IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 8, 2021, accepted October 19, 2021, date of publication October 26, 2021, date of current version November 9, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3123098

GMMPLS: A Novel Automatic State-Based Algorithm for Continuous Decoding in BMIs

REZA FOODEH, VAHID SHALCHYAN, AND MOHAMMAD REZA DALIRI, (Member, IEEE)
Neuroscience and Neuroengineering Research Laboratory, Biomedical Engineering Department, School of Electrical Engineering, Iran University of Science & Technology (IUST), Tehran 16846-13114, Iran

Corresponding author: Mohammad Reza Daliri (daliri@iust.ac.ir)

ABSTRACT In this paper, a novel fully-automated state-based decoding method has been proposed for continuous decoding problems in brain-machine interface (BMI) systems, called Gaussian mixture of model (GMM)-assisted PLS (GMMPLS). In contrast to other state-based and hierarchical decoders, the proposed method does not demand any prior information about the desired output structure. Instead, GMMPLS uses the GMM algorithm to divide the desired output into a specific number of states (clusters). Next, a logistic regression model is trained to predict the probability membership of each time sample for each state. Finally, using the concept of the partial least square (PLS) algorithm, GMMPLS constructs a model for decoding the desired output using the input data and the achieved membership probabilities. The performance of the GMMPLS has been evaluated and compared to PLS, the nonlinear quadratic PLS (QPLS), and the bayesian PLS (BPLS) methods through a simulated dataset and two different real-world BMI datasets. The achieved results demonstrated that the GMMPLS significantly outperformed PLS, QPLS, and BPLS overall datasets.

INDEX TERMS Brain–machine interfaces (BMIs), partial least square (PLS), state-based decoding, continuous decoding, Gaussian mixture of model (GMM).

I. INTRODUCTION

Brain-machine interfaces (BMIs) are technologies for constructing an external pathway between the brain and a machine [1]. These systems capture the neural activities and translate them into understandable commands for prostheses and exoskeleton robots [2]–[5], quadcopters [6], or any external device.

Translating the neural activities has consisted of extracting beneficial features and decoding them to the desired commands [7]. A BMI application may requests discrete or continuous commands. Many researchers focus on decoding continuous parameters like limb movement [8], applied force, and grasp trajectories [9]. Therefore, much research has been conducted on the machine learning aspect of BMIs for decoding improvement.

Usually, For feature extraction in BMI, each channel of the recorded brain signals is decomposed to the specific frequency ranges, e.g., delta 0.5-4 Hz, theta 4-8 Hz, alpha 8-13 Hz, beta 12-30 Hz, low-gamma 30-60 Hz, etcetera.[1] Afterward, the envelope of each band per channel is derived. Sometimes, some specific previous lags in each envelope are considered too. Therefore, the feature space dimension equals the multiplication of three factors: the number of channels, frequency bands, and lags [10]–[12]. From the machine learning aspect, this procedure usually leads to the high-dimensionality problem and overfitting phenomena.

Except for the high-dimensionality problem, another concern is choosing linear or nonlinear models in continuous decoding. In many cases, a simple linear regression can not express the relation between the input and the desired output. However, nonlinear regression methods increase the risk of overfitting. Besides that, choosing a proper nonlinear function for the decoder is a critical issue.

To deal with overfitting obstacles, many researchers utilize feature reduction and feature selection techniques. These techniques reduce the dimensionality of the input space by generating reduced-rank combinations of the features (recorded channels) or selecting an optimal and relevant subset of features (recorded channels) via specific criteria to improve the decoder performances. For example, Jin *et al.* employed bispectrum analysis for channel selection in an Electroencephalogram (EEG)-based BMI system [13]. In another study, they proposed to use l_1-norm and

The associate editor coordinating the review of this manuscript and approving it for publication was Larbi Boubchir.

[1]Frequency division depends on the sampling frequency of the recorded signals.

Dempster–Shafer Theory for feature selection in the same application. Nazari *et al.* demonstrated that selecting informative features using a feature-ranking approach based on the Wilcoxon criterion led to performance improvement in a local field potential (LFP) based-cognitive BMI application [14].

Partial least square (PLS) and its extensions have been widely used in BMI systems for continuous decoding [10]–[12], [15]–[25]. PLS is a linear regression technique that reduces the risk of overfitting phenomena by reducing collinearity in the input data [26]. To overcome this obstacle, PLS builds a low-rank linear relation between the extracted features (input data) and the desired output [25]. However, PLS still tends to overfit, and because of that, some research introduced sparsity and regularization to the PLS cost function [18], [25], [26].

A simple linear regression model in BMI applications may not always explain the relationship between extracted features and the desired output [27]. Several nonlinear kernel-based PLS versions have been proposed and utilized in BMI applications to deal with this difficulty. Kernel-based PLS methods established a linear relation between a nonlinear map of the input and the desired output. However, these methods impose high computational costs and storage if the number of samples in the training set be high since they require building a square kernel matrix with the dimension equals to the number of samples [28]–[30]. On the other hand, other nonlinear PLS methods, such as quadratic PLS (QPLS), depend on choosing a suitable function for the nonlinear mapping of the input data.

Recently, some researchers introduced state-based and hierarchical algorithms for decoding the desired output in BMI applications. These studies assert that the desired output may have been composed of different processes. Therefore, the desired output samples are divided into a certain number of states (groups), and based on this segmentation, the input data samples are assigned to the related states. Afterward, a specific decoder has trained to learn the relation between the input and output samples in each group. Moreover, switching between different continuous decoding models is performed based on a classifier [31].

For example, a hierarchical PLS regression model was proposed in [32] for decoding hand speed, velocity, and position using humans electrocorticogram (ECoG) signals by Bundy *et al.*. The authors were aware that the designed task in their application contained rest and movement periods. Therefore, these periods were separated, and a logistic regression classifier with elastic net regularization[2] was trained to recognize whether the subject's hand was moving or not in every 300 ms time window. Next, a PLS model was trained for each of these states separately. In other words, the output of the classifier was used for switching between the two PLS models.

[2]Elastic net regularization contains both L1-norm and L2-norm constraints on the independent variables.

Ahmadi *et al.* proposed a state-based decoder for estimating applied force using recorded LFP signals in rats [33]. This study asserted that the applied force time series includes resting and active phases. Active phases were included time segments wherein rats were applying force, while in the resting phases, they did not perform the task. In this research, the filter bank common spatial patterns algorithm (FBCSP) was employed in conjunction with the linear discriminant analysis (LDA) for classifying the recorded LFP signals to the rest and active segments. Two different PLS models were trained then to decode each state of the desired output. This method led to higher performance compared with the conventional PLS. It is worth mentioning that a 500 ms window of LFP signals were required to classify the input data into rest/active group in this study. Thus, employing this method leads to at least a 500 ms delay in online decoding applications like Bundy *et al.* work in [32].

Farrokhi *et al.* proposed a state-based probabilistic decoding method for estimating 3D hand position trajectories of monkeys using recorded electrocorticogram (ECoG) signals [24]. The authors expressed that the desired output traces consisted of three states in their task: idle, right-hand movement, and left-hand movement states. Therefore, the authors introduced a feature extraction schema based on linear discriminant analysis (LDA) and PLS. Three LDA were trained to discriminant one state versus the others. LDA filters output fed to seven different PLS models: three models for predicting left-hand movement, three models for right-hand movement, and one model for estimating the movement's state (left, right, or idle). Afterward, a probabilistic model based on the conditional expectation operator was trained to decode the desired output using the output of the PLS models. The achieved results in this research indicated that their proposed method led to better performances compared to the naïve PLS and Kalman filter.

To the best of our knowledge, all the proposed hierarchical and state-based decoding algorithms in the BMI area relied on prior information about the desired output structure. For instance, we should know that the desired output contains resting/non-resting or different movement (task) types periods, and we should be able to separate them in the first step of the state-based decoding algorithm. Therefore, these algorithms suffer from generalization capabilities for utilizing in a different types of applications.

In this paper, a generalized state-based decoding algorithm has been proposed, called Gaussian mixture of model (GMM)-assisted PLS (GMMPLS). First, GMMPLS discriminates the desired output to a specific number of clusters using the GMM algorithm, and it calculates the membership probability of each sample for each cluster. Next, a regularized logistic regression model has been trained for each cluster to estimate an input sample's probability belonging to that cluster. Finally, a novel extended PLS algorithm has been developed to decode each extracted feature sample with respect to its probabilities.

Unlike other state-based decoding algorithms, the proposed GMMPLS is fully automated and does not rely on prior knowledge about the desired output, and could be employed in different BMI applications (and even other types of continuous decoding paradigms). To illustrate this advantage, two different BMI datasets were used to validate the efficiency of the proposed method compared to PLS, the nonlinear QPLS [34] and the bayesian PLS (BPLS) [35] methods. In addition to these real-world datasets, a simulation study was performed for more rigorous analysis. In all scenarios, the achieved results indicated that the proposed method outperformed the others regression techniques in terms of decoding correlation of coefficient, coefficient of determination, and mean absolute error metrics.

The paper is structured as follows: In Section II, a detailed description of conventional PLS is given, and then, the framework of the proposed GMMPLS is described. In Section III, the employed synthetic and real-world BMI datasets, performance metrics, evaluation, and hyper-parameters tuning procedure are introduced. Finally, section IV describes the obtained evaluation results, and sections V and VI conclude the paper.

II. METHODS

First, a detailed description of conventional PLS is given. Next, the structure of the proposed method has been demonstrated.

A. NOTATIONS

Throughout this manuscript, let $\mathbb{R}$ and T denote the set of real numbers and the transpose operator, respectively. Matrices are denoted by boldface capital letters ($\mathbf{X}$ and $\mathbf{Y}$), vectors are denoted by boldface lower-case letters ($\mathbf{x}$ and $\mathbf{y}$), row and column dimensions are denoted by italic upper-case letters (M and L) except for T, and scalar numbers are denoted by italic lower-case letters (m and l). A matrix can be present via its elements, i.e., $\mathbf{X} = \{x_{l,k}\}$ where l and m are the row and column index, respectively.

B. PLS ALGORITHM

Let $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_M] \in \mathbb{R}^{L\times M}$ and $\mathbf{Y} = [\mathbf{y}_1, \cdots, \mathbf{y}_N] \in \mathbb{R}^{L\times N}$ be the mean-centered extracted features (input) and desired output matrices, where L is the number of samples, and M and N are the input and output dimensions, respectively. PLS seeks a reduced-rank linear relation between $\mathbf{X}$ and $\mathbf{Y}$ to decrease the chance of overfitting occurrence. Suppose $\mathbf{X}$ and $\mathbf{Y}$ could be decomposed as follows:

$$\begin{aligned}\mathbf{X} &= \mathbf{T}\mathbf{P}^T + \mathbf{E} = \sum_{r=1}^{R} \mathbf{t}_r \mathbf{p}_r^T + \mathbf{E}\\ \mathbf{Y} &= \mathbf{U}\mathbf{Q}^T + \mathbf{F} = \sum_{r=1}^{R} \mathbf{u}_r \mathbf{q}_r^T + \mathbf{F}\end{aligned} \tag{1}$$

where $\mathbf{T} = [\mathbf{t}_1, \cdots, \mathbf{t}_R] \in \mathbb{R}^{L\times R}$ and $\mathbf{U} = [\mathbf{u}_1, \cdots, \mathbf{u}_R] \in \mathbb{R}^{L\times R}$ include the input's and output's latent vectors, respectively, $\mathbf{P} = [\mathbf{p}_1, \cdots, \mathbf{p}_R] \in \mathbb{R}^{M\times R}$ and $\mathbf{Q} = [\mathbf{q}_1, \cdots, \mathbf{q}_R] \in \mathbb{R}^{N\times R}$ consist of the loading vectors, $\mathbf{E}$ and $\mathbf{F}$ are the residual matrices, and R is the decomposition rank. For this purpose, PLS tries to find projections of the input $\mathbf{t} = \mathbf{X}\mathbf{w} \in \mathbb{R}^L$ and the output $\mathbf{u} = \mathbf{Y}\mathbf{q} \in \mathbb{R}^L$, in a way that the cross-covariance between them become maximum:

$$\max_{\mathbf{w},\mathbf{q}} \mathbf{t}^T\mathbf{u} = \mathbf{w}^T\mathbf{X}^T\mathbf{Y}\mathbf{q}, \quad \text{s.t. } \|\mathbf{w}\|_2 = \|\mathbf{q}\|_2 = 1 \tag{2}$$

where $\mathbf{w} \in \mathbb{R}^M$ is the normalized basis vector and $\|\cdot\|_2$ is the l_2-norm operator. PLS usually solves this problem via the nonlinear iterative partial least squares (NIPALS) algorithm [36]. After finding a maximally correlated[3] latent vectors of $\mathbf{X}$ and $\mathbf{Y}$, the loading vector $\mathbf{p} \in \mathbb{R}^M$ could be calculated as follows:

$$\mathbf{p} = \arg\min \left\|\mathbf{X} - \mathbf{t}\mathbf{p}^T\right\|_2^2 = \frac{\mathbf{X}^T\mathbf{t}}{\mathbf{t}^T\mathbf{t}} \tag{3}$$

