ONLINE INCREMENTAL HIGHER-ORDER PARTIAL LEAST SQUARES REGRESSION FOR FAST RECONSTRUCTION OF MOTION TRAJECTORIES FROM TENSOR STREAMS

Ming Hou, Brahim Chaib-draa

Laval University, Canada

ABSTRACT

The higher-order partial least squares (HOPLS) is considered as the state-of-the-art tensor-variate regression modeling for predicting a tensor response from a tensor input. However, the standard HOPLS can quickly become computationally prohibitive or merely impossible, especially when huge and time-evolving tensorial streams arrive over time in dynamic application environments. In this paper, we present a computationally efficient online tensor regression algorithm, namely incremental higher-order partial least squares (IHOPLS), for adapting HOPLS to the setting of infinite time-dependent tensor streams. By incrementally clustering the projected latent variables in latent space and summarizing the previous data, IHOPLS is able to recursively update the projection matrices and core tensors over time, resulting in greatly reduced costs in terms of both memory and running time while maintaining high prediction accuracy. To show the effectiveness and scalability of our approach for large databases, we apply IHOPLS to two real-life applications as reconstruction of 3D motion trajectories from video and ECoG streaming signals.

Index Terms— Tensor, Online Tensor-variate Regression, Higher-order Partial Least Squares, Motion Trajectory

1. INTRODUCTION

Recently, tensor-variate regression approaches have emerged as very promising tools for large-scale high-order structured data and have been gaining increasing momentum in many fields, including computer vision [1], neuroimaging data analysis [2, 3, 4] and signal processing [5, 6, 7]. Among them, the N-way partial least squares (NPLS) [8] performs a simultaneous canonical/parallel factor analysis (CP) decomposition [9] of independent and dependent tensorial data into rank-one tensors, ensuring that the latent variables have the maximum pairwise covariance. To overcome the inferior fitness ability and slow convergence of NPLS, Zhao [5] proposed a powerful generalized multilinear framework, called higher-order partial least squares (HOPLS). It conducts a multilinear projection of both independent and dependent tensors onto a new latent space based on the orthogonal block Tucker decomposition [10, 11], providing enhanced predictability with optimal balance between fitness and model complexity. Nevertheless, both NPLS and HOPLS are offline batch methods that require all the input to be available in advance, which are not well suited for the time-evolving and time-critical applications in dynamic environments. Even though we can apply above batch methods to the entire data every time that a new tensor pair arrives, the computational and storage cost will be prohibitively expensive or impossible owing to the unlimited bound of time series of tensor steams. For this purpose, the recursive N-way partial least squares (RNPLS) [6] was introduced to take into account the time-dependent changes by combining the recursive calculation scheme of recursive partial least squares (RPLS) [12] with the multiway data representation of NPLS. However, similar to NPLS, RNPLS is developed based on the CP model that needs to be solved by an alternating least square (ALS) style approach, which is known to be a suboptimal procedure with slow convergence rate [5]. Additionaly, RNPLS also inherits the limitation of poor fitness ability from NPLS with an inadequate predictability.

In this paper, we present a computationally efficient online tensor-variate regression algorithm, named incremental higher-order partial least square (IHOPLS), which extends the standard HOPLS to the setting of infinite time-dependent and time-critical tensor streams. Compared to HOPLS, IHOPLS aims to track the time-dependent changes and recursively update the projection factors over time via an efficient incremental clustering strategy for the latent variables. In this way, the computational burden for very large-scale data can be greatly reduced and remains in a low approximately constant level. Moreover, benefiting from the advantages of HOPLS over NPLS model, IHOPLS is much faster and exhibits a better predictive ability than RNPLS. Experimental results of the real-world applications, namely reconstruction of 3D motion trajectories from video and ECoG streams, demonstrate the effectiveness and efficiency of our approach.

