ONLINE LOCAL GAUSSIAN PROCESS FOR TENSOR-VARIATE REGRESSION: APPLICATION TO FAST RECONSTRUCTION OF LIMB MOVEMENTS FROM BRAIN SIGNAL

Ming Hou, Yali Wang, Brahim Chaib-draa

Laval University, Canada

ABSTRACT

Tensor-variate regression approaches have been spotlighted over the past years, due to the fact that many challenging regression tasks in the real world involve in high-order tensorial data. However, these approaches are often computationally prohibitive, which limits the predictive performance for large data sets. In this paper, we propose a computationally-efficient tensor-variate regression approach in which the latent function is flexibly modeled by using online local Gaussian process (OLGP). By doing so, the large data set is efficiently processed by constructing a number of small-sized GP experts in an online fashion. Furthermore, we introduce two efficient search strategies to find local GP experts to make accurate predictions with a Gaussian mixture representation. Finally, we evaluate our approach on a real-life regression task, reconstruction of limb movements from brain signal, to show its effectiveness and scalability for large data sets.

Index Terms— Tensor, Tensor-Variate Regression, Online Local Gaussian Process.

1. INTRODUCTION

Over the past years, tensor-variate regression approaches have been investigated due to the fact that the data sets in many real-life regression tasks are often associated with high-order tensorial structures [1, 2, 3, 4, 5]. Among these approaches, tensor-variate Gaussian process regression (tensor GP) proposed in [3] is promising because, instead of linear assumptions in [1, 2, 4, 5], tensor GP can flexibly model the nonlinearity of the tensorial data by using the powerful Bayesian nonparametric Gaussian process (GP). However, the computation load caused by GP, $O(N^3)$, often makes tensor GP computationally prohibitive in practice, since the number of the training points N is often required to be quite large in the context of tensor-variate regression applications (where the data points in high-order tensor space tend to be sparse due to the curse of dimensionality [6]).

In this paper, we propose a computationally-efficient tensor GP framework by introducing online local Gaussian process (OLGP) for tensor-variate regression (we denote our approach as tensor OLGP). Compared to tensor GP in [3], our tensor OLGP takes advantage of OLGP by assigning the data points to a number of the small-sized GP experts in an online fashion, thus naturally reducing the computation burden for large data sets. We complete this by two efficient search strategies, namely input-based and input-output-based search, to find the local GP experts in order to maintain the predictive accuracy. Finally, we demonstrate the effectiveness of our tensor OLGP on a large-scale tensor regression task, i.e., the limb motion reconstruction using brain signal [7].

2. TENSOR GP REGRESSION

In a standard tensor-variate regression task, we are given a training set $\mathcal{D} = \{(\mathcal{X}_n, y_n)\}_{n=1}^N$ where the scalar output $y_n \in \mathbb{R}$ is generated by a nonlinear function $f(\mathcal{X}_n)$ of the D-order tensor input $\mathcal{X}_n \in \mathbb{R}^{I_1 \times \cdots \times I_D}$ with an additive Gaussian noise $\epsilon_n \sim \mathcal{N}(0, \sigma^2)$

$$y_n = f(\mathcal{X}_n) + \epsilon_n. \tag{1}$$

For simplicity, we concatenate all the tensor inputs into a $(D+1)$-order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{N \times I_1 \times \cdots \times I_D}$ and put all the outputs into a vector $\mathbf{y} = [y_1, ..., y_N]^T$.

The tensor GP approach [3] is based on the idea that the latent function in (1) can be modeled by a GP, i.e.,

$$f(\mathcal{X}) \sim \mathcal{GP}(m(\mathcal{X}), k(\mathcal{X}, \mathcal{X}')|\theta), \tag{2}$$

where $m(\mathcal{X})$ is the mean function, $k(\mathcal{X}, \mathcal{X}')$ is the covariance function and θ is the associated hyperparameter vector. In this paper, we follow a standard GP setting in [3, 8] where $m(\mathcal{X}) = 0$. For $k(\mathcal{X}, \mathcal{X}')$, we follow [3] and adopt the following product probabilistic kernel:

$$k(\mathcal{X}, \mathcal{X}') = \alpha^2 \prod_{d=1}^{D} \exp\left(\frac{KL(p(\mathbf{x}|\Omega_d^{\mathcal{X}}) \parallel q(\mathbf{x}'|\Omega_d^{\mathcal{X}'}))}{-2\beta_d^2}\right), \tag{3}$$

where α is the the magnitude hyperparameter, and β_d denotes the d-mode length-scales hyperparameter. The distributions p and q in the Kullback-Leibler (KL) divergence are characterized by the hyperparameter Ω_d (for multivariate Gaussian $\Omega_d = \{\mu_d, \Sigma_d\}$) which can be estimated from the d-mode unfolding matrix $\mathbf{X}_d$ of tensor $\mathcal{X}$ by treating each $\mathbf{X}_d$ as a

generative model with I_d number of variables and $I_1 \times \cdots \times I_{d-1} \times I_{d+1} \times \cdots \times I_D$ number of observations.

The goal of a tensor GP regression aims to infer the predictive distribution of the latent function value $f(\mathcal{X}_*) = f_*$ at a new test point $\mathcal{X}_*$ given the training data $\mathcal{D}$. According to the definition of GP, we can obtain that any finite number of latent function values at the tensorial inputs are Gaussian distributed [8]. Consequentially, the joint distribution

$$p(f_*, \mathbf{y}|\mathcal{X}_*, \boldsymbol{\mathcal{X}}, \theta, \sigma^2) \tag{4}$$

is Gaussian. Moreover, based on the conditional property of Gaussian distribution, the predictive distribution

$$p(f_*|\mathcal{X}_*, \boldsymbol{\mathcal{X}}, \mathbf{y}, \theta, \sigma^2) = \mathcal{N}(m_*, \sigma_*^2) \tag{5}$$

is also Gaussian with

$$m_* = \mathbf{k}_{\mathcal{X}}^T(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{y} \tag{6}$$

$$\sigma_*^2 = k_* - \mathbf{k}_{\mathcal{X}}^T(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{k}_{\mathcal{X}}, \tag{7}$$

where $k_* = k(\mathcal{X}_*, \mathcal{X}_*)$ and $\mathbf{k}_{\mathcal{X}} = k(\mathcal{X}_*, \boldsymbol{\mathcal{X}})$. However, as we mentioned, the computational complexity of GP, $O(N^3)$, often makes tensor GP computationally expensive because N is usually required to be quite large to achieve a reliable result for a tensor-variate regression task [6].

In the following section, we propose a computationally efficient tensor GP framework where a fast data processing mechanism, which is inspired by online local Gaussian process (OLGP) in [9, 10], is designed for tensor-variate regression (tensor OLGP).

3. TENSOR OLGP REGRESSION

In order to deal with large-scale tensor regression tasks, we take advantage of online local GP (OLGP) to present a computationally-efficient tensor OLGP regression approach that consists of the following two stages:

- **Stage 1 (GP Experts Construction)**: Using the covariance function of GP (3) as a similarity measurement to sequentially partition the training data points into a number of small-sized experts.
- **Stage 2 (Local Prediction)**: Finding a fixed-number of local GP experts to make predictions (for given test tensorial inputs) with a Gaussian mixture.

3.1. GP Experts Construction

To achieve the computation efficiency, we propose to use the covariance function of GP (as a similarity measurement) to sequentially allocate the tensorial data into a collection of local experts. The whole mechanism to construct GP experts is shown in Algorithm 1. Specifically, when a new training data pair $\{\mathcal{X}_{new}, y_{new}\}$ arrives, we first calculate the similarity between $\mathcal{X}_{new}$ and the center of each local expert $\mathcal{C}_k$ using the probabilistic tensor kernel (Line 3). Subsequently, we choose the closest expert t whose center has the highest similarity measure sim_t with $\mathcal{X}_{new}$ according to (3) (Line 5). If sim_t is greater than a predefined threshold w_{gen}, then we insert this new data pair into that local expert t, and update the kernel matrix of expert t accordingly (Line 7-12). Otherwise, we defined $\mathcal{X}_{new}$ as the center of new expert $R+1$ (R is the total number of local experts), and thus initialize a new kernel matrix (Line 14-16).

