_n U_n^{s'} P_n (U^T)_s^n \tag{5}$$

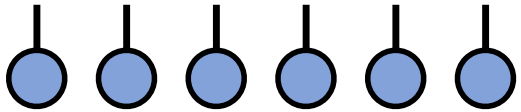
- Thus, U diagonalizes the *feature space covariance matrix* ρ
- Truncate directions along which ρ has a very small projection to rapidly reduce the size of the space needed to carry out learning tasks.

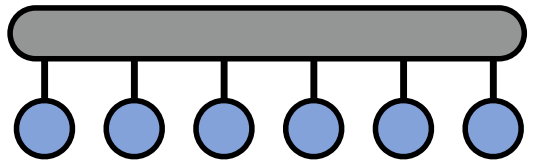
Generalization to Tensors

- We define a local feature map $\phi : \mathbb{R} \to \mathbb{R}^d$, and

$$\Phi(\boldsymbol{x}) = \phi(x_1) \circ \cdots \circ \phi(x_N), \tag{6}$$

so that now W is a tensor of order N with d^N weight parameters.

(a) $\Phi(\mathbf{x}) =$ 

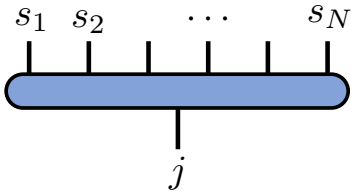
(b) $f(\mathbf{x}) =$  W $\Phi(\mathbf{x})$

(c) $W =$ 

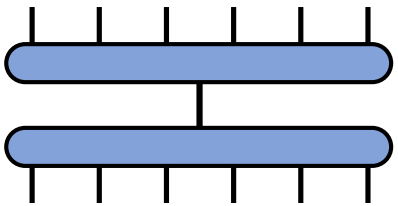
Back to ρ

- As before, the idea is to compute the eigenvectors of ρ, then discard those with smallest eigenvalues
- We think of the collection of feature vectors $\{\Phi(\boldsymbol{x}_j)\}_{j=1}^{N_T}$ as a single tensor of order $N+1$, so that ρ is formed by contracting Φ and Φ^T over the index j

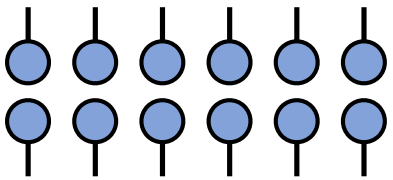
$$\Phi^{s_1 s_2 \cdots s_N}(\mathbf{x}_j) = \Phi_j^{s_1 s_2 \cdots s_N} =$$



$$\rho = \Phi\Phi^\dagger = \frac{1}{N_T}$$

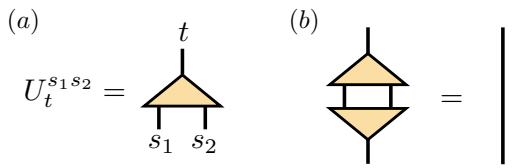


$$= \frac{1}{N_T} \sum_{j=1}^{N_T}$$



Local Isometry

- As it is not feasible to diagonalize ρ directly, we look at local isometries, which are third-order tensors $U_t^{s_1 s_2}$ satisfying $\sum_{s_1 s_2} U_t^{s_1 s_2} U_{s_1 s_2}^{t'} = \delta_t^{t'}$



- We define U_1 such that when it acts on the first two feature space indices, it maximizes the fidelity

