Supervised Learning for Non-Sequential Data with the Canonical Polyadic Decomposition

Plan

A. tensor preliminaries
B. tensor network framework for supervised learning
C. the interpretability of the model and its universal function approximation property
D. Algorithms of learning and prediction
E. Local feature mapping
F. Model initialization
G. Experiments
H. Bibliography

A. Tensor preliminaries

CP-decomposition :

$$\mathcal{X} = \sum_{r=1}^{R} \mathbf{a}_r^{(1)} \circ \cdots \circ \mathbf{a}_r^{(N)} = \sum_{r=1}^{R} \bigcirc_{k=1}^{N} \mathbf{a}_r^{(k)}$$

Factor matrices :

$$\mathbf{A}^{(n)} = [\mathbf{a}_1^{(n)}, \ldots, \mathbf{a}_R^{(n)}] \in \mathbb{R}^{I_n \times R}$$

Tensor-entries :

$$x_{i_1,\ldots,i_N} = \sum_{r=1}^{R} a_{i_1,r}^{(1)} a_{i_2,r}^{(2)} \cdots a_{i_N,r}^{(N)}$$

$$= \left(\mathop{\circledast}_{k=1}^{N} \hat{\mathbf{a}}_{i_k}^{(k)} \right) \mathbf{1},$$

$$\mathbf{1} = [1, \ldots, 1]^T \in \mathbb{R}^R$$

B tensor network framework for supervised learning

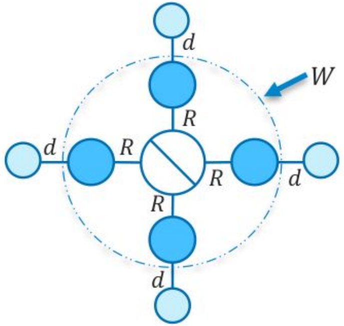
- A supervised learning task where each sample is represented as a set of N features.
- Local feature mapping applied to every feature

Feature mapping $\phi : \mathbb{R} \rightarrow \mathbb{R}^d$

- Interaction between the features :

$$\Phi(\mathbf{x}) = \bigcirc_{k=1}^{N} \phi(x_k) \in \mathbb{R}^{d^N}$$

Supervised learning model $\Rightarrow$ $f(\mathbf{x}) = \langle \Phi(\mathbf{x}), \mathcal{W} \rangle$



C Model interpretability

- the coefficient for each (transformed) feature interaction obtained by performing a Hadamard product of the corresponding row vectors of the learned factor matrices and then by summing over all entries of the resulting vector

$$f(\mathbf{x}) = \sum_{i_1,\ldots,i_N} w_{i_1,\ldots,i_N} \prod_{k=1}^{N} \phi(x_k)_{i_k}$$
$$= \sum_{i_1,\ldots,i_N} \left(\bigcircledast_{k=1}^{N} \hat{\mathbf{a}}_{i_k}^{(k)} \right) \mathbf{1} \prod_{k=1}^{N} \phi(x_k)_{i_k}$$

===> **Enhanced interpretability** over classic deep learning architectures (ANN, CNN, RNN, …)

D Algorithms for learning

Model prediction

$$
\begin{aligned}
f(\mathbf{x}) &= \langle \Phi(\mathbf{x}), \mathcal{W} \rangle \\
&= \langle \mathrm{vec}\left(\Phi(\mathbf{x})\right), \mathrm{vec}\left(\mathcal{W}\right) \rangle \\
&= \mathrm{vec}\left(\Phi(\mathbf{x})\right)^T \mathrm{vec}\left(\mathcal{W}\right) \\
&= \left(\bigodot_{k=N}^{1} \phi(x_k) \right)^T \left(\bigodot_{k=N}^{1} \mathbf{A}^{(k)} \right) \mathbf{1} \\
&= \left(\mathop{\circledast}_{k=1}^{N} \phi^T(x_k) \mathbf{A}^{(k)} \right) \mathbf{1}.
\end{aligned}
$$

Partial derivative of prediction

$$
\begin{aligned}
f(\mathbf{x}) &= \langle \Phi(\mathbf{x}), \mathcal{W} \rangle \\
&= \mathrm{Tr}\left(\mathbf{A}^{(n)} \left(\mathop{\circledast}_{\substack{k=1 \\ k \neq n}}^{N} \mathbf{A}^{(k)T} \phi(x_k) \right) \phi^T(x_n) \right)
\end{aligned}
$$