A linear inner-relation between the latent vectors $\mathbf{t}$ and $\mathbf{u}$ is assumed to exist in PLS, i.e., $\mathbf{u} = d\mathbf{t} + \mathbf{e}$, where $d = \mathbf{u}^T\mathbf{t}/(\mathbf{t}^T\mathbf{t})$ would be a scalar and $\mathbf{e}$ represents the residual vector. After extracting r-th latent and loading vectors, the input and desired output matrices are deflated by their rank-one estimation for seeking the subsequent latent vectors:

$$\begin{aligned}\mathbf{X}_{r+1} &= \mathbf{X}_r - \mathbf{t}_r\mathbf{p}_r^T\\ \mathbf{Y}_{r+1} &= \mathbf{Y}_r - d\mathbf{t}_r\mathbf{q}_r^T\end{aligned} \tag{4}$$

where $\mathbf{X}_1 = \mathbf{X}$ and $\mathbf{Y}_1 = \mathbf{Y}$. Finally, after extracting R components, the linear relation between $\mathbf{X}$ and $\mathbf{Y}$ could be expressed as:

$$\mathbf{Y} \approx \mathbf{T}\mathbf{D}\mathbf{Q}^T \tag{5}$$

where $\mathbf{D} = \text{diag}(d_1, \cdots, d_R)$ represents a diagonal matrix and diag (,) is a diagonal matrix. Furthermore, the following relation could be proved:

$$\mathbf{T} = \mathbf{X}\mathbf{G} = \mathbf{X}\mathbf{W}\left(\mathbf{P}^T\mathbf{W}\right)^{-1} \tag{6}$$

where $\mathbf{G} \in \mathbb{R}^{M\times R}$. By substituting (6) in (5), the PLS regression coefficient matrix could be derived as follows [37], [38]:

$$\mathbf{Y} \approx \mathbf{X}\mathbf{B} \approx \mathbf{X}\mathbf{G}\mathbf{D}\mathbf{Q}^T \approx \mathbf{X}\mathbf{W}\left(\mathbf{P}^T\mathbf{W}\right)^{-1}\mathbf{D}\mathbf{Q}^T \tag{7}$$

where $\mathbf{B} = \mathbf{W}\left(\mathbf{P}^T\mathbf{W}\right)^{-1}\mathbf{D}\mathbf{Q}^T \in \mathbb{R}^{M\times N}$ is the PLS regression coefficient matrix. It is worth mentioning that R is a hyper-parameter that could be tuned using cross-validation (CV) techniques. The PLS algorithm is presented in the appendix section.

C. MAIN IDEA OF GMMPLS

The main idea behind the GMMPLS is that the desired output may compose of several different processes, and the contribution of each process in forming the desired output

[3]Correlation and covariance are equal for zero-mean vectors.

may vary over time. Suppose that the desired output has formed from K different components:

$$\mathbf{y}(l) = \sum_{k=1}^{K} \gamma_k(l)\, \mathbf{y}_k(l) \tag{8}$$

where $\mathbf{y}_k(l) \in \mathbb{R}^N$ is the k-th constituent component of the desired output and $0 \leq \gamma_k(l) \leq 1$ is a scalar representing the participation probability of the k-th component at the l-th sample. With this assumption, (8) can be written as follows for decoding the desired output:

$$\mathbf{y}(l) = \sum_{k=1}^{K} \gamma_k(l) f_k(\mathbf{x}(l)) \tag{9}$$

where $\mathbf{x}(l) \in \mathbb{R}^M$ is the l-th sample of the extracted features and $f_k(\cdot)$ is the k-th decoder. In other words, unlike employing a binary manner to switch between decoders for the fibal decoding, we suggest a probabilistic framework for such a purpose.

We offer to utilize GMM algorithms for estimating $\gamma_k(l)$ in (9). GMM is a model-based clustering technique, which fits K independent Gaussian distributions to the data based on the expectation-maximization (EM) algorithm. In summary, with an initialized mean vector and covariance matrix for each cluster, GMM estimates the membership probability of a sample for each cluster via the expectation step, and thereupon in the maximization step, GMM updates mean vectors and covariance matrices based on the obtained probabilities [39]. The GMM algorithm is presented in the appendix section.

It should be noted that the GMM initialization is done via the K-means algorithm in this study. See [40]–[42]. In addition, we used the z-scored version of $\mathbf{Y}$ in GMM algorithm to prevent the effect of data magnitudes in learning the GMM model.

After applying the GMM algorithm to the desired output $\mathbf{Y}$, the membership probability matrix for all clusters $\boldsymbol{\Gamma} = [\boldsymbol{\gamma}_k, \cdots, \boldsymbol{\gamma}_K] = \{\gamma_k(l)\} \in \mathbb{R}^{L\times K}$ is calculated. Fig. 1 illustrates the main idea of the proposed method.

D. MEMBERSHIP PROBABILITY PREDICTION

The membership probabilities could be obtained using the available desired output and the GMM algorithm in the training phase. Nevertheless, this approach will not be executable in the testing phase or real-time applications since the desired output is absent. Hence, we proposed using a logistic regression model to decode the membership probabilities from the extracted features.

Consider a logistic regression for decoding k-th membership probabilities:

$$\hat{\gamma}_k(l) = \sigma_k\left(\mathbf{x}^T(l)\,\mathbf{h}_k + h_{0k}\right) = \frac{1}{1+e^{-\left(\mathbf{x}^T(l)\mathbf{h}_k + h_{0k}\right)}} \tag{10}$$

where $\sigma(z) = 1/\left(1+e^{-z}\right)$ is the sigmoid function, $\mathbf{h}_k \in \mathbb{R}^M$ is the k-th linear regression vector, and h_{0k} is the scalar intercept term. We augment the intercept h_{0k} in $\mathbf{h}_k$ by $\mathbf{h}_k \leftarrow \left[h_{0k}, \mathbf{h}_k^T\right]^T \in \mathbb{R}^{M+1}$ and include a constant $+1$ in the input vector as $\mathbf{x}(l) \leftarrow \left[+1, \mathbf{x}^T(l)\right]^T \in \mathbb{R}^{M+1}$. Finally, in the rest of the paper, $\sigma_k(l)$ will denote $\sigma_k\left(\mathbf{x}^T(l)\,\mathbf{h}_k\right)$ for simplicity.

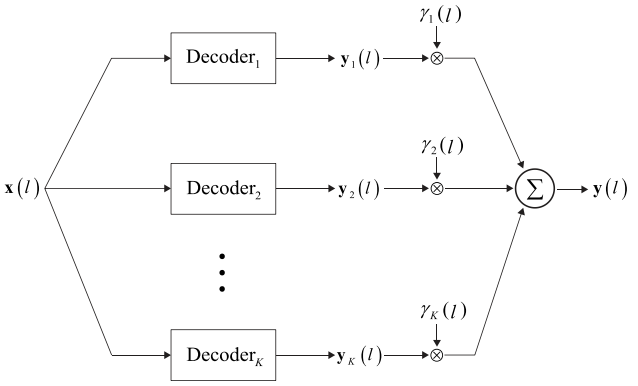

FIGURE 1. The main idea of the proposed method. The separate decoders contribute to forming the desired output with a specific membership probability, varying from sample to sample.

Typically, the cross-entropy error is adopted as the logistic regression's cost function since it is convex rather than the mean square error [39], [43], [44]. This cost function could be represented to estimate $\mathbf{h}_k$ in (10) as follows:

$$\mathrm{J}(\mathbf{h}_k) = -\frac{1}{L}\sum_{l=1}^{L}\left[\gamma_k(l)\ln\left(\sigma_k(l)\right) + \left(1-\gamma_k(l)\right)\ln\left(1-\sigma_k(l)\right)\right] \tag{11}$$

where $\gamma_k(l)$ and $\sigma_k(l)$ are the actual and the predicted membership probability of the k-th cluster at the l-th sample. The derivative of this cost function could be represented as follows:

$$\mathbf{g}_k = -\sum_{l=1}^{L}\mathbf{x}(l)\left(\gamma_k(l)-\sigma_k(l)\right) = -\mathbf{X}^T\left(\boldsymbol{\gamma}_k - \hat{\boldsymbol{\gamma}}_k\right) \tag{12}$$

where $\mathbf{g}_k = \nabla_{\mathbf{h}_k}\mathrm{J}(\mathbf{h}_k)$ is the gradient of (11) with respect to $\mathbf{h}_k$.

To solve (11), adaptive moment estimation (ADAM) optimization has been employed in this study. In short, ADAM is an optimization algorithm that uses the first and second moments of the gradient vector to estimate an adaptive learning rate for each of the optimization parameters [45]. In addition, to prevent overfitting during the learning procedure, the weight decay schema in ADAM is utilized in this study which is a l_2-norm regularization priocedure in ADAM optimization [46]. The details of this method are given in the appendix section. It is worth mentioning that ADAM converges faster than traditional gradient descent algorithms.

E. GMMPLS ALGORITHM

After estimating membership probabilities $\boldsymbol{\Gamma} = \left[\boldsymbol{\gamma}_1, \cdots, \boldsymbol{\gamma}_K\right]$ in the previous section, we should build a model for

decoding the desired output with the aid of these probabilities. Therefore, we proposed a novel method using the concept of the PLS algorithm, called GMMPLS.

Assume the input and the desired output are zero-centered. Suppose each component of the desired output in (8) could be decomposed to latent and loading vectors in the form of equation (1):

$$\mathbf{y}_k(l) = \sum_{r=1}^{R} u_{r,k}(l)\,\mathbf{q}_{r,k}^T + \mathbf{f}_k(l) \tag{13}$$

By Substituting $\mathbf{y}_k(l)$ from (13) in (8), we have:

$$\begin{aligned}\mathbf{y}(l) &= \sum_{k=1}^{K} \gamma_k(l)\left(\sum_{r=1}^{R} u_{r,k}\mathbf{q}_{r,k}^T + \mathbf{f}_k(l)\right)\\ &\approx \sum_{r=1}^{R}\sum_{k=1}^{K} \gamma_k(l)\,u_{r,k}(l)\,\mathbf{q}_{r,k}^T\end{aligned} \tag{14}$$

where $\left\|\mathbf{q}_{r,k}\right\|_2 = 1$. It is worth mentioning that $\mathbf{f}_k(l)$ is the residual term in (13). We assume that all loading vectors in all clusters for each r are equals for simplification, i.e., $\mathbf{q}_r = \mathbf{q}_{r,1} = \cdots = \mathbf{q}_{r,K}$. This assumption means that each desired output's latent variable could be expressed and decomposed in the form of (8). Therefore, (14) can be rewritten as:

$$\begin{aligned}\mathbf{y}(l) &\approx \sum_{r=1}^{R} z_r(l)\,\mathbf{q}_r^T = \sum_{r=1}^{R}\left(\sum_{k=1}^{K}\gamma_k(l)\,u_{r,k}(l)\right)\mathbf{q}_r^T\\ \text{where } z_r(l) &= \sum_{k=1}^{K}\gamma_k(l)\,u_{r,k}(l)\end{aligned} \tag{15}$$

Finally, we assume a linear relation between $u_{r,k}(l)$ and a linear projection of the input data:

$$\begin{aligned}u_{r,k}(l) &= d_{r,k,1}t_{r,k}(l) + d_{r,k,0} + \psi_{r,k}(l)\\ \text{where } t_{r,k}(l) &= \mathbf{x}_k^T(l)\,\mathbf{w}_{r,k}\end{aligned} \tag{16}$$

where $\mathbf{w}_{r,k}$ is a normalized basis vector, and $d_{r,k,1}$ and $d_{r,k,0}$ are scale and intercept scalars, respectively, and $\psi_{r,k}(l)$ is the error term. Thus, equations (15) and (16) are the main framework of the proposed GMMPLS.

