IFT 6760A - Lecture 11
Tensor Train Decomposition

Scribe(s): Nicolas Laliberte, Jimmy Leroux **Instructor:** Jacob Miller

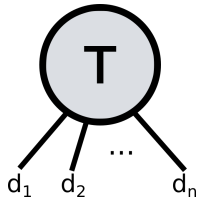
1 Summary

In the previous lecture tensor networks were introduced followed by a presentation of various tensor decomposition: Tensor Train (TT), Tensor Ring (TR) and Hierarchical Tucker (HT) decomposition. Two SVD based algorithms were presented for the former and the latter.

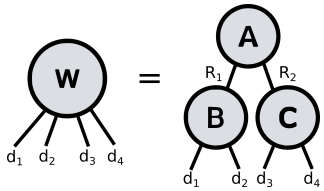
This lecture began with a quick review of tensor network and the tensor train decomposition. We briefly remark that TT is a form of data compression setting the stage for applications in machine learning, which are presented at the end of this lecture. Most of this lecture was dedicated to algebraic operations for TT. More precisely, we looked at how we can define basic linear algebra operations (addition, pointwise multiplication, inner product) on TT such that the structure of TT is preserved and the TT-rank of the results is somewhat bounded by the TT-ranks of the inputs. The lecture ends with application of TT in machine learning. We look at linear regression in this setting and how we can choose bond dimension of the TT with the Density Matrix Renormalization Group (DMRG) algorithm.

2 Tensor networks review

We have seen previously that we can represent a tensor $\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_n}$ in a diagramatic way using the tensor network notation. The simplest representation is the following,



where each leg represent an index in the original tensor. This form allows us to better visualize the interaction between various tensors in big equations where calculating a particular tensor element can become very cumbersome. This representation gives an easy way to represent tensor/matrix multiplication through the connection of matching legs. When two legs representing a matching dimension are connected, this means that we sum over this dimension and we say that we "contracted" the dimension. The summed indices don't appear in the final tensor, only the hanging one does. Using this logic, it is easy to imagine many differents way to represent a given tensor with the product of many elements. We can then decompose a tensor into product of smaller tensors and matrices of the form,



We call $\{d_1, d_2, d_4, d_4\}$ the visible indices and $\{R_1, R_2\}$ the hidden indices (bond).

3 Tensor-trains

Equipped with the tensor network notation, we can imagine many different decompositions. One of them that we will describe below is the tensor-train.

Definition 1 (Tensor-train)**.** *Given a tensor $\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_n}$ and a collection of core $\{\mathcal{G}^s\}_{s=1}^n$ where $\mathbf{G}^1 \in \mathbb{R}^{d_1 \times R_1}$, $\mathcal{G}^s \in \mathbb{R}^{R_{s-1} \times d_s \times R_s}$ and $\mathbf{G}^n \in \mathbb{R}^{R_{n-1} \times d_n}$. We can decompose $\mathcal{T}$ using the decomposition in figure 1, where the components are given by,*

$$\mathcal{T}_{i_1,i_2,\ldots,i_n} = \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \cdots \sum_{r_n=1}^{R_n} \mathbf{G}^1_{i_1,r_1} \mathcal{G}^2_{r_1,i_2,r_2} \cdots \mathbf{G}^n_{r_{n-1},i_n} = \mathbf{G}^{(1)}[i_1] \cdot \mathbf{G}^{(2)}[i_2] \cdot \ldots \cdot \mathbf{G}^{(n)}[i_n]$$

This form is also called Matrix product state.

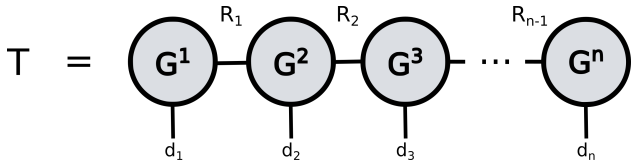
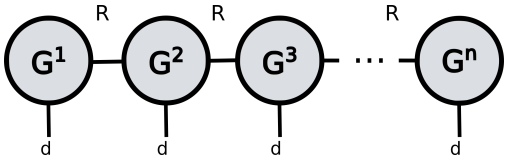


Figure 1: Tensor-train decomposition.

