HJB-Equation-Based Optimal Learning Scheme for Neural Networks With Applications in Brain–Computer Interface

Article · August 2018

DOI: 10.1109/TETCI.2018.2858761

CITATION
1

READS
102

3 authors, including:

Tharun Reddy
Indian Institute of Technology Kanpur
9 PUBLICATIONS **14** CITATIONS

SEE PROFILE

Vipul Arora
Indian Institute of Technology Kanpur
23 PUBLICATIONS **99** CITATIONS

SEE PROFILE

HJB-Equation-based Optimal Learning Scheme for Neural Networks with Applications in Brain-Computer Interface

Tharun Kumar Reddy, Vipul Arora and Laxmidhar Behera, *Senior Member, IEEE*

***Abstract*—This paper proposes a novel method for training Neural Networks. It uses an approach from optimal control theory, namely Hamilton-Jacobi-Bellman (HJB) equation, which optimizes system performance along the trajectory. This formulation leads to a closed form solution for an optimal weight update rule, which has been combined with per-parameter adaptive scheme AdaGrad to further enhance its performance. To evaluate the proposed method, the neural networks (NN) are trained and tested on two problems related to EEG classification, namely, mental imagery classification (multi-class) and eye state recognition (binary-class). In addition, a novel dataset with the name EEG eye-state, for benchmarking learning methods is presented. The convergence proof for the proposed approach is also included, and performance is validated on many small to large scale, synthetic datasets (UCI, LIBSVM datasets). The performance of NNs trained with the proposed scheme is compared with other state-of-the-art approaches. Evaluation results substantiate the improvements brought about by the proposed scheme regarding faster convergence and better accuracy.**

***Index Terms*—Neural network training, Optimal control, HJB equation, Brain-computer interface, Eye state recognition**

I. Introduction

Currently, NN's can be trained using various algorithms like Gradient descent, AdaGrad [1], Rmsprop, Adam [2] etc. Learning algorithms based on extended Kalman filtering [3], Lyapunov stability theory [4] and recursive least squares [5] are few examples of methods developed in other fields like signal processing, optimal control theory, etc. In [6], Vipul et al., proposed the use of Hamilton-Jacobi-Bellman (HJB) equation to derive an optimal update rule for training feed forward NNs (FFNNs). The HJB equation comes from dynamic programming [7] [8], which is a popular approach for optimal control of dynamical systems [9], [10]. It is based on the principle of optimizing a sequence of decisions by defining an optimal control policy. In this framework, given a dynamical system, a cost-to-go objective function is defined based on the predicted 'future trajectory' of the system. Here, 'future trajectory' refers to the sequence of states one obtains when the system evolves from the current state while being controlled by an optimal control law or an optimal weight update scheme in the case of NNs. The HJB equation gives an optimal control input, i.e. an optimal weight update scheme for NNs which optimizes the cost-to-go objective function.

Tharun Kumar Reddy and Laxmidhar Behera are with the Department of Electrical Engineering, Indian Institute of Technology, Kanpur, Kanpur 208016, India (e-mails: tharun@iitk.ac.in, lbehera@iitk.ac.in)

Vipul Arora is Research Scientist at Amazon Alexa Machine Learning Boston, Massachusetts (e-mail: vipular.iitk@gmail.com).

As per [11], NN learning can be modeled as a dynamical system. Further, principles of dynamic programming can be applied to derive an optimal update law for NN. A similar approach is proposed in this paper. The HJB formulation [6] is extended by deriving a closed form solution without using eigenvalue decomposition (EVD). This significantly reduces the number of computations, which is otherwise needed for EVD in each iteration. This framework is applied in the context of training the feed-forward NNs, where updates in network weights form the control input for manipulating the output error. Hence, the proposed approach provides a principled way of training the NNs. Moreover, the proposed weight update rule has a simple form and is computationally light. A theoretical convergence analysis of the HJB algorithm is provided in section II-C under typical assumptions of strong convexity and Lipschitz continuity imposed on the loss function. Convergence analysis reveals that in the general case, where loss function is convex and smooth, convergence time decays inversely with optimization error ϵ which is comparable in performance to Stochastic Gradient Descent. For a strongly convex loss function, convergence time decays exponentially as a function of optimization error ϵ. Experiments were repeated on a variety of publicly available datasets with varying sizes, and a performance comparison is provided with several representative NN training algorithms present in the literature. Another advantage of this algorithm is that it is easily integrable with state-of-the-art adaptation schemes like adding a momentum term, AdaGrad, etc. It is to be noted that while AdaGrad scheme optimizes over the actual past trajectory of weights, the proposed scheme optimizes over the predicted future trajectory, given the current weights. Thus, integrating the two schemes enhances the performance.

The proposed learning algorithm for neural network classifiers not only adapts effectively, but is also computationally efficient, and hence, suitable for online learning. It is an adaptable sequential learning algorithm in the following sense, 1) The training observations are sequentially (sample-by-sample or block-by-block with varying or fixed block length) presented to the learning algorithm. 2) At any time, only the newly arrived sample or block of patterns (instead of the entire prior data) are looked at and learned from. Again, HJB based algorithm presented is a mini-batch sequential learning algorithm (with mini-batch size much smaller than the total number of training points). The mini-batch sequential version demonstrates faster convergence without compromising accuracy in comparison to a per-sample sequential version. Hence-

forth, we consider a mini-batch sequential learning problem. In sections III and IV, the proposed NN based methods are also implemented for several problems in brain-computer interface (BCI). These problems are used as benchmarks for evaluating the performance of the proposed training scheme. Moreover, proposed algorithms are also tested on UCI datasets [12].

A. EEG Brain Computer Interface (EEG-BCI)

Since brain signals are very complex in nature, certain simple tasks are designed to analyze and study them. In such studies, the subject is asked to imagine one of a few selected tasks and his/her brain signals are recorded. These signals are then studied with the help of various signal processing and machine learning techniques. Various linear and non-linear methods[13] like radial basis function network (RBFN), support vector machine (SVM) [14] and NNs [15] are proposed to classify the signals.

In this context, it is notable to mention works where NNs are deployed successfully for EEG signal pre-processing [16][17] and EEG signal classification [13],[18–20]. To take care of temporal evolution of signals, researchers have proposed methods like time-embedded representation [21] and recurrent NNs[22].

Due to subject dependent nature of brain signals, subject-independent optimization is found to be difficult [15]. To overcome this problem, Millán [23] proposed to use online learning for adapting the models on the fly to a particular subject while (s)he handles the BCI device. In addition, Millán [23] proposed simple classifiers to minimize the computational time for adaptation as well as classification.

Researchers invested in BCIs initially with therapeutic applications in mind which include improving the scope of the movement for physically disabled, locked-in patients [24][25] and replacing any other disoriented motor abilities etc. In time, BCI's even started impacting the lives of able-bodied users in the form of hands-free games [26], entertainment, and applications requiring restricted movement.

Recent developments in the area of machine learning have opened up new opportunities for various fields, including BCI. Problems in BCI provide challenging scenarios for machine learning with high variability of brain signals, and possibilities of incorporating multi-channel, multi-view and multi-modal considerations.

In this work, two BCI research problems are considered - mental imagery classification (multi-class) and eye state classification (binary).

1) Mental Imagery Classification: This dataset is collected in the IDIAP Research Institute by the authors of [23]. It was used in the BCI-III competition that took place in 2005. This dataset contains data from 3 normal subjects during 4 non-feedback sessions. The subjects sat in a normal chair with relaxed arms resting on their legs. There are 3 tasks, so this is a three-class classification problem:

- Imagination of repetitive self-paced left hand movements
- Imagination of repetitive self-paced right hand movements
- Generation of words beginning with the same random letter

In sections III-B and III-C, the proposed HJB based NN classifiers are compared with Adagrad based NN classifiers [1] and BCI competition winning algorithm [27] on this dataset.

2) Eye State Recognition: Eye movements and eye states (open or shut) have been used for man-machine interfaces. Eye states, as opposed to eye movements, are long and lasting, and hence, are more effective for controlling machines. They also correlate well for detecting drowsiness in drivers. Electrooculographic (EOG) devices consist of electrodes which are placed close to the subject's eyes and they can detect eye states. Some works [28], [29] use both EEG and EOG signals to detect eye state. However, present EOG detectors are quite inconvenient, and hence, impractical.

On the other hand, EEG devices that can be worn like a cap are greatly useful for practical applications. Thus, detection of eye states from EEG alone can be very useful for practical applications. Some works like [30] analyze the characteristic differences in EEG patterns, observed during the eyes open and eyes closed states. Rösler and Suendermann [31] have created a dataset on which they tested different classifiers, with K* algorithm [32] performing the best, but, it is computationally very slow. Singla and Rana [33] classify eye states from EEG signals using an ensemble of various classifiers. Hamilton *et al.* [34] also use a group of random rotational forest classifiers with adaptive boosting. Recently, Reddy and Behera [35] have proposed NN based methods for eye state recognition.

In section IV, NN classifiers are presented for eye state classification, trained using the proposed HJB learning scheme. A new EEG dataset for eye state recognition is prepared and presented, as there is no large dataset available for this purpose.

II. Proposed optimization Algorithm

Notations: Matrices and vectors are denoted by bold-faced letters in this paper. The bold text in tables represent results of proposed algorithms.

