Control-Tutored Reinforcement Learning: Towards the Integration of Data-Driven and Model-Based Control

Francesco DeLellis FRANCESCO.DELELLIS@UNINA.IT
University of Naples Federico II, Italy

Marco Coraggio MARCO.CORAGGIO@UNINA.IT
Scuola Superiore Meridionale, Italy

Giovanni Russo* GIOVARUSSO@UNISA.IT
University of Salerno, Italy

Mirco Musolesi* M.MUSOLESI@UCL.AC.UK
University College London, UK, and University of Bologna, Italy

Mario di Bernardo* MARIO.DIBERNARDO@UNINA.IT
University of Naples Federico II, Italy, and Scuola Superiore Meridionale, Italy

Abstract

We present an architecture where a feedback controller derived on an approximate model of the environment assists the learning process to enhance its data efficiency. This architecture, which we term as Control-Tutored Q-Learning (CTQL), is presented in two alternative flavours. The former is based on defining the reward function so that a Boolean condition can be used to determine when the control tutor policy is adopted, while the latter, termed as probabilistic CTQL (pCTQL), is instead based on executing calls to the tutor with a certain probability during learning. Both approaches are validated, and thoroughly benchmarked against Q-Learning, by considering the stabilization of an inverted pendulum as defined in OpenAI Gym as a representative problem.

Keywords: Reinforcement learning based control, data-driven control, feedback control.

1. Introduction

Reinforcement learning (RL) is a popular framework used to control systems and devices in a wide range of applications because of its ability to autonomously find control policies to achieve a desired goal without assuming that the dynamics of the environment are known (Sutton and Barto, 2018; Bertsekas and Tsitsiklis, 1996). Despite the many remarkable successes of this type of approaches (Nian et al., 2020), two key problems for RL algorithms remain to be solved: (i) potentially *long* learning times and (ii) the lack of convergence or performance guarantees (important for example in safety-critical applications) during learning (Berkenkamp et al., 2017; Pfeiffer et al., 2018).

To overcome these problems, a possible solution, particularly for control applications of RL, is to adopt model-based solutions where the learning agent derives and refines a data-driven model of the environment during the learning process. Examples in the literature include (Deisenroth and Rasmussen, 2011; Kurutach et al., 2018) among many others. However, in many control applications, some equation-based models of the environment are often available, even though they might not be accurate enough to allow for an entirely model-based solution of the control problem. When

classical RL is used, such approximate or partial models of the environment are often discarded in favour of a completely model-free approach.

In this paper, we investigate the possibility of embedding a feedback control law synthesized by using a partial or uncertain model of the environment to assist the learning process. In the same spirit of human-assisted learning strategies, where data collected from *humans* in the loop are exploited to enhance the learning process (Lien and Pratt, 2009; Suppakun and Maneewarn, 2020; Zhan et al., 2021; Nguyen et al., 2019), here we propose the use of a *feedback controller* in the loop with the aim of steering the learning process, reducing the amount of data samples required and improving the ability of learnt policies to achieve the required stability, robustness and performance. The contributions of this paper can be summarized as follows: (1) we propose a novel algorithm that leverages the use of a feedback controller in the loop to make the RL process more data efficient; (2) we present both a deterministic and a probabilistic approach to implement the strategy above and decide when assistance from the control tutor (the feedback controller in the loop) is invoked during the learning; (3) by using a set of aptly defined metrics, we compare the performance of the novel approaches with those of a classical RL algorithm both from a learning and a control perspective by using the inverted pendulum benchmark implementation from OpenAI Gym (Brockman et al., 2016).

Our results convincingly show that the proposed "control-tutored" learning approaches require fewer data samples and/or obtain higher rewards, while achieving smaller errors in control regulation tasks.

2. Related Work

Several solutions in the existing literature aim at combining control theoretic strategies with reinforcement learning to solve control problems. In particular, various approaches combine RL with model predictive control (MPC). For instance, in (Rathi et al., 2020), a MPC is used to decide the action when the state of the system to control is in a certain region, while the action taken from a Q-table is used otherwise; the table being updated after every action. The use of a (linear) MPC strategy is again suggested in (Zanon and Gros, 2021), where a reinforcement learning module can vary the parameters of the cost function and refine the available model of the system to control.

Other solutions combining control strategies with RL include those in (Abbeel et al., 2006), where a policy gradient algorithm is adopted which uses preexisting knowledge of the system dynamics in the form of an approximate Markov decision process; or that presented in (Li et al., 2021), where a *reference action governor* is used to enforce safety constraints (in the sense of restricting the state space to admissible regions). In so doing, the action is decided via an optimization problem that penalizes deviations from the action suggested by a RL strategy, making these approaches a valuable solution to achieve safe RL.