One of the advantages of representing a tensor in this tensor-train representation is compression. One way to see that is by denoting $d = \max(d_1, d_2, ..., d_n)$ and $R = \max(R_1, R_2, ..., R_n)$ and looking at the following decomposition of $\mathcal{T}$,



Each core $\mathcal{G}^s$ in this decomposition needs $\mathcal{O}(dR^2)$ parameters, so the overall train has $\mathcal{O}(ndR^2)$ parameters. This is a huge improvement from the exponential number of parameters $\mathcal{O}(d^n)$ required to store the original tensor. This seems to good to be true, in fact, in the worst case scenario $R \sim d^{n/2}$ which takes us back to $\mathcal{O}(d^n)$. Fortunately for us, for certain natural tensors, we can always choose R to be constant w.r.t n

4 Algebra with Tensor-trains

Naturally, we are interested in computing basic linear algebra operations within the TT-format. Computing those operations in the original tensor form to afterward come back to the TT-format defy the purpose of the decomposition. Before continuing, let us introduce a definition on the TT-format.

Definition 2 (TT-Rank)**.** *The TT-rank of a tensor $\mathcal{T}$ is the smallest tuple $(R_1, \cdots, R_{n-1})$ such that $\mathcal{T}$ admits a TT decomposition with cores of sizes $R_1, \cdots, R_{n-1}$ (see figure 1).*

In this lecture, we will denote by $\operatorname{rank}(\mathcal{T})$ *the maximum of the elements of the TT-rank, i.e.* $\operatorname{rank}(\mathcal{T}) = \max(R_1, R_2, \ldots, R_{n-1})$.

The TT-rank of a TT decomposition is the expression of the complexity of TT. Back to algebraic operations, here comes the main question: can we define operations on TT's such that the TT-rank of the output somewhat respects the

TT-rank of the inputs. To be more specific, for 2 tensor-trains $\mathcal{T}$, $\tilde{\mathcal{T}}$ can we define $\mathcal{T} + \tilde{\mathcal{T}} = \mathcal{S}$ such that $\text{rank}(\mathcal{S}) \leq \text{rank}(\mathcal{T}) + \text{rank}(\tilde{\mathcal{T}})$? We now show that we have this kind of property for the following algebraic operations: addition, pointwise multiplication and inner product.

4.1 Addition

Suppose $\mathcal{T}, \tilde{\mathcal{T}}$ and $\mathcal{S}$ tensor-trains such that

$$\mathcal{S} = \mathcal{T} + \tilde{\mathcal{T}}$$

with

$$\mathcal{S}_{i_1,i_2,\ldots,i_n} = \mathbf{H}^1[i_1] \cdot \mathbf{H}^2[i_2] \ \ldots \ \mathbf{H}^{n-1}[i_{n-1}] \cdot \mathbf{H}^n[i_n] \tag{1}$$

where the cores of $\mathcal{T}$ and $\tilde{\mathcal{T}}$ are $\mathbf{G}^s[i_s]$ and $\tilde{\mathbf{G}}^s[i_s]$ respectively. We now define $\mathbf{H}^s[i_s]$ and show with this definition that we indeed get $\mathcal{S} = \mathcal{T} + \tilde{\mathcal{T}}$. First, we define central cores and the last one as

$$\mathbf{H}^s[i_s] = \mathbf{G}^s[i_s] \oplus \tilde{\mathbf{G}}^s[i_s] = \left[\begin{array}{c|c} \mathbf{G}^s[i_s] & 0 \\ \hline 0 & \tilde{\mathbf{G}}^s[i_s] \end{array}\right], \quad \mathbf{H}^n[i_n] = \left[\begin{array}{c} \mathbf{G}^n[i_n] \\ \hline \tilde{\mathbf{G}}^n[i_n] \end{array}\right]$$