A. EVD free HJB based Weight Update

Back propagation algorithm attempts to minimize an error cost function computed over a sample or a batch. The optimal rate of convergence of such algorithms and their variants have not been studied analytically in a rigorous way. The present work derives the weight update laws that optimize a global cost function. Using HJB framework, the learning problem that has been comprehensively addressed in this paper tries to ensure optimal convergence while simultaneously optimizing a global cost function. For training the FFNN (Feed Forward Neural Network), the weight update problem can be cast as a control problem with output error $\mathbf{e}(t)$ as the state of the dynamical system and weight updates $\mathbf{u}(t)$ as control inputs.

The control theoretical solution for the weight update law within HJB based optimal framework is derived as follows:

Consider a feedforward NN, with L layers (hidden+output) and N_l neurons in the lth layer ($l \in \{0, 1, ..., L\}$, $l = 0$ for input layer). Let the input to the network be denoted by $\mathbf{x}_p \in \mathbb{R}^{N_0}$, where p indexes the patterns. Let $\hat{\mathbf{y}}_\mathbf{p} \in \mathbb{R}^{N_L}$

denote the predicted output for the pth input pattern, and $\mathbf{y}_p^d$, the corresponding true output. Let $\hat{\mathbf{w}}, \mathbf{w} \in \mathbb{R}^W$ denote the estimated and actual optimal weights of the NN, respectively, where, W denotes the number of weight parameters. The relation between the input pattern and the predicted output can be given as

$$\hat{\mathbf{y}}_p = f(\hat{\mathbf{w}}, \mathbf{x}_p) \tag{1}$$

Here, all vectors are column vectors. For deriving optimal update in weights, $\hat{\mathbf{y}}_p$ can be differentiated with respect to time t as

$$\dot{\hat{\mathbf{y}}}_p = \frac{\partial \mathbf{f}(\hat{\mathbf{w}}, \mathbf{x}_p)}{\partial \hat{\mathbf{w}}} \dot{\hat{w}} = \mathbf{J}_p \dot{\hat{w}} \tag{2}$$

where, $\mathbf{J}_p$ is the Jacobian matrix whose (i, j) element is given by $\partial \hat{y}_{p,i} / \partial \hat{w}_j$. Thus, the input pattern $\mathbf{x}_p$ is independent of t and $\dot{\mathbf{x}}_p = 0$ i.e. the input pattern do not change with t. The error $\mathbf{e}_p$ for the pattern $\mathbf{x}_p$ is given by

$$\mathbf{e}_p = \mathbf{y}_p^d - \hat{\mathbf{y}}_p \tag{3}$$

and its derivative with respect to t is given by

$$\dot{\mathbf{e}}_p = \dot{\mathbf{y}}_p^d - \dot{\hat{\mathbf{y}}}_p \tag{4}$$

$$= -\mathbf{J}_p \dot{\hat{\mathbf{w}}} \tag{5}$$

$$= -\mathbf{J}_p \mathbf{u} \text{ where, } \mathbf{u} = \dot{\hat{\mathbf{w}}} \tag{6}$$

The weight update law $\mathbf{u} = \dot{\hat{\mathbf{w}}}$ can be derived using control theoretic approach such that the error dynamics in (5) is stable while optimizing a global cost function.

For batch mode of learning, N patterns are used together in each iteration for updating the weights. The system dynamics can be represented as

$$\dot{\mathbf{e}} = -\mathbf{J}\mathbf{u}; \;\; \mathbf{u} = \dot{\hat{\mathbf{w}}} \tag{7}$$

where $\mathbf{e} = [\mathbf{e}_1^\intercal, \mathbf{e}_2^\intercal, \ldots, \mathbf{e}_N^\intercal]^\intercal$, $\mathbf{J} = [\mathbf{J}_1^\intercal, \mathbf{J}_2^\intercal, \ldots, \mathbf{J}_N^\intercal]^\intercal$ is an $N_L N \times W$ matrix. The control theoretic solution of (7) in the optimal framework will result in an optimal weight update law that will minimize the following cost function:

$$V(\mathbf{e}(t), \mathbf{u}(t)) = \int_t^T (L(\mathbf{e}(\tau), \mathbf{u}(\tau)) d\tau \tag{8}$$

$$\text{where, } L(\mathbf{e}, \mathbf{u}) = \frac{1}{2}(\mathbf{e}^\intercal \mathbf{Q} \mathbf{e} + \mathbf{u}^\intercal \mathbf{R} \mathbf{u}), \tag{9}$$

$\mathbf{Q}$ and $\mathbf{R}$ are constant positive definite matrices. A proper choice of $\mathbf{Q}$ ensures that the components of the error vector are 'small'. The matrix $\mathbf{R}$ takes care that the components of the input, $\mathbf{u}$, are 'not too large' or the weights of the neural networks remain bounded. The matrix $\mathbf{Q}$ is set to an identity matrix throughout the experimental simulations done in this paper. It should be noted that although the function V is integrated over finite time horizon, the function V is not an explicit function of T and t.

The control law which is also the weight update law for the neural network will be derived through HJB formulation. The cost function V is novel and different from conventional loss functions encountered in neural networks training. With the optimal control law $u^*(t)$, the optimal cost function V^* will be a function of $e(t)$ only.

The learning problem is thus formulated as follows: Given the error dynamics of a neural network in (7) and the cost function as given in (8), it is required to find the optimal weight update law $\dot{\hat{\mathbf{w}}}$ of the network in real-time. Simultaneously, the convergence time complexity needs to be derived. Now, the goal is to find an optimal weight update law $\mathbf{u}^*(t)$ to minimize the overall cost function V. Essentially, we are trying to minimize the cumulative sum of costs encountered during iterations $\tau \in [t, T]$ with an optimal choice of weight update law $\mathbf{u}(t)$. To solve this sequential optimization problem, we can use the HJB formulation as described in [6, 8, 36]. The optimal cost V^* has the following expression:

$$V^*(\mathbf{e}(t)) = \min_{\mathbf{u}} \{V(\mathbf{e}(t), \mathbf{u}(t))\} \tag{10}$$

Since V^* is not the explicit function of time, and the network dynamics (7) is time invariant, the corresponding HJB equation is given as:

$$\min_{\mathbf{u}} \left\{ \frac{\partial V^*}{\partial \mathbf{e}} \dot{\mathbf{e}}(t) + L(\mathbf{e}(t), \mathbf{u}(t)) \right\} = 0 \tag{11}$$

Putting expressions for $\dot{\mathbf{e}}(\mathbf{t})$ and L in (11), we get,

$$\min_{\mathbf{u}} \left\{ -\frac{\partial V^*}{\partial \mathbf{e}} \mathbf{J}\mathbf{u} + \frac{\mathbf{e}^\intercal \mathbf{e}}{2} + \frac{\mathbf{u}^\intercal \mathbf{R} \mathbf{u}}{2} \right\} = 0 \tag{12}$$

Differentiating with respect to $\mathbf{u}$ and setting the derivative to zero, we get,

$$\mathbf{u}^* = \mathbf{R}^{-1} \mathbf{J}^\intercal \left(\frac{\partial V^*}{\partial \mathbf{e}} \right)^\intercal \tag{13}$$

Plugging $\mathbf{u}^*$ from(13) into(12), we get,

$$\left(\frac{\partial V^*}{\partial \mathbf{e}} \right) \mathbf{J} \mathbf{R}^{-1} \mathbf{J}^\intercal \left(\frac{\partial V^*}{\partial \mathbf{e}} \right)^\intercal = \mathbf{e}^\intercal \mathbf{e} \tag{14}$$

This is a quadratic equation of the form $\mathbf{X}^\intercal \mathbf{X} = y$, where $X \in \mathbb{R}^W$ and $\mathbf{y} \in \mathbb{R}$. It has a solution of the form $\mathbf{X} = \sqrt{y}\mathbf{c}$, where $\mathbf{c} \in \mathbb{R}^W$ is a vector with $\mathbf{c}^\intercal \mathbf{c} = 1$. Thus, we get

$$\mathbf{R}^{-1/2} \mathbf{J}^\intercal \left(\frac{\partial V^*}{\partial \mathbf{e}} \right)^\intercal = \sqrt{\mathbf{e}^\intercal(t) \mathbf{e}(t)} \;\; \mathbf{c} \tag{15}$$

Putting(15) in(13) gives

$$\mathbf{u}^*(\mathbf{t}) = \mathbf{R}^{-1/2} \mathbf{c} \sqrt{\mathbf{e}^\intercal(t) \mathbf{e}(t)} \tag{16}$$

In order to solve for $\mathbf{c}$, the constraint that input $\mathbf{u}^*$ must stabilize the system is used. The stability of system is defined with the help of Lyapunov function

$$\mathcal{V}(\mathbf{e}) = \frac{1}{2} \mathbf{e}^\intercal \mathbf{e} \tag{17}$$

The equilibrium point $\mathbf{e} = 0$ is stable, provided that $\dot{\mathcal{V}}(\mathbf{e})$ is negative semi-definite.

$$\dot{\mathcal{V}}(\mathbf{e}) = \mathbf{e}^\intercal \dot{\mathbf{e}} \tag{18}$$

$$= -\mathbf{e}^\intercal \mathbf{J} \mathbf{u}^*(t) \tag{19}$$

$$= -\mathbf{e}^\intercal(t) \mathbf{J} \mathbf{R}^{-1/2} \mathbf{c} \sqrt{\mathbf{e}^\intercal(t) \mathbf{e}(t)} \tag{20}$$

