Learning By Doing NeurIPS 2021 Competition: Controlling a Dynamical System using Control Theory, Reinforcement Learning, or Causality

LearningByDoing@math.ku.dk

Version: August 4, 2021
(This document may be updated during the challenge.)

Abstract. This competition aims to bring together researchers from control theory, reinforcement learning, and causality. We consider the following setting: The participants obtain pre-existing observational and interventional data generated by unknown dynamical systems. Track CHEM is a closed-loop problem in which a single impulse at the beginning of the dynamics can be set, while Track ROBO is an open-loop problem in which control variables can be set at each time step. The goal in both tracks is to infer actions (or controls or interventions) that optimize a given response variable, that is, actions that drive the system to a desired state. In this white paper, we introduce the competition tasks in more detail.

Synergies. We believe that combining the different views from control, RL, and causality might create synergies. All of the three fields go beyond the classical machine learning task of prediction under i.i.d. observations. Instead, they consider systems that are influenced by perturbations actively chosen by a user. More specifically, the problem of learning how to control an unknown system to achieve a certain effect on a response variable is addressed in all of the three communities. But, arguably, the fields have complementary views on the problem: In control theory, methods for system identification usually aim to first identify the system by excitation strategies and then apply classical methods of control. In (non model-based) reinforcement learning, one directly optimizes the reward. In causality, a lot of focus is put on the aspect of identifiability: Under which conditions can one identify causal structure and/or causal effects from observational (and interventional) data. Given the causal structure, intervention distributions can be computed and therefore optimal interventions can be identified. To foster cross-pollination, when introducing the competition tasks below, we explain them within each framework.

Questions asked by the challenge. In this challenge, the participants are asked to control a system that is not fully known. The challenge is designed to provide input on the following questions. For the particular model class considered in this challenge, is it better to aim to build a model first and then infer the optimal control, or to directly estimate the effect of the applied control? To which extent can we transfer knowledge between systems that are not the same (specified by different parameter vectors) but share structural similarities (zeros in parameter vectors)?

Typos and Errors. If you find typos or believe that some parts of the white paper are incorrect, please let us know (by sending an email to the above email address). We may update the white paper accordingly.

Table of content. This white paper contains two sections:

1 The competition

1.1 Track CHEM: Optimally controlling a chemical reaction

In a chemical reaction, one set of chemical compounds is transformed into another. We usually say that reactants are turned into products (both of which are sometimes called species); we provide more details in the paragraph 'Background on chemical reactions' below. In Track CHEM of the competition, the goal is to find optimal policies (or controls or interventions) on the concentrations of chemical compounds to ensure a specific target concentration on one of the products. The dynamical behaviour of species concentrations involved in chemical reactions is modelled by a principle known as mass action kinetics, which results in an ordinary differential equation (ODE) over the species. More concretely, participants need to select a set of controls depending only on some observed initial concentrations of the observed species. During training, the participants are not able to directly interact with the system and instead only have access to previous observations of the system (where the applied control is known but cannot be chosen by the user). The goal in this track is to learn an optimal strategy of interacting with the system when provided only with observational data.

We first provide a few more details on chemical reaction networks and then explain the data generating process in more detail (parts of the following text are taken from Peters et al. [2020]).

Background on chemical reactions A general reaction [e.g. Wilkinson, 2006] takes the form

$$m_1R_1 + m_2R_2 + \ldots + m_rR_r \rightarrow n_1P_1 + n_2P_2 + \ldots + n_pP_p,$$

where r is the number of reactants and p the number of products. Both R_i and P_j can be thought of as molecules and are often called species. The coefficients m_i and n_j are positive integers, called stoichiometries.