Algorithm 1 GP Experts Construction

1: **Input:** new tensor data pair $\{\mathcal{X}_{new}, y_{new}\}$
2: **for** $k = 1$ to number of local experts R **do**
3: Compute the similarity to the kth expert using probabilistic tensor kernel function (3): $w_k = k(\mathcal{X}_{new}, \mathcal{C}_k)$
4: **end for**
5: Choose the nearest local expert t: $sim_t = \max(w_k)$
6: **if** $sim_t > w_{gen}$ **then**
7: Insert $\{\mathcal{X}_{new}, y_{new}\}$ to the nearest local expert t:
8: $\boldsymbol{\mathcal{X}}_t = [\boldsymbol{\mathcal{X}}_t, \mathcal{X}_{new}]$, $\mathbf{y}_t = [\mathbf{y}_t, y_{new}]$
9: **if** maximum number of data points is reached **then**
10: delete another point by permutation
11: **end if**
12: Update the corresponding kernel matrix $\mathbf{K}_t$ by computing the kernel vector $\mathbf{k}_t(\mathcal{X}_{new}, \boldsymbol{\mathcal{X}}_t)$ for $\mathcal{X}_{new}$
13: **else**
14: Create a new expert:
15: $\mathcal{C}_{R+1} \doteq \mathcal{X}_{new}$, $\boldsymbol{\mathcal{X}}_{R+1} = [\mathcal{X}_{new}]$, $\mathbf{y}_{R+1} = [y_{new}]$
16: Initialize the new kernel matrix $\mathbf{K}_{R+1}$
17: **end if**

In this stage of GP expert construction, we process the large data set into a number of small-sized experts in an online fashion, thus naturally speed up the computation efficiency.

3.2. Local Prediction

After we partition the data sets into a number of small-sized experts, we propose the following two local search strategies for prediction of new test point $\mathcal{X}_*$ in order to further take the tradeoff between accuracy and efficiency into account.

3.2.1. input-based searching strategy

Our first strategy, namely input-based searching strategy, exploits M nearest local experts to make the prediction. These M local experts should have the highest similarities with $\mathcal{X}_*$ among all the local experts according to the kernel function defined in (3). Then the similarity measure $w_k = k(\mathcal{X}_*, \mathcal{C}_k)$ from the input space between the test point $\mathcal{X}_*$ and the expert center $\mathcal{C}_k$ can be used as the weight of local

expert k. Hence, the resulting prediction $\hat{y}_*$ can be formulated as the weighted combination from each local prediction $\bar{y}_k = \mathbf{k}_k(\mathcal{X}_*, \boldsymbol{\mathcal{X}}_k)^T(\mathbf{K}_k + \sigma^2\mathbf{I})^{-1}\mathbf{y}_k$ as follows

$$\hat{y} = \frac{\sum_{k=1}^{M} w_k \bar{y}_k}{\sum_{k=1}^{M} w_k}. \tag{8}$$

3.2.2. *input-output-based searching strategy*

In the regression task the latent function is a mapping between input and output, hence we propose to explore the experts which do not only consider the input space but also the output space. We design the strategy as follows:

Given a test point $\mathcal{X}_*$, in the first step of our second strategy we start by finding its nearest local expert $\mathcal{C}_k \doteq \{\mathcal{X}_{\mathcal{C}_k}, y_{\mathcal{C}_k}\}$ from the input $\mathcal{X}$-space using tensor kernel function defined in (3), where the pair $\{\mathcal{X}_{\mathcal{C}_k}, y_{\mathcal{C}_k}\}$ is some data pair coming from local expert k.