Choosing $\mathbf{c} = \mathbf{J}^\intercal \mathbf{e}(t) / ||\mathbf{J}^\intercal \mathbf{e}(t)||$ makes $\dot{\mathcal{V}}(\mathbf{e})$ negative semi-definite. Here, $||\mathbf{x}||$ represents ℓ^2-norm of vector $\mathbf{x}$. Hence, we get

$$\mathbf{u}^*(t) = \frac{\sqrt{\mathbf{e}^\intercal(t) \mathbf{e}(t)}}{||\mathbf{J}^\intercal \mathbf{e}(t)||} \mathbf{R}^{-1/2} \mathbf{J}^\intercal \mathbf{e}(t) \tag{21}$$

In the general case, $\mathbf{e^\intercal Q e}$ can be replaced by any other cost function say $\mathbf{P}(\mathbf{e}(t))$, and one obtains $\mathbf{u}^*(t)$ as:

$$\mathbf{u}^*(t) = \frac{\sqrt{2\mathbf{P}(\mathbf{e}(t))}}{||\mathbf{J^\intercal e}(t)||}\,\mathbf{R^{-1/2}J^\intercal e}(t) \tag{22}$$

For all practical purposes in this paper, we use $\mathbf{R} = r\mathbf{I}$ where, $\mathbf{I}$ is identity matrix and $r > 0$.

B. Integration with other Learning Schemes

The proposed learning scheme can be integrated with popular variations of back propagation like adding a momentum term. It can also be integrated with per-parameter adaptive learning rate methods for improved performance. Here, we integrate it with AdaGrad [1].

Back propagation algorithm involves

$$\hat{\mathbf{w}}(t+1) = \hat{\mathbf{w}}(t) + \eta\mathbf{u}(t) \tag{23}$$

where $\mathbf{u}(t)$ comes from the negative gradient and $\eta \in \mathbb{R}^+$ is the learning rate. The AdaGrad (AD) formulation modifies the learning rate adaptively

$$\hat{\mathbf{w}}(t+1) = \hat{\mathbf{w}}(t) + \frac{\eta}{\sqrt{\sum_{t'=0}^{t}||\mathbf{u}(t')||^2}}\mathbf{u}(t) \tag{24}$$

Here, instead of using back propagation gradients to find $\mathbf{u}(t)$, we assign $\mathbf{u}(t) = \mathbf{u}^*(t)$ from(21).

$$\hat{\mathbf{w}}(t+1) = \hat{\mathbf{w}}(t) + \frac{\eta}{\sqrt{\sum_{t'=0}^{t}||\mathbf{u}^*(t')||^2}}\mathbf{u}^*(t) \tag{25}$$

Both Equations(21) *and* (25) *derived in this paper are novel contributions towards achieving a weight update rule that is simultaneously faster in convergence as well as accurate in terms of performance.* Learning algorithm with update law in(25) is termed **'HJB-AD'**. The update law derived from HJB equation in [6] (also termed **'EVDHJB'**) is

$$\mathbf{u}^*(t) = \mathbf{R^{-1}J^\intercal C}(t)\mathbf{e}(t) \tag{26}$$

where, $\mathbf{C} = \mathbf{U\Sigma^{-1/2}U^\intercal}$, with $\mathbf{U}$ and $\mathbf{\Sigma}$ being the eigenvectors and the diagonal matrix of eigenvalues of $\mathbf{JR^{-1}J^\intercal}$, respectively. On comparing(26) and(21), we see that the latter is computationally more efficient than the former.

C. Convergence Analysis

1) Notations and Assumptions: The definitions of convexity, strong-convexity, G-Lipschitz, M-smoothness and M-Lipschitz continuity used in the proofs below are taken from [37–39].

To analyze the convergence properties of the proposed algorithm, we discretize the optimal cost function V^* to $V^*(\mathbf{e}(k))$, with k and K as the discrete counterparts of t and T, respectively. Here,

$$\mathbf{e}(k) = \mathbf{y}^d(k) - \hat{\mathbf{y}}(k) \tag{27}$$

with $\hat{\mathbf{y}}(k) \in \mathbb{R}^{N_L}$ denoting the predicted output at k^{th} iteration, and $\mathbf{y}^d$, the corresponding true output. From (1) and (27), $\mathbf{e}(k)$ is a function of estimated weights $\hat{\mathbf{w}}(k)$. Hence, the optimal cost function can be written as a function of $\hat{\mathbf{w}}(k)$.

$$V_1^*(\hat{\mathbf{w}}(k)) = V^*(\mathbf{e}(k))$$

TABLE I: Comparison of convergence rates

Algorithm	Time taken per iteration	Iterations to Accuracyϵ	Time to Accuracyϵ
BG (Batch Gradient descent)	n	$\log\frac{1}{\epsilon}$	$n\log\frac{1}{\epsilon}$
2GD (Second order Gradient descent)	n	$\log\log\frac{1}{\epsilon}$	$n\log\log\frac{1}{\epsilon}$
SGD (Stochastic Gradient descent)	p	$\frac{1}{\epsilon}$	$\frac{p}{\epsilon}$
O-HJB (Online time optimal-HJB)	p	$\frac{1}{\epsilon}$	$\frac{p}{\epsilon}$
B-HJB (Batch time optimal-HJB)	n	$\frac{1}{\epsilon}$	$\frac{n}{\epsilon}$

Theorem 1. *Let the cost function $V_1^*(\hat{\mathbf{w}}(k))$ be differentiable, convex, G-Lipschitz and its gradient is M-Lipschitz continuous. If the step size η_k satisfies*

$$\sum_{k=1}^{K} k{\eta_k}^2(2\mathbf{P}(\mathbf{e}(k))) < \infty \tag{28}$$

and

$$\eta_k = \frac{\eta_0}{\sqrt{2\mathbf{P}(\mathbf{e}(k)) + \epsilon_1}}\,\frac{1}{\sqrt{k}} \tag{29}$$

where $\epsilon_1 \ll 1$ ensures denominator is a non-zero quantity and the cost function $V_1^*(\hat{\mathbf{w}}(k))$ satisfies

$$V_1^*\Big(\frac{2}{K(K+1)}\sum_{k=1}^{K} k\hat{\mathbf{w}}_k\Big) - V_1^*(\mathbf{w}^*) = \epsilon, \tag{30}$$

where $\epsilon \geq 0$, then the proposed HJB based algorithm converges in $\mathcal{O}(\frac{1}{\epsilon})$ iterations.

Proof. Given, $V_1^*(\hat{\mathbf{w}}(k))$ is convex and G-Lipschitz. Convexity implies(31)

$$\begin{aligned} V_1^*(\hat{\mathbf{w}}_k) - V_1^*(\mathbf{w}^*) \leq \nabla^\intercal V_1^*(\hat{\mathbf{w}}_k)(\hat{\mathbf{w}}_k - \mathbf{w}^*) \leq \\ \langle\nabla(V_1^*(\hat{\mathbf{w}}_k)), (\hat{\mathbf{w}}_k - \mathbf{w}^*)\rangle \end{aligned} \tag{31}$$

where $\langle .,. \rangle$ represents the inner product. From Cauchy Schwarz inequality, the inner product in 31 is bounded by $G \cdot D$, where $\|\nabla V_1^*(\hat{\mathbf{w}}_k)\| \leq G$ and $\|\hat{\mathbf{w}}_k - \mathbf{w}^*\| \leq D$ (assuming the weights $\hat{\mathbf{w}}_k$ lie in a convex set whose diameter is bounded.)

$$\|V_1^*(\hat{\mathbf{w}}_k) - V_1^*(\mathbf{w}^*)\| \leq \langle\nabla(V_1^*(\mathbf{w}_k)), (\hat{\mathbf{w}}_k - \mathbf{w}^*)\rangle \leq G \cdot D \tag{32}$$

From Lipschitz continuity property of gradient of $V_1^*(\hat{\mathbf{w}}_k)$, we get

$$\begin{aligned} V_1^*(\hat{\mathbf{w}}_{k+1}) \leq V_1^*(\hat{\mathbf{w}}_k) + \nabla^\intercal V_1^*(\hat{\mathbf{w}}_k)(\hat{\mathbf{w}}_{k+1} - \hat{\mathbf{w}}_k) \\ + \frac{M\|(\hat{\mathbf{w}}_{k+1} - \hat{\mathbf{w}}_k)\|^2}{2} \end{aligned} \tag{33}$$

Re-using(23) and plugging in expression for $\mathbf{u}^*$ from(22), we get

$$(\hat{\mathbf{w}}_{k+1} - \hat{\mathbf{w}}_k) = \eta_k\mathbf{u}(k) = \eta_k\sqrt{(2\mathbf{P}(\mathbf{e}(k)))}\frac{\mathbf{J^\intercal e}(k)}{\sqrt{r}\|\mathbf{J^\intercal e}(k)\|} \tag{34}$$

$${\eta_k}^2\|\mathbf{u}(k)\|^2 = {\eta_k}^2 \cdot \frac{2\mathbf{P}(\mathbf{e}(k))}{r} \tag{35}$$

