Classification Error Correction: A Case Study in Brain-Computer Interfacing

Hasan A. Poonawala, Mohammed Alshiekh, Scott Niekum and Ufuk Topcu

***Abstract*— Classification techniques are useful for processing complex signals into labels with semantic value. For example, they can be used to interpret brain signals generated by humans corresponding to a finite set of commands for a physical device. The classifier, however, may interpret the signal as a command that is different from the intended one. This error in classification leads to poor performance in tasks where the class labels are used to learn some information or to control a physical device. We propose a computationally efficient algorithm to identify which class labels may be misclassified out of a sequence of class labels, when these labels are used in a given learning or control task. The algorithm is based on inference methods using Markov random fields. We apply the algorithm to goal-learning and tracking using brain-computer interfacing (BCI), in which signals from the brain are commonly processed using classification techniques. We demonstrate the proposed algorithm reduces the time taken to identify the goal state in control experiments.**

I. Introduction

In most dynamical systems, the output of the system is a vector of real numbers obtained through sensors. The sensed output is used in estimation and feedback control techniques for achieving various tasks, such as set-point regulation or trajectory tracking. If the sensor is noisy, the noise usually takes the form of an additive numerical term with zero mean. In some systems, a classification process [1] yields the sensed output, which is a member of a discrete finite set of labels instead of a continuous value. The noise in classification processes results in a feature or data sample being assigned a label different from the true label that should have been assigned to the feature. This type of error is different from noise in systems with continuous numeric outputs, since real numbers have a well-known ordering relation but a finite set of labels has no *a priori* order.

The shared control of semi-autonomous devices using brain-computer interfaces (BCIs) [2] is an example where classifier outputs are used for control. Brain-computer interface technology will enable increased effectiveness of human-machine interaction in general. Current research in BCIs is largely motivated by human-machine interaction in the context of rehabilitation devices. The human nervous system generates signals which can be recorded using techniques such as electroencephalography (EEG) or electromyography (EMG). A signicant number of BCI approaches extract signals from the human brain by recording brain activity in the form of EEG signals through the human scalp [3]–[6]. Using only EEG has the benefit of requiring the user to wear only one piece of headgear to allow interaction with the device. However, extracting meaningful signals using EEG is challenging [2]. Determination of the user's intention is achieved either by training the user to generate a fixed set of signals, or using machine learning techniques to classify recorded EEG signals [3], [5], [6].

A common task for such devices is to reach a goal state known by the user but unknown to the device, using only classified brain signals as available feedback [3]–[6]. The user either generates brain signals corresponding to control actions, or performs the role of an expert who evaluates the actions selected by the device. Either way, the information that the human wants to transmit is extracted as complex and noisy EEGs signals in many situations, and a classifier may be required to interpret this information. A technique for learning which goal is intended by the user is presented in [5]. However, no technique to infer when misclassification occurs is attempted. The presence of misclassified commands or evaluations in the goal-learning task results in longer times taken to reach the goal [5].

Contributions

We develop an algorithm to estimate when classification errors occurred in control through BCI. The main intellectual contribution is the insight that Markov random fields (MRFs) can be used to construct a useful prior distribution over the true class labels associated with a set of observed class labels. This prior distribution is then used to perform Bayesian inference. We also show how judicious choice of the Markov random field's structure renders the algorithm computationally efficient and therefore suitable for real-time implementation. Through simulations and experiments, we demonstrate that using the proposed algorithm to identify classification errors results in lower times taken to estimate the unknown goal intended to be reached by the user.

II. Background

A. Finite Transition Systems

A finite transition system (FTS) [7] consists of a tuple (S, A, T) where S is a finite set of states, A is a finite set of actions, and $T : S \times A \to S$ is a transition function. The evolution of a FTS occurs in discrete time. At every time step $t \in \mathbb{N}$, the state is $s(t) \in S$, and an action $a(t) \in$

This work was supported by grants from AFRL (#FA8650-15-C-2546), DARPA (#W911NF-16-1-0001), ARO (#W911NF-15-1-0592), and NSF (#1550212 and #1652113).

Hasan A. Poonawala and Mohammed Alshiekh are with the Institute for Computational Engineering and Science, University of texas at Austin, Austin, TX 78712, USA. hasanp@utexas.edu, malshiekh@utexas.edu

Scott Niekum is with the Department of Computer Science, University of texas at Austin, Austin, TX 78712, USA. sniekum@cs.utexas.edu

Ufuk Topcu is with the Department of Aerospace Engineering, University of texas at Austin, Austin, TX 78712, USA. utopcu@utexas.edu

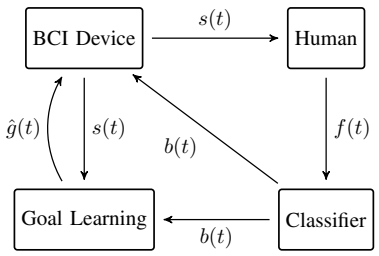


Fig. 1: Human and BCI system together as a classifier-in-the-loop system. Once the goal learning is complete, the BCI system can take over control from the human.

