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 choice of one element out of a known finite set. However, the classifier may assign an element out of the finite set that is different from the intended one. This error in classification can lead 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 that using the proposed algorithm reduces the time taken to identify the goal state in real-time 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, the output is obtained using a classification process [1], and is a member of a discrete set 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.

An example where classifier outputs are used in a control task arises in the shared control of semi-autonomous devices using brain-computer interfaces (BCIs) [2]. 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 can either generate brain signals corresponding to control actions, or perform 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. The increase in time taken to reach the goal can be due to two reasons. The first reason is that the BCI system may move *away* from the goal when it misclassifies the intended action. The second reason is that most estimation algorithms assume that the user is behaving ideally or optimally, and the misclassification becomes misleading evidence for such algorithms.

Contributions

In this paper, we propose 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 real-time 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.

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

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

B. Classifiers

A classifier C consists of a tuple (F, L, C^*) where

- F is a set of features,
- L is a finite set of class labels, and
- C^* is a map that assigns a label to a feature $f \in F$.

The classifier C can be associated with a confusion matrix R that models the imperfection of the classification process. The matrix R is obtained during testing of the classifier. If the classifier is perfect, then R assigns probability 1 to the true class label corresponding to a feature $f \in F$.

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. We will identify the random variable x_i for a node as the node itself, in order to simplify the exposition.

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 are not its neighbors:

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

The random variables x_i for all the nodes together for the random variable $\mathbf{x}$. The joint probabilitiy distribution of the nodes is represented by $p(\mathbf{x})$. In a discrete MRF, the random variables x_i can take on discrete values from a set. A sample of the MRF generated by $p(\mathbf{x})$ consists of an assignment of one of the discrete values to each node in $\mathcal{V}$. Therefore, we will refer to such a sample as an assignment of the MRF, or simply, an assignment.

III. Goal Learning through Brain-Computer Interfaces

The shared control of devices using brain-computer interfaces can be modeled as a classifier-in-the-loop learning and control problem. One of the common tasks to be achieved by such devices is for the state of the system to reach a desired goal state $g \in S$ determined by the human user [3]–[6], where S is the set of states that can be reached by the device.

We can model the dynamics of the device as a FTS. In this section, we will focus on a finite state space arranged in a one dimensional grid as seen below. It turns out that the case of multi-dimensional state spaces can be treated using the method for 1D spaces, which is shown in Section V. The action space A of the device is $\{a_0, a_1, a_2\}$ corresponding to moving left, staying in the same state, and moving right.

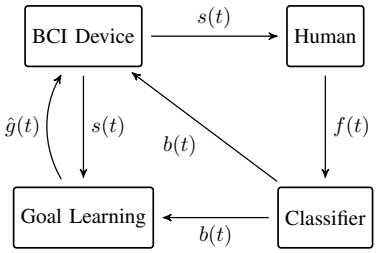
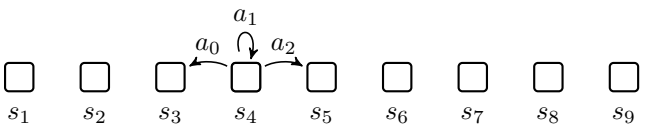


Fig. 1: Human and BCI system together as a classifier-in-the-loop system. Once the goal learning is complete, the BCI system can take over control from the human.



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 $T(s_{|S|}, a_2) = s_{|S|}$.

The state of the system $s(t)$ at time t is represented by a cursor. At each time step t, an action $a(t) \in \{a_0, a_1, a_2\}$ is chosen by the human. The intended action $a(t)$ to be taken in state $s(t)$ is obtained as an EEG-based feature $f(t)$ (see Figure 1). The measured EEG signal $f(t)$ is classified as $b(t)$, using a classifier C^* obtained by prior training. However, there is a non-zero probability that $a(t) \neq b(t)$. The probabilities of making mistakes is captured as the confusion matrix R for classifier C, and is obtained during testing. Therefore, the device is controlled via a classifier $C = (F, L, C^*)$, where F is the set of EEG features that can be obtained, $L = A$, and C has confusion matrix R associated with it. We refer to the pair $(s(t), b(t))$ as a state-label at time t. The elements $s(t)$ and $b(t)$ are referred to as the state and label at time t respectively.

