A New Architecture for Deriving Dynamic Brain-Machine Interfaces

José Fortes[1], Renato Figueiredo[1], Linda Hermer-Vazquez[2], José Príncipe[1] and Justin C. Sanchez[3]

[1] Dep. of Electrical and Computer Engineering, [2] Dep. of Psychology, [3] Dep. of Pediatrics
University of Florida, Gainesville, Florida, USA
{fortes, renatof, lindahv, principe, jcs77}@ufl.edu

Abstract. Great potential exists for future Brain Machine Interfaces (BMIs) to help paralyzed patients, and others with motor disabilities, regain (artificial) motor control and autonomy. This paper describes a novel approach towards the development of new design architectures and research test-beds for advanced BMIs. It addresses a critical design challenge in deriving the functional mapping between the subject's movement intent and actuated behavior. Currently, adaptive signal processing techniques are used to correlate neuronal modulation with known movements generated by the subject. However, with patients who are paralyzed, access to the individual's movement is unavailable. Inspired by motor control research, this paper considers a predictive framework for BMI using multiple adaptive models trained with supervised or reinforcement learning in a closed-loop architecture that requires real-time feedback. Here, movement trajectories can be inferred and incrementally updated using instantaneous knowledge of the movement target and the individual's current neuronal activation. In this framework, BMIs require a computing infrastructure capable of selectively executing multiple models on the basis of signals received by and/or provided to the brain in real time. Middleware currently under investigation to provide this data-driven dynamic capability is discussed.

1 Introduction

The seamless integration of brain and body in the healthy human makes us forget that, in some diseases, brains can be deprived of sensory inputs (e.g. vision and audition) or even become isolated from the body (e.g., due to traumatic injury to the spinal cord). These diseases can be mitigated by Brain Machine Interfaces (BMIs) which can be used to deliver inputs from artificial sensors to the appropriate brain cortices and/or use brain signals to directly command and control external robotic devices (see Fig. 1). The focus of this paper is on BMIs for motor control, impressive examples of which include recent demonstrations of monkeys using brain signals captured by electrodes to control computer cursors and robotic arms [1]. Given the brain plasticity, with sensory feedback, external devices may be assimilated and become a part of the user's cognitive space, on a par with body parts. Great potential exists thus for BMIs to help paralyzed patients, and others with motor disabilities, control devices by the intention of movement and regain lost autonomy.

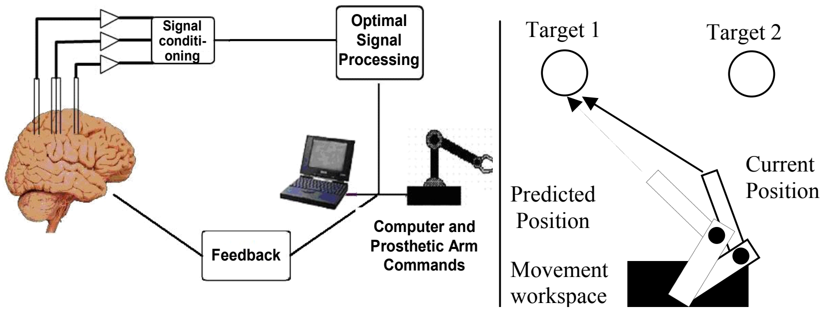

Fig. 1. *(Left)*: BMIs use brain signals to control devices such as computers and robots. Processing is needed to simultaneously execute and adapt the control algorithms as a function of sensed data. *(Right)*: Predictive trajectories based upon neuronal activation and movement kinematics.

The potential of BMIs can only be fulfilled with computer-based systems for not only implementing the decoding algorithms and robotic control but also for improving our understanding of brain function thru modeling. In the specific case of the BMIs, the role of the computer(s) is to dynamically learn and select multiple motor planning and control models to adaptively compute motor control inputs on the basis of brain signals and sensorial feedback. *This paper discusses ideas for facing the challenges of building and using Dynamic Data Driven computing systems to advance the state of the art of research on BMIs.* We use the abbreviation DDDBMI to denote this type of systems.

