Learning optimal switching feedback controllers from data

Laura Ferrarotti * **Alberto Bemporad** *

* *IMT School for Advanced Studies Lucca.*
{*laura.ferrarotti, alberto.bemporad*}*@imtlucca.it*

Abstract In this paper we present a data-driven approach for synthesizing optimal switching controllers directly from experimental data, without the need of a global model of the dynamics of the process. The set of controllers and the switching law are learned by using a coordinate descent strategy: for a fixed switching law, the controllers are sequentially optimized by using stochastic gradient descent iterations, while for fixed controllers the switching law is iteratively refined by unsupervised learning. We report examples showing that the approach performs well when applied to control processes characterized by hybrid or nonlinear dynamics, outperforming control laws that are single-mode (no switching) or multi-mode but with the switching law defined a priori.

Keywords: Consensus and Reinforcement learning control, Optimal control of hybrid systems, Machine Learning

1. INTRODUCTION

Data-driven synthesis of control systems has recently being investigated as an alternative to model-based control design (Formentin et al., 2014). Data-driven techniques have lately been employed to tackle nonlinear control problems (Novara and Formentin, 2018) or constrained control problems (Piga et al., 2018). In the case of switched linear multiple-input multiple-output (MIMO) systems, (Dai and Sznaier, 2018) synthesize a robust switching controller from experimental data, without explicitly identifying a model of the open-loop process. Nonetheless, the method requires the knowledge of the model structure, and therefore of the switching law.

In the robotics literature, contributions can be found related to the synthesis of switching controllers exploiting experience gathered from process/environment interactions, in the way that is typical of Reinforcement Learning (RL), see (Recht, 2018). In particular, *policy search* methods are employed, that rely upon optimizing parameterized policies and can deal with continuous action and state spaces (Desienroth et al., 2011). These methods are mainly related to motion tasks, for instance (Grudic et al., 2003) starts from an existing controller and uses policy gradient techniques to synthesize a switching controller online with improved performance, while (Nagayoshi et al., 2010) uses two control laws, one based on Q-learning and the other on actor-critic, to mimic gross and fine motor skills respectively. However, in both (Grudic et al., 2003) and (Nagayoshi et al., 2010) the switching law is fixed and known a priori.

In this paper we extend the policy gradient approach proposed in (Ferrarotti and Bemporad, 2019) to deal with hybrid control policies, learning both the set of controllers and the switching law directly from experimental data. The parameterized controllers are learned by Stochastic Gradient Descent (SGD) (Robbins and Monro, 1951), while the switching law is iteratively refined. As in (Ferrarotti and Bemporad, 2019), the proposed method needs to compute simple local linear models, that have a double purpose: first, they are used to approximate the gradients required by SGD; second, they are used to approximate the cost of using each of the controllers on a specific region of the space, so to optimize the assignment of areas of pertinence to the controllers. As a consequence, the method is not completely model-free, but it does not first identify a complete model of the open-loop system to then design a control strategy, so it is not model-based either. To the best of our knowledge, (Breschi and Formentin, 2020) is the only other data-driven method estimating a piecewise-affine controller together with the switching law. Although completely model-free, it requires as a critical part of the design phase the tuning of the reference model for the desired closed-loop behaviour.

The paper is organized as follows. The problem of optimal switching policy search is formulated in Section 2, followed by the proposed algorithm in Section 3. In Section 4 the method is tested on numerical examples and Section 5 is dedicated to conclusions.

Notation: Let $\mathbb{R}^n$ be the set of real vectors of dimension n. For each vector x belonging to $\mathbb{R}^n$, x_i indicates the i-th element of the vector. Given a matrix $A \in \mathbb{R}^{n\times m}$ we denote its transpose by A'. We denote by I the identity matrix and by e_i its i-th column. Given a matrix $Q \in \mathbb{R}^{n\times n}$, $\|x\|_Q^2 = x' Q x$. $\mathcal{S}_{M,L}$ is the set of all the sequences of M elements of length L, and $\mathcal{P}_M$ is the set of all the random permutations of M elements. If S is a discrete set, $|S|$ is its cardinality.

2. OPTIMAL SWITCHING POLICY SEARCH PROBLEM

We consider a Markovian signal $s_t \in \mathbb{R}^{n_s}$ representing the dynamics of a strictly causal plant P embedded in its environment, evolving in time according to the (unknown) model

$$s_{t+1} = h(s_t, p_t, u_t, d_t), \tag{1}$$

where $p_t \in \mathbb{R}^{n_p}$ is a vector of measured exogenous signals, $u_t \in \mathbb{R}^{n_u}$ a vector of decision variables (actions), and $d_t \in \mathbb{R}^{n_d}$ a vector of unmeasured disturbances.
The cost of controlling the plant at instant t is represented as a function $\rho : \mathbb{R}^{n_s+n_p+n_u} \to \mathbb{R}$ called *stage cost*.
Given the values $p_0, d_0, p_1, d_1, \ldots$ and the initial condition s_0, the cost of applying a sequence of control actions $u_0, u_1, \ldots$ over an infinite horizon is defined as

$$J_\infty(s_0, \{p_l, d_l, u_l\}_{l=0}^{\infty}) = \sum_{l=0}^{\infty} \rho(s_{l+1}, p_l, u_l), \tag{2}$$