A strategy similar in spirit to the control-tutored reinforcement learning (CTRL) we propose here is reported in (Argerich et al., 2020). Therein, to improve data efficiency, a Deep Q-Network is extended with a policy that, with some probability, can take an action dictated by an "expert", which can solve the control problem. However, differently from (Argerich et al., 2020), in our CTRL approach, we consider the "expert" to be a feedback control law, that if deployed on its own would be in general unable to achieve the control goal. Also, note that contrary to previous approaches, e.g. (Deisenroth and Rasmussen, 2011), where an approximation for the system dynamics is learnt during the control steps, here we assume to possess and exploit some prior information on the

environment before simulations so as to derive some feedback control law to be used to assist the learning agent. An earlier preliminary version of CTRL was recently presented in (De Lellis et al., 2021).

3. Mathematical Preliminaries

Notation. Sets are denoted by calligraphic capital characters and random variables are denoted via capital letters. For example, X is a random variable and we denote its realization by x. The probability density (mass) function of the continuous (discrete) random variable X is denoted by $p(x)$ and we use the notation $x \sim p(x)$ to denote the sampling of a random variable from its probability function. For both continuous and discrete random variables, we always consider the situation where the support of $p(x)$ is compact; $\mathrm{rand}(\mathcal{A})$ denotes the uniform distribution over the set $\mathcal{A}$. The expectation of a function, say $h(\cdot)$, of X is defined as $\mathbb{E}_p[h(X)] \coloneqq \int h(X)p(x)\mathrm{d}x$, when this is continuous; if X is discrete, we have $\mathbb{E}_p[h(X)] \coloneqq \sum h(x)p(x)$. In both cases, the integral/sum is taken on the support of $p(x)$, and we might omit p in $\mathbb{E}_p$ when there is no ambiguity. We denote by $\|\cdot\|$ the Euclidean norm.

Problem set-up. We consider a discrete time dynamical system affected by noise, of the form

$$X_{k+1} = f_k(X_k, U_k, W_k), \quad x_0 = \tilde{x}_0, \tag{1}$$

where $k \in \mathbb{N}_{\geq 0}$ is discrete time, $X_k \in \mathcal{X}$ is the state of the system at time k, with $\mathcal{X}$ being the state space, $\tilde{x}_0 \in \mathcal{X}$ is the initial condition, $U_k \in \mathcal{U}$ is the control input (or action) and $\mathcal{U}$ is the set of feasible inputs. Also, W_k is a random variable representing noise and $f_k : \mathcal{X} \times \mathcal{U} \times \mathcal{W} \to \mathcal{X}$ is the system's dynamics.

Following e.g. (Matni et al., 2019; Recht, 2019), given this set-up, we consider the problem of learning a plan of actions $\pi_1, \dots, \pi_{N-1}$ to solve the following finite-horizon optimization problem:

$$\begin{aligned}
\max_{\pi_1,\dots,\pi_{N-1}} \quad & \mathbb{E}[J^{\bar{\pi}}], && \text{(2a)}\\
\text{s.t.} \quad & X_{k+1} = f_k(X_k, U_k, W_k), \quad k \in \{1, \dots, N-1\}, && \text{(2b)}\\
& U_k = \pi_k(X_k), \quad k \in \{0, \dots, N-1\}, && \text{(2c)}\\
& x_0 \text{ given}, && \text{(2d)}
\end{aligned}$$

where the time horizon is between 0 and N. In (2) the cost is set as the expectation of the *objective function*

$$J^{\bar{\pi}} = r_N(X_N) + \sum_{k=1}^{N} r_k(X_k, X_{k-1}, U_{k-1}), \tag{3}$$

with $r_k : \mathcal{X} \times \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ and $r_N : \mathcal{X} \to \mathbb{R}$ being the *rewards* received, at each k, by the agent. In what follows, whenever we assume a function or quantity is stationary, we drop the subscript k in the notation.

We observe that in many RL scenarios, even if the system dynamics $f_1, \dots, f_{N-1}$ are not perfectly known, some partial knowledge about the plant (from e.g. first-principles) might be available. We propose that this limited information can be exploited to design a feedback control law (or control tutor) that can be used to assist and drive the learning process towards the solution of a control

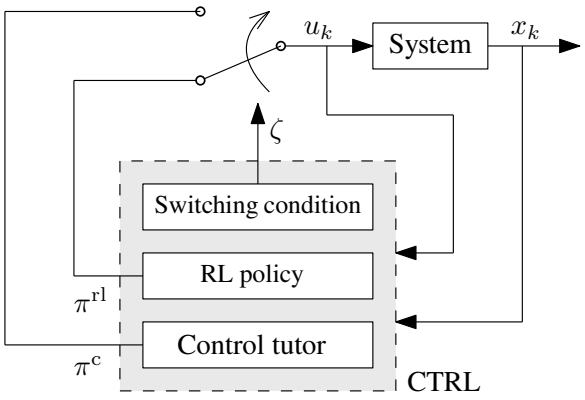

Figure 1: Schematic of the Control-Tutored Reinforcement Learning (CTRL) framework.

problem of interest, reducing the learning times and improving the control performance. In particular, the control tutor can be invoked under certain circumstances during the learning stage to suggest actions that the agent can take as an alternative to those computed using a more traditional approach, e.g. obtained by using a tabular learning strategy.