In mass-action kinetics [Waage and Guldberg, 1864], one usually considers the concentration $[X]$ of a species X, the square parentheses indicating that one refers to the concentration rather than to the integer amount of a given species. The concentration $[X]$ changes over time (but to simplify notation, we sometimes omit the notational dependence on t). The law of mass-action allows to convert the above equations into a system of ODEs of the concentrations of species. Formally, it states: *The instantaneous rate of each reaction is proportional to the product of each of its reactants raised to the power of its stoichiometry.* To better understand how this can be applied to transform a reaction equation into a system of ODEs, it may help to consider an example. The Lotka-Volterra predator-pray model [Lotka, 1909] can be expressed in terms of reactions of the form

$$A \xrightarrow{k_1} 2A \tag{1}$$

$$A + B \xrightarrow{k_2} 2B \tag{2}$$

$$B \xrightarrow{k_3} \emptyset, \tag{3}$$

where A and B describe abundance of prey and predators, respectively. In this model, the prey reproduce by themselves (1), but the predators require abundance of prey for reproduction, see (2). After some time, also the predators die (3). The coefficients k_1, k_2, and k_3 indicate the rates, with which the reactions happen (the larger the rates, the faster the reactions). Applying the law of mass-action yields the following system of ordinary differential equations (ODEs)

$$\tfrac{\mathrm{d}}{\mathrm{d}t}[A] = k_1[A] - k_2[A][B] \tag{4}$$

$$\tfrac{\mathrm{d}}{\mathrm{d}t}[B] = k_2[A][B] - k_3[B]. \tag{5}$$

Data generating process In Track CHEM the d-dimensional process $Z(t)_{t\geq 0}$ is generated in the following way:

$$\begin{aligned} Z(0) &= z \\ \dot{Z}(t) &= F(Z(t)) + BU(t), \end{aligned} \tag{6}$$

where $U(t) \in [-10, 10]$ is the control input at time t, $z \in (0, \infty)^d$ is an initial value, $B \in \mathbb{R}^{d\times p}$ is a matrix specifying where the controls influence the dynamics and $F : \mathbb{R}^d \rightarrow \mathbb{R}^d$

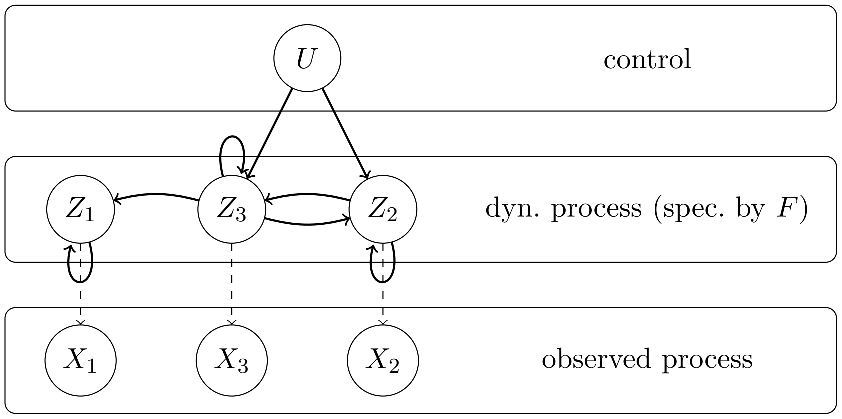

Figure 1: Schematic visualization of the data generating process of Track CHEM. The dynamical process Z is governed by Equation (6), which includes the control variable U. The system is observed directly but only after some observational noise has been added, see Equation (8). The precise graphical structure (solid edges) is unknown and the challenge contains more than 3 variables.

is a function from the function class

$$\mathcal{F} = \left\{ F : \mathbb{R}^d \to \mathbb{R}^d \,|\, F_\ell(Z) = \sum_{j=1}^{d} \theta_j^\ell Z_j + \sum_{k,j=1}^{d} \theta_{j,k}^\ell Z_j Z_k, \quad \ell = 1, \dots, d \right\}. \tag{7}$$

The parameter θ satisfies additional constraints, since we only consider ODE systems that are generated by converting chemical reactions (e.g., (1)-(3)) using the law of mass-action. Furthermore, the rates of the underlying chemical reactions (e.g., k_i in (1)-(3)) are non-negative which adds additional constraints on the coefficients θ. Cyclic relationships are however possible, as an example, it may be that the jth component of F, F_j, depends on Z_k and the kth component of F, F_k, depends on Z_j.

Both the matrix B and the function F are unknown to the participants. Furthermore, participants do not observe the process Z directly. Instead, they only observe a noisy version of the process sampled on a time grid $(t_0, \dots, t_L)$ where $t_0 = 0$. That is, participants have access to samples from a d-dimensional process X at time points $t_0, \dots, t_L$ such that for all $t \in \{t_0, \dots, t_L\}$ it holds that X satisfies

$$X(t) = (Z_1(t), \dots, Z_d(t)) + N(t), \tag{8}$$

where N is a mean zero noise process such that $\{N(t) : t = t_0, \dots, t_L\}$ are independent. The distribution of $N(t)$ may depend on $Z(t)$.