that is the sum of the stage costs accumulated along the trajectory of (s_l, p_l, u_l, d_l) satisfying (1) for all l. Considering a *deterministic policy* $\pi : \mathbb{R}^{n_s+n_p} \to \mathbb{R}^{n_u}$ that associates an action $u_t = \pi(s_t, p_t)$ to each s_t and p_t, we define the overall cost of π as

$$J(\pi) = \mathbb{E}_{S_0, \{P_l, D_l\}_{l=0}^{\infty}}[J_\infty(\pi, S_0, \{P_l, D_l\})], \tag{3}$$

where the expectation of J_∞ is taken with respect to the random variables S_0 and P_l, D_l, $l = 0, 1, \ldots$, representing the initial point of the trajectory and the value of signals p_l, d_l at step l, respectively.
The *optimal policy* with respect to (3) is

$$\pi^* = \arg \min_{\pi \in \mathcal{F}(\mathbb{R}^{n_s+n_p}, \mathbb{R}^{n_u})} J(\pi), \tag{4}$$

where $\mathcal{F}(\mathbb{R}^{n_s+n_p}, \mathbb{R}^{n_u})$ is the set of functions of n_s+n_p real variables taking values in $\mathbb{R}^{n_u}$. Equation (4) represents a general abstract optimal policy search (OPS) problem. In order to find a computable solution to it, we make the following approximations:

- we parametrize the policy π by a set H of parameters, and denote by $\pi_H(s_t, p_t)$ the corresponding policy;
- we consider a finite trajectory of length L for evaluating the cost (2), shortening the sum of stage-costs.

A method presented to solve problem (4) after applying the above simplifications was proposed in (Ferrarotti and Bemporad, 2019) and it is extended here to deal with switching policy parameterizations. A switching (or hybrid) parametric policy can be represented as

$$\pi_K^c(s_t, r_t) = \begin{cases} \pi_{K_1}(s_t, p_t), & \text{if } (s_t, p_t) \in R_1(c), \\ \vdots & \vdots \\ \pi_{K_M}(s_t, p_t), & \text{if } (s_t, p_t) \in R_M(c), \end{cases} \tag{5}$$

where M is a fixed number of policies and $K = \{K_1, \ldots, K_M\}$, $c = \{c_1, \ldots, c_M\}$ the corresponding parameters. The regions $R_1(c), \ldots R_M(c) \in \mathcal{P}(\mathbb{R}^{n_s+n_p})$ are defined as the polyhedra obtained from the Voronoi partition of the points $c_1, \ldots, c_M$, using an opportune distance or semi-distance d over $\mathbb{R}^{n_s+n_p}$, i.e.,

$$\begin{aligned} R_j(c) = \{x \mid\ & (d(x, c_j) = d(x, c_h) \text{ and } j < h) \text{ or} \\ & d(x, c_j) < d(x, c_h), \quad \forall h \neq j,\ h = 1, .., M\} \end{aligned} \tag{6}$$

and K_j is the matrix of parameters that characterize the policy over the j-th region $R_j(c)$. Substituting (5) in the cost of a trajectory of length L, we obtain

$$J_L(K, c, s_0, \{p_l, d_l\}_{l=0}^{L-1}) = \sum_{l=0}^{L-1} \rho(s_{l+1}, p_l, \pi_K^c(s_l, p_l)), \tag{7}$$

from which we derive the optimal switching policy search (OSPS) problem

$$\min_{K,c} \ \mathbb{E}_{\substack{S_0, \\ \{P_l, D_l\}_{l=0}^{L-1}}} [J_L(K, c, S_0, \{P_l, D_l\}_{l=0}^{L-1})] \tag{8a}$$

with

$$S_{l+1} = h(S_l, P_l, D_l) \tag{8b}$$

and S_0, P_l, D_l defined as in (3).
The method proposed in this work addresses problem (8) by coordinate descent: starting from an initial guess K_0, c_0, for $t = 1, \ldots, N_{\text{learn}}$ we iterate

$$K^t = \arg\min_K \ \mathbb{E}[J_L(K, c^{t-1}, S_0, \{P_l, D_l\}_{l=0}^{L-1})] \tag{9a}$$

$$c^t \ = \arg\min_c \ \mathbb{E}[J_L(K^t, c, S_0, \{P_l, D_l\}_{l=0}^{L-1})], \tag{9b}$$

where, to simplify notation, we removed the random variables from the expectations.
The solution of problem (9a) at iteration t is the best set of control parameters $K^t = \{K_1^t, \ldots, K_M^t\}$ to be applied on the regions $R_1(c^{t-1}), \ldots, R_M(c^{t-1})$, generated by the centroids at instant $t-1$; we approach this problem by optimizing sequentially each gain K_j^t via the method proposed in (Ferrarotti and Bemporad, 2019).
Problem (9b), instead, aims at finding the optimal centroids $c^t = \{c_1^t, \ldots, c_M^t\}$ with respect to K^t, that means the best switching law to apply in combination with the most recently updated set of control parameters. To approximate the solution of problem (9b), we sample a mini-batch of initial states, exogenous signals, and disturbance sequences from the mentioned random variables and iteratively divide it into M clusters, based on which of the M gains in K^t is most convenient to apply over each sample. We calculate the barycenters of the assigned clusters and we use the barycenters over the mini-batch to recursively update the estimation of the barycenters over all the space. A detailed description of the algorithm proposed in this paper is provided in Section 3.
In the following we will consider for simplicity a linear subparameterization over each subdomine $R_j(c)$, that is

$$\pi_{K_j}(s_t, p_t) = -K_j' \begin{bmatrix} s_t \\ p_t \end{bmatrix} \qquad j = 1, \ldots, M. \tag{10}$$