The estimate $\hat{g}(t) \in S$ of the goal g at any time t is represented 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 $\delta \in [0, 1]$, it is taken to be the goal state. The values of $P_t(s_i)$ for all $s_i \in S$ are updated after every action is received. The update takes the form

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

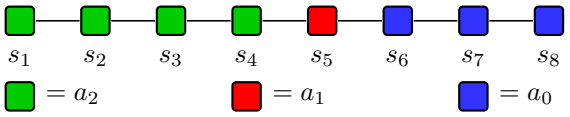


Fig. 2: Optimal actions to be taken in each state in order to reach the goal s_5 and stay there.

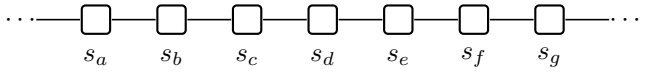


Fig. 3: The grid graph structure of the MRF with each node corresponding to a state-label. The nodes are ordered based on the state of each state-label pair, meaning that the integer indices of the states are such that $a \leq b \leq c \leq d \leq \ldots \leq g$. Equality of the indices occurs when a state is visited multiple times.

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 only be 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 $k_H > k_L > 0$.

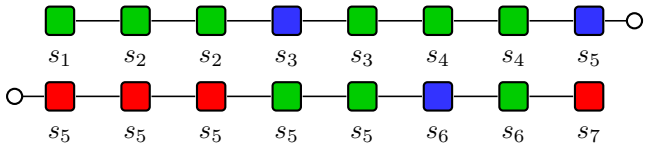
Let the current state be s_i. If $b(t) = a_1$ then $k_t(s_i) = k_H$, and $k_t(s_j) = k_L$ for all other states $s_j \neq s_i$. If $b(t) = a_2$, then $k_t(s_j) = k_H$ for all $s_j \in S$ where $j > i$, and $k_t(s_j) = k_L$ for all $j \leq i$. If $b(t) = a_0$, then $k_t(s_j) = k_H$ for all $s_j \in S$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

If actions are perfectly classified, then $P_t(g)$ will monotonically increase and become larger than the threshold δ, and 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 classification error-free case. To prevent this increase in time taken, we want to identify which actions have been misclassified. This leads to the following problem.

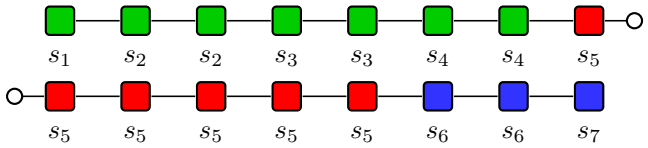
Inference problem: Given a sequence of state-label pairs $(s(t), b(t))$, $t \in \{1, 2, \ldots, N\}$, the task for the device being controlled is to estimate the unknown goal state g selected by the agent.

IV. Inference Using Markov Random Fields

Assume that a class label $x_i \in L$ is intended to be communicated through a classifier, by generating an input feature f_i. The classifier assigns the class label y_i to the feature f_i generated in order to communicate x_i, and it may occur that $y_i \neq x_i$. The class label assigned to a feature f_i can be treated as a noisy measurement of the true class label. The measurements y_i obtained at different time steps can be combined to obtain the set of measurements $\mathbf{y}$. Since the true class labels $\mathbf{x}$ cannot be observed directly, they are treated as random variables. Our goal is to estimate the most likely class labels $\hat{\mathbf{x}}$ given the measured class labels $\mathbf{y}$. The posterior distribution $p(\mathbf{x}|\mathbf{y})$ can be used to compute $\hat{\mathbf{x}}$ using bayesian inference. The likelihood function $p(\mathbf{y}|\mathbf{x})$ is determined by the performance of the classifier, in other words, the error rates in classification. The prior distribution



(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 ideal assignment of the MRF in Figure 4a when the optimal action is assigned in each state. In each state, we assume that ideal classification corresponds to providing the optimal action which is commanded by the user.