A is selected which results in a new state determined by $T(s(t), a(t))$.

B. Classifiers

A classifier $\mathcal{C}$ consists of a tuple (F, L, Q) where F is a set of features, L is a finite set of class labels, and Q is a map that assigns a label from L to a feature $f \in F$. A confusion matrix R captures the imperfection of the classification process, and is obtained during testing of the classifier. If the classifier is perfect, then R is the identity matrix.

C. Markov Random Fields

A Markov random field (MRF) [8] is a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is a set of nodes, and $\mathcal{E}$ is a set of undirected edges in $\mathcal{V} \times \mathcal{V}$. Furthermore, each node $i \in \mathcal{V}$ is associated with a random variable x_i.

The main property of a MRF is that the edge structure $\mathcal{E}$ determines the conditional independence of the nodes. That is, a node is conditionally independent of all nodes that it is not connected to via an edge:

$$p\left(x_i \colon \{x_j\}_{(j)\in\mathcal{V}\setminus\{i\}}\right) = p\left(x_i \colon \{x_j\}_{(i,j)\in\mathcal{E}}\right). \tag{1}$$

The random variables x_i for each node are combined together to form the random variable $\mathbf{x}$. The symbol $p(\mathbf{x})$ denotes the joint probabilitiy distribution of the random variables x_i. The random variables x_i of a discrete MRF have values in a finite discrete set. A sample from $p(\mathbf{x})$ consists of an assignment of one of the discrete values to each node in $\mathcal{V}$. We will refer to such a sample as an assignment of the MRF, or simply, an assignment.

III. Goal-Learning Through Brain-Computer Interfaces

A common task to be achieved by an assistive device is for it to reach a desired goal state $g \in S$ determined by the human user [3]–[6], but unknown to the device, where S is the set of states that can be reached by the device. The device must learn the goal based on the user's commands.

We model the dynamics of the device as a finite transition system. We will focus on a finite state space arranged in a one-dimensional grid, as seen in Figure 2. The set A of actions of the device is $\{a_0, a_1, a_2\}$ corresponding to moving left, staying in the same state, and moving right. As shown in Section V, we can treat the case of multi-dimensional state spaces using the case of one-dimensional state spaces.

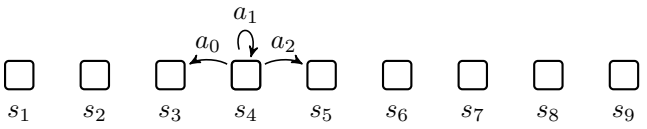


Fig. 2: Example of a finite transition system.

The transition function T of this device is given by three simple rules:

$$T(s_i, a_2) = s_{i+1}, \tag{2}$$
$$T(s_i, a_0) = s_{i-1}, \text{ and} \tag{3}$$
$$T(s_i, a_1) = s_i. \tag{4}$$

Of course, we must impose the end cases $T(s_1, a_0) = s_1$ and $T(s_{|S|}, a_2) = s_{|S|}$.

We denote the state of the system at time t by $s(t)$. At each time step t, the human generates an EEG-based feature $f(t)$ to command an action $a(t) \in \{a_0, a_1, a_2\}$ (see Figure 1). The classifier Q, obtained by prior training, classifies $f(t)$ as label $b(t)$. There is, however, a non-zero probability that $a(t) \neq b(t)$. The probability of making an error in classification is captured by the confusion matrix R of $\mathcal{C}$ obtained during testing. To summarize, the device is controlled via a classifier $\mathcal{C} = (F, L, Q)$, where F is the set of EEG features that can be obtained, $L = A$, and Q has confusion matrix R associated with it. We refer to the pair $(s(t), b(t))$ as the state-label pair at time t, where $s(t)$ and $b(t)$ are the state and label at time t respectively.

We represent the estimate $\hat{g}(t) \in S$ of the goal g at any time t by a probability mass-like function P_t defined on the finite state space S. Note that $g = s_j$ for some $j \in \{1, 2, \ldots, |S|\}$. Once a state $s_i \in S$ uniquely achieves a value of $P_t(s_i)$ greater than a threshold $\delta \in [0, 1]$, we take it to be the goal state. We denote the time step at which a goal state is identified by t_f. We want the value of t_f to be small, implying that the goal is identified quickly.

We update the values of $P_t(s_i)$ for all $s_i \in S$ after every action is received. The update takes the form

$$P_{t+1}(s_i) = \frac{k_t(s_i)}{\eta_t} P_t(s_i), \tag{5}$$

where $s_i \in S$, and η_t is a normalizing factor given by $\eta_t = \sum_{s_i \in S} k_t(s_i) P_t(s_i)$. The term $k_t(s_i)$ acts as a multiplicative update factor of $P_t(s_i)$.The update factor $k_t(s_i)$ can be only one of the elements of $\{k_L, k_H\}$ for each $s_i \in S$. The parameters k_L and k_H are any two real numbers such that $k_H > k_L > 0$.