With the nearest local expert $\mathcal{C}_k \doteq \{\mathcal{X}_{\mathcal{C}_k}, y_{\mathcal{C}_k}\}$ in hand, we aim to find all M candidate experts which are closest to $\mathcal{C}_k$ in output y-space in the second step. More specifically, we search for M local experts $\{\mathcal{C}_m \doteq \{\mathcal{X}_{\mathcal{C}_m}, y_{\mathcal{C}_m}\}\}_{m=1}^{M}$ that are being closest to $y_{\mathcal{C}_k}$ in y-space among all the local expert centers. In other words, this step intends to find M smallest Euclidian distances between $y_{\mathcal{C}_m}$ and $y_{\mathcal{C}_k}$. Then, from these already found $\{y_{\mathcal{C}_m}\}_{m=1}^{M}$, we can easily mark their corresponding local experts $\{\mathcal{C}_m\}_{m=1}^{M}$ as the candidates for prediction. Finally, we use the same weight described in the first strategy to combine the local predictions.

3.3. Computational Complexity

Table 1 shows the comparisons of overall computational complexity. In both methods, we have to evaluate the tensor kernel function between any two points to build the kernel matrix. Such evaluation mainly depends on the estimation of hyperparameter Ω_d whose cost is dominated by the term $\mathcal{O}(I^{D+1})$, where $I = \max\{I_1, ..., I_D\}$.

For standard tensor GP, the computational complexity of learning is $\mathcal{O}(N^3)$, plus the cost of establishing the kernel matrix $\mathcal{O}(N^2 I^{D+1})$. The complexity of prediction requires $\mathcal{O}(NI^{D+1})$ to compute the kernel vector $\mathbf{k}_{\mathcal{X}}(\mathcal{X}_*, \boldsymbol{\mathcal{X}})$, where N is the number of training samples.

In contrast, the cost of tensor OLGP for learning includes finding the nearest local expert $\mathcal{O}(NRI^{D+1})$ and updating the kernel matrix of that local expert $\mathcal{O}(NSI^{D+1} + S^3)$, where R is the number of local experts and S denotes the maximum number of data points contained in each local expert. While the computational complexity of prediction arises from finding M nearest neighbours $\mathcal{O}(RI^{D+1})$ and making M local predictions $\mathcal{O}(M(SI^{D+1} + S^2))$. Note that the cost of our method for learning is only linear in N comparing to N^3 in the case of stand GP.

4. EXPERIMENTAL EVALUATION

In this section, we validate our tensor OLGP on a benchmark tensor regression application, i.e., the reconstruction of limb movements from monkey's brain signals, where each input is the preprocessed ECoG signal that is a 3rd order tensor (frequency $\times$ time $\times$ channel), and the output is the movement distance of the monkey's limb on different markers (shoulder, elbow or hand) along each axis (x, y or z) [1].

More specifically, the whole ECoG food tracking task dataset consists of 15 minutes long experiment with motion data sampled at 120Hz. To illustrate the effectiveness of our approach, we first choose a subsegment of the whole dataset starting from the 2nd minute comprising 10000 data pairs. We then conduct the experiments by randomly selecting a training set with size of 5000. The rest 5000 is used as the test set. As for the input, the wavelet transformed ECoG data are down-sampled to 5 channels and can thus be written as $\mathcal{X} \in \mathbb{R}^{5\times5\times5}$ for each sample. In our experiment, we choose the motion data corresponding to the shoulder marker along the x-axis.

To compare with tensor GP, we perform the evaluation of our tensor OLGP by showing the learning performance in terms of accuracy and efficiency. The hyperparameters contained in the probabilistic tensor kernel are set to the same values empirically for all the methods. We also manually tune the partitioning threshold parameter w_{gen} whose value balances the trade-off between accuracy and efficiency of the final performance. Here, w_{gen} is fixed to 0.5 and the R is set to 6. We repeat the experiment 10 times. Table 2 shows the root mean square error (RMSE), negative log likelihood (NLL) as well as the running time of all the approaches. As expected, our two tensor OLGP variants (tensor x-OLGP and tensor xy-OLGP) achieve the best results in terms of running time. The RMSE of our tensor OLGP is competitive to tensor GP and the NLL of our method outperforms tensor GP. Although tensor GP is slightly better in RMSE, the nonstationarity in the signal makes GP with high uncertainty, that is why tensor OLGP tries to capture the local structure to get lower uncertainty with a few nearby local experts. In particular, the training time of tensor OLGP, which is 321.0 seconds, is about 4 times as fast as that 1279.1 seconds of tensor GP.