The presented methodology can be applied independently of the choice of π_{K_j}, that could also be nonlinear. Moreover, the proposed method is completely general with respect to the task that the plant P has to perform and to the associated stage cost ρ. In this paper, we will focus on an output-tracking task, i.e., we want to learn a switching policy that makes the output y_t of the plant P track a reference signal r_t, using input/output data only. To do so, we consider a state s_t composed by a finite set of past input and output values from P, $n_o \geq 1, n_i \geq 2$, i.e.,

$$x_t = [y_t' \ \ldots \ y_{t-n_o+1}' \ u_{t-1}' \ \ldots \ u_{t-n_i+1}']', \tag{11}$$

an integral term $q_{t+1} = q_t + (y_{t+1} - r_t)$, and set

$$s_t = \begin{bmatrix} x_t \\ q_t \end{bmatrix}, \quad p_t = r_t. \tag{12}$$

The stage cost we consider is

$$\begin{aligned} \rho(s_{t+1}, r_t, u_t) \ &= \|Cs_{t+1} - r_t\|_{Q_y}^2 \ + \ \|u_t\|_{R_u}^2 + \\ &+ \ \|\Delta u_t\|_{R_{\Delta u}}^2 \ + \ \|q_{t+1}\|_{Q_q}^2 \end{aligned} \tag{13}$$

Matrix $C = [\,I\,0 \ldots 0\,]$ is such that $y_{t+1} = Cs_{t+1}$ and $\Delta u_t = u_t - u_{t-1}$. Here we assume $n_i \geq 3$, so u_{t-1} is contained in x_{t+1}, as defined in (11), and in s_{t+1}. In case $n_i = 2$, it is nonetheless possible to add u_{t-1} as additional state in s_t. Matrices Q_y, R_u, $R_{\Delta u}$, Q_q weight the tracking error, the control effort, the input increment and the integral action, respectively.

3. OSPS ALGORITHM

Algorithm 1 summarizes the proposed method for learning an optimal switching policy (5) from input/output data. The method can be applied either offline on a given dataset of input/output collected from the process excited in open loop, or online as new data are collected while the learned policy is already in place. In both settings, at every step t of the learning procedure the policy gains $K^t = \{K_1^t, \dots, K_M^t\}$ are updated by approximating the solution of (9a) via mini-batch SGD. The n-th iteration of mini-batch SGD requires computing $\nabla J_L(K^{n-1}, c^{t-1}, w_h)$ for each element w_h of the sampled mini-batch of data $\{w_h = (s_0^h, \{r_l^h, d_l^h\}_{l=0}^{L-1})\}_{h=1}^{N_b}$. Since dynamics (1) are unknown, as in (Ferrarotti and Bemporad, 2019) for each evaluation of $\nabla J_L(K^{n-1}, c^{t-1}, w_h)$ we replace (1) with a local linear model that approximates the behaviour of the system in a neighborhood of the initial point s_0^h, obtained as described in Section 3.1. The sampling procedure of the mini-batches is summarized in Section 3.2 together with the gradient approximation and the gain update method. Problem (9b) is solved after n_K updates of the control gain matrix K^t, as follows. First, we apply again the sampling procedure to obtain a mini-batch $\{w_k\}_{k=1}^{N_b}$ (see Section 3.2). Then, we divide the sampled elements w_k into M subsets. The assignment of w_k requires computing the following quantities, for $m = 1, \dots, M$,

$$\begin{aligned} F(m, w_k, K^t, c^{t-1}) &= \rho(\, s_1^k, r_0^k, \pi_{K_m^t}(s_0^k, r_0^k)\,) + \\ &+ \hat{J}_{L-1}(\, K^t, c^{t-1}, s_1^k, \{r_l^k, d_l^k\}_{l=1}^{L-1}\,) \end{aligned} \tag{14}$$

Each term $F(m, w_k, K^t, c^{t-1})$ is composed by the stage-cost associated with the application of the m-th policy $\pi_{K_m^t}$ while being in state (s_0^k, r_0^k), plus the approximated cost-to-go. The two terms are approximated using the local linear model fitted in the neighborhood of s_0^k, instead of the unknown system dynamics (1). The cost-to-go is obtained by simulating the remaining $L-1$ steps of the trajectory using the last updated switching law c^{t-1}. After calculating (14) for all $m = 1, \dots, M$, we assign w_k to the m^*-th subset, with m^* being the smallest index corresponding to the most "convenient" gain $K_{m^*}^t$ to be applied in w_k with respect to cost F, that is

$$m^* = \min(\arg\min_m F(m, w_k, K^t, c^{t-1})).$$

After the above unsupervised learning procedure is executed, for each of the M obtained subsets we calculate the barycenters $c^{(1)} = \{c_1^{(1)}, \dots, c_M^{(1)}\}$. The above overall process is iterated n_c times: at every iteration i the most updated version of the mini-batch barycenters $c^{(i-1)}$ is employed in (14) to evaluate the cost-to-go. In this way, we generate a sequence $c^{(0)} = c^{t-1}, c^{(1)}, \dots, c^{(n_c)}$ that refines the barycenters of the regions on the sampled mini-batch. Finally, for $m = 1, \dots, M$ we recursively update the global barycenters estimation as follows:

$$\begin{aligned} N_m^t &= N_m^{t-1} + \bar{N}_{n_c}^m, \\ c_m^t &= \frac{1}{N_m^t}(N_m^{t-1} c_m^{t-1} + \bar{N}_{n_c}^m c_m^{(n_c)}), \end{aligned}$$

where N_m^t and $\bar{N}_k^m$ are the cardinality of the set of states averaged to estimate c_m^t up to time t and in the current sampled mini-batch at iteration k of the described procedure, respectively.

3.1 *Local model*

In principle, evaluating the gradients to solve problem (9a) and the cost-to-go for problem (9b) requires the unknown dynamics (1). In order to avoid identifying first a full model of (1) from data, we substitute it with a local model

$$y_t = \Theta_t \cdot z_t + d_t \tag{15}$$

with $\Theta_t \in \mathbb{R}^{n_y \times n_x}$, that is fitted in a neigborhood of the initial state corresponding to the specific sampled value. We use Kalman filtering (KF) for updating Θ_t, that is modeled as a stochastic process $\Theta_{t+1} = \Theta_t + \xi_t$, where ξ_t is a Gaussian white noise with covariance Q_k and d_t in (15) is a a Gaussian white noise with covariance matrix R_k. The regressor vector is formed by past inputs and outputs collected from the process, namely $z_t = [\, y'_{t-1} \, \dots \, y'_{t-n_o} \, u'_{t-1} \, \dots \, u'_{t-n_i} \,]'$ with n_i and n_o defined in (12). Eventually, it will be enriched by an affine term, i.e., we will set $z_t = [\, y'_{t-1} \, \dots \, y'_{t-n_o} \, u'_{t-1} \, \dots \, u'_{t-n_i} \; 1 \,]'$.

3.2 *Optimization of feedback gains*

To update the feedback gain matrix K^t at time t we apply the following procedure. First, we follow the sampling procedure described in (Ferrarotti and Bemporad, 2019) to obtain a mini-batch $\{w_h = (s_0^h, \{r_l^h, d_l^h\}_{l=0}^{L-1})\}_{h=1}^{N_b}$. To summarize the sampling procedure, we consider the dataset X_t of states x_t defined as in (11), recorded and stored during the evolution of the plant [1]. We randomly sample N_0 states $\{x_{\gamma_t(i)}\}_{i=1}^{N_0}$ from X_t, and perturb them by a (small) normally distributed white noise v_i with variance σ_v^2. We combine the sampled states with N_q samples q_0^j from a normally distributed random variable with zero mean and variance σ_q^2. Hence, we set

$$s_0^{i,j} = \begin{bmatrix} x_{\gamma_t(i)} + v_i \\ q_0^j \end{bmatrix} \quad i = 1, \dots, N_0, \; j = 1, \dots, N_q$$

Each reference trajectory r_l^k is assumed constant between 0 and $L-1$, and we choose N_r constant reference values randomly from the interval $[r_{\min}, r_{\max}]$ of references of interest. A number N_d of disturbance samples d_l^v are randomly generated for $l = 0, \dots, L-1$, $v = 1, \dots, N_d$, from a given interval $[-d_{\max}, d_{\max}]$. All the possible combinations of the sampled states, references and disturbances are built; each of them will be an element $w_h = w_{h(i,j,k,v)}$ of the mini-batch of cardinality $N_b = N_0 \, N_q \, N_r \, N_d$.

After sampling, we assign each w_h to one of the regions $\{R_m(c^{t-1}) \mid m = 1, \dots, M\}$, based on the distance of (s_0^h, r_0^h) from the centroids as defined in (6). In this way, we obtain M sub-batches R_m^t of samples of the mini-batch all belonging to the same region $R_m(c^{t-1})$ for $m = 1, \dots, M$. The sub-batches are non overlapping, because the regions generated by the Voronoi partition associated to c^{t-1}

[1] In the *online* setting X_t is the set of all the states x_t visited by the plant up to the current instant t. In the *offline* setting, instead, at every iteration t we sample from a set X_N containing N states obtained from a previous open-loop data collection phase.

Algorithm 1 Optimal switching policy search

Input: Initial policy $\{K_0, c_0\}$, number N_{learn} of steps.
Online setting: model Θ_0, training reference $\{r_t\}_{t=0}^{N_{\text{learn}}}$.
Offline setting: set $\{X_N, \Theta_N\}$ of states and local models.