Let the current state be s_i. If $b(t) = a_1$ then $k_t(s_i) = k_H$, and $k_t(s_j) = k_L$ for all other states $s_j \neq s_i$. If $b(t) = a_2$, then $k_t(s_j) = k_H$ for all $s_j \in S$ where $j > i$, and $k_t(s_j) = k_L$ for all $j \leq i$. If $b(t) = a_0$, then $k_t(s_j) = k_H$ for all $s_j \in S$ where $j < i$, and $k_t(s_j) = k_L$ for all $j \geq i$.

If actions are classified perfectly, then $P_t(g)$ will increase monotonically and become larger than the threshold δ, and

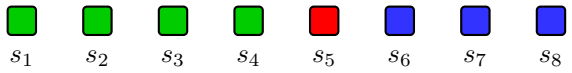


Fig. 3: Optimal actions to be taken in each state in order to reach the goal s_5 and stay there: a_2 is indicated by the color dark green, a_1 by red, and a_0 by blue.

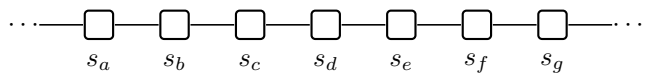
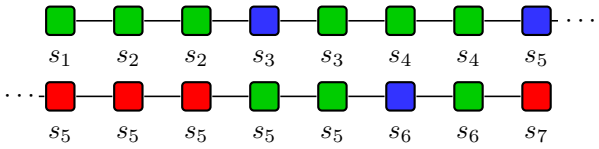
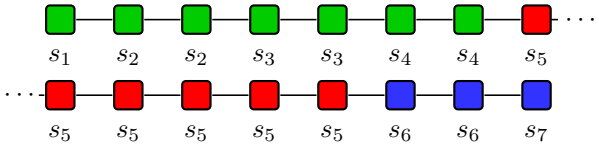


Fig. 4: The grid graph structure of the MRF with each node corresponding to a state-label pair. The nodes are ordered based on the state of each state-label pair (indicated below the node), meaning that the indices of the states are such that $a \leq b \leq c \leq d \leq \ldots \leq g$. Equality of the indices occurs when a state is visited multiple times.



(a) An example of the MRF constructed from the state-label pairs $(s(t), b(t))$ obtained for 16 time steps, where $s(1) = s_1$. There are six time instants when the label is misclassified.



(b) The assignment of the MRF in Figure 5a when the optimal action is assigned in each state.

Fig. 5: Examples of MRFs obtained for a sequence of state-label pairs. Each state-label pair is associated with a node in the MRF. The state associated with a node is indicated below the node, and the label is indicated by the color of the node. a_2 is indicated by the color dark green, a_1 by red, and a_0 by blue.

therefore g will always be identified as the goal state given enough time steps. However, due to misclassification, $P_t(g)$ may not always increase. Even if $P_t(g)$ eventually crosses the threshold, it will take more time steps to do so when compared to the case without any classification errors. To prevent this increase in t_f, we want to identify which actions have been misclassified. We define the following problem.

Inference problem: Given a sequence of state-label pairs $(s(t), b(t))$, $t \in \{1, 2, \ldots, N\}$, estimate the true labels $a(t)$ associated with $b(t)$ for $t \in \{1, 2, \ldots, N\}$.

IV. Inference Using Markov Random Fields

Assume that a human user generates a feature f_i to communicate a class label $x_i \in L$ to a device. The classifier assigns the class label y_i to f_i, and it is possible that $y_i \neq x_i$. The class label y_i is a noisy measurement of the true class label x_i. We combine the measurements y_i obtained at different time steps to obtain the set $\mathbf{y}$ of measuremed class labels. We treat the true class labels $\mathbf{x}$ as random variables since they cannot be observed directly. Our aim is to estimate the most likely class labels $\hat{\mathbf{x}}$ given the measured class labels $\mathbf{y}$. We will use Bayesian inference to estimate $\hat{\mathbf{x}}$. The error rates of the classifier will determine the likelihood function $p(\mathbf{y}|\mathbf{x})$. The prior distribution $p(\mathbf{x})$ is undetermined. We describe the procedure to choose this prior distribution in an intelligent way.

A naive choice for the prior distribution is that all possible values of $\mathbf{x}$ are equally likely; we choose the prior differently. Figure 3 shows an example of the optimal actions in the states given a goal state. Notice that the optimal action to be taken in two adjacent states is almost always the same. The only exceptions occur when one of the two adjacent states is the goal state. This spatial pattern of the true class labels in states is independent of which state the goal is, and we want a prior distribution that associates a high probability to assignments $\mathbf{x}$ that possess such a spatial pattern of state-label pairs.

We obtain this desired prior distribution by using Markov random fields (MRFs). In order to specify the MRF, we must provide the connectivity structure and the clique energy functions [8]. Inference using Markov random fields becomes an energy-minimization problem. In general, this energy-minimization problem is difficult to solve. We show how to choose the connectivity and energy functions in such a way that approximate energy-minimization can be performed using efficient algorithms [9]–[11].