$$F = \mathrm{Tr}[\rho] = \frac{1}{N_T} \sum_j \Phi^T \Phi \tag{7}$$

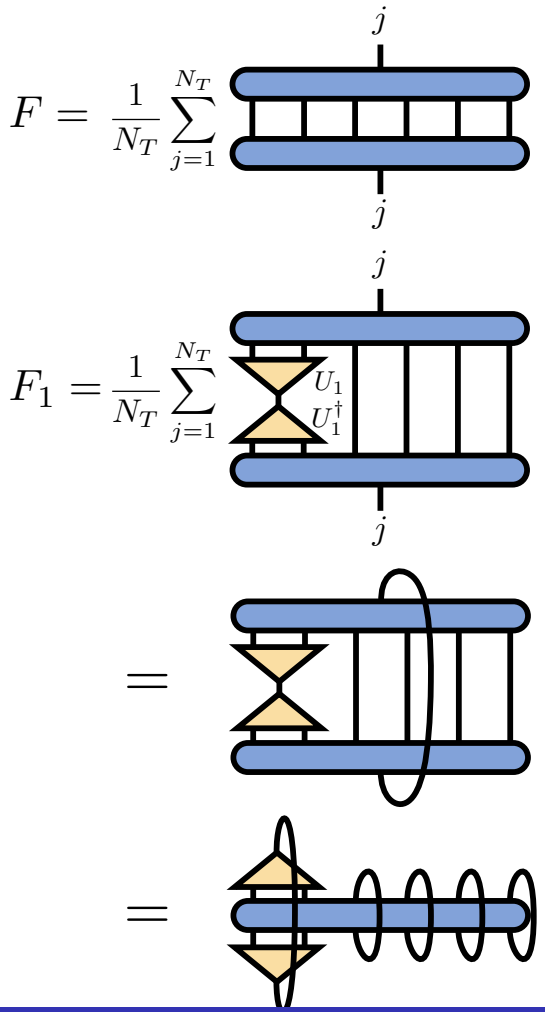
Local Isometry (Contd.)

- The fidelity of the approximated ρ is

$$F_1 = \frac{1}{N_T} \sum_j \Phi^T U_1 U_1^T \Phi \qquad (8)$$

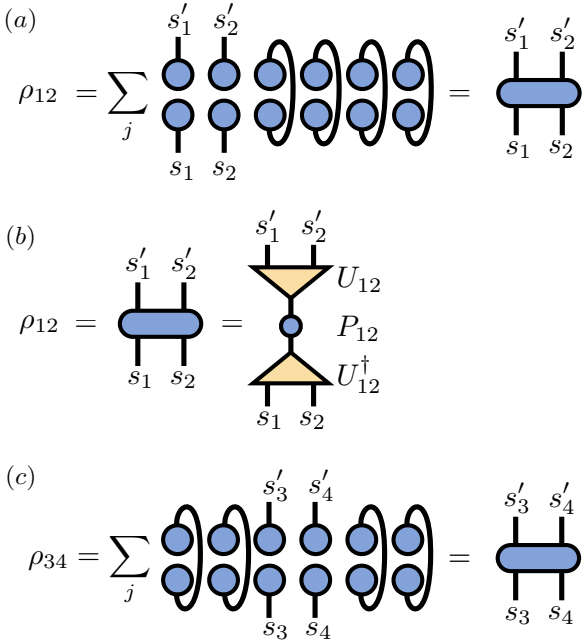
- $F_1 \leq F$.
- The reduced covariance matrix ρ_{12} is defined by tracing over all indices of ρ other than s_1 and s_2, so that

$$F_1 = \sum_{s_1 s_2 s_1' s_2' t} (U_1^T)^t_{s_1' s_2'} \rho_{12}^{s_1' s_2'}{}_{s_1 s_2} U_1^{s_1 s_2}{}_t \qquad (9)$$



Reduced Covariance matrix

- $\rho_{12} = U_1 P_{12} U_1^T$.
- U_1 is truncated keeping the eigenvectors corresponding to the D largest eigenvalues of ρ_{12}, where the choice of D depends on a given truncation error cutoff ϵ.



Diagonalizing ρ

We use the isometry layer to coarse grain the feature vectors, and iterate to diagonalize ρ in $\log_2(N)$ steps.