In this track, the control input only consists of a single vector $u \in \mathbb{R}^p$ which is mapped into the data generating process by

$$U(t) = \begin{cases} u & \text{if } t \in [t_0, t_3) \\ 0 & \text{if } t \geq t_3, \end{cases} \tag{9}$$

such that t_3 is the length of the impulse as $t_0 = 0$.

The data are constructed based on 12 different ODE systems which are specified by the functions $F^1, \ldots, F^{12} \in \mathcal{F}$ (the function class $\mathcal{F}$ is known but $F^1, \ldots, F^{12}$ are unknown to the participants). All of these 12 systems have the same structure, that is, the same parameters θ_j^ℓ and $\theta_{j,k}^\ell$ are zero in all 12 systems. The non-zero values of the θ's may differ between the processes, however, every θ_j^ℓ and $\theta_{j,k}^\ell$ has the same sign in all systems. The parameters of the noise as well as the matrix B are the same in all 12 systems.

Goal One of the processes, $Y := X_d$ (called the noisy response process), is of particular interest. The goal is to choose a value u for the control U such that the corresponding process Z_d, which we call the noise-free response process, is close to a pre-specified target value y_* (see below for details). The value u for the control must be set once, after having observed $X(0)$, and thus, this problem can be considered 'open-loop'.

Training data The training data that are available to the participants are generated by running the data generating process 20 times for each F^i and varying pairs of initial conditions z and controls u. The distribution used to select z and u in the training data is unknown to participants and is different for different systems. For each realization, participants are then provided with values of the observed process X, a time indicator t, the applied control u and an indicator i specifying the system. All training data are provided as data frames saved in CSV-files (additional information on the exact data structure can be found in the online tutorial).

Evaluation For each of the systems $F^1, \ldots, F^{12}$ participants are provided with 50 additional sets of initial vectors, $X(0)$, (indexed by k) as well as an indicator specifying the corresponding system ($i = 1, \ldots, 12$). Below, we use the superscript i, k for the variables of the corresponding process, so $Z_d^{i,k}$, for example, denotes the noise-free response process of the ith system (F^i) under the kth initial condition. For each of these combinations participants are asked to select a control input to minimize the loss function. The loss function measures the proximity of the target process, $Z_d^{i,k}$, to the target, $y_*^{i,k}$, towards the end of the observation interval while also adding a penalty term depending on the size of the control input used. The exact loss function is found below. The participants submit all of their selected controls to CodaLab as a CSV-file compressed as a ZIP-file (there is no additional code submission).

We then evaluate the participants' controls by running the data generating process for each of the provided controls to compute the following loss for all systems

$$J_i := \frac{1}{50}\sum_{k=1}^{50}\left(\sqrt{\frac{1}{40}\int_{40}^{80}\left(Z_d^{i,k}(t)-y_*^{i,k}\right)^2 dt} + c\cdot\sqrt{\frac{||u^{i,k}||_2^2}{8}}\right), \tag{10}$$

where $c = \frac{1}{20}$ and $u^{i,k} \in \mathbb{R}^p$ is the control input provided by the participant corresponding to the kth initial condition in the ith system. The process $Z_d^{i,k}$ of course depends on the provided input, $u^{i,k}$, even though this is not made explicit in the notation. In the competition, we have $d = 15$, that is, $X(t)$ is a 15-dimensional process (with $X_{15} = Y$) and $p = 8$, that is, U has 8 dimensions. For the (preliminary) leader board, which is updated during the competition, only six of the systems are evaluated and the mean loss across those systems is shown on the leader board. The final loss used for selecting a winner is then computed as the average loss on the six held out systems.