Each node in the MRF $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ will correspond to a state-label pair $(s(t), b(t))$ for some $t \in \mathbb{N}$. Therefore, $|\mathcal{V}| = N$. We connect the nodes to form a chain, as seen in Figure 4. Each node has two neighbors in this graph, except the nodes at the ends of the chain. Consider a node associated with state s_i, and let the states associated with its two neighboring nodes be s_j and s_k. Then, $j \leq i \leq k$ or $k \leq i \leq j$. This condition simply says that the graph $\mathcal{G}$ represents a chain of nodes arranged so that the corresponding states s_i of the nodes are in order of the indices i. Figure 5 shows an example of such an MRF. The MRF can accommodate missing state-label pairs, or repeated state-label pairs obtained at different times.

The Hammersley-Clifford Theorem [12] states that the prior distribution $p(\mathbf{x})$ of an MRF can be expressed as the (normalized) exponential of the sum of energy functions defined over the cliques of the graph. A clique of a graph is a subset of nodes of the graph in which all nodes in the subset are connected to all other nodes in the subset. Let $\mathcal{C}$ be the cliques in the MRF. Let $\mathbf{x_c}$ be the set of random variables corresponding to nodes in $c \in \mathcal{C}$. Then, $p(\mathbf{x}) = \frac{1}{Z} \exp\left(-\sum_{c \in \mathcal{C}} V_c(\mathbf{x_c})\right)$, where Z is an appropriate normalizing factor.

The chain graph structure of the MRF ensures that the cliques consist of only pairs of nodes connected by an edge. The clique energy function V_c for cliques in the MRF is given by

$$V_c(x_i, x_j) = \begin{cases} \beta & \text{if } x_i \neq x_j \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

where $\beta > 0$ is an adjustable parameter. The function V_c

is also referred to as the *binary potential energy* function of the MRF. An important property of V_c is that it is submodular [11], since, for any $a, b \in L$,

$$V_c(a,a) + V_c(b,b) \leq V_c(a,b) + V_c(b,a). \tag{7}$$

Informally, an energy function V_c is submodular when its arguments that are more similar are assigned smaller energy. The submodularity of V_c implies that $p(\mathbf{x})$ is higher when connected labels are similar, which a property we want the MRF to possess. Furthermore, efficient inference methods exist for MRFs with submodular binary potential energy functions [11].

We define the *unary potential energy* function D as

$$D(y_i, x_i) = -\log p(y_i|x_i). \tag{8}$$

Equivalently, $p(y_i|x_i) = e^{D(y_i,x_i)}$. Therefore, $p(\mathbf{y}|\mathbf{x}) = e^{\sum_{i\in\mathcal{V}} D(y_i,x_i)}$. The maximum *a posteriori* (MAP) estimate is obtained from the posterior distribution as

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} p(\mathbf{x}|\mathbf{y}).$$

Since $p(\mathbf{y}|\mathbf{x})$ and $p(\mathbf{x})$ can be represented by energy functions D and V_c, computing $\hat{\mathbf{x}}$ is equivalent to the following energy-minimization problem (see [8] for details):

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \sum_{i\in\mathcal{V}} D(y_i, x_i) + \sum_{c\in\mathcal{C}} V_c(\mathbf{x}_c). \tag{9}$$

If L contains only two labels, then the global minimum of the right hand side of (9) can be found exactly in polynomial time [10], [11], provided V_c is submodular. Computing $\hat{\mathbf{x}}$ becomes equivalent to constructing a graph $\mathcal{G}_{min}$ with edge weights determined by the energy functions, and obtaining the min-cut max-flow [10], [11] solution determined by this graph. In general, solving the energy-minimization problem with more than two labels exactly is computationally expensive. However, an approximate minimum within a known factor of the global minimum can be computed using an iterative scheme [9].

To summarize, we develop an inference algorithm that involves two steps. The first step is to construct a MRF from the state-label pairs that encodes the prior distribution over the true class labels associated with the measured labels. The second step is to use an efficient inference algorithm, namely the swap algorithm from [9], to obtain the true class labels given these measured class labels.

V. Multi-dimensional State Spaces

We use the method for goal-learning in one-dimensional state spaces to solve the goal-learning problem in state spaces modeled by regular n-dimensional grids. We reduce the goal-learning in an n-dimensional grid to n goal-learning problems in one-dimensional grids. Figure 6 shows an example of this reduction.

Each state in the n-dimensional grid has n indices, one index per dimension. The index denotes the position of the state along the associated dimension (see Figure 6, for example). For the k^{th} dimension, we create a one-dimensional grid where the states are denoted as s_i^k. The