Output: Optimal switching policy $\{K_{\text{learn}}, c_{\text{learn}}\}$.

for $t = 1, 2, ..., N_{\text{learn}}$ **do**
  online setting: acquire y_t. Recursively update Θ_t;
  - Policy update:
  $|R^t_m| = 0 \quad m = 1, \dots, M$;
  for $h = 1, 2, ..., N_b$ **do**
    sample $w_h = (s^h_0, \{r^h_l,\ d^h_l\}_{l=0}^{L-1})$ (see Sec.3.2);
    compute $j^h = \min \arg\min_j d([s^h, r^h]', c^{t-1}_j)$
    add w_h to $R^t_{j^h}$ and $|R^t_{j^h}| = |R^t_{j^h}| + 1$;
  end for
  $K^{(0)} = K^{t-1}$;
  take a random permutation $\xi_t \in \mathcal{P}_M$;
  for $m = 1, 2, ..., M$ **do**
    for $j = 1, 2, ..., |R^t_{\xi(m)}|$ **do**
      approximate $\nabla_{K_{\xi(m)}} \hat{J}_L(K^{(m-1)}, c^{t-1}, w_j)$;
    end for
    discard the approximations as described in Section 3.3. Update $|R^t_{\xi(m)}|$ accordingly;
    update $\mathcal{D}(K^{(m-1)}_{\xi(m)})$ as in (16)
    update $K^{(m)}_{\xi(m)} \leftarrow K^{(m-1)}_{\xi(m)} - \alpha_t \mathcal{D}(K^{(m-1)}_{\xi(m)})$;
  end for
  - Centroid update:
  if $\text{rem}(t,\ n_K) = 0$ **then**
    sample $\{w_k = (s^k_0, \{r^k_l,\ d^k_l\}_{l=0}^{L-1})\}_{k=1}^{N_b}$ (see Sec.3.2);
    $c^{(0)} = c^{t-1}$
    for $i = 1, 2, ..., n_c$ **do**
      $W_m = 0, \bar{N}^m_i = 0 \quad m = 1, \dots, M$;
      for $k = 1, 2, ..., N_b$ **do**
        $m^k = \min(\arg\min_m F(m, w_k, K^t, c^{(i-1)})$;
        $\bar{N}^{m^k}_i = \bar{N}^{m^k}_i + 1,\ W_{m^k} = W_{m^k} + w_k$;
      end for
      for $m = 1, 2, ..., M$ **do**
        $c^{(i)}_m = W_m / \bar{N}^m_i$;
      end for
    end for
    for $m = 1, 2, ..., M$ **do**
      $N^t_m = N^{t-1}_m + \bar{N}^m_{n_c}$;
      $c^t_m = \frac{1}{N^t_m}(N^{t-1}_m c^{t-1}_m + \bar{N}^m_{n_c} c^{(n_c)}_m)$;
    end for
  else
    $c^t = c^{t-1}$;
  end if
  online setting: apply $u_t \leftarrow -(K^t_{\sigma_{c^t}(s_t, r_t)})' \begin{bmatrix} s_t \\ r_t \end{bmatrix}$;
end for
$\{K_{\text{learn}}, c_{\text{learn}}\} \leftarrow \{K_{N_{\text{learn}}}, c_{N_{\text{learn}}}\}$;
end.

are disjoint, and possibly empty (in case that the mini-batch does not contain any element belonging to a certain region). They satisfy $|R^t_1| + \cdots + |R^t_M| = N_b$.
Then, we start with the sequential update of the feedback gains as follows. First, we take $K^{(0)} = K^{t-1}$ and select a random permutation of M elements $\xi_t \in \mathcal{P}_M$. Then, for $m = 1, \dots, M$:

- we approximate the gradient in $K_{\xi(m)}$ of $\hat{J}_L$, evaluated in $(K^{(m-1)}, c^{t-1}, w)$, for all $w = w_{h(i,j,k,v)} \in R^t_{\xi(m)}$.
  Function $\hat{J}_L$ approximates (7) by using the local model $\Theta_{\gamma_t(i)}$, estimated at the time instant $\gamma_t(i)$ in which the state $x_{\gamma_t(i)}$ was visited by the process. We use finite differences with precision ϵ to estimate each component of the gradient. For $i = 1, \dots, (n_s + n_p)$ we define the set of gains $K(\epsilon, i)^{\xi(m)}$, varying $K^{(m-1)}$ of a quantity ϵ in the i-th element of the $\xi(m)$-th gain, i.e., we set
  $$K(\epsilon, i)^{\xi(m)} = \{K^{(m-1)}_1, \dots, K^{(m-1)}_{\xi(m)} + \epsilon\, e_i, \dots, K^{(m-1)}_M\}$$
  and calculate
  $$\frac{1}{\epsilon}\Big(\hat{J}_L(K(\epsilon, i)^{\xi(m)}, c^{t-1}, w) - \hat{J}_L(K^{(m-1)}, c^{t-1}, w)\Big).$$
- To avoid issues of lack of differentiability, as motivated in Section 3.3, we remove w from $R^t_{\xi(m)}$ and discard the associated gradient estimates if the sequence of regions visited by simulating the policy $K^{(m-1)}$ differs from the sequence of regions visited by simulating the policy $K(\epsilon, i)^{\xi(m)}$ for some $i \in \{1, \dots, (n_s + n_p)\}$. Then $|R^t_{\xi(m)}|$ is decreased accordingly.
- The update direction for SGD is calculated as follows:
$$\mathcal{D}(K^{(m-1)}_{\xi(m)}) = \frac{\sum\limits_{w \in R^t_{\xi(m)}} \nabla_{K_{\xi(m)}} \hat{J}_L(K^{(m-1)}, c^{t-1}, w)}{|R^t_{\xi(m)}|} \tag{16}$$
- Finally, the policy update performed by the SGD algorithm with learning rate α_t, using (16) is
$$K^{(m)}_{\xi(m)} = K^{(m-1)}_{\xi(m)} - \alpha_t \mathcal{D}(K^{(m-1)}_{\xi(m)}). \tag{17}$$