Now, from equation (1) we have

$$\begin{aligned} \mathcal{S}_{i_1,i_2,\ldots,i_n} &= \mathbf{H}^1[i_1] \cdot \mathbf{H}^2[i_2] \ \ldots \ \mathbf{H}^{n-2}[i_{n-2}] \left[\begin{array}{c|c} \mathbf{G}^{n-1}[i_{n-1}] & 0 \\ \hline 0 & \tilde{\mathbf{G}}^{n-1}[i_{n-1}] \end{array}\right] \left[\begin{array}{c} \mathbf{G}^n[i_n] \\ \hline \tilde{\mathbf{G}}^n[i_n] \end{array}\right] \\ &= \mathbf{H}^1[i_1] \cdot \mathbf{H}^2[i_2] \ \ldots \ \mathbf{H}^{n-2}[i_{n-2}] \left[\begin{array}{c} \mathbf{G}^{n-1}[i_{n-1}]\mathbf{G}^n[i_n] \\ \hline \tilde{\mathbf{G}}^{n-1}[i_{n-1}]\tilde{\mathbf{G}}^n[i_n] \end{array}\right] \end{aligned}$$

and applying this product recursively, we get

$$\mathcal{S}_{i_1,i_2,\ldots,i_n} = \mathbf{H}^1[i_1] \left[\begin{array}{c} \mathbf{G}^2[i_2] \ \ldots \ \mathbf{G}^{n-1}[i_{n-1}]\mathbf{G}^n[i_n] \\ \hline \tilde{\mathbf{G}}^2[i_2] \ \ldots \ \tilde{\mathbf{G}}^{n-1}[i_{n-1}]\tilde{\mathbf{G}}^n[i_n]. \end{array}\right] \tag{2}$$

We can easily see that defining

$$\mathbf{H}^1[i_1] = \left[\begin{array}{c|c} \mathbf{G}^1[i_1] & \tilde{\mathbf{G}}^1[i_1] \end{array}\right]$$

the above equation (2) comes down to

$$\begin{aligned} \mathcal{S}_{i_1,i_2,\ldots,i_n} &= \prod_{s=1}^{n} \mathbf{G}^s[i_s] + \prod_{s=1}^{n} \tilde{\mathbf{G}}^s[i_s] \\ &= \mathcal{T}_{i_1,i_2,\ldots,i_n} + \tilde{\mathcal{T}}_{i_1,i_2,\ldots,i_n} \end{aligned}$$

Thus, given that the bond dimensions of $\mathcal{T}$ and $\tilde{\mathcal{T}}$ are $\{R_i\}_{i=1}^n$ and $\{\tilde{R}_i\}_{i=1}^n$ respectively, we see that the bonds dimension of $\{\mathbf{H}^s\}_{s=1}^n$ are $R_s + \tilde{R}_s$. Hence, $\text{rank}(\mathcal{S}) \leq \text{rank}(\mathcal{T}) + \text{rank}(\tilde{\mathcal{T}})$.

4.2 Pointwise multiplication

For the element-wise product, instead of posing, $\mathbf{H}^s[i_s] = \mathbf{G}^s[i_s] \oplus \tilde{\mathbf{G}}^s[i_s]$ we pose,

$$\mathbf{H}^s[i_s] = \mathbf{G}^s[i_s] \otimes \tilde{\mathbf{G}}^s[i_s] = \begin{bmatrix} \mathbf{G}^s[i_s]_{1,1}\tilde{\mathbf{G}}^s[i_s] & \ldots & \mathbf{G}^s[i_s]_{1,R_s}\tilde{\mathbf{G}}^s[i_s] \\ \vdots & & \vdots \\ \mathbf{G}^s[i_s]_{R_{s-1},1}\tilde{\mathbf{G}}^s[i_s] & \ldots & \mathbf{G}^s[i_s]_{R_{s-1},R_s}\tilde{\mathbf{G}}^s[i_s] \end{bmatrix}$$