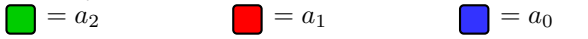


Fig. 4: Examples of MRFs obtained for a sequence of state-label pairs. The hollow circles serve to indicate that the two state-label pairs from different rows are actually connected by a single edge.

$p(\mathbf{x})$ needs to be determined. We now describe the procedure to construct 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. However, notice that given the goal state, the optimal action to be taken in a state s located in a 1D grid only depends on whether the goal state is to the right or the left of state s (see Figure 2). Thus, if the classifier was perfect, we are more likely to observe the same actions commanded in adjacent states. Therefore, we want to choose a prior distribution that encodes this spatial relationship that we expect.

We achieve this encoding by using Markov random fields to model the prior distribution $p(\mathbf{x})$. 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].

A. Connectivity Of The Markov Random Field

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, $|V| = N$. We connect the nodes to form a chain, as seen in Figure 3. Let the neighbors of s_i 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. An example of such an MRF is provided in Figure 4.

Note that the MRF does not require that there be exactly one node for each possible state-label pair. Missing or multiple state-label pairs are allowed.

B. Potential Energy Functions For The Markov Random Field

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 MRF we constructed in the previous subsection ensures that the cliques of the MRF only consist of 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 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].

C. Likelihood Function

The likelihood function $p(\mathbf{y}|\mathbf{x})$ is represented by the energy function D given by

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

where $D(y_i, x_i) \geq 0$ when $p(y_i|x_i) > 0$.

D. Obtaining An Estimate

The MAP estimate is obtained from the posterior distribution as

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} p(\mathbf{x}|\mathbf{y}). \tag{9}$$

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):

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \sum_{i=1}^{n} D(y_i, x_i) + \sum_{c\in\mathcal{C}} V_c(\mathbf{x}_c). \tag{10}$$

If L contains only two labels, then the global minimum of the right hand side of (10) 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 minimum-cut/maximum-flow solution determined by this graph [10], [11].

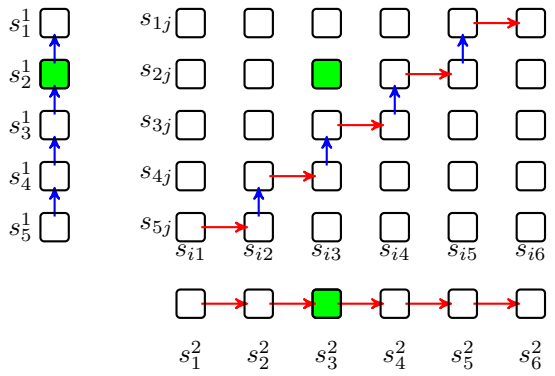


Fig. 5: Example of a decomposition of transitions in a 2D grid into transitions in two 1D grids. The first dimension is along the vertical direction, the second dimension is along the horizontal. A state is denoted by s_{ij}, where i and j are indices that denote the location of s_{ij} along the first and second dimensions. The state is then represented by s_i^1 in the 1D grid associated with the first dimension, and by s_j^2 in the 1D grid associated with the second dimension. The transitions in the 2D grid marked by blue arrows become transitions in the first 1D grid, while those marked with red arrows become transitions in the second 1D grid. The goal s_{33} in the 2D grid becomes the goals s_3^1 and s_3^2 in the two 1D grids respectively.