2. HIGHER-ORDER PARTIAL LEAST SQUARES (HOPLS) REGRESSION

Generally speaking, HOPLS [5] is a generalized multilinear tensor regression model whose objective is to predict a dependent tensor $\mathcal{Y}$ from an independent tensor $\mathcal{X}$ through the extraction of a small number of common latent variables followed by a regression step against them. The principle behind

HOPLS is to sequentially conduct a set of joint block Tucker decompositions of $\mathcal{X}$ and $\mathcal{Y}$ with constraint that the extracted latent variables capture the maximum covariance between $\mathcal{X}$ and $\mathcal{Y}$. Specifically, suppose we have a $(M+1)$th-order independent tensor $\mathcal{X} \in \mathbb{R}^{N \times I_1 \times \cdots \times I_M}$ and a $(L+1)$th-order dependent tensor $\mathcal{Y} \in \mathbb{R}^{N \times J_1 \times \cdots \times J_L}$ which can be obtained by concatenating N pairs of observations $\{(\mathcal{X}^{(n)}, \mathcal{Y}^{(n)})\}_{n=1}^{N}$ that couple in the first mode with equal number of samples. Unlike NPLS, HOPLS decomposes $\mathcal{X}$ into a sum of *rank*-$(1, H_1, ..., H_M)$ Tucker blocks and $\mathcal{Y}$ into a sum of *rank*-$(1, K_1, ..., K_L)$ Tucker blocks, respectively. The model reads

$$\mathcal{X} = \sum_{r=1}^{R} \mathcal{G}_r^{\mathcal{X}} \times_1 \mathbf{t}_r \times_2 \mathbf{P}_r^{(1)} \times \cdots \times_{M+1} \mathbf{P}_r^{(M)} + \epsilon_{\mathcal{X}}, \quad (1)$$

$$\mathcal{Y} = \sum_{r=1}^{R} \mathcal{G}_r^{\mathcal{Y}} \times_1 \mathbf{t}_r \times_2 \mathbf{Q}_r^{(1)} \times \cdots \times_{L+1} \mathbf{Q}_r^{(L)} + \epsilon_{\mathcal{Y}}, \quad (2)$$

where R denotes the number of latent vectors and $\mathbf{t}_r \in \mathbb{R}^N$ corresponds to the rth latent column vector. $\mathcal{G}_r^{\mathcal{X}} \in \mathbb{R}^{1 \times H_1 \times \cdots \times H_M}$ and $\mathcal{G}_r^{\mathcal{Y}} \in \mathbb{R}^{1 \times K_1 \times \cdots \times K_L}$ represent the rth core tensors. $\{\mathbf{P}_r^{(m)}\}_{m=1}^{M} \in \mathbb{R}^{I_m \times H_m}$ and $\{\mathbf{Q}_r^{(l)}\}_{l=1}^{L} \in \mathbb{R}^{J_l \times K_l}$ are the respective rth projection matrices.

The algorithm for solving HOPLS [5] exploits a deflation procedure to sequentially extract R latent vectors. Note that HOPLS uses a closed-form HOSVD [10] solution on the cross-variance tensor to estimate the model parameters, which is more computationally efficient than an iterative NPLS.

3. INCREMENTAL HIGHER-ORDER PARTIAL LEAST SQUARES (IHOPLS) REGRESSION

In order to deal with infinite time-evolving tensor streams and adapt HOPLS to online setting, our strategy aims to perform an incremental clustering in the projected latent subspace and keep track of the summary of previous data in terms of an internal data representation using the clustered latent variables and other model parameters. Because the latent variables are actually the compressed version of data in a low dimensional space, clustering on latent variables can be very efficient and alleviate the curse of dimensionality problem [13]. By doing so, the standard HOPLS can be efficiently applied to the small-scale reconstructed tensors to update model parameters each time a new tensor pair arrives, resulting in much less and roughly constant processing time. The whole mechanism for incrementally updating all the model parameters is summarized in Algorithm 1. Similar to RNPLS, IHOPLS is a sequential algorithm that repeatedly receives and processes a new coming tensor pair over time. More specifically, for a new tensor pair $\{\mathcal{X}_{new} \in \mathbb{R}^{1 \times I_1 \times \cdots \times I_M}, \mathcal{Y}_{new} \in \mathbb{R}^{1 \times J_1 \times \cdots \times J_L}\}$, we first reconstruct the internal tensor representation $\{\tilde{\mathcal{X}} \in \mathbb{R}^{N_{clus} \times I_1 \times \cdots \times I_M}, \tilde{\mathcal{Y}} \in \mathbb{R}^{N_{clus} \times J_1 \times \cdots \times J_L}\}$ from the previous model parameters to summarize all the old data and approximate the current modeling state (Line 3).