4. Control Tutored Reinforcement Learning

We start by assuming that we have an estimate of f, say $\hat{f} : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$, so that the dynamics of system (1) is rewritten as $f(x,u,w) = \hat{f}(x,u) + \delta(x,u,w), \forall x \in \mathcal{X}, \forall u \in \mathcal{U}, \forall w \in \mathcal{W}$, where $\delta : \mathcal{X} \times \mathcal{U} \times \mathcal{W} \to \mathcal{X}$ describes the effect of unknown terms in the dynamics and/or of noise on the system's dynamics. We term the policy based on the use ofx a control law synthesized by considering only $\hat{f}$ as the *control tutor policy*, and denote it by $\pi^{\mathrm{c}} : \mathcal{X} \to \mathcal{U}$.

The architecture of the *Control Tutored Reinforcement Learning* (CTRL) strategy (De Lellis et al., 2021) is schematically shown in Figure 1. The figure highlights the presence of a switching condition ζ that orchestrates, at each k, the use of either a policy coming from a RL algorithm or the tutor policy. The result is the following switching policy used for learning:

$$\pi(x) = \begin{cases} \pi^{\mathrm{rl}}(x), & \text{if } \zeta \text{ is true,} \\ \pi^{\mathrm{c}}(x), & \text{otherwise,} \end{cases} \tag{4}$$

where ζ is a Boolean function (that might depend on time, previous states, etc.) and π^{rl} is the policy of a RL algorithm.

For concreteness, we now provide a simple expression for the control tutor policy π^{c}. First, let $\bar{\mathcal{U}} \supseteq \mathcal{U}$ ($\bar{\mathcal{U}}$ might be a continuous set whose discretization yields $\mathcal{U}$); then, from $\hat{f}$ we can design a feedback control strategy $v : \mathcal{X} \to \bar{\mathcal{U}}$. At this point, from v, letting $\epsilon^{\mathrm{c}} \in (0,1)$, and $\forall x \in \mathcal{X}$, we take the *control tutor policy* in (4) as

$$\pi^{\mathrm{c}}(x) = \begin{cases} \arg\min\limits_{u \in \mathcal{U}} \|v(x) - u\|, & \text{with probability } 1 - \epsilon^{\mathrm{c}}, \\ u \sim \mathrm{rand}(\mathcal{U}), & \text{with probability } \epsilon^{\mathrm{c}}, \end{cases} \tag{5}$$

On the other hand, for the reinforcement learning policy π^{rl} in (4), we adopt an ϵ-greedy Q-Learning solution, i.e.,

$$\pi^{\text{rl}}(x_k) = \begin{cases} \arg\max_{u \in \mathcal{U}} Q_k(x_k, u), & \text{with probability } 1 - \epsilon^{\text{rl}}, \\ u \sim \text{rand}(\mathcal{U}), & \text{with probability } \epsilon^{\text{rl}}, \end{cases} \tag{6}$$

where $\epsilon^{\text{rl}} \in (0, 1)$ and $Q_k : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ is the well-known *state-action value function* (Sutton and Barto, 2018; Bertsekas and Tsitsiklis, 1996) at time k.

At time k, once an action is selected from either π^{rl} or π^{c}, the corresponding reward is obtained and used to update the Q-table according to the law

$$Q_{k+1}(x_k, u_k) = (1 - \alpha) Q_k(x_k, u_k) + \alpha [r(x_{k+1}, x_k, u_k) + \gamma \max_{u \in \mathcal{U}} Q_k(x_{k+1}, u)], \tag{7}$$

where $\alpha \in (0, 1]$ is the *learning rate* and $\gamma \in (0, 1]$ is the *discount factor*.

The remaining term to be defined in (4) is ζ. In the following, we present two alternative choices for ζ that result into two different algorithms.

4.1. Control-Tutored Q-Learning

This first algorithm based on the CTRL framework is the *Control-Tutored Q-Learning* (CTQL), which was first presented in (De Lellis et al., 2021). This algorithm is used to solve regulation problems and uses a reward with a specific structure. In particular, letting $x^* \in \mathcal{X}$ be a *goal state*, $\theta \in \mathbb{R}_{>0}$, and $\bar{\rho} \in \mathbb{R}_{>0}$, we define the *prize function*

$$\rho(x) = \begin{cases} \bar{\rho}, & \text{if } \|x - x^*\| < \theta, \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