Having that we can write,

$$\mathbf{H}^{n-1}[i_{n-1}]\mathbf{H}^{n}[i_n] = \begin{bmatrix} \mathbf{G}^{n-1}[i_{n-1}]_{1,1}\tilde{\mathbf{G}}^{n-1}[i_{n-1}] & \dots & \mathbf{G}^{n-1}[i_{n-1}]_{1,R_{n-1}}\tilde{\mathbf{G}}^{n-1}[i_{n-1}] \\ \vdots & & \vdots \\ \mathbf{G}^{n-1}[i_{n-1}]_{R_{n-2},1}\tilde{\mathbf{G}}^{n-1}[i_{n-1}] & \dots & \mathbf{G}^{n-1}[i_{n-1}]_{R_{n-2},R_{n-1}}\tilde{\mathbf{G}}^{n-1}[i_{n-1}] \end{bmatrix} \begin{bmatrix} \mathbf{G}^{n}[i_n]_{1}\tilde{\mathbf{G}}^{n}[i_n] \\ \vdots \\ \mathbf{G}^{n}[i_n]_{R_{n-1}}\tilde{\mathbf{G}}^{n}[i_n] \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{G}^{n-1}[i_{n-1}]_{1,1}\mathbf{G}^{n}[i_n]_{1}\tilde{\mathbf{G}}^{n-1}[i_{n-1}]\tilde{\mathbf{G}}^{n}[i_n] & +\dots+ & \mathbf{G}^{n-1}[i_{n-1}]_{1,R_{n-1}}\mathbf{G}^{n}[i_n]_{R_{n-1}}\tilde{\mathbf{G}}^{n-1}[i_{n-1}]\tilde{\mathbf{G}}^{n}[i_n] \\ \vdots & & \vdots \\ \mathbf{G}^{n-1}[i_{n-1}]_{R_{n-2},1}\mathbf{G}^{n}[i_n]_{1}\tilde{\mathbf{G}}^{n-1}[i_{n-1}]\tilde{\mathbf{G}}^{n}[i_n] & +\dots+ & \mathbf{G}^{n-1}[i_{n-1}]_{R_{n-2},R_{n-1}}\mathbf{G}^{n}[i_n]_{R_{n-1}}\tilde{\mathbf{G}}^{n-1}[i_{n-1}]\tilde{\mathbf{G}}^{n}[i_n] \end{bmatrix}$$

$$= \mathbf{G}^{n-1}[i_{n-1}]\mathbf{G}^{n}[i_n]\tilde{\mathbf{G}}^{n-1}[i_{n-1}]\tilde{\mathbf{G}}^{n}[i_n]$$

$$= \begin{bmatrix} (\mathbf{G}^{n-1}[i_{n-1}]\mathbf{G}^{n}[i_n])_{1}\tilde{\mathbf{G}}^{n-1}[i_{n-1}]\tilde{\mathbf{G}}^{n}[i_n] \\ \vdots \\ (\mathbf{G}^{n-1}[i_{n-1}]\mathbf{G}^{n}[i_n])_{R_{n-2}}\tilde{\mathbf{G}}^{n-1}[i_{n-1}]\tilde{\mathbf{G}}^{n}[i_n] \end{bmatrix}$$

$$= \mathbf{G}^{n-1}[i_{n-1}]\mathbf{G}^{n}[i_n] \otimes \tilde{\mathbf{G}}^{n-1}[i_{n-1}]\tilde{\mathbf{G}}^{n}[i_n]$$

This shows at the same time the mixed product property of the kronecker product,

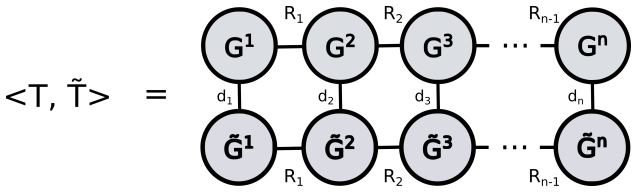
$$(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = (\mathbf{AC}) \otimes (\mathbf{BD}) \tag{3}$$