Algorithm 1 Incremental Higher-order Partial Least Squares (IHOPLS)

1: **Input:** all the old model parameters $\tilde{\mathbf{T}} = [\tilde{\mathbf{t}}_1, ..., \tilde{\mathbf{t}}_R]$, $\{\{\tilde{\mathbf{P}}_r^{(m)}\}_{m=1}^{M}, \{\tilde{\mathbf{Q}}_r^{(l)}\}_{l=1}^{L}\}_{r=1}^{R}$ and $\{\tilde{\mathcal{G}}_r^{\mathcal{X}}, \tilde{\mathcal{G}}_r^{\mathcal{Y}}\}_{r=1}^{R}$, new tensor data pair $\{\mathcal{X}_{new} \in \mathbb{R}^{1 \times I_1 \times \cdots \times I_M}, \mathcal{Y}_{new} \in \mathbb{R}^{1 \times J_1 \times \cdots \times J_L}\}$
2: **Output:** all the new model parameters $\mathbf{T} = [\mathbf{t}_1, ..., \mathbf{t}_R]$, $\{\{\mathbf{P}_r^{(m)}\}_{m=1}^{M}, \{\mathbf{Q}_r^{(l)}\}_{l=1}^{L}\}_{r=1}^{R}$ and $\{\mathcal{G}_r^{\mathcal{X}}, \mathcal{G}_r^{\mathcal{Y}}\}_{r=1}^{R}$
3: Reconstruct the old internal tensor representation $\{\tilde{\mathcal{X}} \in \mathbb{R}^{N_{clus} \times I_1 \times \cdots \times I_M}, \tilde{\mathcal{Y}} \in \mathbb{R}^{N_{clus} \times J_1 \times \cdots \times J_L}\}$ from the old latent matrix $\tilde{\mathbf{T}}$, the loadings $\{\{\tilde{\mathbf{P}}_r^{(m)}\}_{m=1}^{M}, \{\tilde{\mathbf{Q}}_r^{(l)}\}_{l=1}^{L}\}_{r=1}^{R}$ and the core tensors $\{\tilde{\mathcal{G}}_r^{\mathcal{X}}, \tilde{\mathcal{G}}_r^{\mathcal{Y}}\}_{r=1}^{R}$ according to the equations (1) and (2):

$$\tilde{\mathcal{X}} = \sum_{r=1}^{R} \tilde{\mathcal{G}}_r^{\mathcal{X}} \times_1 \tilde{\mathbf{t}}_r \times_2 \tilde{\mathbf{P}}_r^{(1)} \times \cdots \times_{M+1} \tilde{\mathbf{P}}_r^{(M)}$$

$$\tilde{\mathcal{Y}} = \sum_{r=1}^{R} \tilde{\mathcal{G}}_r^{\mathcal{Y}} \times_1 \tilde{\mathbf{t}}_r \times_2 \tilde{\mathbf{Q}}_r^{(1)} \times \cdots \times_{L+1} \tilde{\mathbf{Q}}_r^{(L)}$$

4: Concatenate the $\{\mathcal{X}_{new}, \mathcal{Y}_{new}\}$ to $\{\tilde{\mathcal{X}}, \tilde{\mathcal{Y}}\}$ in the first mode to generate tensor pair:

$$\mathcal{X} = \begin{bmatrix} \tilde{\mathcal{X}} \\ \mathcal{X}_{new} \end{bmatrix} \in \mathbb{R}^{N_{clus}+1 \times I_1 \times \cdots \times I_M}$$

$$\mathcal{Y} = \begin{bmatrix} \tilde{\mathcal{Y}} \\ \mathcal{Y}_{new} \end{bmatrix} \in \mathbb{R}^{N_{clus}+1 \times J_1 \times \cdots \times J_L}$$

5: Apply the standard HOPLS to $\{\mathcal{X}, \mathcal{Y}\}$ to find newly updated model parameters $\mathbf{T}$, $\{\{\mathbf{P}_r^{(m)}\}_{m=1}^{M}, \{\mathbf{Q}_r^{(l)}\}_{l=1}^{L}\}_{r=1}^{R}$ and $\{\mathcal{G}_r^{\mathcal{X}}, \mathcal{G}_r^{\mathcal{Y}}\}_{r=1}^{R}$
6: **for** $k = 1$ to number of clusters N_{clus} **do**
7: Compute the similarity between the projected new data $\mathbf{T}(N_{clus}+1)$ from the $(N_{clus}+1)$th row vector of $\mathbf{T}$ and the projected cluster center in the kth row vector of $\mathbf{T}$:

$$w_k = < \frac{\mathbf{T}(N_{clus}+1)}{||\mathbf{T}(N_{clus}+1)||}, \frac{\mathbf{T}(k)}{||\mathbf{T}(k)||} >$$

