Tensor Decomposition via Core Tensor Networks

Jianfu Zhang, Zerui Tao, Qibin Zhao, and Liqing Zhang
RIKEN AIP, Shanghai Jiao Tong University, Lanzhou University
Assumptions

• The mappings of a bunch of tensors might be shared or highly correlated.

• Leverage the correlated information might be helpful for convergence.
Assumptions

• Train DNN models with auxiliary samples

• With pre-trained models, infer the initialization to fast and accurate tensor decomposition
Gradient Descent

- Tensor completion:
- Tensor denoising:
- Tensor decomposition:
- Objective function:
- Gradient descent:
  \[ G_i^{(k)} \leftarrow G_i^{(k)} - \lambda \nabla G_i^{(k)} L \]
- Convergence:
  \[ |L_t - L_{t-1}| < th_d \]

\[
t_{i_1,\ldots,i_N} = \hat{t}_{i_1,\ldots,i_N} \times w_{i_1,\ldots,i_N} \\
t_{i_1,\ldots,i_N} = \hat{t}_{i_1,\ldots,i_N} + e_{i_1,\ldots,i_N} \\
\mathcal{T} \approx \mathcal{X} = \ll \mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(N)} \gg \\
L = \frac{1}{M} \sum_{i=1}^{M} \| \mathcal{W}_i \ast (\mathcal{T}_i - \mathcal{X}_i) \|^2_F
\]

**Algorithm 1** Gradient Descent for Tensor Decomposition.

**Require:** Input data \( \{\mathcal{T}_1, \ldots, \mathcal{T}_M\} \).

**Ensure:** Model parameters \( \{\mathcal{G}_i^{(1)}, \ldots, \mathcal{G}_i^{(N)}\}_{i=1}^M \).

1: Randomly initialize \( \{\mathcal{G}_i^{(1)}, \ldots, \mathcal{G}_i^{(N)}\}_{i=1}^M \).
2: while Not converged do
3: Calculate loss function \( L \) based on Eq. 5.
4: Update \( \{\mathcal{G}_i^{(1)}, \ldots, \mathcal{G}_i^{(N)}\}_{i=1}^M \) based on Eq. 6.
5: end while
Core Tensor Network

- \( f(\theta^{(k)}, \mathcal{T}) \): (main core tensor) The function representation for the network to learn the k-th core tensor with network parameter \( \theta \)
- Learn the function with multi-layer perceptron
- Combine bias core tensor:
  \[
  g_i^{(k)} = f(\theta^{(k)}, \mathcal{T}_i) + b_i^{(k)}
  \]
Gradient Descent for Core Tensor Networks

• Initialize the core tensors with random projections of the input tensors

• Main core tensors share the same model parameter for all the input tensors

\[ \theta^{(k)} \leftarrow \theta^{(k)} - \lambda \nabla_{\theta^{(k)}} L, B_i^{(k)} \leftarrow B_i^{(k)} - \lambda \nabla_{B_i^{(k)}} L. \]

**Algorithm 2** Gradient Descent for Core Tensor Networks.

**Require:** Input data \( \{\mathcal{T}_1, \ldots, \mathcal{T}_M\} \).

**Ensure:** Model parameters \( \{\theta^{(1)}, \ldots, \theta^{(N)}\} \).

**Ensure:** Model parameters \( \{B_i^{(1)}, \ldots, B_i^{(N)}\}_{i=1}^M \).

1: Randomly initialize \( \{\theta^{(1)}, \ldots, \theta^{(N)}\} \).

2: Initialize \( \{B_i^{(1)}, \ldots, B_i^{(N)}\}_{i=1}^M \) with zeros.

3: **while** Not converged **do**

4: Calculate loss function \( L \) based on Eq. 5.

5: Update \( \theta^{(k)},\{B_i^{(k)}\}_{i=1}^M \) for all \( k \) based on Eq. 6.

6: **end while**
Core Tensor Network with Meta-Learning

Algorithm 3 Transfer Learning for Core Tensor Networks.