We also observe that the tensor xy-OLGP is slightly better than tensor x-OLGP in RMSE but somehow worse than tensor x-OLGP in NLL. In another setting, we increase w_{gen} to 0.6 and obtain a similar result with an more obvious gap in RMSE between tensor x-OLGP and tensor xy-OLGP. This may be because the M most nearby local experts found by the second strategy tend to make a consistent prediction with the local expert closest to the test point $\mathcal{X}_*$. We should notice, however, that the found experts may be faraway from the test point in the input space, leading to an uncertain prediction. This somehow explains why the input-based strategy is better

[1] The data set is available at http://neurotycho.org/food-tracking-task.

	Partation + Training	Prediction
tensor GP	$\mathcal{O}(N^2I^{D+1} + N^3)$	$\mathcal{O}(NI^{D+1} + N^2)$
tensor OLGP	$\mathcal{O}(NRI^{D+1} + NSI^{D+1} + S^3)$	$\mathcal{O}(RI^{D+1} + M(SI^{D+1} + S^2))$

Table 1: Computational complexity of tensor GP and tensor OLGP using product probabilistic tensor kernel.

	Method	RMSE	NLL	Running Time (s)	
				Training	Testing
data size=10000, $w_{gen} = 0.5$	tensor GP	3.05 ± 0.16	7.26 ± 0.57	1279.1 ± 9.2	2480.6 ± 16.7
	tensor x-OLGP	4.71 ± 0.15	2.86 ± 0.10	321.0 ± 3.9	503.5 ± 4.7
	tensor xy-OLGP	4.39 ± 0.18	4.53 ± 0.43	321.0 ± 3.9	492.4 ± 8.3
data size=10000, $w_{gen} = 0.6$	tensor x-OLGP	4.56 ± 0.14	2.66 ± 0.07	511.1 ± 3.2	829.9 ± 6.4
	tensor xy-OLGP	3.82 ± 0.15	4.03 ± 0.41	511.1 ± 3.2	822.0 ± 6.8
data size=36000, $w_{gen} = 0.4$	tensor GP	3.40 ± 0.19	10.15 ± 0.81	19141.9 ± 163.5	39152.4 ± 230.9
	tensor x-OLGP	5.77 ± 0.19	3.18 ± 0.12	2819.9 ± 37.3	5135.2 ± 66.3
	tensor xy-OLGP	5.62 ± 0.24	4.67 ± 0.48	2819.9 ± 37.3	4503.0 ± 48.1

Table 2: Performance comparison for the prediction of movement on shoulder marker along x-axis.

in NLL.

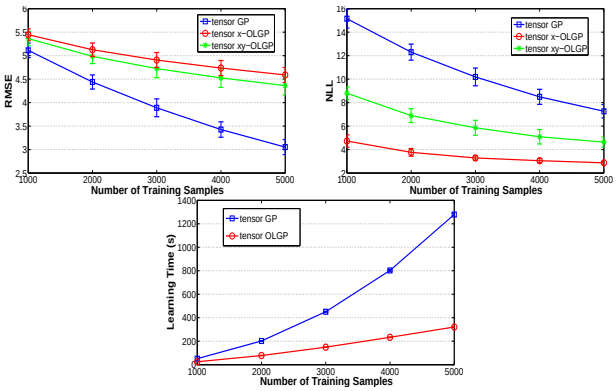

Fig. 1: Performance comparison vs. number of training samples, $w_{gen} = 0.5$, $R = 6$.

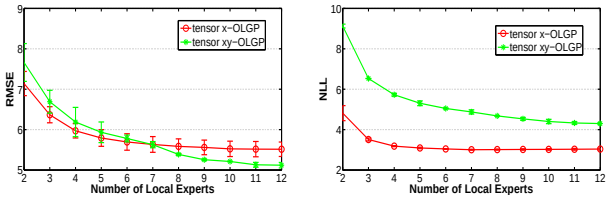

Fig. 2: Performance comparison vs. number of local experts.