To explain the parameters learning strategy in GMMPLS, suppose we want to extract the first latent and loading variables, i.e., $r = 1$. Therefore, we ignore r subscripts to improve the readability of equations. In addition, assume we possess initial values for $\mathbf{w}_k$, $\mathbf{q}_{r,k}$, $d_{k,1}$, and $d_{k,0}$ and subsequently, $t_k(l)$ and $u_k(l)$ can be initialized for $k = 1, \cdots, K$.

Altogether, we have:

$$\begin{aligned}t_k(l) &= \mathbf{x}^T(l)\,\mathbf{w}_k, \quad k = 1, \cdots, K\\ u_k(l) &\approx d_{k,1}t_k(l) + d_{k,0}, \quad k = 1, \cdots, K\\ z(l) &= \sum_{k=1}^{K}\gamma_k(l)\,u_k(l)\\ \mathbf{y}^T(l) &\approx z(l)\,\mathbf{q}^T\end{aligned} \tag{17}$$

Now consider these relations. Due to $\|\mathbf{q}\|_2 = 1$, we have $z(l) \approx \mathbf{y}^T(l)\,\mathbf{q}$. Next, we can claim:

$$\begin{aligned}\varphi(l) &\approx \sum_{k=1}^{K} d_{k,1}\gamma_k(l)\,\mathbf{x}^T(l)\,\mathbf{w}_k\\ \text{where } \varphi(l) &= z(l) - \sum_{k=1,}^{K} d_{k,0}\gamma_k(l)\end{aligned} \tag{18}$$

Now by defining the following augmented matrices:

$$\begin{aligned}\bar{\mathbf{X}} &= \left[d_{1,1}\boldsymbol{\gamma}_1 \otimes \mathbf{X}, \cdots, d_{K,1}\boldsymbol{\gamma}_K \otimes \mathbf{X}\right] \in \mathbb{R}^{L\times KM}\\ \bar{\mathbf{w}} &= \left[\mathbf{w}_1^T, \cdots, \mathbf{w}_K^T\right]^T \in \mathbb{R}^{KM}\end{aligned} \tag{19}$$

equation (18) can be reformulated as:

$$\boldsymbol{\varphi} = \bar{\mathbf{X}}\bar{\mathbf{w}} \in \mathbb{R}^L \tag{20}$$

where $\otimes$ is the element-wise multiplication operation.

To avoid overfitting phenomena, we can solve (20) by a rank-1 PLS as follows:

$$\bar{\mathbf{w}} = \bar{\mathbf{X}}^T\boldsymbol{\varphi} \tag{21}$$

where each $\mathbf{w}_k$ should have a unit norm:

$$\mathbf{w}_k \leftarrow \mathbf{w}_k / \|\mathbf{w}_k\|_2, \quad k = 1, \cdots, K \tag{22}$$

After obtaining $\mathbf{w}_k$ for each $k = 1, \cdots, K$, latent vectors $\mathbf{t}_k = \mathbf{X}\mathbf{w}_k$ are calculated. Now by constructing augmented matrices:

$$\begin{aligned}\bar{\mathbf{T}} &= \left[\boldsymbol{\gamma}_1, \boldsymbol{\gamma}_1 \otimes \mathbf{t}_1, \cdots, \boldsymbol{\gamma}_K, \boldsymbol{\gamma}_K \otimes \mathbf{t}_K\right] \in \mathbb{R}^{L\times 2K}\\ \mathbf{d} &= \left[d_{1,0}, d_{1,1}, \cdots, d_{K,0}, d_{K,1}\right]^T \in \mathbb{R}^{2K}\end{aligned} \tag{23}$$

we have:

$$\begin{aligned}\mathbf{z} &\approx \bar{\mathbf{T}}\mathbf{d}\\ \Rightarrow \mathbf{d} &= \left(\bar{\mathbf{T}}^T\bar{\mathbf{T}}\right)\bar{\mathbf{T}}^T\mathbf{z}\end{aligned} \tag{24}$$

After deriving $\hat{\mathbf{z}}$ using (24), we can estimate $\mathbf{q}$ using the least square error method:

$$\begin{aligned}\mathbf{q} &= \mathbf{Y}^T\hat{\mathbf{z}} \Big/ \left(\hat{\mathbf{z}}^T\hat{\mathbf{z}}\right)\\ \mathbf{q} &\leftarrow \mathbf{q}/\|\mathbf{q}\|_2\end{aligned} \tag{25}$$

The procedure involving (17)-(25) is iterated until the convergence of the parameters.

After fining r-th components of GMMPLS, the desired output is deflated for seeking the subsequent components as follows:

$$\mathbf{Y}_{r+1} \leftarrow \mathbf{Y}_r - \bar{\mathbf{T}}_r\bar{\mathbf{d}}_r\mathbf{q}_r^T \tag{26}$$

After the deflation process in (26), GMMPLS goes back to the iterated process through (17)-(25) to seek the next $(r+1)$-th components. Algorithm 1 demonstrates the GMMPLS procedure in detail.

It is worth mentioning that the number of clusters K and GMMPLS rank R are hyper-parameters, and the process for choosing them will be explained in section III.

F. ESTIMATING THE DESIRED OUTPUT IN GMMPLS

After extracting the GMMPLS parameters in Algorithm 1, the desired output could be estimated via Algorithm 2.

III. EXPERIMENTS

In this section, first, we demonstrated the described datasets. Next, we discussed the hyper-parameter selection procedure, and finally, we introduced evaluation metrics used in this manuscript.

A. DATA DESCRIPTION

In this paper, three different datasets were used to evaluate and to compare the proposed method with PLS, QPLS, and BPLS. The first dataset is a synthetic dataset, the second is the publicly available ECoG dataset for decoding hand trajectories, and the last is our LFP for decoding applied force.

1) SYNTHETIC DATASET

For building this dataset, we generated each column of $\mathbf{X} \in \mathbb{R}^{L\times M}$ randomly with $L = 10000$ and $M = 500$ via the Gaussian distribution with random means and variances. The means and variances were derived from the normal Gaussian distribution. To increase collinearity, we performed the singular value decomposition (SVD) on $\mathbf{X}$ and eliminated 40% of the singular vectors and then $\mathbf{X}$ was reconstructed. We assumed that the desired output was generated as follows:

$$\mathbf{Y} = \sum_{k=1}^{K} \boldsymbol{\gamma}_k \otimes \mathbf{Y}_k \tag{27}$$

where:

$$\begin{cases} \mathbf{Y}_k = \mathbf{X}\,\mathbf{W}_k \\ \boldsymbol{\gamma}_k = \dfrac{\exp\left(\boldsymbol{\sigma}_k\right)}{\sum_{j=1}^{K}\exp\left(\boldsymbol{\sigma}_j\right)} \\ \boldsymbol{\sigma}_k = \mathbf{X}\mathbf{v}_k \\ \boldsymbol{\Gamma} = \left[\boldsymbol{\gamma}_k, \cdots, \boldsymbol{\gamma}_K\right] = \mathbf{X}\mathbf{V} \end{cases} \tag{28}$$

In these relations, $\mathbf{W}_K \in \mathbb{R}^{M\times N}$ and $\mathbf{v}_k \in \mathbb{R}^M$ were generated randomly using Gaussian distribution with random means and covariances. It is worth mentioning that the samples of $\boldsymbol{\gamma}_k$ are bounded between $0-1$ and $\sum_{k=1}^{K}\boldsymbol{\gamma}_k = 1$.

Twelve different cases were considered in this simulation with assuming $N \in \{1, 3, 5\}$ and $K \in \{1, 2, 3, 4\}$. In addition, we ran the simulation 100 times for performance analysis. In each run, the first 90% samples of the synthetic data were used to train the decoders, and the remaining 10% was used for performance evaluation.

2) PUBLIC ECoG DATASET

This public dataset was introduced in 2012 by Shimoda *et al.* for decoding hand movement trajectories using monkeys' recorded ECoG signals [10]. ECoG electrodes with 64 channels were implanted on the epidural space of the prefrontal cortex (PFC) and the primary somatosensory cortex (S1) in the left hemisphere. Each animal performed

Algorithm 1 GMMPLS Algorithm for Estimating Desired $\mathbf{Y}$ From $\mathbf{X}$

Input: $\mathbf{X} \in \mathbb{R}^{L\times M}$, $\mathbf{Y} \in \mathbb{R}^{L\times N}$,
Predicted membership probabilities: $\boldsymbol{\Gamma} = \left[\boldsymbol{\gamma}_k, \cdots, \boldsymbol{\gamma}_K\right] \in \mathbb{R}^{L\times K}$,
Number of PLS components: R.
Output: $\mathbf{W}_r = \left[\mathbf{w}_{r,1}, \cdots, \mathbf{w}_{r,K}\right] \in \mathbb{R}^{M\times K}$,
$k = 1, \cdots, K$,
$\mathbf{d}_r = \left[d_{r,1,0}, d_{r,1,1}, \cdots, d_{r,K,0}, d_{r,K,1}\right]^T \in \mathbb{R}^{2K}$,
$k = 1, \cdots, K$,
$\mathbf{q}_r \in \mathbb{R}^N,\ k = 1, \cdots, K$.
Assign $\mathbf{Y}_r = \mathbf{Y}$.
for r **to** R **do**
  Initial loading vector $\mathbf{q}_r$.
  Initial $\mathbf{d}_r$.
  while not convergence do
    $\mathbf{z}_r = \mathbf{Y}_r\mathbf{q}_r$.
    Calculate $\boldsymbol{\varphi}$ using (19).
    Consturct $\bar{\mathbf{X}}_r = \left[d_{r,1,1}\boldsymbol{\gamma}_1 \otimes \mathbf{X}, \cdots, d_{r,K,1}\boldsymbol{\gamma}_K \otimes \mathbf{X}\right]$.
    $\bar{\mathbf{w}}_r = \left[\mathbf{w}_{r,1}^T, \cdots, \mathbf{w}_{r,K}^T\right]^T = \bar{\mathbf{X}}_r^T\boldsymbol{\varphi}$.
    $\mathbf{w}_{r,k} \leftarrow \mathbf{w}_{r,k} \big/ \left\|\mathbf{w}_{r,k}\right\|_2,\ k = 1, \cdots, K$.
    $\mathbf{t}_{r,k} = \mathbf{X}\mathbf{w}_{r,k},\ k = 1, \cdots, K$.
    Consturct $\bar{\mathbf{T}}_r = \left[\boldsymbol{\gamma}_1, \boldsymbol{\gamma}_1 \otimes \mathbf{t}_{r,1}, \cdots, \gamma_K, \boldsymbol{\gamma}_K \otimes \mathbf{t}_{r,K}\right]$.
    $\mathbf{d}_r = \left(\bar{\mathbf{T}}_r^T\bar{\mathbf{T}}_r\right)\bar{\mathbf{T}}_r^T\mathbf{z}_r$.
    $\hat{\mathbf{z}}_r \approx \bar{\mathbf{T}}_r\mathbf{d}_r$.
    $\mathbf{q}_r = \mathbf{Y}^T\hat{\mathbf{z}}_r \big/ \left(\hat{\mathbf{z}}_r^T\hat{\mathbf{z}}_r\right)$
    $\mathbf{q}_r \leftarrow \mathbf{q}_r \big/ \left\|\mathbf{q}_r\right\|_2$
  end while
  $\mathbf{W}_r \leftarrow \left[\mathbf{w}_{r,1}, \cdots, \mathbf{w}_{r,K}\right]$.
  $\mathbf{d}_r \leftarrow \left[d_{r,1,0}, d_{r,1,1}, \cdots, d_{r,K,0}, d_{r,K,1}\right]^T$.
  Deflation:
  $\mathbf{Y}_{r+1} \leftarrow \mathbf{Y}_r - \hat{\mathbf{z}}_r\mathbf{q}_r^T$
end for