Then, letting $d : \mathcal{X} \to \mathbb{R}_{\geq 0}$ be some distance of the argument with respect to x^*, the reward r_k in (3) is given as

$$r(X_k, X_{k-1}, U_{k-1}) = d(X_{k-1}) - d(X_k) + \rho(X_k), \quad k = 1, \dots, N-1, \tag{9}$$

with $r_N(X_N) = 0$. Note that, in (9), U_{k-1} does not directly affect the reward but its effect is propagated through the system dynamics. Furthermore, the term $d(X_{k-1}) - d(X_k)$ is positive when at time k the agent gets closer to the goal state x^*, and vice versa. The prize term $\rho(X_k)$ gives a strong positive reinforcement when a small distance with respect to the goal state is achieved. The switching criterion ζ in (4) depends on the current state x_k, where

$$\zeta \text{ is } \begin{cases} \text{true}, & \text{if } \max_{u \in \mathcal{U}} Q_k(x_k, u) > 0, \\ \text{false}, & \text{otherwise.} \end{cases} \tag{10}$$

Additionally, $\forall x \in \mathcal{X}, \forall u \in \mathcal{U}$, we initialize $Q_0(x, u) = 0$. Thus, in the first phase of learning, when limited information about the environment is available, the control tutor policy π^{c} drives the learning process. Then, gradually, as the values of the Q-table are updated using (7), the reinforcement learning policy π^{rl} is preferred.

4.2. Probabilistic Control-Tutored Q-Learning

Although we found the CTQL to have better performance with respect to the classical Q-Learning in certain scenarios (see Section 7), the reward (9) does not satisfy the hypotheses used in the classical proof of convergence used for the Q-Learning (see, e.g., (Bertsekas and Tsitsiklis, 1996)), as it is not either non-negative or non-positive.

Moreover, we verified that the CTQL might fail when the reward function is not selected following the structure given in (9). This might depend on the possibility that the reward in (9) shapes a Q-table where state-action pairs that eventually lead to the goal state (following policy π^{rl}) have positive values of Q. This is used in the switching condition (10) ; however, a detailed analytical characterization is beyond the scope of this work and will be the subject of future study. Therefore, we propose next a simpler probabilistic-based choice for the Boolean condition ζ in (4). We name the resulting algorithm as *probabilistic Control Tutored Q-Learning* (pCTQL); differently from CTQL (cf. (9)), we do not use any specific structure for the reward function to derive the switching condition between the two policies.

In particular, letting $\beta \in [0, 1]$,

$$\zeta \text{ is } \begin{cases} \text{true}, & \text{with probability } \beta, \\ \text{false}, & \text{otherwise}, \end{cases} \tag{11}$$

the pCTQL policy is defined as

$$\pi(x) = \begin{cases} \arg\max\limits_{u \in \mathcal{U}} Q_k(x, u), & \text{with probability } \beta(1 - \epsilon^{\mathrm{rl}}), \\ \arg\min\limits_{u \in \mathcal{U}} \|v(x) - u\|, & \text{with probability } \omega \coloneqq (1 - \beta)(1 - \epsilon^{\mathrm{c}}), \\ u \sim \mathrm{rand}(\mathcal{U}), & \text{otherwise}, \end{cases} \tag{12}$$

Note that it is also possible to introduce a dependency of the probability β on the current state, time, or other quantities.

5. Metrics

Here we define several metrics to characterize and compare quantitatively the performance of different control algorithms. Each numerical simulation is run in $S \in \mathbb{N}_{>0}$ independent sessions. Each session is composed of $E \in \mathbb{N}_{>0}$ episodes: the learned quantities (e.g., Q-table) are carried over from one episode to the next, and re-initialized at each session. Each episode consists of a simulation of $N \in \mathbb{N}_{>0}$ time steps. We let J_e^π be the cumulative reward (as given in (3)) obtained in episode e. Moreover, we let the *goal condition* be a Boolean proposition that assesses whether the control goal can be considered as having been achieved in an episode (the specific form of the goal condition depends on the task at hand). We define the following three metrics to assess the learning performance.

Definition 1 (Learning metrics) *(i) The* average cumulative reward *is* $J^\pi_{\mathrm{avg}} \coloneqq \frac{1}{E}\sum_{e=1}^{E} J_e^\pi$. *(ii) Given some integer,* $E^* > 0$, *the* terminal episode E_{t} *is the smallest episode such that the goal condition is satisfied for all* $e \in \{E_{\mathrm{t}} - E^*, \dots, E_{\mathrm{t}}\}$. *(iii) The* average cumulative reward after terminal episode *is* $J^\pi_{\mathrm{avg,t}} \coloneqq \frac{1}{E_{\mathrm{t}}}\sum_{e=E_{\mathrm{t}}}^{E} J_e^\pi$.