1.2 Track ROBO: Controlling a robotic arm in a dynamical environment

Track ROBO is motivated by the challenge of learning skills that can be performed by a diverse set of robots. diverse sets of robots. In the future, one may want to teach robots new skills such as stirring a pot or cutting vegetables for cooking. Such cooking robots will be operating in a variety of different environments, such as cafeterias, restaurants, and home kitchens. The robots will often have different dynamics or even kinematic structures, which will need to be taken into account when acquiring the new skill. Each robot will therefore need to generate an individualized control policy to accurately execute the skill's end-effector trajectory. To allow the skill to be deployed as quickly as possible, without relying on additional training trajectory generation, the robot should use its own prior movement data to compute the skill controller for the new task. In Track ROBO participants are asked to sequentially interact with a robotic arm in such a way that the tip of the robot follows a target process. This task comes with two layers of difficulty: (1) Instead of directly controlling the robot (e.g., via setting the torques of individual joints) participants can only set abstracted control variables. This should imitate a setting in which the robot dynamics are too complicated to explicitly write down. (2) The training data is comprised of different types of trajectories than the later target trajectories. This imitates a setting in which the robot needs to adjust to a new task given previous data of an old task.

Background on robotic arm Track ROBO is based on three simple robot arms: (1) A two joint rotational robot arm, (2) a three joint rotational robot arm, and (3) a prismatic robot arm. This means there are two types of joints, either rotational joints that produce a rotary motion around the joint or prismatic joints that produce a linear motion between links (see Figure 2). Each joint can be controlled by applying a voltage signal to a DC motor located in the joint. In the case of a rotational joint this creates a torque, while in

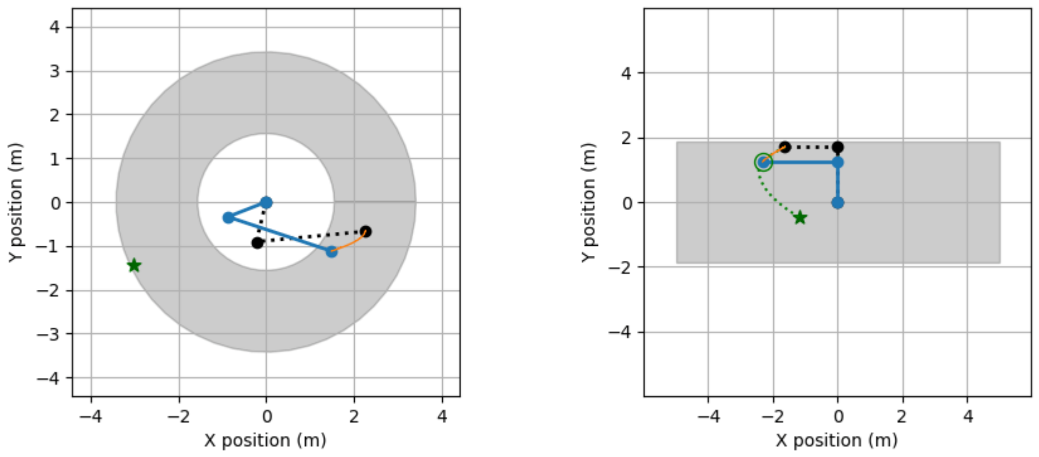

Figure 2: (Left) Rotational 2-link robot moving towards a target position (green star). (Right) Prismatic 2-link robot following a target trajectory (green dotted line) to a target position (green star). In both cases the gray area corresponds to the reachable workspace of the robot, the black dotted lines indicate the initial position of the robots and the orange lines mark the trail of previous positions of the robot tip.

the case of a prismatic joint it creates a linear acceleration. The resulting movement of the joints of the robot arm (and its tip in particular) are then governed by the physical laws of motion. Given a specific robot it is possible to derive exact differential equations that describe the robots' movement. This is known as the dynamics model of the robot and its parameters can depend on various specifications of the robot such as link mass, rotational moment of inertia, link length, location of center of math, and friction coefficients. Even though such a dynamics model is derived under idealized conditions and may not perfectly describe the motion of the robot, these models are often used to accurately control the robot, by inferring and refining some of the parameters based on observed movements of the robot.

This competition considers such a setting by only allowing to control the robots via abstract controls (details are given below), the precise dynamics model of the robot is unknown. This may happen, for example, for complex robots or if there is some complicated dependence structure between different joints.