Here, η_k is the learning rate at the k^{th} time instant. Plugging in expression for $V_1^*(\hat{\mathbf{w}}_k)$ from(31) reduces(33) to

$$V_1^*(\hat{\mathbf{w}}_{k+1}) \leq V_1^*(\mathbf{w}^*) + \nabla^\intercal V_1^*(\hat{\mathbf{w}}_k)(\hat{\mathbf{w}}_k - \mathbf{w}^*) + \nabla^\intercal V_1^*(\hat{\mathbf{w}}_k)(\hat{\mathbf{w}}_{k+1} - \hat{\mathbf{w}}_k) + \frac{M\|\mathbf{u}(k)\|^2}{2} \tag{36}$$

$$V_1^*(\hat{\mathbf{w}}_{k+1}) - V_1^*(\mathbf{w}^*) \leq \nabla^\intercal V_1^*(\hat{\mathbf{w}}_k)(\hat{\mathbf{w}}_{k+1} - \mathbf{w}^*) + \frac{M{\eta_k}^2 \cdot (2\mathbf{P}(\mathbf{e}(k)))}{2r} \tag{37}$$

Multiplying by k and performing summation on both sides of(37) with limits over k from 1 to K and further taking norm on both LHS and RHS, we get

$$\begin{aligned} &\|\sum_{k=1}^{K} k(V_1^*(\hat{\mathbf{w}}_{k+1}) - V_1^*(\mathbf{w}^*))\| \\ &\leq \|\sum_{k=1}^{K} k(\nabla^\intercal V_1^*(\hat{\mathbf{w}}_k)(\hat{\mathbf{w}}_{k+1} - \mathbf{w}^*))\| \\ &\quad + \|\sum_{k=1}^{K} k\Big(\frac{M{\eta_k}^2 \cdot (2\mathbf{P}(\mathbf{e}(k)))}{2r}\Big)\| \end{aligned} \tag{38}$$

According to convexity property,

$$\frac{2}{K(K+1)}\sum_{k=1}^{K} kV_1^*(\hat{\mathbf{w}})_k \geq V_1^*\Big(\frac{2}{K(K+1)}\sum_{k=1}^{K} k\hat{\mathbf{w}}_k\Big) \tag{39}$$

Divide both sides of(38) by $\frac{K(K+1)}{2}$ and use(39). Further, use $V_1^*\Big(\frac{2}{K(K+1)}\sum_{k=1}^{K} k\hat{\mathbf{w}}_k\Big) - V_1^*(\mathbf{w}^*) = \epsilon$. And choose $\eta_k = \frac{\eta_0}{\sqrt{(2\mathbf{P}(\mathbf{e}(k))+\epsilon_1)}\cdot\sqrt{k}}$, this reduces(38) to

$$\epsilon \leq G \cdot D + \frac{M}{2r(K+1)} \tag{40}$$

Thus the following is obtained.

$$K \approx \frac{M}{2r\epsilon} \approx O(\frac{1}{\epsilon}) \tag{41}$$

The number of iterations required to bring the error term on LHS within ϵ is of the order $\frac{1}{\epsilon}$. Assuming a unit time to be taken for computation of gradient $\mathbf{J}^\intercal\mathbf{e}$ per sample, the time complexity of HJB method for a mini-batch size 'p' is $O(\frac{p}{\epsilon})$. A comparison with the standard gradient-based methods is as shown in the Table I.

In Table I, n stands for number of examples, $n = 1$ for 'SGD', 'p' stands for number of examples in a mini-batch. □

Theorem 2. *Let the cost function expression $V_1^*(\hat{\mathbf{w}}(k))$ be differentiable and strongly convex with the convexity parameter $\mu \geq 0$ and also G-Lipschitz. If the step-size η_k satisfies:*

$$\sum_{k=1}^{K}(k^2){\eta_k}^2(2\mathbf{P}(\mathbf{e}(k))) < \infty \tag{42}$$

$$\eta_k = \frac{K(K+1)}{2}\frac{\eta_0}{\sqrt{(2\mathbf{P}(\mathbf{e}(k))+\epsilon_1)}}\frac{\sqrt{r}}{k} \tag{43}$$

where $\epsilon_1 \ll 1$ ensures denominator is a non-zero quantity and the cost function $V_1^*(\hat{\mathbf{w}}(k))$ satisfies (30) *with $\epsilon \geq 0$, then the proposed HJB based algorithm converges in $\mathcal{O}(e^{\frac{-\epsilon}{2G}})$ iterations.*

Proof. Given, $V_1^*(\hat{\mathbf{w}}(k))$ is strongly convex and G-Lipschitz.

From the convex optimization literature (proof provided as a supplementary material), one can obtain the following result for a strongly convex function $V_1^*(\hat{\mathbf{w}}(k))$, where distance from optimal point is always upper bounded by a value given by

$$\|\mathbf{w}^* - \hat{\mathbf{w}}(k)\| \leq \frac{2\|\nabla V_1^*(\hat{\mathbf{w}}(k))\|}{\mu} \tag{44}$$

$$\begin{aligned} &\langle\nabla(V_1^*(\hat{\mathbf{w}}_k)), (\mathbf{w}^* - \hat{\mathbf{w}}_k)\rangle \\ &= \langle\nabla(V_1^*(\hat{\mathbf{w}}_k)), (\mathbf{w}^* - \hat{\mathbf{w}}_{k+1})\rangle \\ &\quad + \langle\nabla(V_1^*(\hat{\mathbf{w}}_k)), (\hat{\mathbf{w}}_{k+1} - \hat{\mathbf{w}}_k)\rangle \end{aligned} \tag{45}$$

Using Cauchy Schwarz inequality for each of the separated inner products in (45), one gets

$$\begin{aligned} &-\|\nabla(V_1^*(\hat{\mathbf{w}}_k))\|\|(\mathbf{w}^* - \hat{\mathbf{w}}_{k+1})\| \\ &\leq \langle\nabla(V_1^*(\hat{\mathbf{w}}_k)), (\mathbf{w}^* - \hat{\mathbf{w}}_{k+1})\rangle \\ &\leq \|\nabla(V_1^*(\hat{\mathbf{w}}_k))\|\|(\mathbf{w}^* - \hat{\mathbf{w}}_{k+1})\| \end{aligned} \tag{46}$$

Strong convexity of $V_1^*(\hat{\mathbf{w}}_k)$ implies (47).

$$\begin{aligned} V_1^*(\mathbf{w}^*) \geq &V_1^*(\hat{\mathbf{w}}_k) + \\ &\nabla^\intercal V_1^*(\hat{\mathbf{w}}_k)(\mathbf{w}^* - \hat{\mathbf{w}}_k) + \frac{\mu\|\mathbf{w}^* - \hat{\mathbf{w}}_k\|^2}{2} \end{aligned} \tag{47}$$

Rewriting (47), one gets to (48)

$$\|\hat{\mathbf{w}}(k) - \mathbf{w}^*\|^2 \leq -\frac{2[A+B]}{\mu} \tag{48}$$

where, $A = \langle\nabla^\intercal(V_1^*(\hat{\mathbf{w}}_k)), (\mathbf{w}^* - \hat{\mathbf{w}}_k)\rangle$, $B = (V_1^*(\hat{\mathbf{w}}(k)) - V_1^*(\mathbf{w}^*))$.

Comparing RHS of (44) and (48), multiplying by k and performing summation with limits over k from 1 to K. Break term A using (46). Also use (39), this implies

$$\sum_{k=1}^{K} k\frac{4G^2}{\mu^2} \approx \sum_{k=1}^{K} k\Big(\frac{-2G\cdot\frac{-2G}{\mu} + 2G\|\eta_k\mathbf{u}(k)\| + \epsilon}{\mu}\Big) \tag{49}$$

Also note that

$$(\hat{\mathbf{w}}_{k+1} - \hat{\mathbf{w}}_k) = \eta_k\mathbf{u}(k) = \eta_k\sqrt{(2\mathbf{P}(e(k)))}\frac{\mathbf{J}^\intercal\mathbf{e}(k)}{\|\mathbf{J}^\intercal\mathbf{e}(k)\|} \tag{50}$$

$$\|\mathbf{u}(k)\| = \frac{\sqrt{(2\mathbf{P}(\mathbf{e}(k)))}}{\sqrt{r}} \tag{51}$$

Choosing η_k from (43) reduces (49) further to

$$\frac{\sum_{k=1}^{K} 2G}{k} + \epsilon \approx 0 \tag{52}$$

The summation on LHS is the n^{th} Harmonic number which can be approximated by $2G\cdot\ln K$. This implies that the number of iterations $K \approx O(e^{\frac{-\epsilon}{2G}})$

□

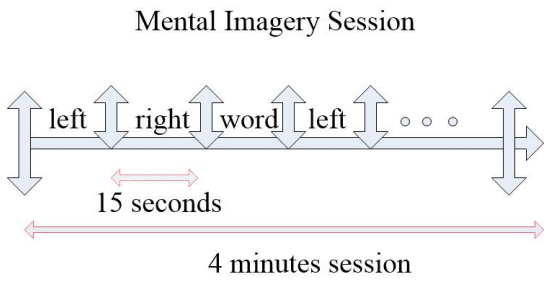

Fig. 1: BCI experiments session

III. Mental Imagery Classification

This section focuses on a multi-class task of classifying EEG signals. Description of the task, implementation of the proposed algorithm and experimental evaluations are described in this section.