In this paper, we have three actions and therefore three labels. In general, solving the energy-minimization problem with more than two labels exactly is computationally expensive. However, an approximate minimum can be computed using an iterative scheme [9], where the iterations are continued until convergence of the total energy to a constant value. Each iteration is based on the energy-minimization methods in [10], [11]. In any iteration, we treat two subsets of L as the two labels over which to perform inference. When we consider two separate labels $\alpha, \gamma \in L$ as the two subsets, the iteration is called an $\alpha - \gamma$ swap [9]. When we consider a label $\alpha \in L$ as one subset and all other labels as the other subset, the iteration is called an α-expansion [9]. Both types of iterations involve solving a minimum-cut/maximum-flow graph problem with weights determined by the energy functions of the MRF.By testing the algorithm using simulations, we determine that performing inference based on α-expansions is not effective. In the rest of the paper, the inference/energy-minimization is performed using iterations of $\alpha - \gamma$ swaps over all possible pairs of labels in L until convergence.

Note that the solution for $\hat{\mathbf{x}}$ depends on the value of β in (6). The possible solutions for $\hat{\mathbf{x}}$ represent a trade-off between obtaining MAP estimates where $\hat{\mathbf{x}}$ is similar to $\mathbf{y}$ and obtaining MAP estimates where the adjacent evaluations are assigned similar labels. A high value of β results in all nodes being assigned the same label.

V. Multi-dimensional State Spaces

The method for goal-learning in 1D state spaces can be used to solve a goal-learning problem in state spaces

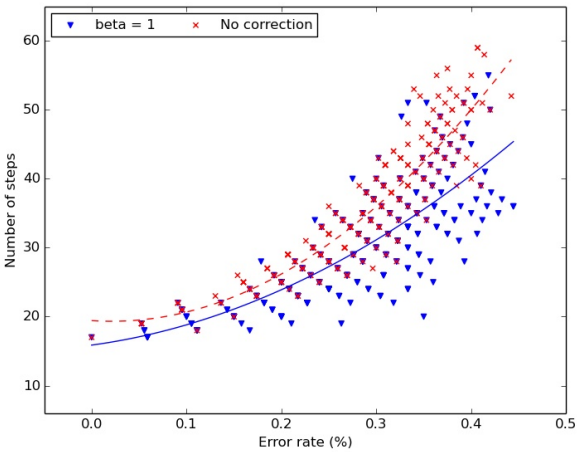


Fig. 6: Scatter plot of the number of time steps taken for $P_t(g)$ to first cross 0.95 vs the error rate, for trials in Sim 1. The red checkmark corresponds to the time taken when measured labels $\mathbf{y}$ are used to compute $P_t(s_i)$ for all $s_i \in S$, and the blue triangles correspond to the time taken when the estimate $\hat{\mathbf{x}}$ of the true class labels– obtained using the proposed inference algorithm – is used. The dashed line and solid lines are parabolic fits to the points corresponding to the results for using $\mathbf{y}$ (no correction) vs the results for using $\hat{\mathbf{x}}$.

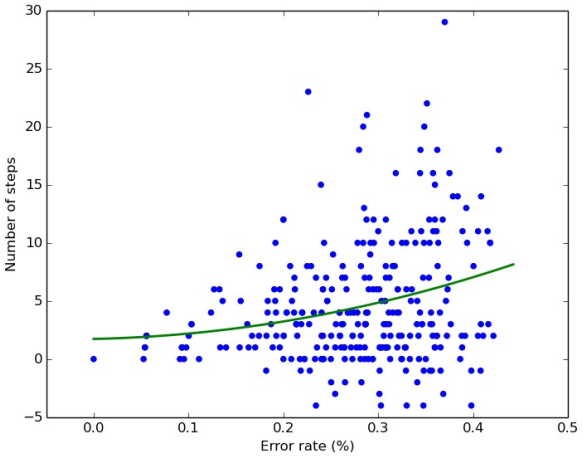


Fig. 7: Scatter plot of the difference in number of time steps taken for $P_t(g)$ to first cross 0.95 vs the error rate, for trials in Sim 1. The difference in steps is between the case when the measured class labels $\mathbf{y}$ is used to compute $P_t(s_i)$ for all $s_i \in S$, and the case when estimate $\hat{\mathbf{x}}$ is used. The green line represents a parabolic fit to the data points.