By continuing this until we have multiplied the whole tensor-train, we can easily show that,

$$\boldsymbol{\mathcal{S}}_{i_1,i_2,\dots,i_n} = (\boldsymbol{\mathcal{T}} * \tilde{\boldsymbol{\mathcal{T}}})_{i_1,i_2,\dots,i_n} = \prod_{s=1}^{n} \mathbf{G}^{s}[i_s] \otimes \prod_{s'=1}^{n} \tilde{\mathbf{G}}^{s'}[i_{s'}] = \boldsymbol{\mathcal{T}}_{i_1,i_2,\dots,i_n}\tilde{\boldsymbol{\mathcal{T}}}_{i_1,i_2,\dots,i_n}$$

4.3 Inner product

Another operation that can be done with tensors in their tensor-train representation is the inner product. Suppose we have two tensors $\boldsymbol{\mathcal{T}}, \tilde{\boldsymbol{\mathcal{T}}} \in \mathbb{R}^{d_1 \times d_2 \times \dots \times d_n}$. Recall that for tensors, as well as for vectors, the inner product can be represented by connecting all the corresponding visible index together. The inner product can then easily be represented in the tensor-train representation by,



We can see that this representation is correct by simply using the definition,

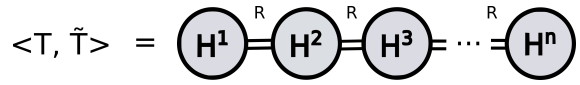
$$\langle \boldsymbol{\mathcal{T}}, \tilde{\boldsymbol{\mathcal{T}}} \rangle = \sum_{i_1=1}^{d_1} \sum_{i_2=1}^{d_2} \sum_{i_3=1}^{d_3} \dots \sum_{i_n=1}^{d_n} \boldsymbol{\mathcal{T}}_{i_1,\dots i_n}\tilde{\boldsymbol{\mathcal{T}}}_{i_1,\dots i_n}$$

From the usual sum ordering in the natural tensor representation, we can encode a "contraction stategy" for the inner product in the tensor-train representation. By replacing the value of the product $\boldsymbol{\mathcal{T}}_{i_1,\dots i_n}\tilde{\boldsymbol{\mathcal{T}}}_{i_1,\dots i_n}$ by the eqation found for the element wise product we have that,

$$\langle \boldsymbol{\mathcal{T}}, \tilde{\boldsymbol{\mathcal{T}}} \rangle = \sum_{i_1=1}^{d_1} \sum_{i_2=1}^{d_2} \sum_{i_3=1}^{d_3} \dots \sum_{i_n=1}^{d_n} \prod_{s=1}^{n} \mathbf{G}^{s}[i_s] \otimes \prod_{s'=1}^{n} \tilde{\mathbf{G}}^{s'}[i_{s'}]$$

By using the mixed product property of the kronecker product that we showed earlier (3), the above expression can we reduced to,

$$\langle \boldsymbol{\mathcal{T}}, \tilde{\boldsymbol{\mathcal{T}}} \rangle = \sum_{i_1=1}^{d_1} \mathbf{G}^1[i_1] \otimes \tilde{\mathbf{G}}^1[i_1] \sum_{i_2=1}^{d_2} \mathbf{G}^2[i_2] \otimes \tilde{\mathbf{G}}^2[i_2] \sum_{i_3=1}^{d_3} \mathbf{G}^3[i_3] \otimes \tilde{\mathbf{G}}^3[i_3] \ \dots \ \sum_{i_n=1}^{d_n} \mathbf{G}^n[i_n] \otimes \tilde{\mathbf{G}}^n[i_n]$$



Where we posed $\mathbf{H}^s[i_s] = \sum_{i_s=1}^{d_s} \mathbf{G}^s[i_s] \otimes \tilde{\mathbf{G}}^s[i_s]$ and pushed the corresponding sums to the corresponding product to reduce computation. It is important to note that since tensor $rank(\boldsymbol{\mathcal{T}})$ is bounded by $\max(R_1, R_2, ..., R_n)$ and that we do n summation of $\mathcal{O}(d)$, the total time to compute the inner-product is of $\mathcal{O}(ndR^4)$. This is a huge improvement, compared to the standard way of computing $\langle \boldsymbol{\mathcal{T}}, \tilde{\boldsymbol{\mathcal{T}}} \rangle$ which is of $\mathcal{O}(d^n)$