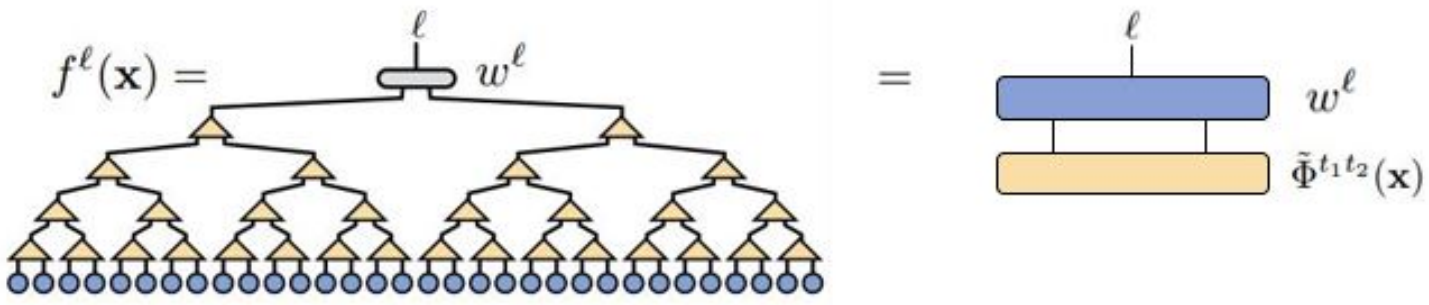
Defining the model

- Having determined $\mathcal{U}$, our model is:

$$f(x) = \sum_{t_1 t_2} w_{t_1 t_2} \tilde{\Phi}^{t_1 t_2}(x) \tag{10}$$

where

$$\tilde{\Phi}^{t_1 t_2}(x) = \sum_{t_1 t_2} \mathcal{U}^{t_1 t_2}_{s_1 s_2 \cdots s_N} \Phi^{s_1 s_2 \cdots s_N}(x) \tag{11}$$



Experiments

- The local feature map $\phi^{s_n}(x_n)$ is defined by

$$\phi^{s_n=1}(x_n) = 1$$
$$\phi^{s_n=2}(x_n) = x_n$$

- We use conjugate gradient to optimize the top tensor $\mathcal{W}$

ϵ	t_1	t_2	**Accuracy on training set** (%)	**Accuracy on test set** (%)
10^{-3}	107	151	98.75	97.44
$6x10^{-4}$	328	444	99.68	98.08

Table: Results on MNIST dataset using unsupervised / supervised algorithm

Mixed task-specific / unsupervised algorithm

- Mix the feature space covariance matrix ρ with another matrix based on a specific task:

$$\rho_\mu = \mu\hat{\rho}_W + (1-\mu)\hat{\rho} \tag{12}$$

- Given a prior guess for supervised task weights:

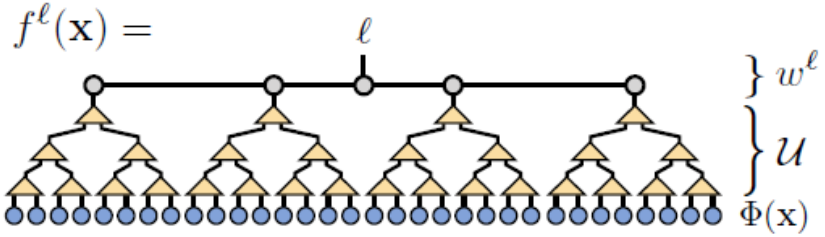
$$\hat{\rho}_W = \frac{1}{Tr(W^T W)} W^T W, \qquad \hat{\rho} = \frac{1}{Tr(\rho)}\rho$$

μ	ϵ	t_1	t_2	**Accuracy on training set**(%)	**Accuracy on test set**(%)
0.5	4x10^{-4}	279	393	99.798	98.110

Table: Results on MNIST dataset using mixed task-specific / unsupervised algorithm

Partial coarse graining: tree curtain model

- Consider the weights $\mathcal{W}$ as a matrix product state



μ	ϵ	**Accuracy on training set**(%)	**Accuracy on test set**(%)
0.9	2x10^{-9}	95.38	88.97

Table: Results on fashion-MNIST dataset using partial coarse graining / unsupervised algorithm

Approach	**Accuracy** (%)
XGBoost	89.8
AlexNet	89.9
Two-layer convolutional neural network trained with Keras	87.6
GoogLeNet	93.7

Table: Results for state-of-the-art approaches without preprocessing

- Constructing a model using a tree tensor network $\mathcal{U}$ and a top tensor $\mathcal{W}$
- The algorithm scales linearly in both training set size and input space dimension
- This can be reduced to sublinear using stochastic optimization techniques
- Experimentation can be done with different choices of the covariance matrix ρ and feature map
- Stochastic Gradient Descent can be used for optimization of the top tensor to improve accuracy
- Instead of using tree tensor network, use MERA tensor network

References

[1] Jacob Biamonte and Ville Bergholm. Quantum tensor networks in a nutshell. 2017.

[2] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. Neural Comput., 10(5):1299–1319, July 1998.