3.3 Batch reduction

Consider $J^c_w(K) = \hat{J}_L(K, c, w)$, where c and w are fixed. Let us introduce the function $\sigma_c : \mathbb{R}^{n_s+n_p} \to \{1, \dots, M\}$,
$$\sigma_c(s, r) = \min(\arg \min_{i \in \{1,\dots,M\}} d(\begin{bmatrix} s \\ r \end{bmatrix}, c_i)),$$
that assigns each state and reference to the corresponding region index, with respect to the centroids c. Define $\Sigma(K, c, w) = \{\sigma_c(s_l, r_l) \mid l = 0, \dots, L-1\}$ as the sequence of indices of the regions $R_1(c), \dots, R_M(c)$ visited following a trajectory of length L induced by K, given initial state, reference trajectory, and disturbances indicated by w. $\Sigma(K, c, w)$ is an ordered sequence of length L, composed by indices belonging to $\{1, \dots, M\}$. Merging σ_c with (7) we can write the function in (7) as
$$J^c_w(K) = \begin{cases} \sum\limits_{l=0}^{L-1} \rho(s_{l+1}, r_l, -K'_{\chi_1(l)} \begin{bmatrix} s_l \\ r_l \end{bmatrix}), & \text{if } K \in R^K_1 \\ \vdots & \vdots \\ \sum\limits_{l=0}^{L-1} \rho(s_{l+1}, r_l, -K'_{\chi_n(l)} \begin{bmatrix} s_l \\ r_l \end{bmatrix}), & \text{if } K \in R^K_n \end{cases}$$
where $\mathcal{S}_{M,L} = \{\chi_1, \dots, \chi_n\}$ is the set of all the possible sequences of M objects of length L, having cardinality $n = M^L$ and R^K_j is the set of gains K that, given w and c, is characterized by the sequence $\Sigma(K, c, w) = \{\chi_j(l) \mid$

Table 1. Parameters

	n_y	n_u	n_i	n_o	Q_k	R_k	L
PWL	1	1	2	2	$10 \cdot I$	0.01	10
NL	2	1	2	1	$10 \cdot I$	0.1	10
	N_0	N_r	N_q	$r_{\min}$	$r_{\max}$	σ_q	σ_v
PWL	50	10	5	-10	10	10	0.01
NL	50	5	2	$-\pi$	π	10	0.1

$l = 0, \ldots, L-1\}$. The regions R_j^K are disjoint and their union is equal to the whole space $\mathbb{R}^{n_s+n_p}$ of gains. In the interior of each region R_j^K, function J_w^c is continuous and differentiable for all values of K, being the composition of continuous and differentiable functions, while this is not granted for the values of K belonging on the border ∂R_j^K of the region.
At the t-th iteration of the proposed algorithm we approximate $\nabla_{K_{\xi(m)}} \hat{J}_L(K^{(m-1)}, c^{t-1}, w) = \nabla_{K_{\xi(m)}} J_w^{c^{t-1}}(K^{(m-1)})$, using finite differences with fixed precision ϵ, unless the function is not differentiable in $K^{(m-1)}$. If $K^{(m-1)}$ belongs to R_j^K, we check the sequence $\Sigma(K(\epsilon, i)^{\xi(m)}, c^{t-1}, w)$ of regions visited by $K(\epsilon, i)^{\xi(m)}$: if it is not equal to $\chi(j)$, then $K(\epsilon, i)^{\xi(m)}$ does not belong to the same region R_j^K of $K^{(m-1)}$ for the fixed ϵ. The gain $K(\epsilon, i)^{\xi(m)}$ is defined by adding ϵ to a specific element of $K^{(m-1)}$, which implies that moving away from $K^{(m-1)}$ by a distance ϵ leads to reaching a different region. Given that ϵ is "small", then there is a high chance of $K^{(m-1)}$ laying on ∂R_j^K.
For this reason, we choose to use only the gradients related to those samples w such that $K^{(m-1)}$ and $K(\epsilon, i)^{\xi(m)}$ belong to the same region for $i = 1, \ldots, (n_s + n_p)$, for a given ϵ.

4. NUMERICAL RESULTS

We analyze the ability of our Algorithm 1 in synthesizing optimal switching controllers in offline setting on two simple examples: a piecewise-linear (PWL) plant and a nonlinear (NL) one. The parameters of the model of the KF, and of the mini-batch sampling are reported in Table 1. We design the local model (15) by neglecting the dependence on the disturbances d_t; in the first example we use a linear local model while in the second one we take an affine one. We update the gains using the AMSGrad algorithm [2] (Reddi and Kumar, 2018), a faster variant of SGD. Together with the performance of the presented method, alternating $n_K = 10$ steps of gain update with a step containing $n_c = 10$ iterations of centroid optimization, as a comparison we show the behaviour of the method in case we just optimize the gains, maintaining the centroids unaltered. Both cases (fixed and optimized centroids) are compared to learning a single controller, synthesized using the same technique as in (Ferrarotti and Bemporad, 2019) (corresponding to setting $M = 1$) on the same dataset. Closed-loop performance is measured using the sum of stage costs defined in (13), with $Q_q = 0$.