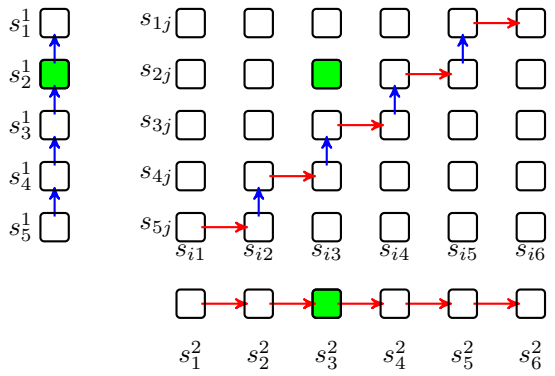


Fig. 6: Example of the decomposition of a two-dimensional grid into two one-dimensional grids. A state s_{ij} in the two-dimensional grid has index i and j corresponding to the first and second dimensions of the grid. The states s_i^1 and s_j^2 in the vertical and horizontal one-dimensional grids respectively represent s_{ij}. The transitions in the two-dimensional grid marked by blue arrows become transitions in the vertical one-dimensional grid, while those marked with red arrows become transitions in the horizontal one-dimensional grid. The goal s_{23}, marked green, in the two-dimensional grid becomes the goals s_2^1 and s_3^2 in the two one-dimensional grids respectively.

subscript i indicates that this state is the i^{th} state in the one-dimensional grid associated with the k^{th} dimension. Moreover, this state represents all those states in the n-dimensional grid which have their k^{th} index equal to i. In a regular n-dimensional grid, a transition results in the next state differing from the previous state by only one index, say the k^{th} index. Therefore, this transition in the n-dimensional grid induces a transition only in the one-dimensional grid associated with the k^{th} dimension. In this way, we create n one-dimensional goal-learning problems. We combine the solution to the n goal-learning problems in one dimension to obtain the solution to the goal-learning problem in n dimensions.

VI. Simulations

We simulate a goal-learning scenario for one-dimensional and two-dimensional state spaces in multiple trials. In each trial, we update the state based on a sequence of actions. The probability of choosing the optimal action in a state is 0.7 in the one-dimensional case, and 0.6 in the two-dimensional case. The remaining non-optimal actions in a state are equally likely to be chosen. Choosing a non-optimal action in a state simulates a classification error. This process yields a sequence of state-label pairs in a trial.

At each time step, the available sequence of state-label pairs yields a set of measured labels $\mathbf{y}$, and a set of estimated true labels $\hat{\mathbf{x}}$ obtained by using our inference algorithm. We choose β to be 1.0 in our algorithm. We compute the energy function (8) using the simulated error probabilities mentioned in the previous paragraph. We use both $\mathbf{y}$ and $\hat{\mathbf{x}}$ to identify the goal using separate probability mass functions, as described in Section III, for each set of labels. The threshold δ is 0.95. We denote the time taken to identify the goal when using the measured labels $\mathbf{y}$ as $t_f(\mathbf{y})$, and that when using

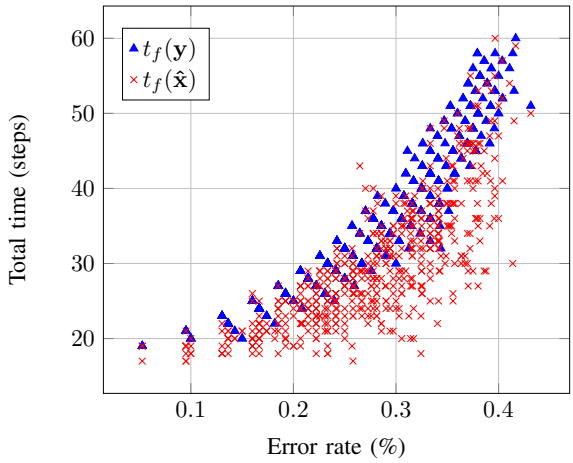


Fig. 7: Scatter plot of time $t_f(\mathbf{y})$ taken to identify the goal using the measured labels and time $t_f(\hat{\mathbf{x}})$ taken to identify the goal using the labels estimated by our inference algorithm, vs the error rate associated with $t_f(\mathbf{y})$, for trials in Sim 1.

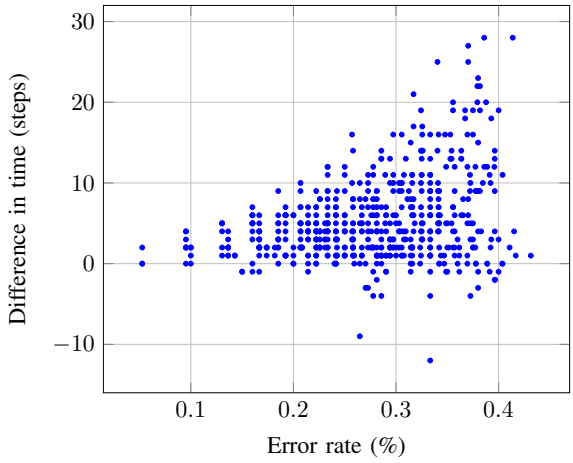


Fig. 8: Scatter plot of $t_f(\mathbf{y}) - t_f(\hat{\mathbf{x}})$ versus the error rate, for trials in Sim 1.

the estimated labels $\hat{\mathbf{x}}$ as $t_f(\hat{\mathbf{x}})$. The error rate for a trial is the ratio of the total number of errors and the value of t_f. We stop a trial after 60 time steps have passed.