modeled by regular n-dimensional grids. The idea is that each dimension of the n-dimensional grid induces a goal-learning problem in a 1D grid. Goal-learning in an n-dimensional grid is therefore reduced to goal-learning in n 1D grids. An example of this is depicted in Figure 5.

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 5, for example). For the k^{th} dimension, we can create a 1D grid where the states are denoted as s_i^k. The subscript i indicates that this state is the i^{th} state in the 1D 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. This means that this transition in the n-dimensional grid induces a transition in the 1D grid associated with the k^{th} dimension, and no transition in any of the other 1D grids. In this way, we can create n 1D goal-learning problems. The solution to the n goal-learning problems can be combined to solve the goal-learning problem in n-dimensions.

The number of graph-cut problems that must be solved for an n-dimensional state space is n. The graph-cut algorithm for computing the MAP estimate is polynomial in the number of nodes in the problem. Therefore, decomposing the n-dimensional state space problem as described does not lead to a prohibitive increase in computation. In fact, it decreases the computations required since the number of state-label pairs are split amongst the n problems.

VI. Simulations

We simulate the goal-learning scenario for $1D$ and $2D$ grids assuming that the optimal actions are chosen in each state, however the implemented action may not be the same as the optimal action. When the implemented action in a state is not the same as the optimal one in that state given the goal state, an error is said to have occurred at that time step. We record the time taken to identify the correct state as the goal, if done so within 60 time steps. We also note down the effective error rate. The probability that an error occurs for the $1D$ case is chosen to be 0.3, and for the $2D$ case the probability is chosen to be 0.4. When an error occurs, the non-optimal actions in a state are equally likely to be chosen.

The value of β for the simulations is taken to be 1. For the simulations, the energy function (8) is computed using the simulated error probabilities mentioned in the prior paragraph. That is, the confusion matrix for classification determines the values of the energy D for a label.

We refer to these $1D$ and $2D$ simulations as Sim 1 and Sim 2 respectively. The results for Sim 1 are shown in Figures 6 and 7. Each point denotes a single trial. The ordinate is the number of time steps taken for $P_t(g)$ to reach 0.95 in a trial. The abscissa is the fraction of the total number of time steps taken for which the action was erroneous in a state. In Figure 7, the ordinate is the difference in the number of time steps taken for $P_t(g)$ to reach 0.95 in a trial, when comparing the case where $P_t(g)$ is computed using the measured class labels $\mathbf{y}$, and the case where $P_t(g)$ is computed using the MAP estimate $\hat{\mathbf{x}}$ using the proposed inference algorithm. Similarly, the results for Sim 2 are shown in Figures 8 and 9.

The null hypothesis for our simulation experiment is that using our proposed inference algorithm to obtain $\hat{\mathbf{x}}$ and then

	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.