Algorithm 2 Estimation of the Desired Output in GMMPLS Algorithm

Input: $\mathbf{X} \in \mathbb{R}^{L\times M}$,
Predicted membership probabilities:
$\boldsymbol{\Gamma} = \left[\boldsymbol{\gamma}_k, \cdots, \boldsymbol{\gamma}_K\right] \in \mathbb{R}^{L\times K}$,
$\mathbf{W}_r = \left[\mathbf{w}_{r,1}, \cdots, \mathbf{w}_{r,K}\right] \in \mathbb{R}^{M\times K},\ k = 1, \cdots, K$
$\mathbf{d}_r = \left[d_{r,1,0}, d_{r,1,1}, \cdots, d_{r,K,0}, d_{r,K,1}\right]^T \in \mathbb{R}^{2K}$,
$k = 1, \cdots, K$
$\mathbf{q}_r \in \mathbb{R}^N,\ k = 1, \cdots, K$.
Output: $\hat{\mathbf{Y}}$.
Assign $\hat{\mathbf{Y}} = \mathbf{0}$.
for r **to** R **do**
  $\mathbf{t}_{r,k} = \mathbf{X}\mathbf{w}_{k,r},\ k = 1, \cdots, K$
  $\hat{\mathbf{u}}_{r,k} = d_{r,k,1}\mathbf{t}_{r,k} + d_{r,k,0},\ k = 1, \cdots, K$
  $\hat{\mathbf{z}} = \sum_{k=1}^{K}\boldsymbol{\gamma}_k \otimes \hat{\mathbf{u}}_{r,k}$
  $\hat{\mathbf{Y}} = \hat{\mathbf{Y}} + \hat{\mathbf{z}}\mathbf{q}_r^T$
end for

ten 15-minute sessions of food-tracking task while ECoG signals were recorded with a 1000 Hz sampling rate, and hand motions were recorded with a sampling rate of 120 Hz.

For feature extraction, similar to [18], the ECoG signals were down-sampled to 250 Hz, and a common average reference (CAR) was applied to the recorded channels. Next, ECoG signals were divided into six different frequency bands [1-4, 4-8, 8-12, 12-30, 30-60, and 60-120 Hz] using an eight-order Butterworth filter for extracting delta, theta, alpha, beta, low-gamma, and high-gamma1 brain rhythms. It is worth mentioning that the upper bound in the high-gamma1 was set to 120 Hz due to the frequency range in the epidural ECoG signals (<120 Hz). Next, each frequency band in each channel was then rectified and smoothed using third-order Savitzky–Golay moving average with 0.3s width to obtain its envelope. The extracted envelopes represent the variations through each frequency band. Eventually, for each sample of time, the sample itself and its' nine previous ones with a 0.1s interval were collected as the features in each frequency band of each channel. This manner yielded feature vectors with the dimension of $3840 = 64$ (channels) $\times$ 6 (bands) $\times$ 10 (lags).

The objective of this manuscript was to decode three-dimensional right arm trajectories of the monkeys from the extracted features using PLS, QPLS, BPLS and the proposed GMMPLS.

3) LFP DATASET

This dataset was collected in our neuroscience lab in 2016 by Khorasani *et al.* for decoding rats' applied force from the 16 channels of LFP signals [11]. Thirsty rats were trained to apply a certain amount of force on a load cell to receive a drop of water as the reward.

16-channel microwire array was implanted in the forelimb sensorimotor cortex in the left hemisphere of three Wistar rats. The neural signals were then recorded with a 10 kHz sampling rate during the task. First, a low-pass filter with a 500 Hz cutoff frequency was applied to the recorded signals, and then the signals were down-sampled to 1000 Hz. Simultaneously, the output of the load cell was recorded at a 30 Hz sampling rate.

Again similar to our previous work [18], the recorded LFP channels were firstly filtered out using CAR. Next, eight different frequency bands [1-4, 4-8, 8-12, 12-30, 30-60, 60-120, 120-200, and 200-400 Hz] were extracted from each channel using an eight-order Butterworth filter to extract delta, theta, alpha, beta, low-gamma, high-gamma1, high-gamma2 and high-gamma3 brain rhythms. It is worth mentioning that the spectral range in LFP signals is up to 400 Hz.

Envelope extraction of each channel's frequency bands was achieved via rectifying and smoothing by third-order Savitzky–Golay moving average with 0.3s width. Finally, analogously to the previous sub-section, each sample of time and its lags were collected as the features in each channel's frequency band. Therefore, the dimension of the extracted features was $1280 = 16$ (channels) $\times$ 8 (bands) $\times$ 10 (lags) for each time sample.

The goal of this manuscript was to continuously decode one-dimensional applied force values using extracted features for each rat.

B. HYPER-PARAMETER SELECTION

In the PLS, QPLS, and BPLS algorithms, the only hyper-parameter is the decomposition rank R. We used the BPLS toolbox presented in https://github.com/vidaurre/bpls. On the other hand, GMMPLS involves other hyper-parameters: the decomposition rank R, the number of clusters K, and the regularization parameter λ for membership probabilities estimation (see Algorithm 5 in appendix section). In addition, β_1, β_2, ε and the step size α are also hyper-parameters in ADAM (see Algorithm 5 in appendix section) which have to tune.

Usually, fix values $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\varepsilon = 10^{-8}$ are used in ADAM algorithms in the manuscript. Therefore, we employed these values and ignored tuning these hyper-parameters. On the other hand, the step size α can be tuned within the ADAM optimization. Therefore, we considered the list $\alpha = [0, 0.0001, 0.001, 0.01, 0.1]$ and performed a line search procedure to test different values of α in updating $\mathbf{h}_k(t)$ and selected the one that yielded a lower cost value in equation (11) in each iteration.

For tuning R, K, and λ, we performed an internal 10-fold CV procedure during the GMMPLS algorithm. More precisely, we divided the training data into new training and validation sets using 10-fold CV and chose the optimal hyper-parameters from several values; those that minimized the mean square error (MSE) of the desired output estimation in the validation set. The optimal values for the hyper-parameters were selected among the predefined set $R \in [1, \cdots, 50]$, $K \in [1, \cdots, 5]$ and $\lambda \in [0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50, 100]$. In the PLS, QPLS, and BPLS algorithms, the same procedure was performed to optimize R.

C. PERFORMANCE EVALUATION

To evaluate and compare the proposed method with the other regression techniques performances, we performed a 10×10-fold CV procedure in the ECoG and the LFP datasets to divide data into training and test sets. Briefly, the recorded brain signals (ECoG and LFP) and the desired output (hand trajectories and applied force) were randomly split into ten non-overlapping folds. At each stage, nine folds were used to train the decoders, and the remaining one was used to test the decoding performance. All of the folds were used as the test set one time. Finally, this procedure was repeated ten times, and all 100 gained performances were averaged for each decoding method.

In this manuscript, we employed three different metrics to analyze and compare PLS, QPLS, BPLS, and GMMPLS performance. The first one is the correlation coefficient which

TABLE 1. Decoding correlation coefficients obtained in different scenarios for PLS, QPLS, BPLS and the proposed GMMPLS. *N* represents the dimension of the desired output, and *K* represents the number of clusters in generating the dataset. GMMPLS-*K* stands for training GMMPLS with *K* clusters. Bold values indicate the highest performance in each column. The higher value is better.

Method	*N*=1				*N*=3				*N*=5				Overall	
	K=				*K*=				*K*=					
	1	2	3	4	1	2	3	4	1	2	3	4	mean	std
PLS	**1**	0.72	0.63	0.61	**1**	0.72	0.63	0.59	**1**	0.75	0.67	0.61	0.74	0.1
QPLS	**1**	0.73	0.65	0.62	**1**	0.72	0.64	0.59	**1**	0.75	0.67	0.6	0.75	0.1
BPLS	**1**	0.71	0.62	0.59	**1**	0.71	0.62	0.57	**1**	0.74	0.65	0.59	0.73	0.11
GMMPLS-1	**1**	0.72	0.64	0.61	**1**	0.72	0.64	0.59	0.99	0.75	0.67	0.61	0.74	0.1
GMMPLS-2	0.95	0.83	0.88	**0.9**	0.97	**0.81**	**0.84**	0.85	0.97	**0.86**	0.77	**0.81**	**0.87**	0.09
GMMPLS-3	0.94	0.86	0.86	0.88	0.96	0.8	0.8	0.84	0.96	0.85	**0.78**	0.77	0.86	0.08
GMMPLS-4	0.91	**0.87**	0.87	0.88	0.95	0.8	0.79	0.81	0.95	0.85	0.77	0.77	0.85	0.09
GMMPLS-5	0.92	0.86	**0.91**	0.88	0.95	0.77	**0.84**	**0.88**	0.95	0.82	0.77	0.79	0.86	0.09

ranges between -1 and 1:

$$\rho = \frac{\text{cov}\left(\mathbf{y}, \hat{\mathbf{y}}\right)}{\text{std}\left(\mathbf{y}\right) \text{std}\left(\hat{\mathbf{y}}\right)} \tag{29}$$

where cov (.) and std (.) are the cross-covariance and the standard deviation operators, respectively. The next metric is the coefficient of determination, denoted by R^2 which represents the ''goodness of fit'' and ranges between $-\infty$ and 1. $R^2 = 1$ means the perfect fit and $R^2 = 0$ means that the estimated desired output is equal to 0. On the other hand, negative values of R^2 indicates a disaster in the decoding procedure. R^2 is defined as:

$$R^2 = 1 - \frac{\text{var}\left(\mathbf{y} - \hat{\mathbf{y}}\right)}{\text{var}\left(\mathbf{y}\right)} \tag{30}$$

where var (.) is the variance operator. It is worth mentioning that $\text{var}\left(\mathbf{y} - \hat{\mathbf{y}}\right)/\text{var}\left(\mathbf{y}\right)$ has been defined as the normalized mean square error (NMSE) in manuscripts.

The last metric used in this paper is the mean absolute error:

$$\text{MAE} = \text{mean}\left(\left|\mathbf{y} - \hat{\mathbf{y}}\right|\right) \tag{31}$$

where mean $(\cdot)$ and $|\cdot|$ are the average and the absolute operators, respectively. Compared to MSE based metrics (like R^2 and NMSE), MAE is less sensitive to outliers and large error values since it does not involve squaring values [47].

D. RUN-TIME COMPARISON

For complexity comparison, we performed a run-time analysis for each decoding method. To achieve this goal, we used three minutes of ECoG and LFP datasets to train PLS, QPLS, BPLS, and the proposed GMMPLS and calculated the training run-times separately. This process was repeated 100 times, and the obtained values were averaged.

To avoid the effect of hyperparameter tuning and CV procedure in this analysis, we used fix hyper-parameter values in all decoding methods. We set $R = 20$ for all methods, and we used a fixed regularization parameter value $\lambda = 10$ in training GMMPLS. In addition, we trained the proposed method using a different number of clusters K to evaluate the effect of this hyperparameter in the computational cost.

IV. RESULTS

A. SYNTHETIC DATASET

As mentioned before, we generated the synthetic dataset 100 times randomly. At each time, 90% of the generated data was used to train PLS, QPLS, BPLS, and GMMPLS, and the rest of the data was used for performance evaluation. Tables 1, 2 and 3 demonstrates achieved correlation coefficients, R^2 and MAE decoding performance using each decoding method in this simulation study. Different cases were considered in this simulation. $N = 1, 3, 5$ and $K = 1, 2, 3, 4$ led to 12 different scenarios. In addition, we performed the decoding using a different number of clusters in GMMPLS, e.g., GMMPLS-3 means that we supposed $K = 3$ in training GMMPLS. These results indicated that when the desired output is composed of more than one process, i.e., $K > 1$, our proposed algorithm with $K > 1$ led to higher performance. On the other hand, GMMPLS with $K = 1$ resulted in approximately the same performance compared to PLS. In other words, GMMPLS is simplified to the ordinary PLS when we consider $K = 1$ in the training of GMMPLS.

It is evident that increasing the number of states in the generation of the desired output has caused decreasing in decoding performance. However, the results illustrate that the proposed GMMPLS could stand against this phenomenon more than others. GMMPLS with $K = 2$ achieved the highest performance in this simulation study.