Example 1 Let the plant P in (1) be the (unknown) single-input single-output (SISO) PWL system

$$x_{t+1} = 0.8 \begin{bmatrix} \cos\alpha_t & -\sin\alpha_t \\ \sin\alpha_t & \cos\alpha_t \end{bmatrix} x_t + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_t, y_t = \begin{bmatrix} 0 & 1 \end{bmatrix} x_t \quad (18)$$

[2] AMSGrad is tuned as follows: in the PWL example we take $\beta_1 = 0.5$, and $\beta_2 = 0.6$, while in the NL one $\beta_1 = 0.8$ and $\beta_2 = 0.6$. We take $\alpha = 0.1$.

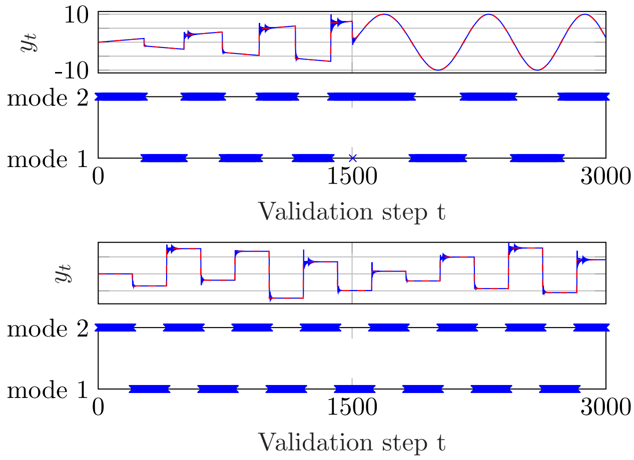

Figure 1. PWL example (offline learning): validation tracking tasks: task A (upper plot) and task B (lower plot).

Table 2. PWL example (offline learning)

	centroids	task A	task B
$M = 1$		11679.54	3065.04
$M = 2$	fixed	179.55	80.49
$M = 2$	optimized	155.81	71.96

and $\alpha_t = \frac{\pi}{3}$ if $x_{t_1} > 0$ and $\alpha_t = -\frac{\pi}{3}$ otherwise (Bemporad and Morari, 1999, Example 4.1). We assume that there is no disturbance d_t affecting P and we parameterize the control increment Δu_t, i.e., $u_t = u_{t-1} + \pi_K^c(s_t, r_t)$. After choosing stage-cost weights $Q_y = 1$, $R_{\Delta u} = Q_q = 0.01$, $R_u = 0$, we fix $M = 2$ and synthesize a switching controller for reference tracking. The policy gains are initialized with K_1^0 and K_2^0 randomly generated from a normally distributed vector with mean zero and standard deviation 0.001. For the optimization of the centroids, it is noticeable that in this plant the switching law is based on the first state x_{t_1}, a piece of information that we assume not to know, and we do not even measure x_{t_1}. We consider regions in $\mathbb{R}^{n_s+p_s}$ defined using the semidistance $d(x, y) = \|x_1 - y_1\|_2$. Based on the definition of s_t given in (12), this means that we base the controller switching law on the output y_t of the plant P. The initial centroids c_0 are picked randomly in $[r_{\min}, r_{\max}]$. The learning procedure is executed for $N_{\text{learn}} = 7000$ iterations over a dataset of cardinality $N = 5000$ collected in open-loop from the plant. The behaviour of the resulting policy $\{K_{\text{learn}}^{\text{off}}, c_{\text{learn}}^{\text{off}}\}$ in validation, while performing two different tracking tasks (indicated as task A and task B) is shown in Figure 1, and the cost of both tasks is compared in Table 2 with the cost of applying a single control law, and with the cost of a switching controller synthesized with fixed centroids $[-4, 0.5]$.

Example 2 To test our approach on a simple nonlinear system, we consider the inverted pendulum, consisting of a mass $\bar{m} = 1$ kg rotating by an angle θ at a fixed distance $\ell = 0.5$ m from the central joint, subject to earth gravity $g = 9.81$ m/s^2 and experiencing a viscous friction governed by the viscosity coefficient $\beta = 0.5$ Nms. The physical model of the inverted pendulum dynamics, when subject to the action of the torque u, is the nonlinear ordinary differential equation (ODE)

Table 3. NL example (offline learning)

	task C		task D	
centroids	fixed	optimized	fixed	optimized
$M = 1$	511.60	//	1097.62	//
$M = 2$	401.29	367.43	988.18	966.80

$$\ell^2\, \bar{m}\ddot{\theta} = \bar{m}\, g\, \ell\, \sin\theta - \beta\dot{\theta} + u.$$