compute $P_t(s_i)$ for all $s_i \in S$ does not affect the number of time steps taken for $P_t(g)$ to become greater than $\delta = 0.95$, when compared to using the measured labels $\mathbf{y}$ to compute $P_t(s)$. In both Figure 6 and 8 we see that on average, the number of time steps taken taken to identify the goal with confidence 0.95 is lower for the case when the proposed inference method is used to identify misclassifications. This conclusion is validated by the t-statistic values (penultimate row of Table II) for Sim 1 and Sim 2 being greater than the required paired t-test values (bottom row of Table II) for rejecting the null hypothesis with 99.95% confidence. Furthermore, the reduction in time steps increases with the error rate, as seen in Figures 6-9. 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, the inferred labels are such that the time taken for $P_t(g)$ to cross δ is greater than when using just the measured class labels. When checking the state-label pairs corresponding to such trials, we observe that when too many errors occur in the same state in consecutive time steps, the algorithm is less likely to correct these errors. Furthermore, for Sim 2, the t-statistic is not as high as for Sim 1. We simulated a case where the parameters for the energy function (8) are chosen to be different from the confusion matrix implied by the simulated probabilities of choosing actions in a state. In particular, 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. This simulation is referred to as Sim 3. From Table II, we see that the performance of the inference algorithm improves for this simulation. 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 control of a virtual device using a brain-computer interface. This section describes the details of the experiment and presents the results. The null hypothesis for the experiment is, again, that using our proposed inference algorithm does not affect the number of time steps taken for $P_t(g)$ to become greater than $\delta = 0.95$.

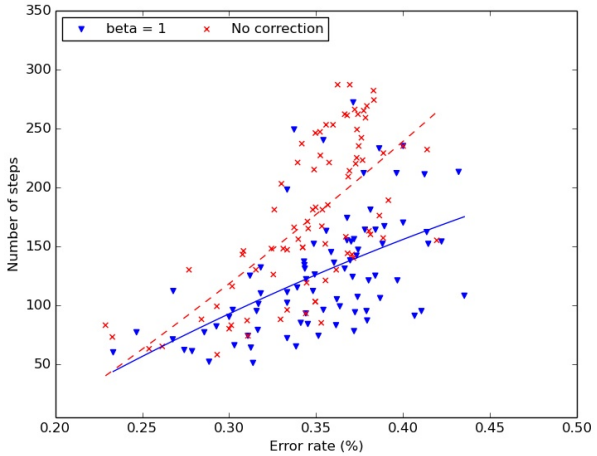


Fig. 8: Scatter plot of the number of time steps taken for $P_t(g)$ to first cross 0.95 vs the error rate, for trials in Sim 2. The red checkmark corresponds to the time taken when $\mathbf{y}$ is used to compute $P_t(s_i)$ for all $s_i \in S$, and the blue triangles correspond to the time taken $\hat{\mathbf{x}}$ is used. The dashed line and solid lines are parabolic fits to the two sets of data points.

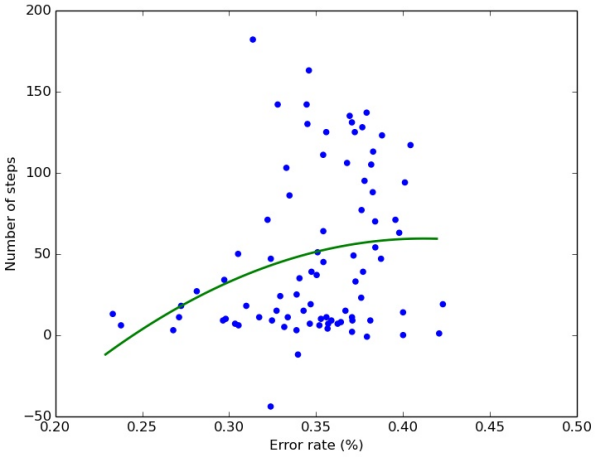


Fig. 9: Scatter plot of the difference in number of time steps taken for $P_t(g)$ to first cross 0.95 vs the error rate, for trials in Sim 2. The difference in steps is between the case when $\mathbf{y}$ is used to compute $P_t(s_i)$ for all $s_i \in S$ and the case when estimate $\hat{\mathbf{x}}$ is used. The green line represents a parabolic fit to the data points.