J^{π}_{avg} is a common metric typically used in RL (Duan et al., 2016; Wang et al., 2019). E_{t} is used to assess when the learning phase might be considered concluded, and thus to evaluate data efficiency; additionally, in the definition of E_{t}, we chose the value $E^* = 30$ as that can be considered an effective indicator of the stabilization of the pendulum in the problem we considered. $J^{\pi}_{\text{avg,t}}$ describes how performing the controller is, in terms of rewards, once training is completed. Next, we define two metrics inspired by those commonly used in control theory to assess the transient and steady-state performance of an algorithm. Let again x^* be a goal state, let $\eta \in \mathbb{R}_{\geq 0}$, $N^- \in \mathbb{N}_{>0}$ with $N^- < N$, and let the goal condition be true if

$$\exists \bar{k} \in [0, N^-] : \|x_k - x^*\| \leq \eta, \quad \forall k \in [\bar{k}, N]. \tag{13}$$

Definition 2 (Control metrics) *(i) In an episode, the* settling time k_{g} *is the smallest value of* $\bar{k}$ *that fulfills* (13)*. (ii) The* steady state error *is* $e_{\text{g}} \coloneqq \frac{1}{N-k_{\text{g}}+1} \sum_{k=k_{\text{g}}}^{N} \|x - x^*\|$.

6. Benchmark Description

6.1. Control Problem

As a benchmark problem to compare the performance of the proposed algorithms, we consider the problem of stabilizing a pendulum in its inverted position, provided by the OpenAI Gym framework (Brockman et al., 2016; OpenAI, 2019). This problem is particularly representative for two reasons. (i) As the upward position is unstable and the the system dynamics is nonlinear, this problem is typically used in control theory as a test for new control strategies (Khalil, 2002). (ii) We will select a linear feedback controller (v in (5)), which by itself cannot stabilize the pendulum. This means that any benefit observed when using CTQL and pCTQL will be due to the combination of the reinforcement learning policy and the model-based one, and not just the latter.

Environment. The pendulum is a rigid rod of length $l = 1$ m, with a homogeneous distribution of mass $m = 1$ kg; its moment of inertia is $I = ml^2/3$ and it is affected by gravity, with acceleration g. We let $x_k = [x_{1,k} \;\; x_{2,k}]^{\mathsf{T}}$, where $x_{1,k}$ and $x_{2,k}$ are the angular position and angular velocity of the pendulum, respectively; $x_{1,k} = 0$ corresponds to the unstable vertical position. The control input u_k is a torque applied to the pendulum. The discrete-time dynamics is obtained by discretizing the continuous-time dynamics with a sampling time $T = 0.05$ s using the forward Euler method. Unless noted otherwise, the initial condition is the downward stable position $\tilde{x}_0 = [\pi \;\; 0]^{\mathsf{T}}$.

State and control spaces. The spaces for states and control variable are bounded, so that $x_k \in [-\pi, \pi] \times [-8, 8]$, and $u_k \in [-2, 2]$. Both spaces are discretized non uniformly, employing a finer discretization close to the origin of the state space and for small control actions. We verified that this allows to select values more precisely when close to the regulation point, reducing regulation error and control energy; on the other hand, a coarser discretization far from the regulation point yields shorter learning time. Concerning $x_{1,k}$, the interval $\left[-\pi, -\frac{\pi}{9}\right]$ is discretized into 8 equally spaced values, $\left(-\frac{\pi}{9}, -\frac{\pi}{36}\right]$ into 7 values, and $\left(-\frac{\pi}{36}, 0\right]$ into 5 values; $[0, \pi]$ is discretized in an analogous fashion. Concerning $x_{2,k}$, $[-8, -1]$ is discretized into 10 values, and $(-1, 0]$ into 9 values (analogously for $[0, 8]$). Concerning u_k, $[-2, -0.2]$ is discretized into 9 values, and $(-0.2, 0]$ into 4 values (analogously for $[0, 2]$.

Iterations. For each set-up, we run $S = 10$ sessions and average the results. For each session, we run $E = 10000$ episodes, composed of $N = 400$ time steps.

Goal and rewards. The objective is to stabilize the pendulum in its upward position, $x^* = [0\ \ 0]^\mathsf{T}$. Concerning the goal condition in (13), we take $N^- = 300$ and $\eta = 0.05 x_{\max}$, where $x_{\max} \coloneqq \|[\pi\ \ 8]\|$. This goal is encoded in two reward functions. The first one is

$$r^{\mathrm{a}}(X_k, X_{k-1}, U_{k-1}) = d(X_{k-1}) - d(X_k) + \rho(X_k), \tag{14}$$

where $d(x) \coloneqq x_1^2 + 0.1 x_2^2$, and ρ was given in (8), with $\bar{\rho} = 5$ and $\theta = 0.05$. The second reward function we will consider is the standard Gym reward, i.e.,

$$r^{\mathrm{g}}(X_k, X_{k-1}, U_{k-1}) = X_{1,k}^2 + 0.1 X_{2,k}^2 + 0.001 U_{k-1}^2. \tag{15}$$

Hyperparameters. In (7), we take $\gamma = 0.97$ and $\alpha = \left(1 + \frac{e}{1000}\right)^{-\frac{1}{2}}$, where e is the current episode (Even-Dar et al., 2003), so that the learning rate decays approximately from 0.7 to 0.3, over 10000 episodes. In (5) and (6), we take $\epsilon^{\mathrm{c}} = \epsilon^{\mathrm{rl}} = 0.03$. Concerning β in (11), we tested $\beta \in \{0.9990, 0.9948, 0.9897, 0.9485, 0.8969\}$, which approximately corresponds to $\omega \in \{0.001, 0.005, 0.010, 0.050, 0.100\}$ in (12).