A. Task and Dataset Description

Dataset V corresponding to mental imagery classification, a standard task from BCI competition III, [1] is used in the experiments. The subjects (total 3) were asked to imagine one of the three tasks: I) left-hand movement, II) right-hand movement, and III) words beginning with same random letter. Four sessions were held per subject, each session of 4 minutes in length. Each session comprised of randomly prompted tasks, with each task lasting about 15 s. (Fig. 1)

The dataset consists of pre-computed features calculated from raw EEG data collected at a sampling rate of 512 Hz. These features consist of power spectral density (PSD) in the band 8-30 Hz over a data window of length 1 s with a resolution of 2 Hz for eight channels. It is calculated every 62.5 ms (16 times in a second) using the Welch method with 5 overlapping (offset-25%) Hanning windows of duration 500 ms. Thus, each feature vector is 96-dimensional (8 channels $\times$ 12 frequency components). Each feature vector is normalized such that the features of each channel have unit ℓ^1-norm. The class label is estimated for each feature vector. The dataset is balanced for all the classes.

The performance was measured using classification accuracy, i.e., number of samples correctly classified over the total number of samples.

B. Implementation of the Proposed Scheme

Two NNs with architectures $96\times50\times30\times3$, $104\times50\times30\times3$ are denoted as the initial and terminal classifiers respectively as represented in Fig. 2. The first classifier uses the 96-dimensional feature vector of the current time sample as input. The first NN is trained on the labeled feature vectors from various training sessions (as per column II, Table III). The terminal NN accepts inputs of dimension 104, where labels obtained from 8-time samples and feature vector calculated from current time sample are concatenated. The terminal NN is also trained on feature vectors from various sessions (as per column II Table III). All hidden layers use the rectified linear unit (ReLU) as non-linearity. The first hidden layer uses drop out with the probability of turning off a neuron set to 0.2. Each of the 3 neurons in the output layer corresponds to one class and uses sigmoid non-linearity i.e. the class corresponding to the neuron with maximum activation is chosen as the estimated class.

[1] http://www.bbci.de/competition/iii/desc_V.html

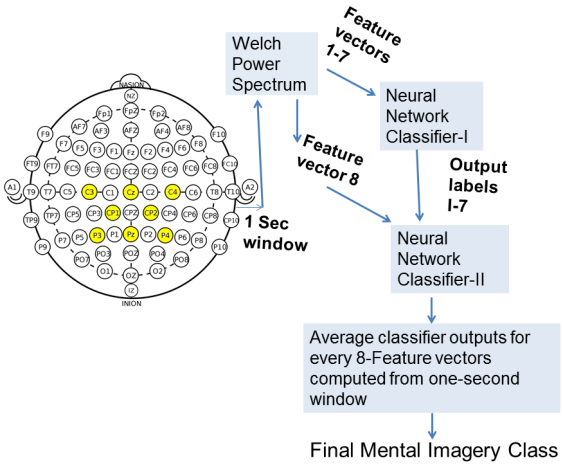

Fig. 2: Algorithm with two neural networks used for mental imagery classification

The learning scheme minimizes cross entropy error over the training data. The proposed algorithm uses $R = 100\mathbf{I}$ with $\eta = 0.01$ in (21) and (25). These parameters were chosen using grid search. A smoothing term of 10^{-4} is added to the denominators of (21) and (25) to avoid division by zero.

C. Baseline

The results of the competition[2] are used as a baseline. The top-performing algorithm by Galán *et al.* [27] used precomputed features with Fisher's discriminant for feature selection and a distance-based classifier for multi-class classification.

To compare the proposed learning scheme with a state-of-the-art NN learning scheme, the NN described in Sec. III-B is trained using the AdaGrad method, with $\eta = 0.01$.

IV. Eye State Recognition

This section focuses on the problem of eye state recognition. It describes the EEG dataset prepared by the authors for the purpose of this application. Then, the proposed implementation is detailed and experimental evaluation is discussed.

A. Task and Dataset Description

This section is focused on the analysis of datasets recorded in relaxed state situation with both eyes closed (EC) and open (EO). For preparing the dataset[3], the subjects were asked to sit on a chair under a faintly lit environment in a room and to relax for few minutes while staying awake, avoiding stressed focusing and concentration. During the eyes in open condition, the subjects were asked to fix the eyes on a red cross on a

[2] http://www.bbci.de/competition/iii/results/#martigny

[3] https://goo.gl/jS8psW

black screen in order to reduce artifacts due to eyes movement. 20 subjects have participated in the study. All information is from multiple persistent EEG recordings collected with the Emotiv EPOC EEG Neuroheadset at a sampling rate of 128 Hz. The minimum duration of recording from each subject is 260 s and the maximum is 583 s.

Eye movements, blinks and muscle artifacts that interferes with the EEG channels and are generally removed. Mostly EOG recordings are used to identify and remove ocular artifacts. As pointed out earlier, EOG devices are inconvenient for practical use. There are methods that can automatically remove ocular [40] as well as muscular [41] artifacts from EEG data alone, with the help of blind source separation techniques. On the eye-state EEG data, authors do not perform any signal processing for artifact rejection. The classifiers are allowed to implicitly learn to deal with the variations in EEG signals caused by any artifacts. This makes them more robust for practical use.

Short term spectra are extracted from each EEG channel, using a window length of 1.5 s and a hop size of 0.5 s. The EEG spectrum is split into four bands: Delta wave (0.5-4 Hz), Theta wave (4-7 Hz), Alpha wave (8-15 Hz) and Beta wave (16-31 Hz). This also removes the baseline drift as well as the electric supply frequency (50Hz) interference with the EEG signal. Further, each band is divided into 8 sub-bands using linearly equi-spaced and non-overlapping rectangular filters. The energy values from each sub-band of all the bands across all the channels are concatenated to form the feature vector, which is $(14 \times 8 \times 4 =)$ 448 dimensional. The binary ground truth values are known from the prompts supplied to the subjects.

Recognition performance is analyzed for detecting EC state, which is useful for drowsiness detection. Correct detection of EC state is called true positive (TP) and its incorrect detection is called false positive (FP). Similarly, correct detection of EO is called true negative (TN), while false negative (FN) refers to its incorrect detection. Accuracy is defined as number of correctly recognized samples (TP+TN) over total number of samples. Precision is defined as TP/(TP+FP) and recall is TP/(TP+FN).

B. Implementation

The feature vectors in the training set are decorrelated using principal component analysis (PCA), by choosing the smallest number (i.e., 90 here) of principal components such that 95% of variance is retained. The PCA-transformed feature vectors are normalized to zero mean and unit standard deviation. Further, a NN is trained to classify them. The hidden layers have ReLU non-linearity and drop out with a turn-off probability of 0.2. The output layer has sigmoid non-linearity for two neurons corresponding to the two output classes.

The parameter settings used for the proposed algorithm are the same as that used in Sec. III-B.

C. Baseline

As a baseline, rotational random forest and adaptive rotational J48 forest (RJ48F) classifiers are implemented, which have previously been reported [34] to work well for eye state recognition, but on a much smaller dataset. Weka[4] machine learning toolkit is used to implement these classifiers. For K*, global blend value is set to 39. For RJ48F, 10 rotational forests are used with 100 trees each.

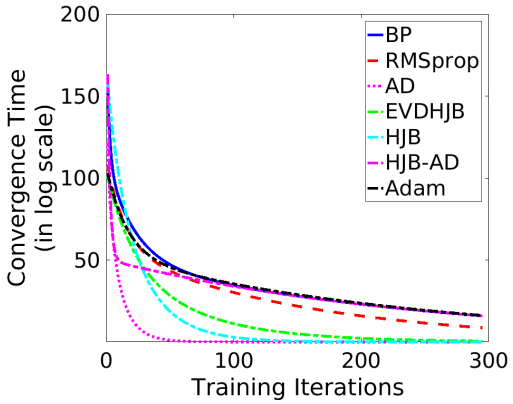

Fig. 3: Convergence rates of different algorithms on Eye-state dataset

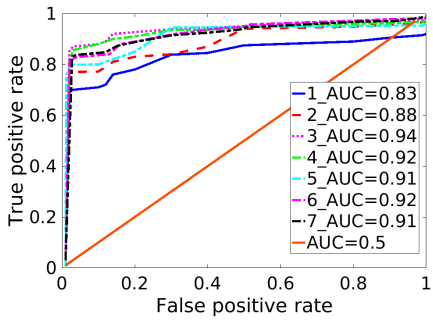

Fig. 4: ROC plot for training with different algorithms on Eye-state dataset

To evaluate the performance of the proposed learning scheme, NN classifiers are implemented and trained with different learning schemes like Back Propagation (BP), Adam, RMSprop, HJB, EVDHJB, HJB+Adagrad(HJB-AD) and AdaGrad (AD). Ens1 denotes an ensemble of HJB+AD$^{(1)}$ and AD$^{(2)}$, Ens2 denotes an ensemble of EVDHJB$^{(1)}$ and EVDHJB$^{(2)}$, Ens3 is an ensemble of HJB$^{(1)}$ and HJB $^{(2)}$, while Ens4 is an ensemble of BP$^{(1)}$ and BP$^{(2)}$. Here, the superscript $^{(1)}$ represents DNN architecture $91 \times 50 \times 30 \times 10 \times 2$, while the superscript $^{(2)}$ represents DNN architecture $91 \times 50 \times 30 \times 2$.