Derevative wrt factor matrices

$$
\frac{\partial f(\mathbf{x})}{\partial \mathbf{A}^{(n)}} = \phi(x_n) \left(\mathop{\circledast}_{\substack{k=1 \\ k \neq n}}^{N} \phi^T(x_k) \mathbf{A}^{(k)} \right)
$$

D Algorithms for learning

Algorithm 1: Model Prediction

Input: Data point $\mathbf{x} \in \mathbb{R}^N$ and factor matrices $\mathbf{A}^{(n)} \in \mathbb{R}^{d \times R}$, $1 \leq n \leq N$

Output: Prediction $\hat{y} \in \mathbb{R}$

begin

  // Construct $\phi(x_n) \in \mathbb{R}^{d \times 1}$ for $1 \leq n \leq N$

  $\mathbf{p} = \mathbf{1}^T \in \mathbb{R}^{1 \times R}$ // Initialize (row) vector of ones

  for $n = 1, \ldots, N$ **do**

    $\mathbf{p} \leftarrow \mathbf{p} \circledast \phi^T(x_n)\mathbf{A}^{(n)}$

  end

  $\hat{y} = \text{sum}(\mathbf{p})$ // Sum entries of $\mathbf{p}$

end

Algorithm 2: Partial Derivative of Prediction

Input: Data point $\mathbf{x} \in \mathbb{R}^N$ and factor matrices $\mathbf{A}^{(n)} \in \mathbb{R}^{d \times R}$, $1 \leq n \leq N$

Output: Partial derivative $\frac{\partial f(\mathbf{x})}{\partial \mathbf{A}^{(n)}} \in \mathbb{R}^{d \times R}$ for $1 \leq n \leq N$

begin

  // Construct $\phi(x_n) \in \mathbb{R}^{d \times 1}$ for $1 \leq n \leq N$

  $\mathbf{p} = \mathbf{1}^T \in \mathbb{R}^{1 \times R}$ // Initialize (row) vector of ones

  for $n = 1, \ldots, N$ **do**

    $\mathbf{m}_n = \phi^T(x_n)\mathbf{A}^{(n)} \in \mathbb{R}^{1 \times R}$ // Store matrix-vector products

    $\mathbf{p} \leftarrow \mathbf{p} \circledast \mathbf{m}_n$ // Store Hadamard products

  end

  for $n = 1, \ldots, N$ **do**

    $\mathbf{d} = \mathbf{p} \oslash \mathbf{m}_n \in \mathbb{R}^{1 \times R}$ // Divide element-wise

    $\frac{\partial f(\mathbf{x})}{\partial \mathbf{A}^{(n)}} = \phi(x_n)\mathbf{d}$

  end

end

$\mathcal{O}(NRd)$

D Algorithms for learning (Regularization)

Order regularization : penalizing large coefficients for higher-order terms more than those for lower-order ones

$$\mathcal{B} = \bigcirc_{k=1}^{N} \mathbf{b} \qquad \mathbf{b} = [1, \beta, \beta^2, \ldots, \beta^{d-1}]^T$$

$$\mathbf{B} = [\mathbf{b}, \ldots, \mathbf{b}] \in \mathbb{R}^{d \times R}$$

$$P(\mathcal{B}, \mathcal{W}) = \langle \mathcal{B} \circledast \dot{\mathcal{W}}, \mathcal{B} \circledast \dot{\mathcal{W}} \rangle$$

$$= \mathbf{1}^T \left(\bigotimes_{k=1}^{N} \mathbf{Y}^{(k)T} \mathbf{Y}^{(k)} \right) \mathbf{1}$$

$$\mathbf{Y}^{(n)} = \mathbf{A}^{(n)} \circledast \mathbf{B}$$

Algorithm 3: Order Regularization Penalty

Input: Factor matrices $\mathbf{A}^{(n)} \in \mathbb{R}^{d \times R}$, $1 \leq n \leq N$, order regularization parameter $\mathbf{b} \in \mathbb{R}^{d \times 1}$, and regularization constant $\alpha \in \mathbb{R}$

Output: Penalty $P \in \mathbb{R}$

begin

  $\mathbf{B} = [\mathbf{b}, \ldots, \mathbf{b}]^T \in \mathbb{R}^{d \times R}$

  $\mathbf{P} = [\mathbf{1}, \ldots, \mathbf{1}]^T \in \mathbb{R}^{R \times R}$ // Initialize all-ones matrix

  for $n = 1, \ldots, N$ **do**

    $\mathbf{P} \leftarrow \mathbf{P} \circledast \left(\left(\mathbf{A}^{(n)} \circledast \mathbf{B} \right)^T \left(\mathbf{A}^{(n)} \circledast \mathbf{B} \right) \right)$

  end

  $P = \alpha * \mathrm{sum}(\mathbf{P})$ // Sum entries of $\mathbf{P}$

end

$\mathcal{O}(NR^2 d)$

D Algorithms for learning (Regularization)