Feedback control law. We assume we have partial information on the pendulum dynamics, in the form of an approximate dynamics $\hat{f}$. In particular, $\hat{f}$ is the linear dynamics that is topologically equivalent to the nonlinear dynamics of the pendulum, close to the origin $[0\ \ 0]^\mathsf{T}$ (also the goal state). Namely, $\hat{f}\,(x_k, v_k) = Ax_k + Bv_k$, where $A = \begin{bmatrix} 0 & 1+T \\ 3Tg/2l & 1 \end{bmatrix}$ and $B = \begin{bmatrix} 0 \\ T/I \end{bmatrix}$. From $\hat{f}$, we synthesize the linear controller $v_k = -Kx_k$, where $K = [5.83\ \ 1.83]^\mathsf{T}$. This controller can locally stabilize the pendulum in its inverted position from nearby initial conditions, and is obtained, for the sake of simplicity, via a pole placement technique, assigning poles to have an acceptable settling time. Note that this controller if used on its own is in unable to swing up the pendulum from its downward asymptotically stable position.

7. Comparison of Learning Performance

Case of reward (14). First, we compare Q-Learning, CTQL and pCTQL with different values of ω, when using reward (14). The results are reported in Figure 2 in terms of the cumulative reward per episode J_e^π and the frequency with which the control tutor is used. Also a quantitative comparison via the learning and control metrics is reported in Table 1. For the sake of clarity, in Figure 2 the results of the pCTQL were only plotted for $\omega = 0.01$, as we found that value to give the best performance overall.

From Table 1, comparing CTQL and pCTQL to Q-Learning, we observe that E_{t}—a measure of data efficiency—is smaller (by a statistically significant margin) for the CTQL and for the pCTQL with $\omega = 0.05$; however, the presence of a constant bias from the control tutor in the pCTQL worsens the overall performances which shows a decreasing trend of J_{avg}^π and J_{avg}^π as ω increases. Furthermore, the positive effects of the tutored approaches are also captured in 2.(a), as the reward curves of pCTQL and CTQL grow earlier than that of Q-Learning. Finally, Figure 2.(b) shows that CTQL uses the control tutor policy more in the beginning, and progressively less as episodes are completed.

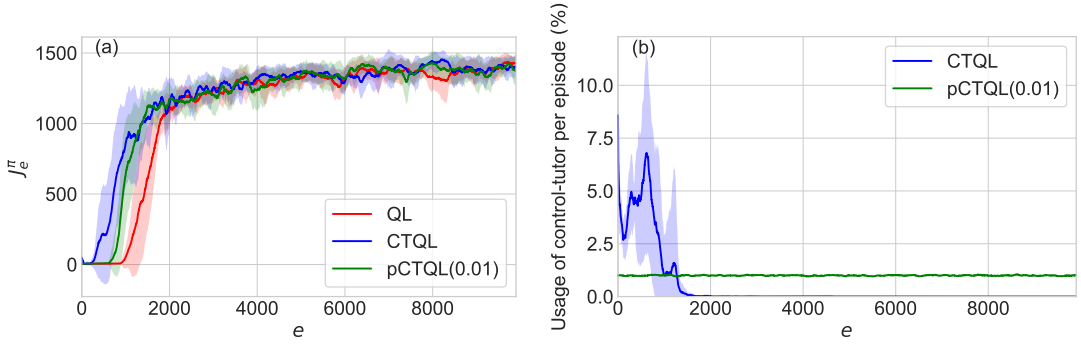

Figure 2: (a) Cumulative reward per episode J_e^π, obtained with reward (14). (b) Percentage of steps the control-tutor policy π^c was used in each episode. In both (a) and (b) the solid curves are the mean of the results of S sessions; for readability, the curves are averaged with a moving average of 100 samples (taken on the right); shaded areas correspond to the means plus or minus the standard deviations.

	Training metrics			Control metrics	
	E_T	J^π_{avg}	$J^\pi_{\text{avg},t}$	k_{g}	$e_{\text{g}}/x_{\max}[\cdot 10^{-3}]$
QL	2726 ± 742	1110 ± 39	1332 ± 45	112 ± 20	2.6 ± 1
CTQL	$\mathbf{2028 \pm 241}$	$\mathbf{1195 \pm 58}$	1336 ± 38	118 ± 19	$\mathbf{1.5 \pm 0.6}$
pCTQL (0.1%)	2670 ± 562	$\mathbf{1157 \pm 45}$	1358 ± 40	125 ± 37	2.4 ± 1.4
pCTQL (0.5%)	2439 ± 823	$\mathbf{1182 \pm 30}$	1372 ± 45	120 ± 25	2.5 ± 1.5
pCTQL (1%)	2106 ± 507	$\mathbf{1164 \pm 57}$	1327 ± 51	110 ± 25	$\mathbf{3.8 \pm 1}$
pCTQL (5%)	$\mathbf{1907 \pm 493}$	1112 ± 16	$\mathbf{1234 \pm 15}$	111 ± 41	1.9 ± 0.8
pCTQL (10%)	2952 ± 739	$\mathbf{992 \pm 26}$	$\mathbf{1152 \pm 24}$	105 ± 21	2.5 ± 0.8