Extant BMI designs and experiments (e.g. for rodents and monkeys) rely on the subjects' ability to move their limbs (to train the computational models). Through the training of a model (linear adaptive filter or neural network), an optimization criteria such as the mean square error (MSE) is used to minimize the difference between the model output and known movement trajectory (desired response) generated by the subject. In contrast, BMIs that are tested with paralyzed individuals do not have access to the movement trajectory (desired response used for training) because the patient cannot move. One alternative approach to this problem is to incrementally update a *set* of possible movement trajectories using knowledge of the individual's current neuronal activation and the movement target as shown in Fig. 1 (right). Here the goal is to move the robot actuator to one of two targets. However the most appropriate trajectory that is related to the generation of neural activity (intent) is unknown.

We are seeking to model the control loop involving the motor cortex, spinal cord and muscle actuators with greater realism using recent theories of motor control neurophysiology. In this framework, much more intelligence and computing power is required from the assistive device than in present approaches. One must define and choose the set of incremental predictive positions in real-time. Moreover, the set of available trajectories should be seamlessly perceived by the patients. This means that the BMI should operate with low latency and should produce trajectories that are biologically compatible with commands generated by the motor cortex. To avoid instability-causing delays in the sensory feedback loop, the training of the models must be done in a predictive framework using supervised or reinforcement learning.

Due to the demand-based, highly dynamical and distributed nature of brain processing, realistic modeling is nontrivial, involving real-time low-latency computation of many adaptive models that need to be turned on and off throughout the task. Therefore, BMI design and performance will be significantly impacted, in feasibility and capability, by on-demand data-driven adaptation and computation of multiple models of brain learning and motor control. This presents stringent real-time data-driven computational demands which, while potentially deliverable through Grid-computing, require new middleware to be researched and developed. While online DDDBMIs are challenging, offline processing is also needed to replay experiments and conduct ancillary modeling studies using "extended" time. They too require significant compute and storage resources, and middleware for dynamically adaptive computation.

2 Envisioned DDDBMI Architecture

Pioneering research in motor physiology suggested the existence of *internal models* of motor control. A similar model-based approach to motor control, strongly inspired by the works of Kawato and Wolpert [2][3], is the basis of our design methodology for enhanced BMIs. The first key observation is that a goal-directed movement can be formulated as a sophisticated feedback control problem suggesting that humans visually monitor kinematic events to detect errors in execution (visual cortex), and the parietal cortex appears to be involved in visuomotor aspects of manual manipulative movements. Feedforward control is a skill learned by self-observation of feedback-controlled movements, which involves inverse model learning (e.g. feedback error learning). According to this model, action recognition is mediated by unconscious or implicit mental movement simulation that is implemented in the premotor cortex.

When motor mechanisms are impaired and need to be replaced by assistive technology (e.g. an artificial limb), computational modeling requires particular knowledge of how the cortex may adapt to the new assistive technology. It will be necessary for the different brain areas to relearn how to issue and sequence commands to properly control motion of the assistive device(s). Although the brain plasticity is extraordinary, the assistive device (robotic) interface should act as close as possible to the normal physiology to decrease the training times and preserve as much as possible the accuracy of human movement. We propose to design the interface by specifying the details of the feedforward control loop (in premotor cortex) and its relation with the movement execution in the primary motor cortex. For this purpose, we use a concept called the Multiple Paired Forward-Inverse Model (MPFIM) proposed by Wolpert & Kawato. The MPFIM consists of multiple pairs of models, each comprising a forward model (for movement planning in the premotor cortex) and an inverse model (for movement execution in the primary motor cortex). Individual *model-pairs* or combinations of model-pairs are used to control motion on the basis of real-time feedback data from sensors (visual or proprioceptive).