Require: Training data \( \{ T'_1, \ldots, T'_M \} \).
Require: Test data \( \{ T_1, \ldots, T_M \} \).
Ensure: Model parameters \( \{ \theta^{(1)}, \ldots, \theta^{(N)} \} \).
Ensure: Model parameters \( \{ B^{(1)}_i, \ldots, B^{(N)}_i \}_{i=1}^M \).
1: Randomly initialize \( \{ \theta^{(1)}, \ldots, \theta^{(N)} \} \).
2: for \( \text{iter} \) in 1, \ldots, \( \text{iter}_{\max} \) do
3: Sample a batch \( \{ T'_{b_1}, \ldots, T'_{b_m} \} \) from training set.
4: Initialize \( \{ B^{(1)}_i, \ldots, B^{(N)}_i \}_{i=1}^M \) with zeros.
5: for \( p \) in 1, \ldots, \( \gamma \) do
6: Calculate \( L \) for \( \{ T'_{b_1}, \ldots, T'_{b_m} \} \) based on Eq. 5.
7: Update \( \{ B^{(1)}_i, \ldots, B^{(N)}_i \}_{i=1}^M \) based on Eq. 6.
8: end for
9: Calculate \( L \) for \( \{ T'_{b_1}, \ldots, T'_{b_m} \} \) based on Eq. 5.
10: Update \( \theta^{(k)} \) for all \( k \in [N] \) based on Eq. 6.
11: end for
12: Initialize \( \{ B^{(1)}_i, \ldots, B^{(N)}_i \}_{i=1}^M \) with zeros.
13: while Not converged do
14: Calculate \( L \) for \( \{ T_1, \ldots, T_M \} \) based on Eq. 5.
15: Update \( \theta^{(k)}, \{ B^{(k)}_i \}_{i=1}^M \) for all \( k \) based on Eq. 6.
16: end while

- Pre-train the main core tensors and finetune the model to the test set.
Experiments

- CTN converges much faster than GD

<table>
<thead>
<tr>
<th>Metrics/Algorithms</th>
<th>GD</th>
<th>ALS</th>
<th>CTN</th>
<th>tCTN</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSE↓</td>
<td>0.1243</td>
<td>0.1229</td>
<td>0.1205</td>
<td>0.1201</td>
</tr>
<tr>
<td>Second per Image↓</td>
<td>22.2</td>
<td>109.3</td>
<td>6.55</td>
<td>0.57</td>
</tr>
</tbody>
</table>

- Higher $\gamma$ costs more time to train, $\gamma = 20$ performs best

<table>
<thead>
<tr>
<th>Metrics/Settings</th>
<th>0</th>
<th>5</th>
<th>10</th>
<th>20</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSE↓</td>
<td>0.1284</td>
<td>0.1246</td>
<td>0.1212</td>
<td>0.1201</td>
<td>0.1211</td>
</tr>
<tr>
<td>Batch per Second↑</td>
<td>26.53</td>
<td>19.53</td>
<td>13.44</td>
<td>9.93</td>
<td>4.68</td>
</tr>
</tbody>
</table>
Experiments

<table>
<thead>
<tr>
<th>Rate</th>
<th>Metric</th>
<th>BCPF</th>
<th>TT-WOPT</th>
<th>TR-ALS</th>
<th>TRLRF</th>
<th>CTN</th>
<th>tCTN</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.9</td>
<td>PSNR↑</td>
<td>19.69</td>
<td>23.21</td>
<td>22.22</td>
<td>22.27</td>
<td>23.50</td>
<td>23.53</td>
</tr>
<tr>
<td></td>
<td>RSE↓</td>
<td>0.1868</td>
<td>0.1262</td>
<td>0.1396</td>
<td>0.1388</td>
<td>0.1205</td>
<td>0.1201</td>
</tr>
<tr>
<td>0.7</td>
<td>PSNR↑</td>
<td>25.18</td>
<td>25.36</td>
<td>24.51</td>
<td>26.82</td>
<td>27.78</td>
<td>27.79</td>
</tr>
<tr>
<td></td>
<td>RSE↓</td>
<td>0.0993</td>
<td>0.0972</td>
<td>0.1072</td>
<td>0.0822</td>
<td>0.0736</td>
<td>0.0735</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Level</th>
<th>Metrics</th>
<th>TT-WOPT</th>
<th>TR-ALS</th>
<th>CTN</th>
<th>tCTN</th>
</tr>
</thead>
<tbody>
<tr>
<td>10dB</td>
<td>PSNR↑</td>
<td>19.24</td>
<td>19.69</td>
<td>20.17</td>
<td>20.33</td>
</tr>
<tr>
<td></td>
<td>RSE↓</td>
<td>0.0849</td>
<td>0.0774</td>
<td>0.0686</td>
<td>0.0682</td>
</tr>
<tr>
<td>20dB</td>
<td>PSNR↑</td>
<td>19.48</td>
<td>20.03</td>
<td>20.23</td>
<td>20.34</td>
</tr>
<tr>
<td></td>
<td>RSE↓</td>
<td>0.0804</td>
<td>0.0723</td>
<td>0.0682</td>
<td>0.0675</td>
</tr>
</tbody>
</table>
Future Works

• Replace MLP with CNN, which can go deeper and preserve local structures of the input tensors.

• Analyze the patterns of the core tensors.

• Theoretical analyses of the core tensor network.

• Thanks for listening! E-mail: jianfu.zhang@riken.jp