B. ECoG DATASET

In this sub-section, first, an example of estimated hand trajectories in the first monkey using PLS and GMMPLS was depicted in Fig. 2. To make this figure more clear, QPLS and BPLS predicted traces were not drawn. In this figure, a 70 s time window of the observed and predicted hand trajectories was shown for three dimensions. In this example, all hyper-parameters were optimized through the CV procedure. The optimized number of clusters in GMMPLS was $K = 4$.

TABLE 2. Decoding coefficient of determination (R^2) obtained in different scenarios for PLS, QPLS, BPLS and the proposed GMMPLS. *N* represents the dimension of the desired output, and *K* represents the number of clusters in generating the dataset. GMMPLS-*K* stands for training GMMPLS with *K* clusters. Bold values indicate the highest performance in each column. The higher value is better.

Method	N=1				N=3				N=5				Overall	
	K=				K=				K=					
	1	2	3	4	1	2	3	4	1	2	3	4	mean	std
PLS	**1**	0.54	0.42	0.39	**1**	0.52	0.42	0.37	0.99	0.59	0.46	0.38	0.59	0.14
QPLS	**1**	0.55	0.44	0.4	0.99	0.52	0.42	0.37	0.99	0.58	0.45	0.37	0.59	0.13
BPLS	**1**	0.52	0.38	0.36	**1**	0.5	0.39	0.32	**1**	0.57	0.43	0.34	0.57	0.15
GMMPLS-1	**1**	0.54	0.41	0.39	0.96	0.53	0.43	0.37	0.95	0.59	0.47	0.39	0.59	0.14
GMMPLS-2	0.89	0.69	0.78	**0.82**	0.94	**0.66**	**0.72**	0.74	0.93	**0.74**	0.6	**0.66**	**0.76**	0.15
GMMPLS-3	0.84	**0.74**	0.75	0.78	0.9	0.64	0.65	0.72	0.91	0.73	**0.61**	0.61	0.74	0.14
GMMPLS-4	0.78	**0.74**	0.76	0.77	0.87	0.63	0.63	0.68	0.88	0.72	0.59	0.6	0.72	0.16
GMMPLS-5	0.79	0.71	**0.83**	0.78	0.86	0.57	0.71	**0.78**	0.88	0.67	0.58	0.63	0.73	0.15

TABLE 3. Decoding mean absolute error (MAE) obtained in different scenarios for PLS, QPLS, BPLS and the proposed GMMPLS. *N* represents the dimension of the desired output, and *K* represents the number of clusters in generating the dataset. GMMPLS-*K* stands for training GMMPLS with *K* clusters. Bold values indicate the highest performance in each column. The lower value is better.

Method	N=1				N=3				N=5				Overall	
	K=				K=				K=					
	1	2	3	4	1	2	3	4	1	2	3	4	mean	std
PLS	**0.21**	12.57	14.43	17.03	0.75	15.83	16.26	16.41	1.19	11.6	15.1	17.46	11.57	3.87
QPLS	0.29	15.54	19.93	20.93	**0.49**	20.19	20.61	19.39	**0.91**	16.51	19.15	21.67	14.64	5.6
BPLS	1.41	12.9	14.95	17.64	1.3	16.23	16.73	17.05	1.3	11.75	15.51	18.03	12.07	4.04
GMMPLS-1	2.04	12.58	14.55	17.17	3.01	15.69	16.14	16.44	3.68	11.67	15.01	17.31	12.11	3.97
GMMPLS-2	7.08	9.71	7.51	**6**	4.27	**12.82**	9.75	8.25	4.44	**8.86**	**11.88**	**9.66**	**8.35**	5.23
GMMPLS-3	8.88	7.6	8.84	7	5.9	13.72	11.2	8.99	5.37	9.25	**11.88**	11.4	9.17	5.03
GMMPLS-4	10.54	**6.67**	8.19	8.02	6.83	13.84	11.43	9.71	6.49	9.55	12.25	11.93	9.62	5.27
GMMPLS-5	10.44	7.41	**5.97**	7.61	7.49	15.43	**9.44**	**7.48**	6.67	10.85	12.64	10.73	9.35	5.41

This example demonstrates that the GMMPLS performed much better decoding in comparison to PLS. The achieved performances in this example were $\rho_{\text{PLS}} = [0.29, 0.79, 0.62]$, $R^2_{\text{PLS}} = [0.15, 0.54, 0.22]$, $\text{MAE}_{\text{PLS}} = [0.57, 0.49, 0.6]$ for PLS and $\rho_{\text{GMMPLS}} = [0.72, 0.95, 0.76]$, $R^2_{\text{GMMPLS}} = [0.65, 0.77, 0.53]$, $\text{MAE}_{\text{GMMPLS}} = [0.36, 0.24, 0.43]$ for GMMPLS in X, Y and Z directions. In addition, this example depicts that the proposed algorithm decoded the steady intervals in the desired output more accurately compared to PLS.

The achieved performances in this dataset were shown in Fig. 3. The correlation coefficients, the coefficient of determinations, and the mean absolute errors were given for monkeys 1 and 2 in three dimensions for PLS, QPLS, BPLS, and GMMPLS methods (mean $\pm$ standard error). All the hyper-parameters in each method were tuned using described CV technique for obtaining these results.

This figure demonstrates that except in the Z dimension, the proposed method outperformed other methods in all directions in both animals. The Friedman non-parametric test with the post-hoc Bonferroni correction was applied to the achieved performance for more rigorous analysis. This statistical test revealed that the superiority of GMMPLS performance is significant compared to PLS, QPLS, and BPLS ($p < 0.001$ for all metrics).

Finally, the overall performances in the ECoG dataset were $\rho_{\text{PLS}} = 0.55$, $R^2_{\text{PLS}} = 0.25$, $\text{MAE}_{\text{PLS}} = 0.67$ for PLS, $\rho_{\text{QPLS}} = 0.57$, $R^2_{\text{QPLS}} = 0.27$, $\text{MAE}_{\text{QPLS}} = 0.66$ for QPLS, $\rho_{\text{BPLS}} = 0.56$, $R^2_{\text{BPLS}} = 0.27$, $\text{MAE}_{\text{BPLS}} = 0.66$ for BPLS, and $\rho_{\text{GMMPLS}} = 0.63$, $R^2_{\text{GMMPLS}} = 0.41$, $\text{MAE}_{\text{GMMPLS}} = 0.55$ for GMMPLS. These results indicated that the proposed method improved performance metrics by about 15%, 12% and 13% in ρ, 64%, 52% and 52% in R^2, and 22%, 20% and 20% in MAE compared to PLS, QPLS and BPLS, respectively.

An example for explaining the effect of the number of used components (R) and the number of clusters (K) in decoding performance in the ECoG dataset is given in Fig. 4. The rank-R was varied from 0 to 20 for decoding hand trajectories, and the achieved performances were averaged over three dimensions for PLS, QPLS, BPLS, and GMMPLS. In addition, the GMMPLS was trained using a different number of clusters. i.e., from $K = 1$ to $K = 5$, which was denoted as GMMPLS-K in this example. Fig. 4 demonstrates that the performance of GMMPLS was approximately equal to PLS when $K = 1$ in lower R values. GMMPLS-2 behaved slightly better than PLS, QPLS, BPLS, and GMMPLS-1. However, the best results were achieved when the decoding was performed using $K = 3$ to $K = 5$ in GMMPLS. Finally,

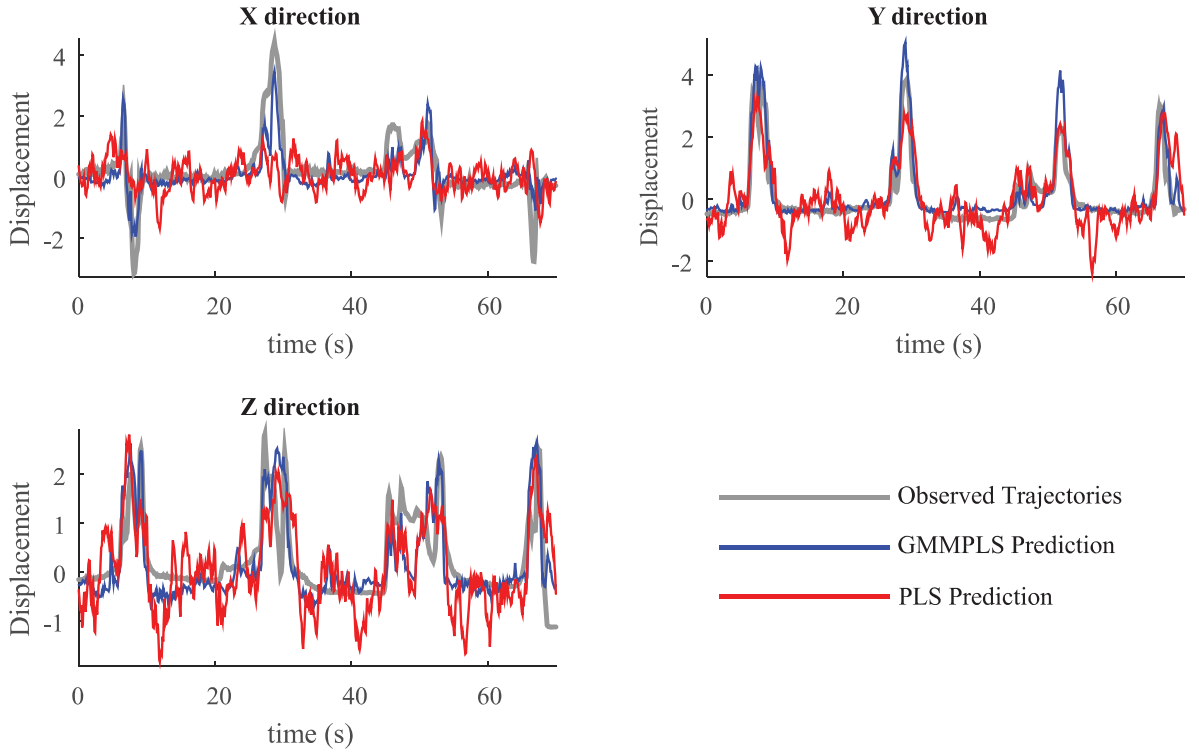

FIGURE 2. Decoding comparison between PLS and GMMPLS methods in an example time window. The hyper-parameters for each method were optimized using the CV technique for generating this figure. The optimized number of clusters in GMMPLS was K = 4, and the decomposition rank was R = 20 for both methods in this example. This figure indicates that the proposed method estimated the desired output more precisely in all three dimensions compared to PLS.