The left side of Fig. 2 represents in more detail the function of the premotor cortex (forward model, likelihood model and responsibility predictor), where we can recognize the input from prefrontal cortex (efferent copy, and contextual signal), and the visual feedback coming from the parietal cortex. The right side of the figure contains details of the motor cortex (inverse model) and control scheme. The body moves with its own feedback control loop implemented by an inverse model that receives inputs from the premotor cortex (specifying the desired trajectory) and is self adjusted by the feedback command created by proprioceptive feedback. However, a scheme to select the forward model is still needed. This is accomplished by computing locally the errors of each model (likelihood model) and weighting them across all the models (responsibility predictor). This scheme can be thought as a global weighting gate in a so-called mixture-of-experts paradigm. It is responsible for one of the dynamic data-driven aspects of the DDDBMI architecture. Since each model is a linear filter or a neural network, further adaptation of parameters may be necessary as a result of the input. Our approach uses MPFIM as the basis of DDDBMIs, where, some of the signals of Fig. 2 are generated by the motor, premotor and parietal cortices, and other signals are generated by the robot controller and movement sensors, while the computational components are implemented using Grid computing resources. The premotor and motor cortex electrodes supply the efferent copy and the desired trajectory respectively, while the posterior parietal provides the contextual signals of Fig. 2. Movement sensors (e.g. a camera and object recognition software) will supply the desired trajectory (sensory feedback signals) of Fig. 1.

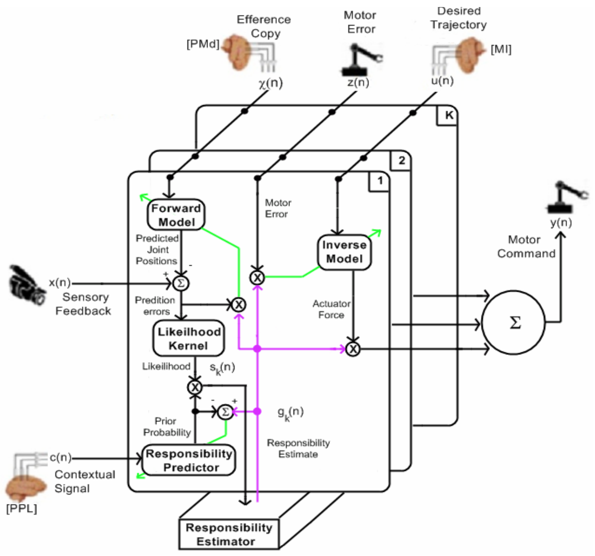

Fig. 2. (Adapted from [2]). The robot controller receives the feedforward motor command and provides the feedback motor command from position sensors

Our approach calls for a *symbiotic simultaneous adaptation* of both the patient motor cortex and the computational models. Since the architecture is anthropomimetic, we conjecture that the patient will be able to learn how to modify his or her brain activity such that it can control the switching of the computational models and ultimately learn the dynamics of the robotic arm. Since the signals for adaptation are being generated on line, we propose to adapt the models in a predictive framework using either a mixture model or a full blown sequential Bayesian estimation. This will allow the models to track changes in neuronal importance over time. Since the proposed BMI architecture mixes brain signals with robotic control signals, we consider types of models, competitive frameworks and several approaches to the training of individual models that are distinct from those used for other BMIs.

3 Algorithms and Systems Software for Training Mixture Models

We now turn to the question of how to train the gating mechanism (i.e. the individual likelihood models and the responsibility estimator) and forward inverse models of Fig. 2. Let K denote the number of forward models and N denote the number of samples (which define a time series) of the sensory feedback signal x(n) during a given time window. The vector of N samples in the rth window is represented by the vector $\mathbf{X}_r$. $\chi(n)$ and the corresponding vector $\boldsymbol{\chi}_\mathbf{n}$ denote the input to all the models (efferent copy). We choose the free parameters of the predictors (i.e. forward models) and gate that maximize the process log-likelihood.

We consider a system consisting of one hundred paired (inverse controller – forward) linear models, in a mixture-model competitive configuration. The size of this system can be estimated by the following back-of-the-envelope computation. Each model is a MIMO (Multiple Input Multiple Output) system, with one output per coordinate of the output space, and with a number of inputs given by the number of neurons being monitored in the motor cortex. To predict the next position, one needs to use 500 milliseconds of neuronal firing data (which are binned every 50 ms); therefore, the number of parameters is given by ten times the product of the number of neurons by the number of outputs. For 100 neurons this results in M=3,000 parameters per linear model, leading to a total of 300,000 parameters total for 100 models. The parameters need to be continuously adapted in real time.