8: **end for**
9: Choose the nearest cluster center v: $sim_v = \max(w_k)$
10: **if** $sim_v > w_{gen}$ **then**
11: Update the number of data points in cluster v: $C_v = C_v + 1$
12: Set the timestamp of cluster v: $TS_v = 1$
13: Update the timestamps of other clusters v': $TS_{v'} = TS_{v'} + 1$
14: Update the cluster center row vector:

$$\mathbf{T}(v) = \frac{(C_v - 1) * \mathbf{T}(v) + 1 * \mathbf{T}(N_{clus}+1)}{C_v}$$

15: Remove the last row vector $\mathbf{T}(N_{clus}+1)$ from $\mathbf{T}$
16: **else**
17: Set the number of data points in cluster $(N_{clus}+1)$: $C_{N_{clus}+1} = C_{N_{clus}+1} + 1$
18: Set the timestamp of cluster $(N_{clus}+1)$: $TS_{N_{clus}+1} = 1$
19: Update the timestamps of other clusters v': $TS_{v'} = TS_{v'} + 1$
20: Update the number of row vectors in $\mathbf{T}$: $N_{clus} = N_{clus} + 1$
21: **if** maximum number of clusters N_{max} is reached **then**
22: Remove the cluster center represented by the ith row vector with minimum ratio of C_i/TS_i, $i = 1, ..., N_{clus}$
23: **end if**
24: **end if**

Note that the latent matrix $\tilde{\mathbf{T}}$ represents all clusters and each row vector in $\tilde{\mathbf{T}}$ stands for a projected cluster center or a projected data sample in latent subspace, while all the projection matrices and core tensors can be regarded as tensorial bases onto which the tensorial data projects. Thus, the number of row vectors N_{clus} is in fact the number of clusters. We then concatenate the new tensor pair $\{\mathcal{X}_{new}, \mathcal{Y}_{new}\}$ to $\{\tilde{\mathcal{X}}, \tilde{\mathcal{Y}}\}$ along the first mode to generate tensor pair $\{\mathcal{X} \in \mathbb{R}^{N_{clus}+1 \times I_1 \times \cdots \times I_M}, \mathcal{Y} \in \mathbb{R}^{N_{clus}+1 \times J_1 \times \cdots \times J_L}\}$ with sample size $N_{clus}+1$ (Line 4). Subsequently, we apply the standard HOPLS to $\{\mathcal{X}, \mathcal{Y}\}$ to extract new model parameters by incorporating the information contained in $\{\mathcal{X}_{new}, \mathcal{Y}_{new}\}$ into the model (Line 5). Having extracted all parameters, we next allocate the projected new data, represented by the extra row vector $\mathbf{T}(N_{clus}+1)$ in latent matrix $\mathbf{T}$, into a collection of clusters that are expressed by the cluster centers using row vectors from 1 to N_{clus}. To this end, we choose the nearest cluster v whose center $\mathbf{T}(v)$ has the highest similarity measure sim_v with row vector $\mathbf{T}(N_{clus}+1)$ (Line 6-9). If sim_v is greater than a predefined threshold w_{gen}, then we insert this new projected data into that cluster v, and update the cluster center $\mathbf{T}(v)$ accordingly (Line 10-15). Otherwise, we define $\mathbf{T}(N_{clus}+1)$ to be the new cluster center and increase the number of clusters by one (Line 17-20). Meanwhile, we keep tracking both the number of data points and timestamps for each cluster i whose ratio C_i/TS_i indicates the relative importance of that cluster in participating in summarizing the current state of the model. This implies that the cluster with more data points and more recent timestamps is more important than those far in the past with less data points. if the number of clusters exceeds the maximum value, then the cluster with minimum ratio should be removed from $\mathbf{T}$ (Line 21-23).