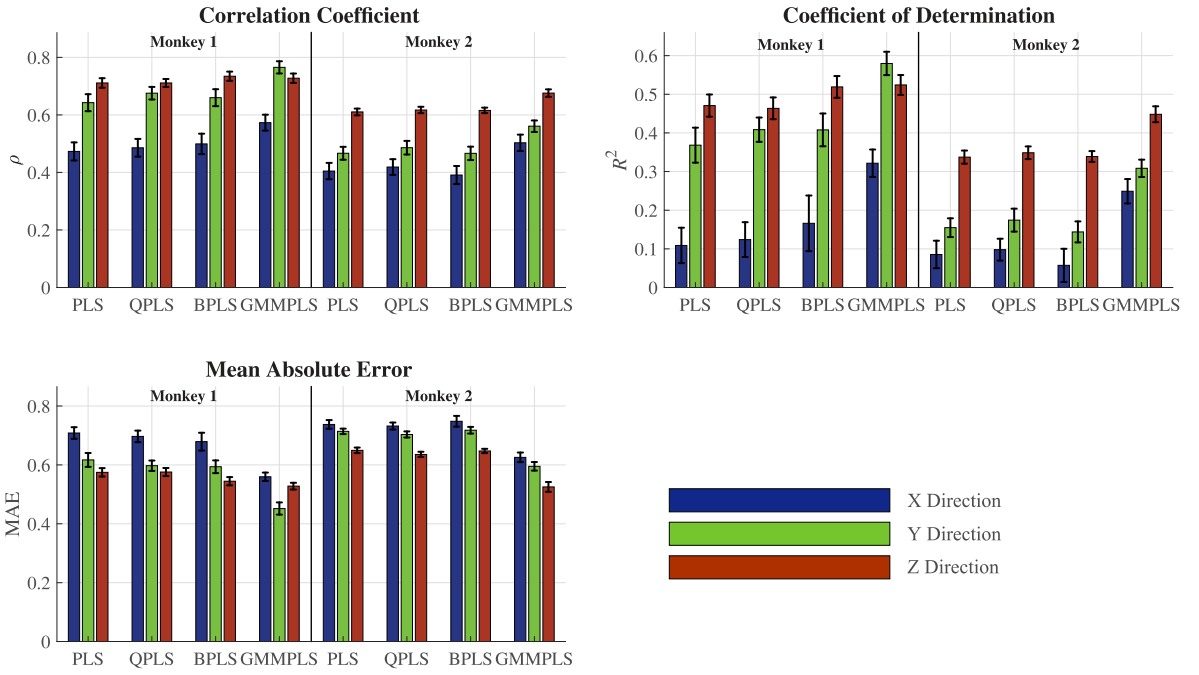

FIGURE 3. The achieved performances (mean ± standard error) for PLS, QPLS, BPLS and GMMPLS in the ECoG dataset. The left and right sides of each sub-figure represent the results for the first and second monkeys, respectively, in X, Y, and Z directions. The overall performances in this dataset were $\rho_{PLS} = 0.55$, $R^2_{PLS} = 0.25$, $MAE_{PLS} = 0.67$ for PLS, $\rho_{QPLS} = 0.57$, $R^2_{QPLS} = 0.27$, $MAE_{QPLS} = 0.66$ for QPLS, $\rho_{BPLS} = 0.56$, $R^2_{BPLS} = 0.27$, $MAE_{BPLS} = 0.66$ for BPLS, and $\rho_{GMMPLS} = 0.63$, $R^2_{GMMPLS} = 0.41$, $MAE_{GMMPLS} = 0.55$ for the proposed method.

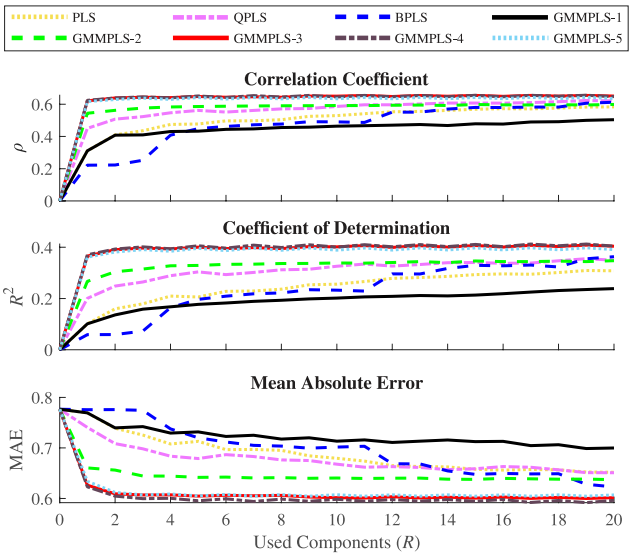

FIGURE 4. An example for illustrating the effect of the used components (R) in all decoding methods and the number of clusters (*K*) in GMMPLS for decoding hand trajectories in the ECoG dataset. The achieved performances were averaged over three dimensions for each metric. GMMPLS-*K* stands for running GMMPLS with *K* clusters.

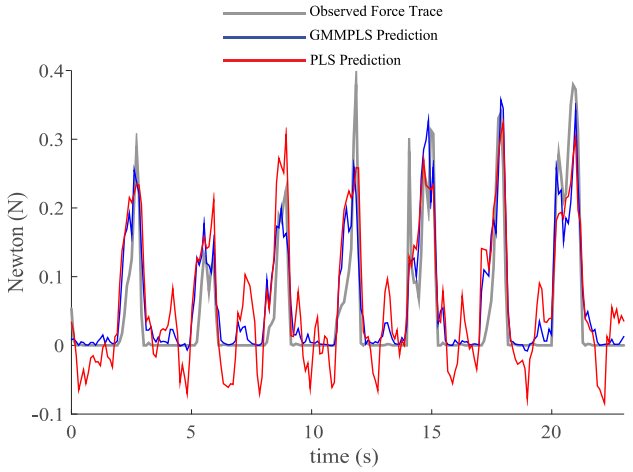

FIGURE 5. An example of decoding in the LFP dataset using PLS and GMMPLS in the first rat. All the hyper-parameters were optimized using CV technique for generating this figure. The optimized number of clusters in GMMPLS was K = 3, and the decomposition ranks were R = 10 for GMMPLS and R = 5 for PLS in this example.

this example showed that all the decoders' performances become steady for $R \geq 4$ except BPLS.

C. LFP DATASET

Fig. 5 shows an example of the decoded applied force trace using PLS and GMMPLS in the LFP dataset. To make this figure more clear, QPLS and BPLS predicted traces were not drawn. A time window (≈30) of the observed and predicted desired output in the first rat was depicted in this example. All hyper-parameters were optimized through the CV procedure. The optimized number of clusters in GMMPLS was $K = 3$, and the number of used components were $R = 5$ and $R = 10$ for PLS and GMMPLS, respectively. The gained performances in this example were $\rho_{\text{PLS}} = 0.82$, $R^2_{\text{PLS}} = 0.62$, and $\text{MAE}_{\text{PLS}} = 4.84$ for PLS and $\rho_{\text{GMMPLS}} = 0.89$, $R^2_{\text{GMMPLS}} = 0.79$ and $\text{MAE}_{\text{GMMPLS}} = 2.85$ for GMMPLS. Again similar to the ECoG dataset, it can be noticed that the proposed method won over PLS when it came to decoding steady intervals in the desired output.

Fig. 6 demonstrates the obtained performances in the LFP dataset via the correlation coefficient, the coefficient of determination, and the mean absolute error metrics for the three rats using all decoding methods. Again, the hyper-parameters were optimized through the CV procedure. From this figure, it seems that the proposed method slightly outperformed PLS, QPLS, and BPLS in all metrics. The Friedman non-parametric test with the post-hoc Bonferroni correction was employed to investigate this hypothesis. This test revealed that the proposed method's results were significantly higher than PLS, QPLS, and BPLS in all metrics ($p < 0.001$ in all metrics).

The overall mean performances in this LFP dataset were $\rho_{\text{PLS}} = 0.71$, $R^2_{\text{PLS}} = 0.46$, $\text{MAE}_{\text{PLS}} = 5.34$ for PLS, $\rho_{\text{QPLS}} = 0.71$, $R^2_{\text{QPLS}} = 0.45$, $\text{MAE}_{\text{QPLS}} = 7.64$ for QPLS, $\rho_{\text{BPLS}} = 0.72$, $R^2_{\text{BPLS}} = 0.45$, $\text{MAE}_{\text{BPLS}} = 5.20$ for BPLS, and $\rho_{\text{GMMPLS}} = 0.75$, $R^2_{\text{GMMPLS}} = 0.53$, $\text{MAE}_{\text{GMMPLS}} = 3.99$ for the proposed method. The GMMPLS performance improvements were about 6%, 6% and 4% in ρ, 15%, 18% and 15% in R^2, and 34%, 91% and 30% in MAE over PLS, QPLS and BPLS, respectively.

It is worth mentioning that the improvement in MAE was more remarkable than other metrics since the desired output in this dataset includes intervals with high-value peaks. Prediction error in these intervals causes dramatic effects in l_2-norm based performance metrics (like ρ and R^2). On the other hand, since MAE is a l_1-norm based metric, it is less sensitive to large error values. In addition, the proposed method caused more minor prediction errors in steady intervals compared to PLS, QPLS, and BPLS, and this feature led to a lower MAE.

Finally, the effect of the decomposition rank (R) and the number of clusters (K) in decoding performance for the second rat in the LFP dataset was explicated via an example in Fig. 7. The rank-R was varied from 0 to 20 for all decoding methods. A different number of clusters were used for training the proposed GMMPLS, which is denoted as GMMPLS-K. This figure demonstrates that the best performance in this example was achieved by using GMMPLS-2.

As it can be deduced from this figure, PLS, QPLS, and GMMPLS-1 behaved similarly for $R \leq 5$. However, unlike GMMPLS, the metrics R^2 and MAE achieved for PLS and QPLS fell down when more than $R = 5$ components were used for training. On the other hand, the performance of BPLS was lower than others for $R \leq 5$. Unlike PLS and QPLS, BPLS performance did not decrease for $R > 5$ and it behaved almost similar to GMMPLS-1. However, the best performances belonged to GMMPLS with $K = 2$ to $K = 5$.

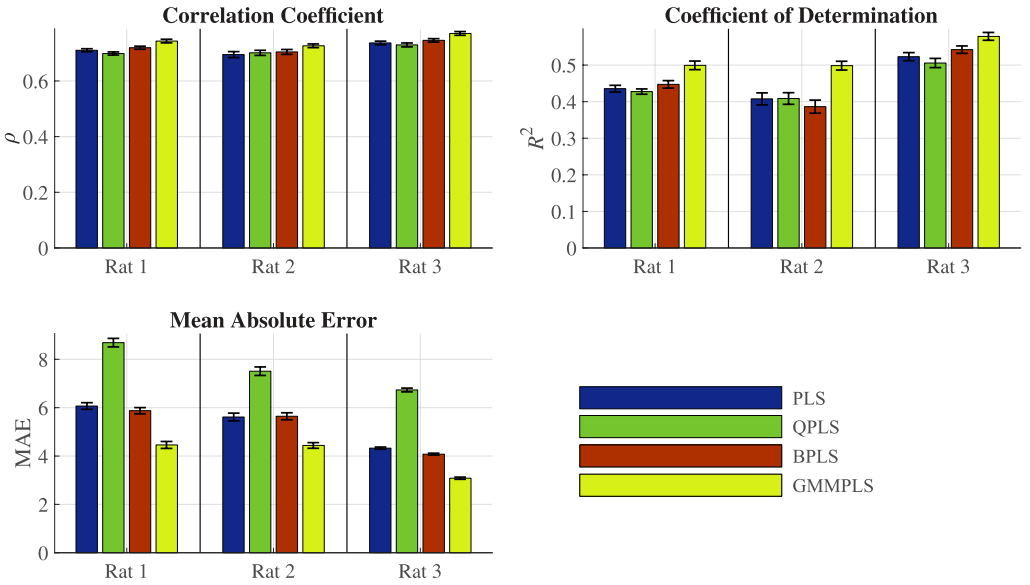

FIGURE 6. The obtained performances (mean ± standard error) for PLS, QPLS, BPLS, and GMMPLS in the LFP dataset. Symbols ρ, R^2 and MAE stand for the correlation coefficient, the coefficient of determination, and the mean absolute error metrics. The overall mean performances in this dataset were $\rho_{\text{PLS}} = 0.71$, $R^2_{\text{PLS}} = 0.46$, $\text{MAE}_{\text{PLS}} = 5.34$ for PLS, $\rho_{\text{QPLS}} = 0.71$, $R^2_{\text{QPLS}} = 0.45$, $\text{MAE}_{\text{QPLS}} = 7.64$ for QPLS, $\rho_{\text{BPLS}} = 0.72$, $R^2_{\text{BPLS}} = 0.46$, $\text{MAE}_{\text{BPLS}} = 5.20$ for BPLS, and $\rho_{\text{GMMPLS}} = 0.75$, $R^2_{\text{GMMPLS}} = 0.53$, $\text{MAE}_{\text{GMMPLS}} = 3.99$ for the proposed method.

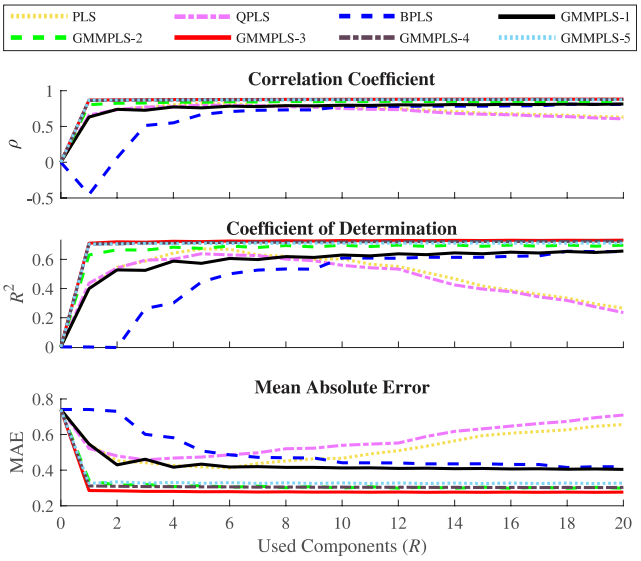

FIGURE 7. The effect of the rank of decomposition (R) in all decoding methods and the number of clusters (K) in GMMPLS for decoding applied force in the LFP dataset. GMMPLS-K stands for running GMMPLS with K clusters.