DDDBMI systems for the envisioned applications will have computational demands that exceed the capacity of a single workstation. Grid-computing infrastructures can potentially deliver these necessary resources on demand. However, a DDDBMI system has stringent real-time requirements, as a result of the need for low latency between brain signaling and sensory feedback. We are developing an infrastructure (using In-VIGO[1] Grid middleware) that can aggregate resources, create appropriate parallel execution environments and also guarantee the necessary Quality-of-Service (QoS) in computation and communication to meet response deadlines associated with the above-mentioned latencies. Known parallel implementations of the BMI algorithms will be deployed and managed by this Grid middleware every time a BMI research experiment needs to be done.

The computational component of the proposed DDDBMI architecture is responsible for selectively computing and adapting all the models according to the control flow shown in Fig. 2, and the data signals and algorithms discussed above. The simplified diagram in Fig. 3 shows the broad computational steps associated with

[1] In-VIGO version 1.0 and a prototype portal have been operational and available for public use at http://invigo.acis.ufl.edu since 7/2002. A detailed discussion of its design appears in [4].

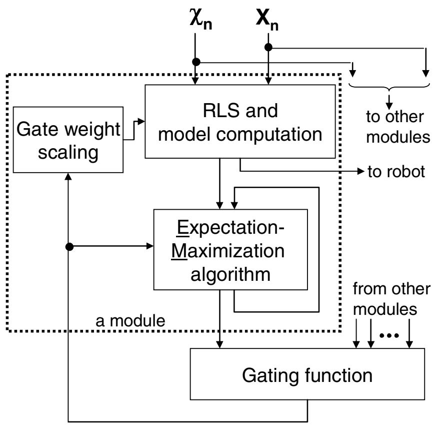

Fig. 3. Procedure for (1) computing and gating models; and (2) adapting weights and contributions of individual models as a function of efferent copy and sensory signals *(U_n, χ_n and X_n)*

each one of the K sets of models in Fig. 2 when linear models are used. Nonlinear models require additional computation but, for simplicity of exposition, we will not consider them in this part of the paper. As apparent from Fig. 3, with exception of the gating function, all computation is local to a set of models. We will refer to this local computation as a module (it roughly corresponds to an "expert" in the previously-discussed "mixture-of-experts" paradigm). Overall computation starts with the broadcast of $\mathbf{U_n}$, χ_n and $\mathbf{X_n}$ to all modules, followed by the independent computation of as many as K modules in parallel. Each module generates two kinds of outputs. One is combined with similar outputs from other modules to generate the three new position coordinates for the robot controller. This output is generated by applying the learned models to the inputs and is hereon referred to as "model computation". The second kind of output is used by the gating function, which gets similar inputs from other modules in order to determine which modules need to be used to process future inputs, and how the outputs of the modules should be combined to drive the robotic device. The overall computation is dynamically data-driven because the training computation determines through the gating function which modules should be computed and how to weight their outputs.

The model computation and the training computation can be done in parallel for all models. Both occur periodically with new input vectors (corresponding to windows of brain and sensory signal samples) arriving every 50 ms. Each model computation can be computed independently, requires O(M) floating-point multiplications and must be done in less than 50 ms. This is a hard deadline in the sense that live subjects cannot perform in the presence of longer sensory feedback times. Each learning computation can also be computed independently and requires O(M^2) multiplications due to the Recursive Least Squares (RLS) step. The time spent in the training computation and gating function must be less than 100 ms, but this is a soft deadline in the sense that model adaptation can still take place with basis on older inputs. Each module needs to send out the (new and predicted) position coordinates and receives a weight to scale its outputs. Communication needs are thus modest, estimated as less than 1 ms on a cluster with a 1-Gbps Ethernet. On the basis of our back-of-the-envelope complexity estimates and our experience in implementing single models of similar complexity, an effective computational rate of 100 MFLOPS per module comfortably suffices to do both the model computation and the training computation for a single model with M=3,000 parameters with latencies below 50 ms and 100ms, respectively. These rates are well within the reach of optimized implementations of our algorithms on Grid-accessible 3.2GHz-Intel-Xeon machines. The challenge is then for middleware to be able to find and aggregate enough resources to compute and connect all necessary modules.