V. Experiments on UCI datasets

This section brings light to the experiments repeated on the datasets from UCI repository and LIBSVM datasets. Breast Cancer.wisc (BC), Adult dataset (ADU), Bank Marketing (BM), Mammographic Mass (MMG) are the datasets taken from UCI repository for experimentation. LIBSVM datasets include IJCNN[5], Webspam (WS)[6], HTRU2[7], Covtype[8]. Other

[4]http://www.cs.waikato.ac.nz/ml/weka/

TABLE II: Weight update laws of various algorithms

Algorithm	Update law, $\dot{\hat{\mathbf{w}}} = \mathbf{u}$
BP	$\mathbf{u} = \eta \mathbf{J}^\intercal \mathbf{e}$
LF	$\mathbf{u} = \mu \frac{\|\|\mathbf{e}\|\|^2}{\|\|\mathbf{J}^\intercal \mathbf{e}\|\|^2} \mathbf{J}^\intercal \mathbf{e}$
LM	$\mathbf{u} = [\mathbf{J}^\intercal \mathbf{J} + \mu \mathbf{I}]^{-1} \mathbf{J}^\intercal \mathbf{e}$
EVDHJB	$\mathbf{u} = \mathbf{R}^{-1} \mathbf{J}^\intercal \mathbf{C} \mathbf{e}$
HJB with time optimality	$\mathbf{u} = \frac{\sqrt{2\mathbf{P}(\mathbf{e})}}{\|\|\mathbf{J}^\intercal \mathbf{e}\|\|} \mathbf{R}^{-1/2} \mathbf{J}^\intercal \mathbf{e}$

small scale datasets like 8-bitparity (8-BP), Modulo-2 function (Mod-2) and 2D-Gabor function (Gabor-2D) are also considered for experiments.

A. Implementation

Experiments are performed as per the standard data preprocessing steps followed in [42]. This comprises of two stages: hold out stage and 4-fold cross validation stage. In the hold-out stage, input patterns from the datasets are randomly split into 50% each for training and testing. Then, they are mean subtracted and standard deviation normalized. The trained neural network model obtained after this stage is further validated using four fold cross validation. The obtained test accuracies and convergence times in each of the four folds are averaged and tabulated in Table V. The convergence time is summation of average convergence time during 4-fold CV and the one during hold-out validation stage. Performance is reported for the maximum step-size η^* which satisfies the following criterion with the smallest ϵ. Here, $V(\mathbf{w};\eta)$ is the mean value of training loss for a particular step-size η.

$$\eta^* = \underset{\eta}{\operatorname{argmin}}\, V(\mathbf{w};\eta), \text{ where, } V(\mathbf{w};\eta) \leq V(\mathbf{w}^*) + \epsilon \quad (53)$$

B. Baseline

Various representative neural network learning algorithms are implemented as baseline. Several gradient based algorithms are proposed in the literature with varying rates of convergence, stability and generalization abilities. Some of the representative algorithms are Back Propagation (SGD), Adagrad(AD), Adam, RMSprop etc.

VI. Results and Discussion

Experiments are performed on the tasks presented in sections III-V. The sizes of the datasets for these tasks range from small to large. For each algorithm and each experiment, a constant learning rate (step-size), which achieves the best performance, is used.

1) Mental Imagery Classification: Training and testing of algorithms are performed subject-wise in both on-line and off-line conditions. Under off-line condition, the models are trained on data from one (or more) sessions and tested using data from another session. With on-line condition, initial 1 minute data from the session to be used for testing is implemented for training in an on-line fashion, i.e., samples arrive in a sequence and the model is updated mini-batch size at a time. Four fold cross validation is used while, training to validate the performances of all the algorithms used. For each algorithm, the model with the least validation error is used for testing. Table III shows the classification performance with different algorithms. The proposed HJB algorithm significantly outperforms the baseline methods, with a p-value < 0.05 in paired t-test over off-line as well as on-line conditions. In each cell, the first element indicates the accuracy for offline training/testing for each subject, while the second element signifies the accuracy for online setting, where an offline trained classifier is finetuned on a portion of the subsequent session's data. It is noticed that performance achieved after finetuning an offline trained classifier upon a portion of streaming data (online adaptation) is statistically equivalent to that of an offline classifier trained on the whole dataset (previous and current session's EEG data). Under online learning scenario, a proper functional BCI takes into account constant user feedback responses and this needs a complex adaptive step size for faster learning.

2) Eye-state Recognition: Four-fold cross validation is performed, i.e., the training set consists of 15 subjects, selected randomly, while the remaining subjects comprise the test set, and this split is repeated 4 times. Results are reported by averaging over multiple splits.

Table IV shows the performance of different learning algorithms for eye state recognition. All methods, except K* and RJ48F, use a NN with architecture $91 \times 50 \times 30 \times 10 \times 2$. Fig. 3 shows convergence rates with different learning algorithms. The proposed HJB-AD (HJB+Adagrad) converges faster as compared to others. Fig. 4 shows the ROC curve (receiver operator characteristics) for training with different algorithms. The plots in Fig. 4 sequentially correspond to BP, EVDHJB, HJB-AD, HJB, Adagrad, RMSprop and Adam algorithms. Area under the ROC (AUC) is highest for the HJB-AD algorithm. Paired t-test is conducted to obtain statistically significant $(p < 0.01)$ improvement in precision and recall rates for both HJB and HJB-AD algorithms in comparison to BP.

It is known that ensemble averaging of models improves the classification performance. Two networks are implemented with different architectures (architecture 1 is $91 \times 50 \times 30 \times 10 \times 2$ and architecture 2 is $91 \times 50 \times 30 \times 2$) and trained with various algorithms. Their outcomes are then averaged to get the final outcome (ensemble averaging). K* and Rotational Random Forests constitute the baseline methods. Random forests do not have many hyper-parameters to fine-tune (only, the number of trees; in general, more trees imply better performance). During the experiments, random forests take long-time to train in comparison to a multi layered neural network. This is visible in the wall clock time column of Table IV. Also, experiments reveal that random forests are not as good as neural network models for online learning in BCI's. This is because, in the latter case, the trees have to fall apart with every arrival of a new sample and be re-constructed leading to greater computational complexity. Here, NNs perform superior owing to the large data available in the form of multiple subjects.

3) UCI, LIBSVM and synthetic Datasets: Five independent trials are performed for each algorithm. Maximum number of iterations is fixed to $30, 50, 100, 150$ and 200, respectively

TABLE III: Classification Accuracy of different algorithms for Mental Imagery Classification task

Subject	Train → Test (sessions used)					Train → Test (sessions used)				
	1→2	1→3	2→3	1+2→3	1+2+3→4†	1→2	1→3	2→3	1+2→3	1+2+3→4†
	Algorithm[27]					**DNN with HJB**				
	67.8	69.5	72.4	75.2	75.3	73.5	68.5	74.9	77.4	82.8
1	72.8	72.3	75.4	76.3	79.5	82.2	74.3	85.4	87.3	88.4
	52.1	60.3	59.1	62.2	62.6	69.1	67.3	64.1	70.2	78.6
2	59.8	68.3	64.7	68.8	70.1	70.8	70.2	71.7	70.8	83.8
	52.5	43.4	38.9	42.6	51.0	65.4	66.4	63.6	73.1	76.2
3	61.7	46.2	39.4	43.8	56.2	74.4	69.3	68.3	75.8	83.1
	57.5	57.7	56.8	60.0	63.0	69.3	67.4	67.5	73.6	79.2
	64.8	62.3	59.8	63.0	68.6	75.8	71.3	75.1	77.9	85.1
	DNN with AD					**DNN with HJB + AD**				
	71.5	73.4	74.8	75.3	81.3	74.2	76.5	79.1	79.7	83.1
1	79.5	75.5	78.3	85.0	85.2	83.2	84.1	86.0	87.1	87.5
	65.3	67.9	64.2	68.7	78.2	66.0	68.5	66.2	69.7	79.7
2	68.1	69.9	68.9	70.3	81.6	69.4	70.9	69.2	70.1	83.2
	63.9	62.5	57.6	71.0	75.5	66.8	65.0	64.9	74.1	75.8
3	68.6	66.2	58.9	72.0	77.7	73.1	68.3	67.8	75.2	82.8
	66.9	67.9	65.5	71.7	78.3	69.0	70.0	70.1	74.5	79.5
	72.1	70.5	68.7	75.8	81.5	75.2	74.4	74.3	77.5	84.5

Each subject (and average) has two rows - the first row corresponds to offline training and the second row corresponds to online adaptation

†this last column corresponds to BCI competition III condition

*DNN has two hidden layers with architecture $96 \times 50 \times 30 \times 3$

TABLE IV: Evaluation Performance of algorithms and Ensemble Models for Eye State Recognition. In the first row, WCtime is wall clock time, TrAcc is training accuracy and TsAcc is test accuracy

Algorithm	Epochs	WCtime (in s)	TrAcc (in %)	TsAcc (in %)	Precision	Recall	Algorithm	Epochs	WCtime (in s)	TrAcc (in %)	TsAcc (in %)	Precision	Recall
K*	-	-	-	78.8	0.78	0.79	RJ48F	200	562.3	-	82.7	0.82	0.81
BP	270	38.2	98.9	91.2	0.91	0.92	LF	250	50.5	98.8	62.0	0.61	0.60
AD	200	22.5	98.0	96.0	0.96	0.92	EVDHJB	220	37.4	98.6	95.2	0.94	0.96
HJB	200	34.1	99.1	96.2	0.95	0.94	HJB+AD	180	20.2	99.2	97.0	0.98	0.95
Adam	187	25.5	98.2	96.3	0.96	0.97	RMSprop	192	26.5	98.4	96.8	0.97	0.95
Ens1	-	-	99.3	98.9	0.98	0.96	Ens3	-	-	99.0	98.2	0.97	0.96
Ens2	-	-	98.6	97.8	0.96	0.95	Ens4	-	-	98.7	96.1	0.93	0.94

TABLE V: Evaluation Results (convergence time in s, test error in %) for synthetic and UCI datasets. In the first column, N is number of samples, D is feature dimension, and k is 10^3. The proposed algorithms are in the last two columns.