Data generating process In Track ROBO, participants are able to sequentially interact with the robotic arm and need to learn how to control it to follow a provided target process (i.e., trajectory). Instead of allowing participants to directly interact with the system, we only give access to abstract controls of the robot. The controls to which the participants have access will be linked to the torques of the robot through a function which is unknown to the participants. No explicit dynamical model for the robot is available. Participants need to be able to learn, based on offline training data, how to use the abstract control variables to control the robot's movement. They are not able to directly interact with the robot during the training phase and need to submit a fully automated controller that is able to track unknown trajectories. More precisely, the robot dynamics are given by

$$\begin{aligned}(Z(0), W(0)) &= (z, w) \\ (\dot{Z}(t), \dot{W}(t)) &= F(Z(t), W(t), C(t)),\end{aligned} \tag{11}$$

where $Z(t) \in \mathbb{R}^2$ is the position of the tip of the robot, $W(t) \in \mathbb{R}^{2d}$ is the position of other joints of the robot (the dimension d depends on the underlying robot), $z \in \mathbb{R}^2$ and $w \in \mathbb{R}^{2d}$ are the initial values and $C(t) \in \mathbb{R}^q$ are the underlying robot controls (such as torques applied to each joint). The (unknown) function $F : \mathbb{R}^{2(d+1)+q} \to \mathbb{R}^{2(d+1)}$ specifies the true dynamics model of the robot. The robot can only be controlled on a linearly spaced discrete time grid $(t_0, t_1, \ldots, t_{200})$ with $t_0 = 0$ and $t_{200} = 2$, that is, for each time step $\ell \in \{0, \ldots, 199\}$ it holds that $C(t) \equiv \text{const}$ for all $t \in [t_\ell, t_{\ell+1})$. Furthermore, instead of setting the robot controls $C(t)$ directly, participants can only control abstract control variables $U(t) \in \mathbb{R}^p$ (on the same discrete time grid). These are related to true controls via an (unknown) function $G : \mathbb{R}^p \to \mathbb{R}^q$ given by

$$C(t) = G(U(t)).$$

The function G is called interface function. Such a setting becomes relevant whenever the underlying dynamics model (in this case F) is unknown and one can no longer rely on methods which make use of the explicit structure of the system. The competition is based on 24 different systems, each corresponding to a specific underlying robot and an interface function (i.e., $(F^1, G^1), \ldots, (F^{24}, G^{24})$). There are 3 different types of robots, a 2- and 3-link rotational and a 2-link prismatic planar robot, each with different dimensions d and q, and several different interface functions.

Goal The goal is to control the tip of the robot $Z(t)$ to follow a given target process $z_*(t)$ (see below for details). More specifically, participants should learn a controller that for each time step $\ell \in \{0, \ldots, 199\}$ takes as input the current positions $Z(t_\ell), W(t_\ell)$, their derivatives $\dot{Z}(t_\ell), \dot{W}(t_\ell)$, and the next target position $z_*(t_{\ell+1})$ and optimally sets the value of the abstract controls $U(t_\ell)$ to follow the target process. As the controller only gets access to the next time step and not the entire target trajectory, the task does not involve any planning. The control can use the information it gathers during the control process, however only under time and computational constraints (see below for details). Since the control variable $U(t)$ needs to be set in each step and feedback is available, this can be considered a 'closed-loop' problem.

Training Data The available training data for each system (F^i, G^i) consists of 50 repetitions each generated by controlling the robot using an LQR-controller [Anderson and Moore, 2007] and the true robot interface function to transition from a random starting position to a different target position. The target positions used for to generate the trainings trajectories are not meaningful for the task and are not provided to participants. For each of these repetitions, participants are provided with the observed processes W and Z, their derivatives $\dot{W}$ and $\dot{Z}$, a time indicator t, the applied controls U and an indicator i specifying the system. The training movements were generated by using an LQR-controller to move the tip of the robot arm to random positions (not contained in the training data) in the work space. All training data are provided as data frames saved in CSV-files (additional information on the exact data structure can be found in the online tutorial).