From the Fig.1, it is straightforward to see that both RMSE and NLL of the test set decrease gradually when the proportion of data used in training increases from 1000 to 5000, implying the stability of the proposed methods. Comparing to the polynomial growth in $\mathcal{O}(N^3)$ of tensor GP, the learning time of tensor OLGP only grows linearly in $\mathcal{O}(N)$.

We are also interested in the situation that how the performance can be affected by distinct number of local experts R. In this context, w_{gen} is set to 0.3, and R is listed from 2 to 12. These results are demonstrated in the Fig.2. Observe that both RMSE and NLL reduce significantly before reaching their optimal values when R goes up. The result reflects that fact that a certain number of the most nearby local experts are required to guarantee a more accurate and reliable prediction. The further increase of the R brings no improvement in performance when the number of local experts becomes saturated.

Finally, we show the performance of scalability to a very large dataset when comparing all the methods. Here, we use the first 5 minutes of ECoG data with total number of 36000 data points, and randomly select 18000 points as training set and use the rest as the testing set. As is shown in Table 2, the results confirm the great superiority of our method to tensor GP in terms of scalability. Note that we select w_{gen} as 0.4 and R as 20, the which makes our method relatively much faster than the case of 10000 at only a small cost of accuracy loss.

5. CONCLUSION

In this paper, we have introduced a new tensor-variate local GP regression framework which successfully adapts the local GP modeling to the tensor input space. By doing so, our method is able to handle the applications of tensor steams in an online fashion. Furthermore, we have explored two different searching strategies to find the nearest neighbouring local experts for a reliable prediction. The experimental results have demonstrated the effectiveness and scalability of our method with very large-scaled data.

6. REFERENCES

[1] Q. Zhao, C.F. Caiafa, D.P. Mandic, Z.C. Nagasaka, Y. Fujii, and A. Cichocki, "Higher order partial least squares (hopls): A generalized multilinear regression method," *Pattern Analysis and Machine Intelligence, IEEE Transactions*, vol. 35(7), pp. 1660–1673, 2013.

[2] Q. Zhao, G. Adali, T. Zhang, and A. Cichocki, "Kernelization of tensor-based models for multiway data analysis: Processing of multidimensional structured data," *Signal Processing Magazine, IEEE Transactions*, vol. 30(4), pp. 137–148, 2013.

[3] Q. Zhao, G. Zhang, and A. Cichocki, "Tensor-variate gaussian processes regression and its application to video surveillance.," *In Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference*, pp. 1265–1269, 2014.

[4] H. Zhou, L. Li, and H. Zhu, "Tensor regression with applications in neuroimaging data analysis," *Journal of the American Statistical Association*, vol. 108(502), pp. 540–552, 2013.

[5] A. Eliseyev and T. Aksenova, "Recursive n-way partial least squares for brain-computer interface," *PloS one 8(7)*, 2013.

[6] T. Oommen, D. Misra, N.K.C. Twarakavi, A. Prakash, B. Sahoo, and S. Bandopadhyay, "An objective analysis of support vector machine based classification for remote sensing," *Mathematical Geosciences*, 2008.

[7] Z.C. Chao, Y. Nagasaka, and N. Fujii, "Long-term asynchronous decoding of arm motion using electrocorticographic signals in monkeys," *Frontiers in Neuroengineering 3*, 2010.

[8] C. E. Rasmussen and C. K. I. Williams, *Gaussian Process for Machine learning*, MIT Press, 2006.

[9] D. Nguyen-Tuong, J. Peters, and M. Seeger, "Local gaussian process regression for real time online model learning," *In Advances in Neural Information Processing Systems*, pp. 1193–1200, 2009.

[10] R. Urtasun and T. Darrell, "Sparse probabilistic regression for activity-independent human pose inference," *In Computer Vision and Pattern Recognition, CVPR*, 2008.