Therefore, we can claim that the proposed method is more robust to over-fitting when the number of used components is overvalued compared to PLS and QPLS.

D. RUN-TIME COMPARISON

In this sub-section, we presented the run-time analysis results as the metric of computational complexity. Table 4 presented each decoder training computational time using three minutes of LFP and ECoG datasets. Again, a different number of clusters were used for training GMMPLS, which is denoted as GMMPLS-K.

TABLE 4. Averaged computational time (s) for training decoding methods using three minutes of LFP and ECoG signals.

Method	Averaged Run Time (s)	
	LFP	ECoG
PLS	0.21	0.76
QPLS	0.22	0.82
BPLS	2.65	15.73
GMMPLS-1	0.17	0.92
GMMPLS-2	1.57	7.14
GMMPLS-3	1.67	11.32
GMMPLS-4	3.53	12.24
GMMPLS-5	5.32	17.03

It can be seen that the computational complexity of GMMPLS-1 is almost equal to PLS and QPLS. However, the run-time is increased with increasing the number of clusters (K) used in the GMMPLS structure, which is not unexpected.

V. DISCUSSION

The simulation and the experimental results represented the efficacy of the proposed method in the continuous decoding problems. This method led to higher performances in the sense of correlation coefficient, coefficient of determination, and mean absolute error compared to ordinary PLS, QPLS, and BPLS with statistical significance.

GMMPLS achieved more success than other employed methods in decoding steady intervals. This may occur because of the state-based and hierarchical nature of the GMMPLS. The proposed method divides the desired output into a specific number of clusters. Then, the membership probabilities of each sample are estimated for each cluster, and these probabilities are employed in the GMMPLS algorithm. Therefore, the aforementioned steady intervals are recognized as a specific cluster, which helps GMMPLS perform more accurately for these samples.

As mentioned before, all the previously state-based algorithms have been introduced for specific applications. Thus, they are dependent on the existence of some prior information about the structure of the desired output. On the contrary, the proposed method is more generalized for such a purpose since the clustering and the decoding procedures are fully automated.

The proposed method includes three hyper-parameters which have to tune using the CV procedure: the regularization parameter (λ), the decomposition rank (R), and the number of clusters (K). Regularization parameter (λ) is used to avoid overfitting phenomena in the ADAM algorithm. We tuned this parameter using the CV procedure through this manuscript. However, our investigation reveals that setting $\lambda = 10$ or $\lambda = 20$ may be a good choice in most scenarios. The decomposition rank is a common hyper-parameter in all variants of PLS algorithms. We showed that GMMPLS is more robust to R increment compared to PLS and QPLS in Fig. 7. On the other hand, we illustrated that the proposed method leads to higher performances in lower R compared to other employed methods.

GMMPLS with one cluster ($K = 1$) downgrades to the ordinary PLS. On the other hand, our simulation study illustrated that increasing K does not necessarily lead to improved performances. In other words, choosing optimal K is depended on the structure of the desired output, and it should be tuned using a CV procedure.

In the end, we used a simple logistic regression with the regularized ADAM optimizer to decode the membership probabilities. However, based on our experiments, we informed that the membership probability estimation in GMMPLS significantly influences decoding precision. Therefore, improving this estimation will be the goal of our future studies.

It should be noted that the MATLAB code of the proposed GMMPLS is available upon request from the authors.

VI. CONCLUSION

This paper presents a novel fully automated state-based decoder called GMMPLS for continuous decoding in various BMI applications. Unlike the other state-based decoders, GMMPLS does not rely on prior information about the desired output structure. We illustrated and evaluated the generalization of GMMPLS in two different real-world datasets and a synthetic dataset. In all cases, the proposed method outperformed the ordinary PLS, quadratic PLS (QPLS), and Bayesian PLS (BPLS) in terms of decoding correlation coefficient, coefficient of determination, and mean absolute error metrics.

APPENDIX

NIPALS algorithm for PLS is outlined in Algorithm 3. Algorithm 4 explains the detail of the GMM method imple-

Algorithm 3 PLS Algorithm for Estimating Desired $\mathbf{Y}$ From $\mathbf{X}$

Input: $\mathbf{X} \in \mathbb{R}^{L\times M}$, $\mathbf{Y} \in \mathbb{R}^{L\times N}$,
number of PLS components: R
Output: $\mathbf{T} = [\mathbf{t}_1, \cdots, \mathbf{t}_R] \in \mathbb{R}^{L\times R}$,
$\mathbf{P} = [\mathbf{p}_1, \cdots, \mathbf{p}_R] \in \mathbb{R}^{M\times R}$,
$\mathbf{Q} = [\mathbf{q}_1, \cdots, \mathbf{q}_R] \in \mathbb{R}^{N\times R}$,
$\mathbf{W} = [\mathbf{q}_1, \cdots, \mathbf{q}_R] \in \mathbb{R}^{M\times R}$.
$\mathbf{D} = \text{diag}\,(d_1, \cdots, d_R) \in \mathbb{R}^{R\times R}$ and $\mathbf{B} \in \mathbb{R}^{M\times N}$
Assign $\mathbf{X}_1 = \mathbf{X}$ and $\mathbf{Y}_1 = \mathbf{Y}$
for r **to** R **do**
  Initial $\mathbf{q}_r$
  while not convergence do
    $\mathbf{u}_r = \mathbf{Y}_r\mathbf{q}_r$
    $\mathbf{w}_r = \mathbf{X}_r^T\mathbf{u}_r \big/ \left(\mathbf{u}_r^T\mathbf{u}_r\right)$
    $\mathbf{w}_r \leftarrow \mathbf{w}_r \big/ \|\mathbf{w}_r\|_2$
    $\mathbf{t}_r = \mathbf{X}_r\mathbf{w}_r$
    $\mathbf{q}_r = \mathbf{Y}_r^T\mathbf{t}_r \big/ \left(\mathbf{t}_r^T\mathbf{t}_r\right)$
    $\mathbf{q}_r \leftarrow \mathbf{q}_r \big/ \|\mathbf{q}_r\|_2$
  end while
  $\mathbf{p}_r = \mathbf{X}_r^T\mathbf{t}_r \big/ \left(\mathbf{t}_r^T\mathbf{t}_r\right)$
  $d_r = \mathbf{u}_r^T\mathbf{t}_r \big/ \left(\mathbf{t}_r^T\mathbf{t}_r\right)$
  $\mathbf{X}_{r+1} \leftarrow \mathbf{X}_r - \mathbf{t}_r\mathbf{p}_r^T$
  $\mathbf{Y}_{r+1} \leftarrow \mathbf{Y}_r - d_r\mathbf{t}_r\mathbf{q}_r^T$
end for
$\mathbf{B} = \mathbf{W}\left(\mathbf{P}^T\mathbf{W}\right)^{-1}\mathbf{D}\mathbf{Q}^T \in \mathbb{R}^{M\times N}$

Algorithm 4 GMM Algorithm for Estimating Membership Probability

Input: $\mathbf{Y} = [\mathbf{y}(1), \cdots, \mathbf{y}(L)]^T \in \mathbb{R}^{L\times N}$
number of clusters: K
Output: $\mathbf{\Gamma} = \{\gamma_k(l)\} \in \mathbb{R}^{L\times K}$
Initialization the means $\boldsymbol{\mu}_k \in \mathbb{R}^N$, covariance $\mathbf{\Sigma}_k \in \mathbb{R}^{N\times N}$ using k-means algorithm and the mixing coefficients $\pi_k = 1$ for $k = 1, \cdots, K$
while not convergence do
  E-step

$$\gamma_k(l) = \frac{\pi_k \mathcal{N}\left(\mathbf{y}(l)\,\middle|\,\boldsymbol{\mu}_k, \mathbf{\Sigma}_k\right)}{\sum_{j=1}^{K}\pi_j \mathcal{N}\left(\mathbf{y}(l)\,\middle|\,\boldsymbol{\mu}_j, \mathbf{\Sigma}_j\right)}$$

  where $\mathcal{N}(\cdot)$ is the Gaussian distribution operator
  M-step

$$\boldsymbol{\mu}_k^{new} = \frac{1}{s_k}\sum_{l=1}^{L}\gamma_k(l)\,\mathbf{y}(l)$$

$$\mathbf{\Sigma}_k^{new} = \frac{1}{s_k}\sum_{l=1}^{L}\gamma_k(l)\left(\mathbf{y}(l)-\boldsymbol{\mu}_k^{new}\right)\left(\mathbf{y}(l)-\boldsymbol{\mu}_k^{new}\right)^T$$

$$\pi_k^{new} = \frac{s_k}{L}$$

  where $s_k = \sum_{l=1}^{L}\gamma_k(l)$
end while

mentation. Note that we used the K-means for initializing the GMM algorithm. In addition, the z-scored version of the desired output is used as the input of this algorithm to prevent the effect of data magnitudes in learning the GMM model.

Algorithm 5 demonstrates the ADAM algorithm with weight decay. In this algorithm, β_1 and β_2 are exponential decay rates for the moment estimates, ε is a small constant, α is a step size parameter, and λ is the regularization hyper-parameter. Choosing these parameters is explained in section III.

Algorithm 5 Adam Optimization With Weight Decay

Input: $\mathbf{X} \in \mathbb{R}^{L\times(M+1)}$, $\boldsymbol{\gamma}_k \in \mathbb{R}^L$
Hyperparameter scalers β_1, β_2 and ε
Regularization hyperparameter λ
Step size α
Output: $\mathbf{h}_k \in \mathbb{R}^{M+1}$
Initialize parameter vector: $\mathbf{h}_k\,(0)$
Initialize iteration: $t \leftarrow 0$
Initialize 1st moment vector: $\mathbf{m}\,(0) \leftarrow 0$
Initialize 2st moment vector: $\mathbf{v}\,(0) \leftarrow 0$
while not convergence do
$t \leftarrow t+1$
calculate $\mathbf{g}_k = \nabla_{\mathbf{h}_k} \mathrm{J}\,(\mathbf{h}_k\,(t))$ from eq. (12)
$\mathbf{m}\,(t) = \beta_1 \mathbf{m}\,(t-1) + (1-\beta_1)\,\mathbf{g}_k$
$\mathbf{v}\,(t) = \beta_2 \mathbf{v}\,(t-1) + (1-\beta_2)\,\mathbf{g}_k^2$
$\hat{\mathbf{m}}\,(t) = \mathbf{m}\,(t) \big/ (1-\beta_1)$
$\hat{\mathbf{v}}\,(t) = \mathbf{v}\,(t) \big/ (1-\beta_2)$
$\mathbf{h}_k(t) = \mathbf{h}_k(t-1) - \alpha\left(\hat{\mathbf{m}}(t) \Big/ \left(\sqrt{\hat{\mathbf{v}}\,(t)} + \varepsilon\right) + \lambda \mathbf{h}_k(t-1)\right)$
end while

REFERENCES

[1] M. Xu, J. Han, Y. Wang, T.-P. Jung, and D. Ming, "Implementing over 100 command codes for a high-speed hybrid brain-computer interface using concurrent P300 and SSVEP features," *IEEE Trans. Biomed. Eng.*, vol. 67, no. 11, pp. 3073–3082, Nov. 2020.

[2] N. Fatima, A. Shuaib, and M. Saqqur, "Intra-cortical brain-machine interfaces for controlling upper-limb powered muscle and robotic systems in spinal cord injury," *Clin. Neurol. Neurosurg.*, vol. 196, Sep. 2020, Art. no. 106069.

[3] E. Nsugbe, "Brain-machine and muscle-machine bio-sensing methods for gesture intent acquisition in upper-limb prosthesis control: A review," *J. Med. Eng. Technol.*, vol. 45, no. 2, pp. 115–128, Feb. 2021.