Some comments are now appropriate. We should mention that our approach is specially designed for the online tensor stream setting where it is not feasible to search for the globally optimal model parameters since the new observations keep arriving. Rather than fixing the projection matrices and core tensors, we incrementally update them by assuming that the characteristics of the data might change over time. In practice, the number N_{clus} turns out to be very small such that the HOPLS can be executed efficiently owing to that the HOPLS is particularly suitable for small sample size [5]. IHOPLS only needs dozens or tens of samples to extract the initial model parameters. Moreover, our method handles the time dependency by combining an appropriate maximum cluster number with an eliminating stategy based on an importance ratio.

For HOPLS, the overall computational cost is dominated by $\mathcal{O}(\max(NI^{M+L}, I^{M+L+1}))$ whose two terms correspond to establishing and decomposing the cross-covariance tensor, where I is the maximum index of all modes and is not large in practice. Evidently, as N increases over time, the first term will eventually surpass second term owing to the unbounded data streams. Since the clusters number N_{clus} is bounded and small, the complexity of IHOPLS, on the other hand, is mainly affected by a constant $\mathcal{O}(\max(N_{clus}I^{M+L}, I^{M+L+1}))$.

4. EXPERIMENTAL EVALUATION

We compared our approach with RNPLS [6] as well as HOPLS [5] in an online fashion, that is, first making a prediction for the new input and then updating the model using the new ground truth. The performance was quantitatively evaluated in terms of the root mean squares of prediction (RMSEP) [14] and the Q index which is $Q = 1 - \|\mathcal{Y} - \hat{\mathcal{Y}}\|_F / \|\mathcal{Y}\|_F$, where $\hat{\mathcal{Y}}$ is the prediction of $\mathcal{Y}$. The optimal hyperparameters of RNPLS were selected by cross-validation on the first $1/3$ data sequence, including the forgetting factor γ. In addition, the convergence criterion and the maximum iteration number for estimating the loadings were set to 10^{-5} and 40. On the other hand, we empirically fixed number of latent vectors R and number of loading λ for both HOPLS and IHOPLS. We manually tuned the partitioning threshold parameter w_{gen} and the maximum cluster number N_{max} whose combination actually controls the number of clusters responsible for the recursive update. We repeatedly run the experiment 10 times. Unlike RNPLS, data are not centered for HOPLS and IHOPLS.

4.1. Utrecht Multi-Person Motion Database

The first experiment was carried out on the Multi-Person Motion (UMPM) benchmark [15] that contains temporally synchronized video sequences and human motion capture data. In our experiment, the input was an intensity image sequence from the front camera with a downsized resolution of 32×24 pixels, taking form of a 3rd-order predictor tensor (i.e., frames $\times$ width $\times$ height). To test on large-sized data, we concatenated videos from four scenarios, including "triangle", "grab", "chair" and "table", into one big sequence with total 5250 frames at 25 fps. The corresponding output containing 3D positions of 37 reflective markers can be represented as a 3rd-order tensor (i.e., samples $\times$ 3D positions $\times$ markers).

The averaged predictive performance as well as the learning time of the IHOPLS, HOPLS and RNPLS are summarized in Table 1. As we see, our approach significantly outperformed HOPLS and the best of RNPLS in prediction accuracy with respect to both high and low frame rate situation, implying the usefulness of IHOPLS for adjacent observations with relatively less overlapped features. In particular, the averaged improvement by IHOPLS over HOPLS and RNPLS at 5fps when using 4 latent vectors were significant 0.25, 0.08 in terms of Q, and 199.8, 66.1 in terms of RMSEP. This indicates the capability of IHOPLS in capturing local variation in dynamic streams. In the meantime, the averaged total learning time of IHOPLS, namely 505.3 seconds, is extremely faster than that 6901.3 seconds of RNPLS. Fig.1 illustrates the learning time and prediction error versus elapsed frames. Overall, the CPU cost for both IHOPLS and RNPLS exhibited a constant trend, while cost of offline HOPLS keeps in-