We simulate the inverted pendulum using the ODE solver `ode45` from MATLAB ODE Toolbox with sampling time $T_s = 0.05$ seconds to obtain $y_t = [\theta_t,\, \dot{\theta}_t]'$, where θ_t and $\dot{\theta}_t$ are the angular position and velocity at time instant t. To simulate noisy measurements, additive Gaussian white noise with standard deviation $\sigma_n = 0.01$ is added on both signals. The control objective is to optimally make θ_t track the set point r_t. The considered stage-cost weights are

$$Q_y = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 0 \end{smallmatrix}\right], \quad R_u = 0.01, \quad R_{\Delta u} = 0, \quad Q_q = \left[\begin{smallmatrix} 0.01 & 0 \\ 0 & 0 \end{smallmatrix}\right],$$

where we weight the control input norm u_t to penalize the torque amplitude. The centroid optimization in $\mathbb{R}^{n_s+n_p}$ is realized with respect to the semidistance

$$d(x, y) = \left\| \begin{bmatrix} \cos(x_1) \\ \sin(x_1) \end{bmatrix} - \begin{bmatrix} \cos(y_1) \\ \sin(y_1) \end{bmatrix} \right\|_2,$$

i.e., we define the switching law as a function of the angular position θ_t. As for the PWL case, we generate the initial policy gains K^0 randomly from a normally distributed vector with mean zero and standard deviation 0.001. We consider $M = 2$ control modes and synthesize a switching controller, performing $N_{\text{learn}} = 7000$ iterations of the proposed method, using a dataset of cardinality $N = 5900$, collected in open-loop from the plant. The centroids are initialized as $c_0 = [\,0,\ \pi\,]$ and the same values are used as centroids in the fixed-centroid case. The results in validation on two different tracking tasks (indicated as task C and task D) of 3000 steps presented in Table 3 show that both the switching controllers with $M = 2$ modes outperform the single controller. In particular, the policy $\{\,K^{\text{off}}_{(2)},\, c^{\text{off}}_{(2)}\,\}$, obtained by the procedure with centroids optimization, results to be the best of the three, showing the importance of learning the partition. In Figure 2 the validation tasks C and D are shown, together with the behaviour in validation of the synthesized controller with $M = 2$ modes.

5. CONCLUSIONS

We have presented an approach to synthesize optimal switching feedback controllers that learns both the set of control laws and the switching law directly from experimental data, without requiring a system identification phase of the open-loop process. Simple local linear models are recursively updated with the purpose of approximating both the local optimization direction for the controllers parameters, and the cost of each controller on different areas of the space, so as to optimize the assignment of control domains to control laws. Current research is devoted to extending the approach in several directions, including the investigation of stability conditions for the method and the use of more general parameterizations of both the control laws and the switching law.

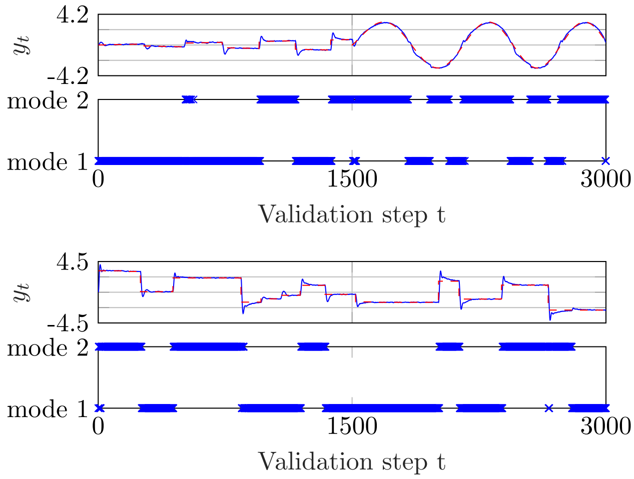

Figure 2. NL example (offline learning), validation tracking tasks: task C (upper plot) and task D (lower plot).

REFERENCES

Bemporad, A. and Morari, M. (1999). Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3), 407–427.

Breschi, V. and Formentin, S. (2020). Direct data-driven design of switching controllers. *International Journal of Robust and Nonlinear Control*. doi:10.1002/rnc.4821.

Dai, T. and Sznaier, M. (2018). A moments based approach to designing mimo data driven controllers for switched systems. In *2018 IEEE Conference on Decision and Control (CDC)*, 5652–5657.

Desienroth, M.P., Neumann, G., and Peters, J. (2011). A survey on policy search for robotics. *Foundations and Trends in Robotics*, 2(1-2), 1–142.

Ferrarotti, L. and Bemporad, A. (2019). Synthesis of optimal feedback controllers from data via stochastic gradient descent. In *Proceedings of the 18th European Control Conference (ECC)*, 2486–2491.

Formentin, S., Heusden, K., and Karimi, A. (2014). A comparison of model-based and data-driven controller tuning. *International Journal of Adaptive Control and Signal Processing*, 28.

Grudic, G., Kumar, R., and Ungar, L. (2003). Using policy gradient reinforcement learning on autonomous robot controllers. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. Las Vegas, Nevada, USA.

Nagayoshi, M., Murao, H., and Tamaki, H. (2010). A reinforcement learning with switching controllers for a continuous action space. *Artificial Life and Robotics*, 15(1), 97–100.

Novara, C. and Formentin, S. (2018). Data-driven inversion-based control of nonlinear systems with guaranteed closed-loop stability. *IEEE Transactions on Automatic Control*, 63(4), 1147–1154.

Piga, D., Formentin, S., and Bemporad, A. (2018). Direct data-driven control of constrained systems. *IEEE Transactions on Control Systems Technology*, 26(4), 1422–1429.

Recht, B. (2018). A Tour of Reinforcement Learning: The View from Continuous Control.

Reddi, S. and Kumar, S. (2018). On the convergence of Adam and beyond. In *Proc. International Conference on Learning Representation*. Vancouver, CA, USA.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3), 400–407.