[4] Z. Li, J. Li, S. Zhao, Y. Yuan, Y. Kang, and C. L. P. Chen, "Adaptive neural control of a kinematically redundant exoskeleton robot using brain–machine interfaces," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 12, pp. 3558–3571, Dec. 2019.

[5] D. F. Collazos-Huertas, A. M. Álvarez-Meza, and G. Castellanos-Dominguez, "Spatial interpretability of time-frequency relevance optimized in motor imagery discrimination using deep & wide networks," *Biomed. Signal Process. Control*, vol. 68, Jul. 2021, Art. no. 102626.

[6] N. Yan, C. Wang, Y. Tao, J. Li, K. Zhang, T. Chen, Z. Yuan, X. Yan, and G. Wang, "Quadcopter control system using a hybrid BCI based on off-line optimization and enhanced human-machine interaction," *IEEE Access*, vol. 8, pp. 1160–1172, 2020.

[7] Y. Huang, J. Jin, R. Xu, Y. Miao, C. Liu, and A. Cichocki, "Multi-view optimization of time-frequency common spatial patterns for brain-computer interfaces," *J. Neurosci. Methods*, vol. 365, Jan. 2022, Art. no. 109378.

[8] N. Ahmadi, T. G. Constandinou, and C.-S. Bouganis, "Robust and accurate decoding of hand kinematics from entire spiking activity using deep learning," *J. Neural Eng.*, vol. 18, no. 2, Apr. 2021, Art. no. 026011.

[9] R. D. Flint, J. M. Rosenow, M. C. Tate, and M. W. Slutzky, "Continuous decoding of human grasp kinematics using epidural and subdural signals," *J. Neural Eng.*, vol. 14, no. 1, Feb. 2017, Art. no. 016005.

[10] K. Shimoda, Y. Nagasaka, Z. C. Chao, and N. Fujii, "Decoding continuous three-dimensional hand trajectories from epidural electrocorticographic signals in Japanese macaques," *J. Neural Eng.*, vol. 9, no. 3, Jun. 2012, Art. no. 036015.

[11] A. Khorasani, N. H. Beni, V. Shalchyan, and M. R. Daliri, "Continuous force decoding from local field potentials of the primary motor cortex in freely moving rats," *Sci. Rep.*, vol. 6, no. 1, pp. 1–10, Dec. 2016.

[12] C. Chen, D. Shin, H. Watanabe, Y. Nakanishi, H. Kambara, N. Yoshimura, A. Nambu, T. Isa, Y. Nishimura, and Y. Koike, "Prediction of hand trajectory from electrocorticography signals in primary motor cortex," *PLoS ONE*, vol. 8, no. 12, Dec. 2013, Art. no. e83534.

[13] J. Jin, C. Liu, I. M. Y. Daly, S. Li, X. Wang, and A. Cichocki, "Bispectrum-based channel selection for motor imagery based brain-computer interfacing," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 28, no. 10, pp. 2153–2163, Oct. 2020.

[14] M. R. Nazari, A. M. Nasrabadi, and M. R. Daliri, "Single-trial decoding of motion direction during visual attention from local field potential signals," *IEEE Access*, vol. 9, pp. 66450–66461, 2021.

[15] A. Khorasani, R. Foodeh, V. Shalchyan, and M. R. Daliri, "Brain control of an external device by extracting the highest force-related contents of local field potentials in freely moving rats," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 26, no. 1, pp. 18–25, Jan. 2018.

[16] A. Khorasani, V. Shalchyan, and M. R. Daliri, "Adaptive artifact removal from intracortical channels for accurate decoding of a force signal in freely moving rats," *Frontiers Neurosci.*, vol. 13, p. 350, Apr. 2019.

[17] H. Choi, J. Lee, J. Park, S. Lee, K.-H. Ahn, I. Y. Kim, K.-M. Lee, and D. P. Jang, "Improved prediction of bimanual movements by a two-staged (effector-then-trajectory) decoder with epidural ECoG in nonhuman primates," *J. Neural Eng.*, vol. 15, no. 1, Feb. 2018, Art. no. 016011.

[18] R. Foodeh, S. Ebadollahi, and M. R. Daliri, "Regularized partial least square regression for continuous decoding in brain-computer interfaces," *Neuroinformatics*, vol. 18, no. 3, pp. 465–477, Jun. 2020.

[19] R. Foodeh, A. Khorasani, V. Shalchyan, and M. R. Daliri, "Minimum noise estimate filter: A novel automated artifacts removal method for field potentials," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 25, no. 8, pp. 1143–1152, Aug. 2017.

[20] S. Morishita, K. Sato, H. Watanabe, Y. Nishimura, T. Isa, R. Kato, T. Nakamura, and H. Yokoi, "Brain-machine interface to control a prosthetic arm with monkey ECoGs during periodic movements," *Frontiers Neurosci.*, vol. 8, p. 417, Dec. 2014.

[21] A. Eliseyev and T. Aksenova, "Stable and artifact-resistant decoding of 3D hand trajectories from ECoG signals using the generalized additive model," *J. Neural Eng.*, vol. 11, no. 6, Dec. 2014, Art. no. 066005.

[22] N. H. Beni, R. Foodeh, V. Shalchyan, and M. R. Daliri, "Force decoding using local field potentials in primary motor cortex: PLS or Kalman filter regression?" *Phys. Eng. Sci. Med.*, vol. 43, no. 1, pp. 175–186, Mar. 2020.

[23] Z. C. Chao, Y. Nagasaka, and N. Fujii, "Long-term asynchronous decoding of arm motion using electrocorticographic signals in monkey," *Frontiers Neuroeng.*, vol. 3, p. 3, Mar. 2010.

[24] B. Farrokhi and A. Erfanian, "A state-based probabilistic method for decoding hand position during movement from ECoG signals in non-human primate," *J. Neural Eng.*, vol. 17, no. 2, May 2020, Art. no. 026042.

[25] M. A. J. van Gerven, Z. C. Chao, and T. Heskes, "On the decoding of intracranial data using sparse orthonormalized partial least squares," *J. Neural Eng.*, vol. 9, no. 2, Apr. 2012, Art. no. 026017.

[26] S. Wold, H. Ruhe, H. Wold, and W. Dunn, "The collinearity problem in linear regression. The partial least squares (PLS) approach to generalized inverses," *J. Sci. Stat. Comput.*, vol. 5, no. 3, pp. 735–743, Jan. 1984.

[27] M. Kashefi and M. R. Daliri, "A stack LSTM structure for decoding continuous force from local field potential signal of primary motor cortex (M1)," *BMC Bioinf.*, vol. 22, no. 1, pp. 1–19, Dec. 2021.

[28] R. Fazai, M. Mansouri, K. Abodayeh, H. Nounou, and M. Nounou, "Online reduced kernel PLS combined with GLRT for fault detection in chemical systems," *Process Saf. Environ. Protection*, vol. 128, pp. 228–243, Aug. 2019.

[29] H. Liu and X. Sun, "A partial least squares based ranker for fast and accurate age estimation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2016, pp. 2792–2796.
[30] X. Liu and S. Zhou, "Approximate kernel partial least squares," *Ann. Math. Artif. Intell.*, vol. 88, no. 9, pp. 973–986, Sep. 2020.
[31] K. Volkova, M. A. Lebedev, A. Kaplan, and A. Ossadtchi, "Decoding movement from electrocorticographic activity: A review," *Frontiers Neuroinform.*, vol. 13, p. 74, Dec. 2019.
[32] D. T. Bundy, M. Pahwa, N. Szrama, and E. C. Leuthardt, "Decoding three-dimensional reaching movements using electrocorticographic signals in humans," *J. Neural Eng.*, vol. 13, no. 2, Apr. 2016, Art. no. 026021.
[33] A. Ahmadi, A. Khorasani, V. Shalchyan, and M. R. Daliri, "State-based decoding of force signals from multi-channel local field potentials," *IEEE Access*, vol. 8, pp. 159089–159099, 2020.
[34] S. Wold, N. Kettaneh-Wold, and B. Skagerberg, "Nonlinear PLS modeling," *Chemometrics Intell. Lab. Syst.*, vol. 7, nos. 1–2, pp. 53–65, 1989.
[35] D. Vidaurre, M. A. J. van Gerven, C. Bielza, P. Larrañaga, and T. Heskes, "Bayesian sparse partial least squares," *Neural Comput.*, vol. 25, no. 12, pp. 3318–3339, Dec. 2013.
[36] P. Geladi and B. R. Kowalski, "Partial least-squares regression: A tutorial," *Anal. Chim. Acta*, vol. 185, pp. 1–17, Jan. 1986.
[37] R. Rosipal and N. Krämer, "Overview and recent advances in partial least squares," in *Proc. Int. Stat. Optim. Perspect. Workshop*, 2005, pp. 34–51.
[38] I. S. Helland, "On the structure of partial least squares regression," *Commun. Statist.-Simul. Comput.*, vol. 17, no. 2, pp. 581–607, 1988.
[39] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer-Verlag, 2006
[40] T. M. Nguyen and Q. M. J. Wu, "Gaussian-mixture-model-based spatial neighborhood relationships for pixel labeling problem," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 1, pp. 193–202, Feb. 2012.
[41] C. Chokwitthaya, Y. Zhu, S. Mukhopadhyay, and A. Jafari, "Applying the Gaussian mixture model to generate large synthetic data from a small data set," in *Proc. Construct. Res. Congr.*, 2020, pp. 1251–1260.
[42] K. Y. Lee, "Local fuzzy PCA based GMM with dimension reduction on speaker identification," *Pattern Recognit. Lett.*, vol. 25, no. 16, pp. 1811–1817, Dec. 2004.
[43] V. K. Ayyadevara, "Logistic regression," in *Pro Machine Learning Algorithms*. Berkeley, CA, USA: Apress, 2018, pp. 49–69.
[44] Y. Bengio, N. Le Roux, P. Vincent, O. Delalleau, and P. Marcotte, "Convex neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 18, Apr. 2006, p. 123.
[45] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[46] I. Loshchilov and F. Hutter, "Fixing weight decay regularization in Adam," in *Proc. ICLR*, 2018, pp. 1–4.
[47] T. Chai and R. Draxler, "Root mean square error (RMSE) or mean absolute error (MAE)," *Geosci. Model Develop. Discuss.*, vol. 7, no. 1, pp. 1525–1534, 2014.

REZA FOODEH was born in Esfahan, Iran, in 1990. He received the M.S. degree in biomedical engineering from the Department of Electrical Engineering, Iran University of Science & Technology (IUST), Tehran, Iran, in 2016, where he is currently pursuing the Ph.D. degree in electrical engineering. His research interests include neural signal processing, brain–computer interfaces, and machine learning.

VAHID SHALCHYAN received the M.Sc. degree in biomedical engineering from the Amirkabir University of Technology, Tehran, Iran, in 2002, and the Ph.D. degree in biomedical science and engineering from Aalborg University, Aalborg, Denmark, in 2013. From 2011 to 2013, he was a Visiting Researcher with the University Medical Center Göttingen, Georg-August University, Göttingen, Germany. He has been working as an Assistant Professor with the Department of Biomedical Engineering, School of Electrical Engineering, Iran University of Science & Technology (IUST), Tehran. His main research interests include biomedical signal processing and pattern recognition, with emphasis on their application to neural signals, for neuroscience, neurotechnology, and brain–computer interface researches.

MOHAMMAD REZA DALIRI (Member, IEEE) received the M.Sc. degree in medical radiation engineering from the Amirkabir University of Technology, Tehran, Iran, in 2001, and the Ph.D. degree in cognitive neuroscience from the International School for Advanced Studies (SISSA/ISAS), Trieste, Italy, in 2007. From 2007 to 2009, he was a Postdoctoral Fellow with the International School for Advanced Studies (SISSA/ISAS), Trieste, and the German Primate Center (DPZ), Göttingen, Germany. He is currently a Full Professor with the Department of Biomedical Engineering, School of Electrical Engineering, Iran University of Science & Technology (IUST), Tehran. His main research interests include neural signal processing, brain–computer interfaces, computational and cognitive neuroscience, pattern recognition, and computer vision.