We refer to these one-dimensional and two-dimensional simulations as Sim 1 and Sim 2 respectively. Figures 7 and 8 show the results for Sim 1. Each point denotes a single trial. The ordinate is the value of t_f for the trials. The abscissa is the error rate in the trial associated with $t_f(\mathbf{y})$. Similarly, Figures 9 and 10 show the results for Sim 2.

The null hypothesis for our simulation experiment is that the value of t_f is not affected by whether the estimated true labels $\hat{\mathbf{x}}$ or the measured labels $\mathbf{y}$ are used to identify the goal. Figures 7 and 9 show that on average, $t_f(\hat{\mathbf{x}})$ is smaller than $t_f(\mathbf{y})$. The t-statistic values (penultimate row of Table I) for Sim 1 and Sim 2 are greater than the required paired t-test values (bottom row of Table I) for rejecting the null hypothesis with 99.95% confidence. We conclude that the proposed inference algorithm enables faster identification of the goal on average, when compared to using the measured

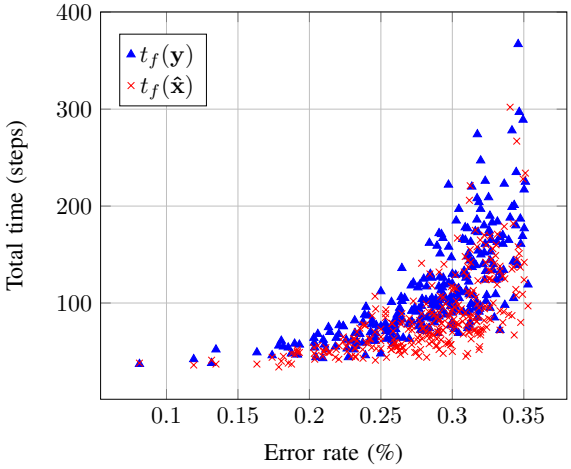


Fig. 9: Scatter plot of time $t_f(\mathbf{y})$ taken to identify the goal using the measured labels and time $t_f(\hat{\mathbf{x}})$ taken to identify the goal using the labels estimated by our inference algorithm, vs the error rate associated with $t_f(\mathbf{y})$, for trials in Sim 2.

	Sim 1	Sim 2	Sim 3	Exp 1
Mean	4.63	48.06	44.14	3.59
Std. dev.	5.46	49.91	34.77	4.14
Min	−9	−44	0	−1
Max	31	182	190	18
# of samples	285	84	483	43
t-statistic	14.56	8.83	27.899	5.67
$t_{.9995}$	3.39	3.416	3.39	3.551

TABLE I: Results of statistical analysis of the data for the simulations (Sim 1-3) and experiment (Exp 1). The unit for the first four rows is the number of time steps.

class labels. Figures 7-10 also show that the reduction in time steps increases with the error rate. This matches our expectation that the inference algorithm results in faster goal-learning when the number of errors are higher, by identifying classification errors.

For some trials, $t_f(\hat{\mathbf{x}})$ is larger than $t_f(\mathbf{y})$. The state-label pairs corresponding to these trials contain several errors that occur in the same state in consecutive time steps, and our inference algorithm is less likely to correct these errors. Furthermore, for Sim 2, the t-statistic is not as high as for Sim 1. We simulate a case where the parameters for the energy function (8) are different from the values determined by the simulated probabilities of choosing actions in a state. We decrease the probability of choosing the optimal action to 0.36, and the probability of choosing the non-optimal actions in a state to be 0.16. The resulting simulation is referred to as Sim 3. Table I shows that the performance of the inference algorithm improves for Sim 3. Therefore, choosing a more conservative success rate for the classifier rather than the one given by the confusion matrix associated with the classifier may be one way to overcome the issue related to consecutive errors in the same state.

VII. Experiments

We implement the proposed inference algorithm for a classifier-in-the-loop system in an experiment involving the

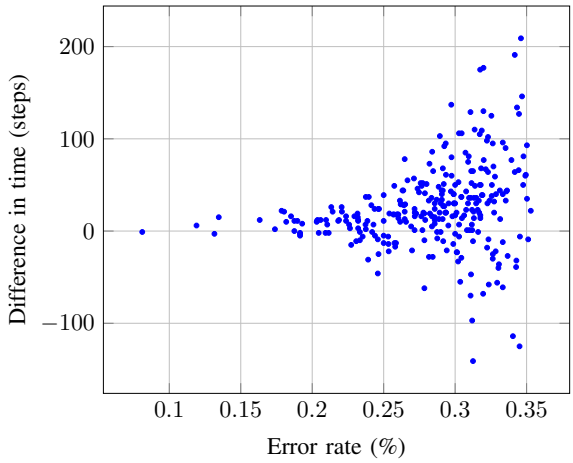


Fig. 10: Scatter plot of $t_f(\mathbf{y}) - t_f(\hat{\mathbf{x}})$ versus the error rate, for trials in Sim 2.

control of a virtual device using a brain-computer interface. The null hypothesis for the experiment is, again, that using our proposed inference algorithm does not affect the number of time steps taken to identify the goal.