Dataset	Size and Frequency	Method	Rank and Hyperparameter	Q	RMSEP	Total Time (s)
UMPM	1050, 5fps	IHOPLS	$R = 4, \lambda = 2, w_{gen} = 0.2, N_{max} = 20$	**0.8240**	**149.6**	**505.3**
			$R = 8, \lambda = 4, w_{gen} = 0.2, N_{max} = 20$	**0.8274**	**145.2**	**902.7**
		HOPLS	$R = 4, \lambda = 2$	0.5700	349.4	812.5
			$R = 8, \lambda = 4$	0.5864	332.4	1471.8
		RNPLS	$F = 4, \gamma = 0.8$	0.7433	215.7	6901.3
	5250, 25fps	IHOPLS	$R = 4, \lambda = 2, w_{gen} = 0.2, N_{max} = 20$	**0.9645**	**29.5**	**2316.3**
			$R = 8, \lambda = 4, w_{gen} = 0.2, N_{max} = 20$	**0.9664**	**28.5**	**4775.2**
		HOPLS	$R = 4, \lambda = 2$	0.5778	342.8	21145.3
			$R = 8, \lambda = 4$	0.6008	320.7	36396.6
		RNPLS	$F = 4, \gamma = 0.6$	0.9337	50.4	19923.0
ECoG	900, 1hz	IHOPLS	$R = 8, \lambda = 8, w_{gen} = 0.4, N_{max} = 100$	0.6962	34.4	**665.2**
		HOPLS	$R = 8, \lambda = 8$	0.7003	33.7	1137.0
		RNPLS	$F = 5, \gamma = 1$	**0.7036**	**33.2**	1647.8
	18000, 20hz	IHOPLS	$R = 8, \lambda = 8, w_{gen} = 0.4, N_{max} = 10$	**0.9204**	**9.0**	**17357.3**
		HOPLS	$R = 8, \lambda = 8$	0.7214	30.3	586710.0
		RNPLS	$F = 5, \gamma = 0.8$	0.9011	11.0	160113.4

Table 1: Performance comparison of IHOPLS, HOPLS and RNPLS for the averaged Q, RMSEP and total learning time.

creasing over time. In contrast to HOPLS and RNPLS, IHOPLS was highly computationally efficient no matter which frame frequency was used. Note carefully that, for both cases, the RMSEP of IHOPLS was relatively large at the beginning learning period but was quickly reduced to and reamained in a low level as the test went on, except for a few bursts or bumps that correspond to the points when we concatenated different video sequences or significant changes of motions in the input streams, which is more obvious in the case of 5250 samples.

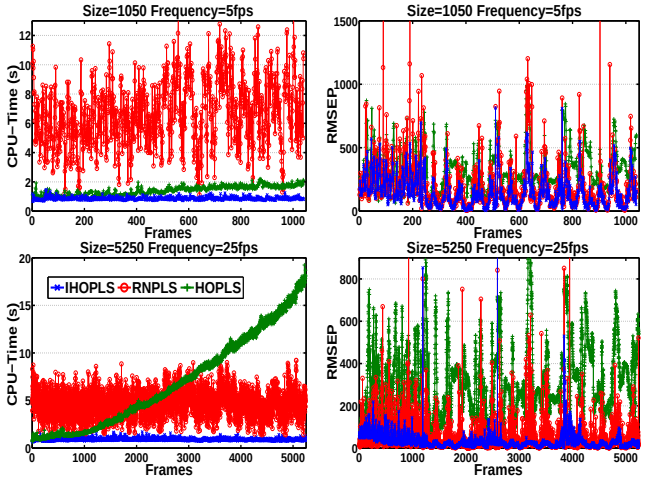

Fig. 1: CPU cost and prediction error of three methods over time, $R = 8$ and $\lambda = 4$ for IHOPLS and HOPLS on UMPM.

4.2. Neurotycho Electrocorticography Database

We also evaluated our approach on an application of reconstruction of limb movements from monkey's brain signals using Neurotycho food tracking Electrocorticography (ECoG) dataset [16]. Particularly, the wavelet transformed ECoG signal, a 4th-order tensor $\mathcal{X} \in R^{N \times 10 \times 10 \times 32}$ (i.e., samples $\times$ time $\times$ frequency $\times$ channels), was employed as the input. The output was a 3rd-order tensor of 3D movement distances of the monkey's limb on different markers. In Table 1, our approach showed a remarkable advantage regarding computational burden over other methods. We also observed that IHOPLS achieved the best accuracy in the case of 20Hz while remaining competitive with HOPLS and RNPLS in the 1Hz case where the successive samples were selected with non-overlapped features. Note that the accuracy was improved in the Fig.2 as we increased the maximum cluster number N_{max} from 10 to 400 for non-overlapped feature case, reflecting the fact that we need more clusters to characterize the variation of data. On the contrary, we may need fewer clusters for highly-overlapped samples so as to obtain a better predictive ability.