A. Setup

A $1D$ grid with 10 states is presented on a computer screen to the subject (see Figure 10). The current state of the system is marked by a green cursor. The goal state (taken as the state s_4 of the grid) is marked red. The cursor can move horizontally in the grid. The task for the subject is to command the cursor to move towards the goal 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

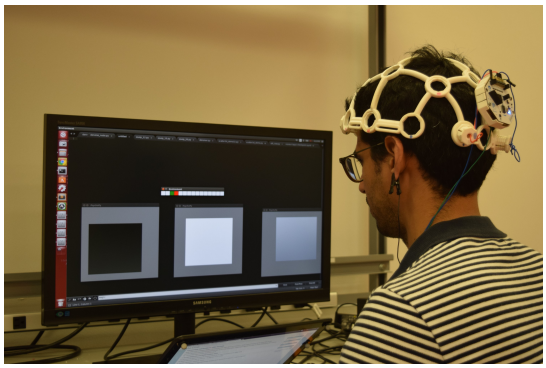


Fig. 10: Setup for experiments involving the control of a virtual device using a brain-computer interface with a classifier in the loop.

		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 C^*.

looking at the flashing square corresponding to an assigned action. The EEG signals associated with this selection are extracted using the OpenBCI R&D Kit [13]. The headset has 16 channels, and uses a 32 bit microprocessor to process the measured potentials.

The simulation and processing for the experiment is performed on a Lenovo Thinkpad with an Intel Core i7 2.1GHz CPU and 8GB of RAM running Ubuntu 14.04 as the operating system. The flashing squares are generated using the software program Psychopy [14]. The real-time processing of the labels and simulation of the virtual device is done using Robot Operating System (Indigo version).

B. Classification

As mentioned in the previous section, the EEG signals corresponding to an action are generated when the user looks at the associated flashing square. Such EEG signals are known as steady-state visually evoked potentials (SSVEP). We use the $O1$ and $O2$ electrodes to obtain the EEG signals at a sampling rate of 250Hz. Each feature vector (observation) is constructed 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 results in a feature vector with 6 elements.

The resulting dataset contained 1,191 labeled observations after processing 119,100 EEG labeled samples. The dataset was split into a training set and a testing set, with 893 and 298 observations respectively. We trained a logistic regression model to obtain our classifier. The confusion matrix obtained after testing the classifier is shown in Table II.

In the experiments, we observe a higher occurrence of errors than predicted by the confusion matrix in Table II. The training data was obtained by showing only one flashing square at a time, and without the requirement to track a

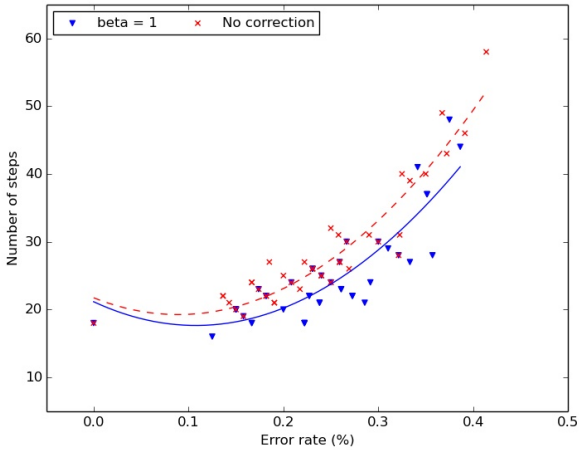


Fig. 11: Scatter plot of the number of time steps taken for $P_t(g)$ to first cross 0.95 vs the error rate, for trials in Exp 1. The red checkmark corresponds to the time taken when $\mathbf{y}$ is used to compute $P_t(s_i)$ for all $s_i \in S$, and the blue triangles correspond to the time taken $\hat{\mathbf{x}}$ is used. The dashed line and solid lines are parabolic fits to the two sets of data points.