Derivative of penalty wrt to factor matrices

$$\frac{\partial P}{\partial \mathbf{A}^{(n)}} = 2\mathbf{B} \circledast \left(\mathbf{Y}^{(n)} \left(\mathop{\circledast}_{\substack{k=1 \\ k\neq n}}^{N} \mathbf{Y}^{(k)T}\mathbf{Y}^{(k)} \right) \right)$$

$\mathcal{O}(NRd + NR^2)$

Algorithm 4: Partial Derivative of Order Regularization Penalty

---

Input: Factor matrices $\mathbf{A}^{(n)} \in \mathbb{R}^{d\times R}$, $1 \leq n \leq N$, order regularization parameter $\mathbf{b} \in \mathbb{R}^{d\times 1}$, and regularization coefficient $\alpha \in \mathbb{R}$

Output: Partial derivative $\frac{\partial P}{\partial \mathbf{A}^{(n)}} \in \mathbb{R}^{d\times R}$ for $1 \leq n \leq N$

begin

  $\mathbf{B} = [\mathbf{b}, \ldots, \mathbf{b}]^T \in \mathbb{R}^{d\times R}$

  $\mathbf{P} = [\mathbf{1}, \ldots, \mathbf{1}]^T \in \mathbb{R}^{R\times R}$ // Initialize all-ones matrix

  for $n = 1, \ldots, N$ **do**

    $\mathbf{P} \leftarrow \mathbf{P} \circledast \left(\left(\mathbf{A}^{(n)} \circledast \mathbf{B}\right)^T \left(\mathbf{A}^{(n)} \circledast \mathbf{B}\right) \right)$ // Store Hadamard products

  end

  for $n = 1, \ldots, N$ **do**

    $\mathbf{D} = \mathbf{P} \oslash \left(\left(\mathbf{A}^{(n)} \circledast \mathbf{B}\right)^T \left(\mathbf{A}^{(n)} \circledast \mathbf{B}\right) \right) \in \mathbb{R}^{R\times R}$ // Divide element-wise

    $\frac{\partial P}{\partial \mathbf{A}^{(n)}} = 2\alpha \mathbf{B} \circledast \left(\left(\mathbf{A}^{(n)} \circledast \mathbf{B}\right) \mathbf{D} \right)$

  end

end

E Feature mapping

$$\phi_d(x_n) = [1, x_n, x_n^2, \ldots, x_n^{(d-1)}]^T$$

(contains inherently solution for linear problem)

$$\hat{\phi}_d(x_n) = \frac{1}{\sqrt{\sum_{k=0}^{d-1} x_n^{2k}}}[1, x_n, x_n^2, \ldots, x_n^{(d-1)}]^T$$

$$\phi(x_n) = \left[1, \mathbf{v}_n^T\right]^T \qquad \mathbf{v}_n \in \mathbb{R}^{K_n}$$

one-hot-encoded representation of the n-th categorical feature, and Kn is the number of values that the feature can assume

$$\phi(x_n) = \left[1, \psi^{(1)}(x_n), \ldots, \psi^{(d-1)}(x_n)\right]^T$$

this form enables new initialization procedure

F Model initialization of weights

- <u>Classique</u> : initialize the factor matrices is to use independent zero-mean Gaussian noise, with a tunable standard deviation
- <u>This paper:</u>

$$\phi(x_n) = \left[1, \psi^{(1)}(x_n), \ldots, \psi^{(d-1)}(x_n)\right]^T$$

$$a_{1,r}^{(n)} = \begin{cases} \frac{b}{N}, & \text{for } r = n \\ 1, & \text{for } 1 \le r \le N, r \neq n \ , \\ 0 & \text{otherwise} \end{cases}$$

$$a_{j,r}^{(n)} = \begin{cases} w_{r,j-1}, & \text{for } r = n, j = 2, \ldots, d \\ 0 & \text{otherwise} \end{cases}$$