A. Setup

We present the human subject with a one-dimensional grid containing 10 states displayed on a computer screen (see Figure 11). We mark the current state of the system is by a green cursor. The cursor can move horizontally in the grid. The task for the subject is to command the cursor to move towards the goal (taken as s_4) and stay there upon reaching it. We also present three squares which flash (between black and white) at 12Hz, 15Hz, and 20Hz respectively. Each square flashing at a unique frequency corresponds to a unique action. The user selects an action by looking at the flashing square corresponding to an assigned action. We extract EEG signals from the human using the OpenBCI R&D Kit [13]. The headset has 16 channels, and uses a 32 bit microprocessor to process the measured potentials.

We perform the simulation and processing for the experiment on a Lenovo Thinkpad with an Intel Core i7 2.1GHz CPU and 8GB of RAM running Ubuntu 14.04 as the operating system. We generate the flashing squares using the software program Psychopy [14]. We simulate the virtual device using the Robot Operating System (Indigo version).

B. Classification

The user commands an action by looking at the flashing square associated with that action. These flashing squares result in EEG signals known as steady-state visually evoked potentials (SSVEP). We use the $O1$ and $O2$ electrodes to record the EEG signals at a sampling rate of 250Hz. We construct each feature vector (observation) by calculating the average spectral power magnitudes (SPMs) across three disjoint bandwidths centered at 12Hz, 15Hz, and 20Hz with a width of $\pm 1.5Hz$ for 100 consecutive samples resulting in three averages per channel. This procedure results in a feature vector with 6 elements.

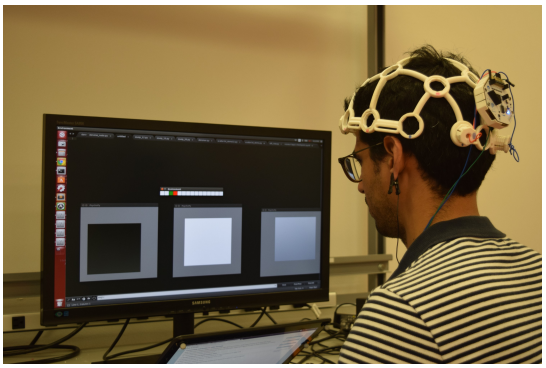


Fig. 11: Setup for experiments involving the control of a virtual device using a brain-computer interface with a classifier in the loop.

		Predicted label		
		a_0	a_1	a_2
Actual label	a_0	0.85	0.07	0.07
	a_1	0.10	0.80	0.10
	a_2	0.09	0.04	0.87

TABLE II: Confusion matrix for Q.

The resulting dataset contains 1,191 labeled observations after processing 119,100 samples. We split the dataset into a training set and a testing set, containing 893 and 298 observations respectively. We train a logistic regression model to obtain our classifier. Table II shows the confusion matrix obtained after testing of the classifier.

C. Procedure

A trial consists of simulating a sequence of actions taken by the cursor at discrete instants of time. The interval between two actions is two seconds. The feature corresponds to data collected for one second just before the cursor is moved. As mentioned, the cursor takes an action based on the output of the classifier. We assume that the user looks at the flashing square corresponding to the optimal action in the current state when commanding an action.

We compute the value of $P_t(g)$ is at every time step after an action is performed, using all state-label pairs obtained in the trial so far. We use both the measured class labels $\mathbf{y}$ and the estimated class labels $\hat{\mathbf{x}}$ obtained by using the proposed inference algorithm to compute two separate estimates of $P_t(g)$. We end the trial either when both values reach 0.95, or 60 time steps have been completed. We discard trials in which $P_t(g)$ does not cross 0.95 when using either $\mathbf{y}$ or $\hat{\mathbf{x}}$. We refer to this experiment as Exp 1 in Table I.

We choose parameter β to be 1 based on the simulations. Based on the discussion at the end of Section VI, the energy function in (8) is not defined by using the confusion matrix in Table II. Instead, we set $p(y_i|x_i)$ as 0.55 when $y_i = x_i$, and 0.225 otherwise. We select these values through trial and error using simulations.

D. Results

We present the experimental data collected over 43 trials. The t-statistic for Exp 1 (5.67) is higher than that required to reject the null hypothesis with 99.95% confidence (3.55).

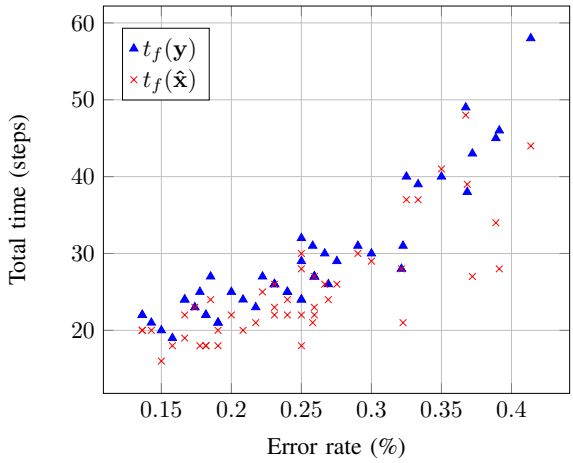


Fig. 12: Scatter plot of time $t_f(\mathbf{y})$ taken to identify the goal using the measured labels and time $t_f(\hat{\mathbf{x}})$ taken to identify the goal using the labels estimated by our inference algorithm, vs the error rate associated with $t_f(\mathbf{y})$, for trials in Exp 1.