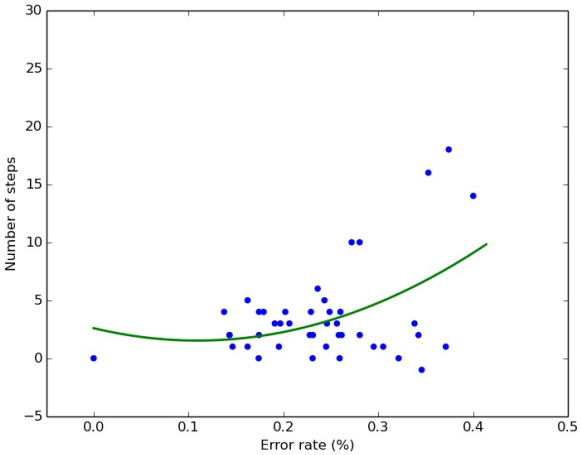


Fig. 12: Scatter plot of the difference in number of time steps taken for $P_t(g)$ to first cross 0.95 vs the error rate, for trials in Exp 1. The difference in steps is between the case when $\mathbf{y}$ is used to compute $P_t(s_i)$ for all $s_i \in S$ and the case when estimate $\hat{\mathbf{x}}$ is used. The green line represents a parabolic fit to the data points.

cursor moving in the presented $1D$ grid. It is possible that the additional flashing squares and the task of observing the current state of the cursor increases the misclassification rate.

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. It is assumed that the user always looks at the flashing square corresponding to the optimal

action in the current state of the cursor.

The value of $P_t(g)$ is computed in real-time at every time step when an action is performed, using all state-label pairs obtained during the trial up till the current time step. The value of $P_t(g)$ is computed using both the raw measured class labels $\mathbf{y}$ and the estimated class labels $\hat{\mathbf{x}}$ obtained by using the proposed inference algorithm. The trial is stopped either when both values reach 0.95, or 60 time steps have been completed. The trial is discarded if $P_t(g)$ does not cross 0.95 when using either $\mathbf{y}$ or $\hat{\mathbf{x}}$. We refer to this experiment as Exp 1 in Table II.

Based on the simulations, the parameter β is taken to be 1. The energy function in (8) is not defined by the confusion matrix in Table II, based on the discussion at the end of Section VI. Instead, we set $p(y_i|x_i)$ as 0.55 when $y_i = x_i$, and 0.225 otherwise. These values were based on 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). Therefore, even in experiment, the proposed inference algorithm is effective in reducing the time taken to identify the goal. As seen in Figures 11 and 12, the reduction in time taken increases as the error rate increases.

When conducting the experiments, it was observed that the correction of labels was sensitive to the states and actions in which an error occurred, just as was observed in the simulation. As discussed earlier, the successful identification and correction of misclassified labels depends on the parameters that define the energy functions of the MRF. When the proposed inference algorithm is applied to the experimental data with varying values of β, the misclassified labels are corrected with high reliability for certain values of β. However, at present we have not formulated an algorithmic method to choose the value of β.

VIII. Conclusion

In this paper 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 who commands the actions of the device. The action is recorded using EEG signals, leading to the presence of a classifier in the loop. 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 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. These relationships can be used to overcome the errors in assigning individual class labels. 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 real-time experiments feasible. It may be possible to apply the proposed method to other situations in which the correct class labels in states are related based on the spatial relationship of those states.

The simulations and experiments provided show that the proposed 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. It may be worthwhile to solve the graph-cut problem over a discrete set of values of β, until the labels match a pattern that is expected.

References

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

[2] G. Dornhege, J. del R. Milln, T. Hinterberger, D. J. McFarland, and K.-R. Mller, *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 *Twenty-Eighth AAAI Conference on Artificial Intelligence*, Quebec, Canada, July 2014, pp. 1–8. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00984068

[4] J. d. R. Millan, "Brain-controlled devices: the perception-action closed loop," in *2016 4th 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 *Engineering in Medicine and Biology Society (EMBC), 2013 35th Annual International Conference of the IEEE*, 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 Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 11, pp. 1222–1239, November 2001. [Online]. Available: http://dx.doi.org/10.1109/34.969114

[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.