Dataset (N, D)	RMSprop	BP	EVDHJB	AD	Adam	**HJB**	**HJB+AD**
BC (699, 9)	(4.47, 3.00)	(47.56, 4.28)	(30.65, 4.27)	(29.56, 3.18)	(8.54, 3.00)	(3.51, 3.14)	(20.83, 2.85)
8-BP (256, 8)	(0.26, 0.25)	(0.27, 0.25)	(0.87, 0.25)	(0.55, 0.25)	(0.35, 0.25)	(0.36, 0.25)	(0.26, 0.25)
Mod-2 (9, 2)	(3.34, 0.20)	(12.42, 0.20)	(1.53, .0.20)	(2.68, 0.20)	(2.63, 0.20)	(1.64, 0.20)	(7.69, 0.20)
Gabor-2D (100, 2)	(0.19, 0.02)	(0.47, 0.02)	(0.15, 0.02)	(0.54, 0.02)	(0.14, 0.02)	(0.19, 0.02)	(0.14, 0.02)
ADU (33k, 14)	(891.1, 15.89)	(976.2, 14.98)	(921, 16.29)	(871, 16.05)	(881.2, 15.93)	(398.7, 15.42)	(866.9, 15.83)
BM (4.5k, 16)	(351.12, 11.61)	(357.88, 11.54)	(411.34, 11.61)	(341.68, 11.61)	(347.44, 11.61)	(319.93, 10.77)	(322.54, 11.61)
MMG (961, 6)	(21.6, 16.04)	(22.96, 15.42)	(29.4, 19.06)	(22.26, 16.67)	(25.42, 15.72)	(23.11, 16.14)	(24.55, 16.87)
IJCNN (142k, 22)	(3479, 9.50)	(890, 9.79)	(6713, 9.57)	(5406.8, 9.57)	(3307, 9.50)	(739.7, 0.83)	(7497, 1.23)
WS (350k, 254)	(7196, 3.35)	(2848, 1.65)	(16082, 2.85)	(7437, 3.22)	(7451, 1.80)	(3704, 2.74)	(7711, 3.18)
HTRU2 (18k, 8)	(1110.93, 1.99)	(1268.3, 9.67)	(1247.6, 1.96)	(1085.6, 9.15)	(1053.2, 1.12)	(1051, 2.08)	(1062.6, 2.94)
Covtype (581k, 54)	(12228, 20.40)	(21183, 15.21)	(62125, 15.75)	(12497, 25.15)	(20728, 14.37)	(20310, 15.65)	(12200, 13.97)

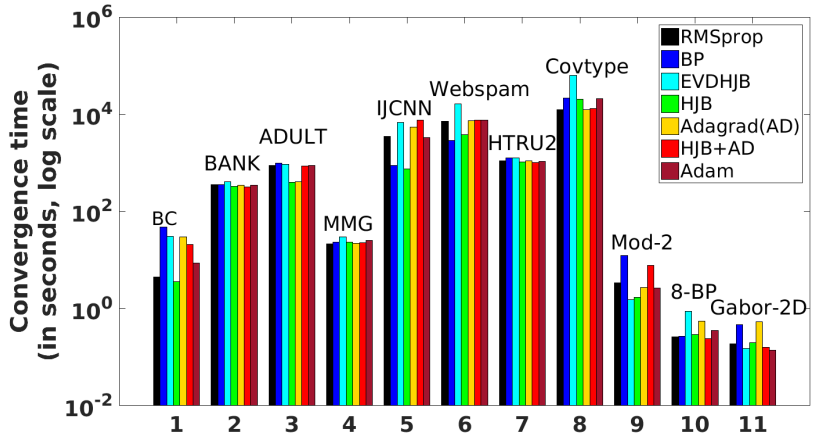

Fig. 5: Logarithmic convergence time over all datasets

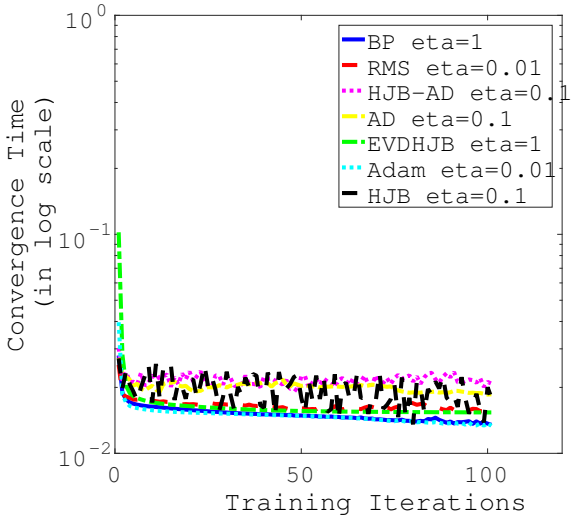

Fig. 6: Training error convergence for stabilized eta values

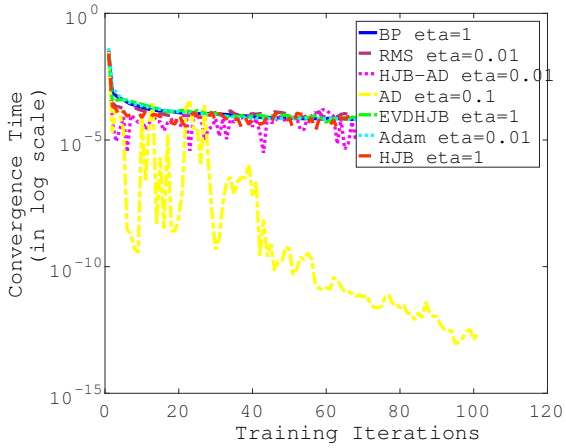

Fig. 7: Generalization error convergence for stabilized eta values

for each trial. Mini-batch size is taken as 8, 128 and 256 for the datasets with sizes of the order of 100, 10^3 and 10^5, respectively. Several properties which are crucial for the learning algorithms like stability, convergence time, generalization ability are analyzed in detail in the experiments. Fig. 5 contains the average convergence time (in *log* scale) for all the algorithms run on UCI and LIBSVM datasets. The proposed HJB & HJB-AD algorithm recorded the least convergence times on all datasets, except webspam dataset. The convergence time of EVD version of HJB is high on the large-scale datasets like Covtype and Webspam due to the computational overhead of eigen value decomposition, but the convergence time performance of EVD version of HJB is better on the IJCNN (another large-scale dataset) due to the dominating property of EVDHJB to converge to global optimum despite the computational overhead. Furthermore, the stability of algorithms is analyzed by the plots of training error over 100 iterations for various step sizes in the set: $\{0.01, 0.1, 1, 5\}$ as per (53).

Figs. 6 and 7 displays the training error and generalization error evaluated on HTRU2 dataset for the stabilizing step sizes in (53). A general trend that has been observed shows faster convergence with small and moderate step-size values $\{0.01, 0.1, 1\}$. Detailed stability analysis plots are provided as a supplementary material.

Several algorithms are prone to be stranded in local minima in the process for moderate and large step sizes of $\{1, 5\}$. Excepting HJB and HJB-EVD schemes, all other algorithms demonstrate a significantly higher value of training and generalization loss value than the optimal value. It is observed that on increasing the number of iterations, oscillations in the plots of HJB algorithm decrease and, finally, converge to optimal value.

Table V summarizes the test data accuracy and convergence times. HJB based algorithms outperform other algorithms on 9 out of 11 datasets with a statistically significant ($p < 0.01$) convergence time. The difference is more appropriately noticed on the datasets with large number of samples like IJCNN, Webspam, Covtype datasets. In order to control over-fitting,

both training and generalization errors are monitored.

VII. Conclusion

A novel learning algorithm is presented in this paper, based on control theoretic principles. It is used for training NNs for brain-computer interface applications, namely, mental imagery classification and eye state recognition. A new dataset is prepared for the latter application. The performance of the proposed algorithm is compared with that of several state of the art algorithms. Experimental results show that the proposed algorithms bring significant improvements in terms of faster convergence as well as better accuracies. Also, the computational complexity of the proposed update rule is linear in the number of NN weights as well as the number of training samples, making it computationally efficient. Although the NNs used in this paper have 2 to 3 hidden layers, the proposed algorithm gives good results even without pre-training using methods like restricted Boltzmann machine. In the future, experiments and weight update law can be optimized to accomodate datasets with large feature dimensions. In addition, extension of the formulation to non-convex optimization problems, for example, deep neural networks is an interesting future prospect. In future, we intend to address the convergence of HJB based scheme for the class of functions known as μ-PL non-convex loss functions [43].