$w_{n,j}$ weight n-th feature of $\psi^{(j)}$

<u>Bias term :</u>

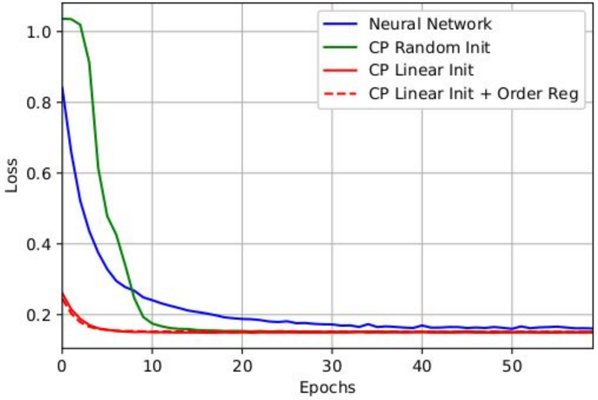
$$\left(\mathop{\circledast}_{k=1}^{N} \hat{\mathbf{a}}_1^{(k)}\right) \mathbf{1} = \Big[\underbrace{\tfrac{b}{N}, \ldots, \tfrac{b}{N}}_{N \text{ times}}, 0, \ldots, 0\Big] \mathbf{1} = b$$

<u>The coefficient for</u> $\psi_j(x_n)$

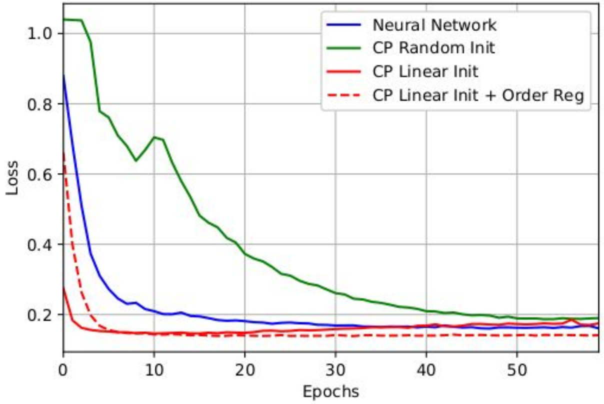
$$\left(\mathop{\circledast}_{\substack{k=1 \\ k \neq n}}^{N} \hat{\mathbf{a}}_1^{(k)} \circledast \hat{\mathbf{a}}_{j+1}^{(n)}\right) \mathbf{1} = w_{n,j}$$

CP-based model produces the same predictions as the linear model

G Experiments ....

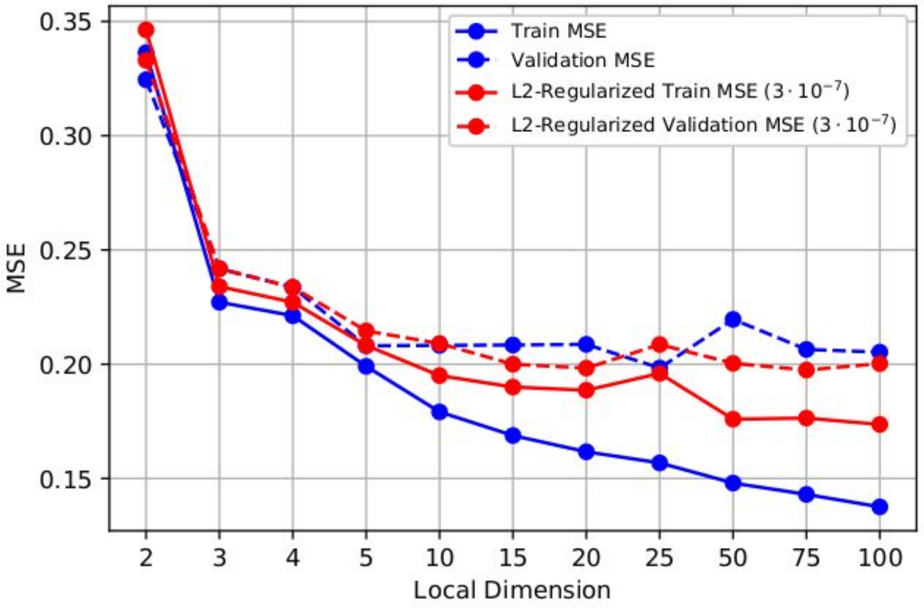


d=2 and R=7



d = 4 and R = 30

G Experiments ....



H Bibliography

- Supervised Learning for Non-Sequential Data with the Canonical Polyadic Decomposition https://arxiv.org/pdf/2001.10109.pdf
- CP decomposition https://en.wikipedia.org/wiki/Tensor_rank_decomposition