In-VIGO makes extensive use of virtualization technology for the creation of dynamic pools of virtual resources that can be aggregated on-demand. Its approach to Grid-computing is unique in that it decouples user environments from physical resources by building upon three layers of virtualization. At the bottom layer (*virtual resources)*, it manages and aggregates virtual instances of machines, networks, applications and data as needed to build virtual computational Grids to serve specific users and their applications. Within virtual computational Grids, tools and other utilities can be provided as Web services which can be aggregated and combined to constitute (virtual) information Grids. This second layer (*services)* exposes services available to upper middleware levels while hiding the kinds of machines used to provide services. At the topmost layer *(presentation)*, In-VIGO users are presented with domain-specific portals (e.g. the nanoHUB nanoelectronics simulation portal) which can be accessed by devices with possibly different interfaces.

Users access the portal through any conventional Web-browser which allows them to log into In-VIGO, request actions and provide data to the User Interface Manager (UIM). The Resource Manager (RM) orchestrates actions of the Virtual Machine System (VMS), Virtual Application System (VAS) and Virtual File System (VFS) that are needed to create the In-VIGO user environment and enable the execution of applications selected by the user. As the names suggest, the VMS, VFS and VAS components provide the virtual instances of machines, file systems and applications needed by a given user's invocation of a tool.

Two broad scenarios are considered for the computational infrastructure needed by DDDBMIs. One is the online scenario extensively discussed in this paper, where real-time computation is needed for in-vivo experimentation. The other is the offline scenario where data from past experiments is "replayed" and analyzed, e.g. to generate statistics or train models. In both scenarios, a key requirement is the computation of hundreds of modules as discussed above in relation to Fig. 3. We envision In-VIGO middleware being used to provide BMI researchers with a Web-based interface that would allow them to specify the models they wish to experiment with and possibly additional information (e.g. desired start time, explicit QoS requirements and data collection/storage). In the offline case they would also specify which experiment(s) to replay. Our current research focuses on the challenge of having In-VIGO automatically set up the necessary resources. In the online case this includes setting up the necessary connectivity among resources and the data acquisition system, and guaranteeing QoS requirements. In the offline case, a virtual application will have to be created to re-create experiments from stored data, but data rates can be slowed down, thus removing hard deadlines on computation.

4 Conclusions

Dynamically data-driven BMI's have the potential to drastically impact the ability of patients who suffer from total/partial paralysis following brainstem injury, stroke, or degenerative diseases such as amyotrophic lateral sclerosis (ALS) regain (artificial) motor control and autonomy. We presented our initial ideas for the design of DDDBMIs and the implementation of a test bed for research of their architectures based on predictive models in a mixture-of-experts configuration inspired by Kawato's work. Our initial analysis of requirements of DDDBMI's indicates that it is possible to use Grid-computing resources to execute and steer the necessary signal processing tasks on-demand and in real time. We are in the early stages of implementation of the test bed described in this paper.

Acknowledgements

The work reported in this paper is supported in part by the National Science Foundation under NSF Grant No. CNS-0540304.

References

1. Carmena, J.M., et al.: Learning to control a brain–machine interface for reaching and grasping by primates. PLoS Biology, 2003. **1**: p. 1-16.
2. Wolpert, D.M., Kawato, M.: Multiple paired forward and inverse models for motor control, Neural Networks 11 (1998) 1317– 1329.
3. Oztop, E., Wolpert, D., Kawato, M.: Mental state inference using visual control parameters Cognitive Brain Research 22 (2005) 129– 151, 2004
4. Adabala, S., Chadha, V., Chawla, P., Figueiredo, R., Fortes, J.A.B., Krsul, I., Matsunaga, A., Tsugawa, M., Zhang, J., Zhao, M., Zhu, L., Zhu, X.: "From Virtualized Resources to Virtual Computing Grids: The In-VIGO System", In Future Generation Computing Systems, 21(6), April 2005.