Table 1: Learning metrics (Definition 1) with reward (14) and control metrics (Definition 2), with reward (14) and nominal conditions. The means and standard deviations of S sessions are reported. Values that are statistically significantly different from those of the Q-Learning are in bold (according to a Welch's t-test with p-value less than 0.05 (Welch, 1947)).

Case of reward (15) We also compared the performance of Q-Learning and pCTQL when using reward (15); from Table 2 we see that pCTQL with $\omega = 0.01$ is comparable to Q-Learning in terms of learning time (E_{t}), yet obtains a larger average reward (J^π_{avg}) and average reward after terminal episode ($J^\pi_{\text{avg,t}}$), confirming the effectiveness of a control tutor-based architecture, even when the reward has a structure different from (9).

8. Comparison of Control Performance

Nominal conditions. We also compared the algorithms in terms of their control performance at the end of the learning stage, using the metrics given in Definition 2. The results, using both rewards (14) and (15) are reported in Tables 1–2 where we show that, as it is desirable, the differences in settling time (k_g) of pCTQL and CTQL with respect to Q-Learning are not statistically significant.

	Training metrics			Control metrics	
	E_T	J^{π}_{avg}	$J^{\pi}_{avg,t}$	k_g	$e_g/x_{\max}[\cdot 10^{-3}]$
QL	3207 ± 767	-1045 ± 18	-734 ± 25	137 ± 78	1.5 ± 0.6
pCTQL (0.1%)	3188 ± 692	-1035 ± 11	-723 ± 36	126 ± 40	$\mathbf{0.9 \pm 0.3}$
pCTQL (0.5%)	3711 ± 835	-1009 ± 11	-688 ± 29	150 ± 76	1.4 ± 1.2
pCTQL (1%)	3684 ± 539	$\mathbf{-1010 \pm 11}$	$\mathbf{-692 \pm 16}$	114 ± 33	$\mathbf{0.8 \pm 0.3}$
pCTQL (5%)	3779 ± 1098	$\mathbf{-1017 \pm 10}$	-743 ± 25	107 ± 20	1.2 ± 0.6
pCTQL (10%)	$\mathbf{4552 \pm 1003}$	$\mathbf{-1028 \pm 11}$	$\mathbf{-777 \pm 23}$	134 ± 27	1.2 ± 0.5

Table 2: Learning metrics (Definition 1) with reward (9) and control metrics (Definition 2) with reward (9) and nominal conditions. The means and standard deviations of S sessions are reported. Values that are statistically significantly different from those of the Q-Learning are in bold (according to a Welch's t-test with p-value less than 0.05 (Welch, 1947)).

However, when using reward (14) we observed that the CTQL achieves the best (lowest) steady state error, whereas when using reward (15) the smallest error is given by the pCTQL with $\omega = 0.01$.

Perturbed conditions. To test the robustness of the learned control strategies to changes in the environment, we generated 1000 set-ups, by varying the initial conditions randomly (with a uniform distribution) in the state/control spaces, and varying the mass and length of the pendulum by $\pm 5\%$ of their nominal values, and using the Latin hypercube method (Loh, 1996). The results, not portrayed here for brevity, show that, as is desirable, we obtain similar settling times for all the set-ups. Also, concerning the steady state error, when using reward (14), for all the set-ups, performance remain centered around the ones obtained under nominal conditions. Instead, when using reward (15), pCTQL displays a larger error when compared to that obtained under nominal condition (which was however lower than that of Q-Learning), whereas Q-Learning retains the same mean.

9. Conclusions

We presented a deterministic and a probabilistic Control-Tutored Q-Learning strategy, that integrate a feedback control law synthesized on a partial model of the plant within a Q-Learning framework to render the learning process faster and improving the performance of the learnt policies in achieving a control goal of interest. We compared the control-tutored strategies with a classical Q-Learning approach using the inverted pendulum stabilization benchmark from OpenAI Gym as a representative control problem. We found that, when compared to Q-Learning, CTQL requires fewer data samples and has a larger average reward, while pCTQL yields higher rewards with a comparable number of data samples; moreover, both CTQL and pCTQL yield lower regulation error when certain reward functions are used. Our numerical results show that both from a learning and a control viewpoint using a control-tutored learning approach might be beneficial.