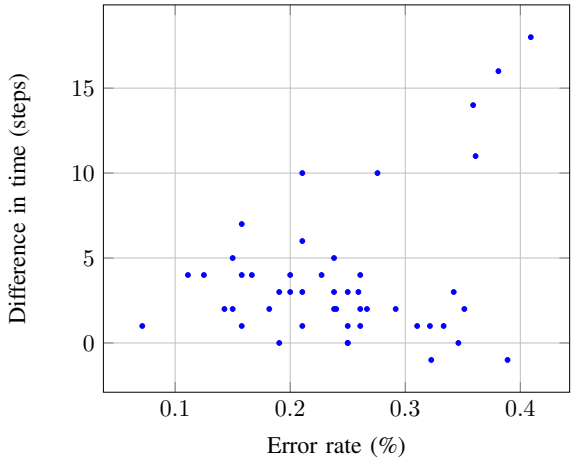


Fig. 13: Scatter plot of $t_f(\mathbf{y}) - t_f(\hat{\mathbf{x}})$ versus the error rate, for trials in Exp 1.

Therefore, even in experiment, the proposed inference algorithm is effective in reducing the time taken to identify the goal. As seen in Figures 12 and 13, the reduction in time taken increases as the error rate increases.

VIII. Conclusion

We have outlined a learning and control task in which the information available for control is in the form of class labels which are the output of a classifier. The class labels are used to determine the unknown goal state selected by a human user. The classification process can make errors, and therefore we proposed a method to detect and correct classification errors, leading to improved performance in the goal-learning task.

The main insight we use is that the correct class labels for each state-action pair are related in a predictable way. In particular, the correct labels in adjacent states are more likely to be the same than to be different, no matter which state the goal is. This property can be used to identify errors in classification at individual time steps. Moreover, the process of inferring the correct labels can be done efficiently by exploiting existing algorithms for optimization of submodular functions. The computational efficiency of the inference algorithm makes its use in experiments feasible. It may be possible to apply the proposed method to other situations where the correct class labels in states possess a known spatial relationship.

The simulations and experiments presented show that the inference algorithm reduces the time taken to identify the true goal on average. However, there are sequences of state-label pairs which may result in no real improvement in the time taken due to a failure to identify all misclassified labels. Often, this failure is linked to a concentration of errors in the same state. We observed that using a higher misclassification rate than given by the confusion matrix as a parameter in the proposed inference algorithm can reduce the instances in which the proposed algorithm does not perform well. One issue that has not been addressed is the selection of β. The MAP estimate is sensitive to the choice of the value of β. Future work will consist of estimating the true labels using multiple values of β, until the labels match an expected pattern associated with perfect classification.

References

[1] E. Alpaydin, *Introduction to Machine Learning*, 2nd ed. The MIT Press, 2010.

[2] G. Dornhege, J. del R. Millán, T. Hinterberger, D. J. McFarland, and K.-R. Müller, *Towards Brain Computer Interfacing*. MIT Press, 2007.

[3] J. Grizou, I. Iturrate, L. Montesano, P.-Y. Oudeyer, and M. Lopes, "Calibration-Free BCI Based Control," in *AAAI Conference on Artificial Intelligence*, Quebec, Canada, July 2014, pp. 1–8.

[4] J. d. R. Millan, "Brain-controlled devices: the perception-action closed loop," in *International Winter Conference on Brain-Computer Interface (BCI)*, Feb 2016, pp. 1–2.

[5] I. Iturrate, L. Montesano, and J. Minguez, "Shared-control brain-computer interface for a two dimensional reaching task using EEG error-related potentials," in *International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS)*, July 2013, pp. 5258–5262.

[6] L. J. Trejo, R. Rosipal, and B. Matthews, "Brain-computer interfaces for 1-D and 2-D cursor control: designs using volitional control of the EEG spectrum or steady-state visual evoked potentials," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 14, no. 2, pp. 225–229, June 2006.

[7] C. Baier and J.-P. Katoen, *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.

[8] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009.

[9] Y. Boykov, O. Veksler, and R. Zabih, "Fast Approximate Energy Minimization via Graph Cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 11, pp. 1222–1239, November 2001.

[10] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1124–1137, Sept 2004.

[11] V. Kolmogorov and R. Zabin, "What energy functions can be minimized via graph cuts?" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 2, pp. 147–159, Feb 2004.

[12] R. Kinderman and S. Snell, *Markov random fields and their applications*. American mathematical society, 1980.

[13] OpenBCI, "http://openbci.com/."

[14] J. Peirce, "Generating stimuli for neuroscience using PsychoPy," *Frontiers in Neuroinformatics*, vol. 2, p. 10, 2009.