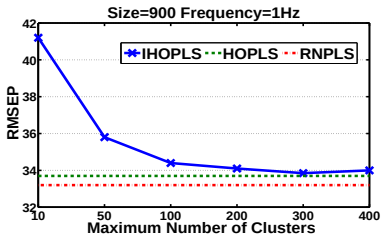

Fig. 2: Errors versus N_{max} for $R = 8$ $\lambda = 8$ $w_{gen} = 0.4$.

5. CONCLUSION

Our new online IHOPLS algorithm efficiently adapts the standard HOPLS to the setting of infinite time-dependent tensor streams. Our approach allows recursively updating the regression coefficients by incrementally clustering the projected latent variables in latent space, leading to a constant computational load over time. We have validated the effectiveness and scalability of IHOPLS on two different real-life tensor regression applications with very large-scale data.

6. REFERENCES

[1] Qibin Zhao, Guoxu Zhou, Liqing Zhang, and Andrzej Cichocki, "Tensor-variate gaussian processes regression and its application to video surveillance," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 1265–1269.

[2] Hua Zhou, Lexin Li, and Hongtu Zhu, "Tensor regression with applications in neuroimaging data analysis," *Journal of the American Statistical Association*, vol. 108, no. 502, pp. 540–552, 2013.

[3] Xiaoshan Li, Hua Zhou, and Lexin Li, "Tucker tensor regression and neuroimaging analysis," *arXiv preprint arXiv:1304.5637*, 2013.

[4] Ming Hou and Brahim Chaib-draa, "Hierarchical tucker tensor regression: Application to brain imaging data analysis," in *Image Processing (ICIP), 2015 IEEE International Conference on*. IEEE, 2015.

[5] Qibin Zhao, Cesar F Caiafa, Danilo P Mandic, Zenas C Chao, Yasuo Nagasaka, Naotaka Fujii, Liqing Zhang, and Andrzej Cichocki, "Higher order partial least squares (HOPLS): a generalized multilinear regression method," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 7, pp. 1660–1673, 2013.

[6] Andrey Eliseyev and Tetiana Aksenova, "Recursive n-way partial least squares for brain-computer interface," *PloS one*, vol. 8, no. 7, pp. e69962, 2013.

[7] Ming Hou, Yali Wang, and Brahim Chaib-draa, "Online local gaussian process for tensor-variate regression: Application to fast reconstruction of limb movements from brain signal," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015.

[8] Rasmus Bro, "Multiway calidration. multilinear PLS," *Journal of chemometrics*, vol. 10, pp. 47–61, 1996.

[9] Richard A Harshman, "Foundations of the parafac procedure: Models and conditions for an "explanatory" multi-modal factor analysis," 1970.

[10] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle, "A multilinear singular value decomposition," *SIAM journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1253–1278, 2000.

[11] Ledyard R Tucker, "Implications of factor analysis of three-way matrices for measurement of change," *Problems in measuring change*, vol. 15, 1963.

[12] S Joe Qin, "Recursive pls algorithms for adaptive data modeling," *Computers & Chemical Engineering*, vol. 22, no. 4, pp. 503–514, 1998.

[13] Thomas Oommen, Debasmita Misra, Navin KC Twarakavi, Anupma Prakash, Bhaskar Sahoo, and Sukumar Bandopadhyay, "An objective analysis of support vector machine based classification for remote sensing," *Mathematical geosciences*, vol. 40, no. 4, pp. 409–424, 2008.

[14] Hyunsoo Kim, Jeff X Zhou, Herbert C Morse III, and Haesun Park, "A three-stage framework for gene expression data analysis by L1-norm support vector regression," *International journal of bioinformatics research and applications*, vol. 1, no. 1, pp. 51–62, 2005.

[15] NP Van Der Aa, X Luo, GJ Giezeman, RT Tan, and R-C Veltkamp, "Utrecht multi-person motion (UMPM) benchmark," Tech. Rep., Citeseer, 2011.

[16] Zenas C Chao, Yasuo Nagasaka, and Naotaka Fujii, "Long-term asynchronous decoding of arm motion using electrocorticographic signals in monkeys," *Frontiers in neuroengineering*, vol. 3, 2010.