VIII. Acknowledgement

The work is funded in part by the TCS fellowship (Tharun Kumar Reddy) under the project TCS/CS/2011191A and also in part by the Ministry of Human Resource Development (MHRD), Government of India under the project 'MHRD-EE-2016150'.

References

[1] J. Duchi, E. Hazan, and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[2] D. P. Kingma and J. B. Adam, "A method for stochastic optimization. 2014," *arXiv preprint arXiv:1412.6980*.

[3] Y. Iiguni, H. Sakai, and H. Tokumaru, "A real-time learning algorithm for a multilayered neural network based on the extended kalman filter," *IEEE Transactions on Signal processing*, vol. 40, no. 4, pp. 959–966, 1992.

[4] L. Behera, S. Kumar, and A. Patnaik, "On adaptive learning rate that guarantees convergence in feedforward networks," *IEEE transactions on neural networks*, vol. 17, no. 5, pp. 1116–1125, 2006.

[5] J. Bilski and L. Rutkowski, "A fast training algorithm for neural networks," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 45, no. 6, pp. 749–753, 1998.

[6] V. Arora, L. Behera, T. Reddy, and A. P. Yadav, "HJB Equation Based Learning Scheme for Neural Networks," in *International Joint Conference on Neural Networks (IJCNN)*, 2017.

[7] S. H. Zak, *Systems and control*. Oxford University Press New York, 2003, vol. 174.

[8] L. Behera and I. Kar, *Intelligent Systems and control: principles and applications*. Oxford University Press, Inc., 2010.

[9] S. Ferrari, J. E. Steck, and R. Chandramohan, "Adaptive feedback control by constrained approximate dynamic programming," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 38, no. 4, pp. 982–987, 2008.

[10] Q. Wei, F.-Y. Wang, D. Liu, and X. Yang, "Finite-approximation-error-based discrete-time iterative adaptive dynamic programming," *IEEE Transactions on Cybernetics*, vol. 44, no. 12, pp. 2820–2833, 2014.

[11] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[12] D. Dheeru and E. Karra Taniskidou, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[13] K.-R. Muller, C. W. Anderson, and G. E. Birch, "Linear and nonlinear methods for brain-computer interfaces," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 11, no. 2, pp. 165–169, 2003.

[14] S. Bhattacharyya, A. Khasnobish, A. Konar, D. Tibarewala, and A. K. Nagar, "Performance analysis of left/right hand movement classification from eeg signal by intelligent algorithms," in *Computational Intelligence, Cognitive Algorithms, Mind, and Brain (CCMB), 2011 IEEE Symposium on*. IEEE, 2011, pp. 1–8.

[15] S. Bhattacharya, R. J. Haddad, and M. Ahad, "A multiuser eeg based imaginary motion classification using neural networks," in *SoutheastCon, 2016*. IEEE, 2016, pp. 1–5.

[16] D. Coyle, "Neural network based auto association and time-series prediction for biosignal processing in brain-computer interfaces," *IEEE Computational Intelligence Magazine*, vol. 4, no. 4, 2009.

[17] D. Coyle, G. Prasad, and T. M. McGinnity, "A time-series prediction approach for feature extraction in a brain-computer interface," *IEEE transactions on neural systems and rehabilitation engineering*, vol. 13, no. 4, pp. 461–467, 2005.

[18] G. Pfurtscheller, C. Neuper, A. Schlogl, and K. Lugger, "Separability of eeg signals recorded during right and left motor imagery using adaptive autoregressive parameters," *IEEE transactions on Rehabilitation Engineering*, vol. 6, no. 3, pp. 316–325, 1998.

[19] E. Haselsteiner and G. Pfurtscheller, "Using time-dependent neural networks for eeg classification," *IEEE transactions on rehabilitation engineering*, vol. 8, no. 4, pp. 457–463, 2000.

[20] C. W. Anderson and Z. Sijercic, "Classification of eeg signals from four subjects during five mental tasks," in *Solving engineering problems with neural networks: proceedings of the conference on engineering applications in neural networks (EANN'96)*. Turkey, 1996, pp. 407–414.

[21] C. Anderson, E. Forney, D. Hains, and A. Natarajan, "Re-

liable identification of mental tasks using time-embedded eeg and sequential evidence accumulation," *Journal of Neural Engineering*, vol. 8, no. 2, p. 025023, 2011.

[22] E. M. Forney and C. W. Anderson, "Classification of eeg during imagined mental tasks by forecasting with elman recurrent neural networks," in *Neural Networks (IJCNN), The 2011 International Joint Conference on*. IEEE, 2011, pp. 2749–2755.

[23] J. d. R. Millan, "On the need for on-line learning in brain-computer interfaces," in *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, vol. 4. IEEE, 2004, pp. 2877–2882.

[24] L. F. Nicolas-Alonso and J. Gomez-Gil, "Brain computer interfaces, a review," *Sensors*, vol. 12, no. 2, pp. 1211–1279, 2012.

[25] U. Chaudhary, N. Birbaumer, and A. Ramos-Murguialday, "Brain-computer interfaces for communication and rehabilitation," *Nature Reviews Neurology*, vol. 12, no. 9, pp. 513–525, 2016.

[26] D. Marshall, D. Coyle, S. Wilson, and M. Callaghan, "Games, gameplay, and bci: the state of the art," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 5, no. 2, pp. 82–99, 2013.

[27] F. Galán, F. Oliva, and J. Guardia, "BCI competition III. data set v: Algorithm description," *Brain Computer Interfaces Competition III*, 2005.

[28] E. L.-A. Vivas, A. García-González, F. Galindo, and V. Sánchez-González, "Algorithm to detect six basic commands by the analysis of electroencephalographic and electrooculographic signals," in *Biomedical Robotics and Biomechatronics (BioRob), 2012 4th IEEE RAS & EMBS International Conference on*. IEEE, 2012, pp. 189–194.

[29] E. López-Arce, R. Fuentes-Aguilar, I. Figueroa-García, and A. García-González, "Analysis and comparison of classification methods in a brain machine interface," in *Human-Computer Systems Interaction: Backgrounds and Applications 3*. Springer, 2014, pp. 63–73.

[30] R. J. Barry, A. R. Clarke, S. J. Johnstone, C. A. Magee, and J. A. Rushby, "Eeg differences between eyes-closed and eyes-open resting conditions," *Clinical Neurophysiology*, vol. 118, no. 12, pp. 2765–2773, 2007.

[31] O. Rösler and D. Suendermann, "A first step towards eye state prediction using eeg," *Proc. of the AIHLS*, 2013.

[32] J. G. Cleary, L. E. Trigg *et al.*, "K*: An instance-based learner using an entropic distance measure," in *Proceedings of the 12th International Conference on Machine learning*, vol. 5, 1995, pp. 108–114.

[33] D. Singla and P. S. Rana, "Eye state prediction using ensembled machine learning models," in *Inventive Computation Technologies (ICICT), International Conference on*, vol. 2. IEEE, 2016, pp. 1–5.

[34] C. R. Hamilton, S. Shahryari, and K. M. Rasheed, "Eye state prediction from eeg data using boosted rotational forests," in *Machine Learning and Applications (ICMLA), 2015 IEEE 14th International Conference on*. IEEE, 2015, pp. 429–432.

[35] T. K. Reddy and L. Behera, "Online eye state recognition from eeg data using deep architectures," in *Systems, Man, and Cybernetics (SMC), 2016 IEEE International Conference on*. IEEE, 2016, pp. 000 712–000 717.

[36] D. E. Kirk, *Optimal control theory: an introduction*. Courier Corporation, 2012.

[37] Y. Mu, W. Liu, X. Liu, and W. Fan, "Stochastic gradient made stable: A manifold propagation approach for large-scale optimization," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 2, pp. 458–471, 2017.

[38] L. Xiao and T. Zhang, "A proximal stochastic gradient method with progressive variance reduction," *SIAM Journal on Optimization*, vol. 24, no. 4, pp. 2057–2075, 2014.

[39] M. Schmidt, N. Le Roux, and F. Bach, "Minimizing finite sums with the stochastic average gradient," *Mathematical Programming*, vol. 162, no. 1-2, pp. 83–112, 2017.

[40] G. Gómez-Herrero, W. De Clercq, H. Anwar, O. Kara, K. Egiazarian, S. Van Huffel, and W. Van Paesschen, "Automatic removal of ocular artifacts in the eeg without an eog reference channel," in *Signal Processing Symposium, 2006. NORSIG 2006. Proceedings of the 7th Nordic*. IEEE, 2006, pp. 130–133.

[41] W. De Clercq, A. Vergult, B. Vanrumste, W. Van Paesschen, and S. Van Huffel, "Canonical correlation analysis applied to remove muscle artifacts from the electroencephalogram," *IEEE transactions on Biomedical Engineering*, vol. 53, no. 12, pp. 2583–2587, 2006.

[42] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3133–3181, 2014.

[43] S. J. Reddi, S. Sra, B. Poczos, and A. J. Smola, "Proximal stochastic methods for nonsmooth nonconvex finite-sum optimization," in *Advances in Neural Information Processing Systems*, 2016, pp. 1145–1153.