The next step is the derivation of proofs of convergence for the control-tutored algorithms presented in this paper. Also, we wish to uncover and formally characterize the relationships among the specific choice of the reward function, the performance of the algorithms and the approximate system dynamics needed to synthesize the control tutor. We wish to emphasize that embedding a control tutor in the loop could be used to render more efficient learning strategies other than Q-Learning. This will also be the subject of future investigation.

References

Pieter Abbeel, Morgan Quigley, and Andrew Y Ng. Using inaccurate models in reinforcement learning. In *ICML*, pages 1–8, 2006.

Mauricio Fadel Argerich, Jonathan Fürst, and Bin Cheng. Tutor4rl: Guiding reinforcement learning with external knowledge. In *AAAI Spring Symposium: Combining Machine Learning with Knowledge Engineering (1)*, 2020.

Felix Berkenkamp, Matteo Turchetta, Angela Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. *NeurIPS*, 30:908–918, 2017.

Dimitri P Bertsekas and John N Tsitsiklis. *Neuro-dynamic programming*. Athena Scientific, 1996.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016.

Francesco De Lellis, Giovanni Russo, and Mario di Bernardo. Tutoring reinforcement learning via feedback control. *accepted to European Control Conference, available on arXiv, arXiv:2012.06863v1*, 2021.

Marc Deisenroth and Carl E Rasmussen. PILCO: A model-based and data-efficient approach to policy search. *ICML*, pages 465–472, 2011.

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *ICML*, pages 1329–1338. PMLR, 2016.

Eyal Even-Dar, Yishay Mansour, and Peter Bartlett. Learning rates for Q-learning. *Journal of Machine Learning Research*, 5(1), 2003.

Hassan K Khalil. *Nonlinear systems; 3rd ed.* Prentice-Hall, 2002.

Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. In *ICML*, 2018.

Yutong Li, Nan Li, H Eric Tseng, Anouck Girard, Dimitar Filev, and Ilya Kolmanovsky. Safe reinforcement learning using robust action governor. In *Learning for Dynamics and Control*, pages 1093–1104. PMLR, 2021.

Jyh-Ming Lien and Emlyn Pratt. Interactive planning for shepherd motion. In *AAAI Spring Symposium: Agents that Learn from Human Teachers*, pages 95–102, 2009.

Wei-Liem Loh. On latin hypercube sampling. *The Annals of Statistics*, 24(5):2058–2080, 1996.

Nikolai Matni, Alexandre Proutiere, Anders Rantzer, and Stephen Tu. From self-tuning regulators to reinforcement learning and back again. *CDC*, pages 3724–3740, 2019.

Hung The Nguyen, Matthew Garratt, Lam Thu Bui, and Hussein Abbass. Apprenticeship learning for continuous state spaces and actions in a swarm-guidance shepherding task. In *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 102–109. IEEE, 2019.

Rui Nian, Jinfeng Liu, and Biao Huang. A review on reinforcement learning: Introduction and applications in industrial process control. *Computers & Chemical Engineering*, 139:106886, 2020.

OpenAI. *OpenAI Gym Pendulum-v0*, 2019. URL https://github.com/openai/gym/blob/master/gym/envs/classic_control/pendulum.py.

Mark Pfeiffer, Samarth Shukla, Matteo Turchetta, Cesar Cadena, Andreas Krause, Roland Siegwart, and Juan Nieto. Reinforced imitation: Sample efficient deep reinforcement learning for mapless navigation by leveraging prior demonstrations. *IEEE Robotics and Automation Letters*, 3(4): 4423–4430, 2018.

Meghana Rathi, Pietro Ferraro, and Giovanni Russo. Driving reinforcement learning with models. *Proceedings of SAI Intelligent Systems Conference*, pages 70–85, 2020.

Benjamin Recht. A tour of reinforcement learning: The view from continuous control. *Annual Review of Control, Robotics, and Autonomous Systems*, 2:253–279, 2019.

Nakarin Suppakun and Thavida Maneewarn. Coaching: accelerating reinforcement learning through human-assisted approach. *Progress in Artificial Intelligence*, 9(2):155–169, 2020.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

Tingwu Wang, Xuchan Bao, Ignasi Clavera, Jerrick Hoang, Yeming Wen, Eric Langlois, Shunshi Zhang, Guodong Zhang, Pieter Abbeel, and Jimmy Ba. Benchmarking model-based reinforcement learning. *available on arXiv, arXiv:1907.02057*, 2019.

Bernard L Welch. The generalization of 'student's'problem when several different population varlances are involved. *Biometrika*, 34(1-2):28–35, 1947.

Mario Zanon and Sébastien Gros. Safe reinforcement learning using robust MPC. *IEEE Transactions on Automatic Control*, 66:3638–3652, 2021.

Huixin Zhan, Feng Tao, and Yongcan Cao. Human-guided robot behavior learning: A gan-assisted preference-based reinforcement learning approach. *IEEE Robotics and Automation Letters*, 6(2): 3545–3552, 2021.