5 Tensor-trains for Machine Learning

This section follows [2] and [1]. First, let's remind quickly what is a linear regression. Given a dataset $\{(x_i, y_i\}_{i=1}^D$ we define

$$\hat{y}(x_i) = \langle W, x_i \rangle$$

The objective is to learn the parameter W which minimizes $\mathcal{L}(\hat{y}(x_i), y_i)$ for a choosen loss function $\mathcal{L}$. Note that W is defined with the bias trick. Furthermore, suppose a feature map $\phi(x)$ defined as the following (model from [2])

$$\phi(x_i) = \begin{bmatrix} 1 \\ x_i \end{bmatrix}$$

where

$$\boldsymbol{\mathcal{X}} := \phi(X) = \phi(x_1) \circ \dots \circ \phi(x_n)$$

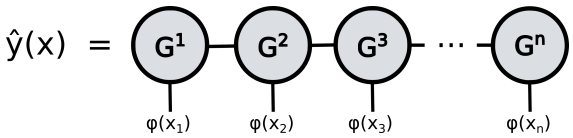
Remark that $\boldsymbol{\mathcal{X}}$ represents a n-dimensional tensor with 2^n elements corresponding to permutations of features such as $x_1x_2x_3$. Each of those elements have exactly one associated weight. In this setting, the above problem comes down to

$$\hat{y}(x_i) = \langle \boldsymbol{\mathcal{W}}, \boldsymbol{\mathcal{X}} \rangle$$

With a weight tensor $\boldsymbol{\mathcal{W}}$. This is where the tensor-trains comes handy, as we saw, we can represent this exponentially large tensor $\boldsymbol{\mathcal{W}}$ in a compact TT-format. We can show (see theorem 1 of [2]) that the model $\hat{y}(x)$ can be computed in $\mathcal{O}(nr^2)$ where r is the TT-rank. Some questions arise from this, note that the TT-rank is an hyperparameters of this model. Suppose this setting

$$\hat{y}(x_i) = \langle \boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{X}} \rangle$$

where $\boldsymbol{\mathcal{T}}$ is a tensor-train.



So, how can we train the model and how do we choose the TT-rank of $\boldsymbol{\mathcal{T}}$? First, we can show the mapping

$$(\mathbf{G}^1, \dots, \mathbf{G}^n; x) \to \hat{y}(x)$$

is differentiable. Then, the gradient descent works. Secondly, the way to choose the bond dimension is inspired by the Density Matrix Renormalization Group (DMRG) algorithm developed in physic. The idea is to train with adaptive bond dimensions. The algorithm is the following with ϵ as an hyperparameter

Repeat:

1. Merge the cores $\mathbf{G}^s$ and $\mathbf{G}^{s+1}$ into $\mathbf{B}$.
2. Train with respect to $\{\mathbf{G}^1, \ldots, \mathbf{G}^{s-1}, \mathbf{B}, \mathbf{G}^{s+1}, \ldots, \mathbf{G}^n\}$
3. Split $\mathbf{B} \rightarrow \tilde{\mathbf{G}}^s, \tilde{\mathbf{G}}^{s+1}$ using SVD.
4. From the singular values, we choose the maximum R such that $\sigma_R > \epsilon$. The new bond dimension is $R'_s = R$.

References

[1] E. Miles Stoudenmire and D. J. Schwab. Supervised Learning with Quantum-Inspired Tensor Networks. *arXiv e-prints*, art. arXiv:1605.05775, May 2016.

[2] A. Novikov, M. Trofimov, and I. Oseledets. Exponential Machines. *arXiv e-prints*, art. arXiv:1605.03795, May 2016.