Evaluation Participants are asked to submit code with their controller to CodaLab (for details see the online tutorial and the starter kit). For each of the 24 systems $(F^1, G^1), \dots, (F^{24}, G^{24})$ the participants' submitted controller is used to follow 10 different target processes. More specifically, for each system $i \in \{1, \dots, 24\}$ and repetition $k \in \{1, \dots, 10\}$, there is a target process $z_*^{i,k} : [0, 2] \to \mathbb{R}^2$ and after, for each system and repetition, running the system using the participants' controller the following loss is computed[1]

$$J_{i,k} := b_{i,k} \cdot \int_0^2 ||Z^{i,k}(t) - z_*^{i,k}(t)||_2^2 dt + c_{i,k} \cdot \int_0^2 U^{i,k}(t)^\top U^{i,k}(t) dt, \tag{12}$$

where $b_{i,k}$ and $c_{i,k}$ are scaling constants which are selected such that $J_{i,k} = 100$ when no controls are applied and $J_{i,k} = 1$ if an oracle LQR-controller is used, that is, an LQR-controller using the true robot and interface function. If $J_{i,k}$ is smaller than 1, it is improving on the oracle LQR-controller; if it is larger than 100 this implies a worse performance than doing nothing. We clip all scores at 100 before averaging them. The scaling is to ensure that losses are comparable across each repetition. For the (preliminary) leader board, which is updated during the competition, only 12 systems are evaluated and the mean loss across those systems (and all corresponding repetitions) is shown on the leader board. For the final ranking the average loss across the 12 held-out systems (and all corresponding repetitions) is used. During an initial trial phase, only 3 out of 10 repetitions are used per each of the 12 non-held out systems.

Mimicking a real-time robot control scenario, the participants' computational resources on updating their controller (or policy), are restricted. More precisely, the runtime and computational resources are limited as follows:

- *Available compute:* 6 CPU cores (Intel Xeon Gold) and 24 GB RAM, no swap

[1]The integrals are approximated numerically.

- *Time constraint:* 16 minutes controller-only runtime for all systems and repetitions (this amounts to on average 8 seconds per trajectory, which is four times the "live time" of one trajectory which lasts 2 seconds) and the controller has to initialise within 60 seconds and provide the first control input, consecutive controls are required to arrive within less than 8 seconds each

If the time constraints are not met, the submission is not scored and returns an error.

2 Three points of view: RL, Control and Causality

We now provide a brief introduction to each of the three fields and a glossary of their terminology and notation (this part may be updated during the challenge).

2.1 Reinforcement learning

From a reinforcement learning (RL) perspective, the problems in Track CHEM and Track ROBO each can be formulated as a Markov Decision Process (MDP). An MDP consists of a tuple (S, A, R, T, γ) [Sutton and Barto, 1998]. S is the set of states of the system and A is a set of actions that the agent can execute. $R(s, a, s')$ is the reward function that expresses the immediate reward for executing action $a \in A$ in state $s \in S$ and then transitioning to the next state $s' \in S$. $T(s'|s, a)$ is the transition distribution which gives the distribution over next states s' given the current state s and action a. $\gamma \in [0, 1]$ is a discount factor that expresses the agent's preference for immediate rewards over long term rewards. To select an action, the agent applies a policy $\pi(a|s)$ that defines the distribution over the next action to execute a given the current state s. Policies can be stochastic or deterministic. The ith sampled transition thus results in a tuple (s_i, a_i, s'_i, r_i), where s_i is the current state, a_i is the sampled action, s'_i is the next state after the transition, and $r_i \in \mathbb{R}$ is the resulting scalar reward. The goal of learning is to acquire an optimal policy, often denoted as π^*, that will maximize the expected return $E_{s' \sim T, a \sim \pi} \left[\sum_t^T \gamma^t r_t \right]$ where T is the duration of the task [Bell et al., 1996].

For track CHEM, the vector u is selected at the start of each trial and then executed for a number of steps. The model fits into the RL framework in various ways. For example, we can model this task as an episodic task with a hierarchical policy. In this case, given the initial value $X(0)$ of a task i, we choose a parameter θ according to a high-level policy $\pi(\theta|X(0))$. This parameter is then used by the low-level policy $\pi_\theta(a|s)$, which is executed during the episode. In track CHEM, the low-level policy is given (and particularly simple, see Equation (9)), so the agent would only need to learn the high-level policy. This problem formulation is closely connected to bandit or contextual bandit problems [Sutton and Barto, 1998], wherein there are no state transitions and the agent simply receives a reward based on the selected action (and context if applicable). However, for the episodic formulation, the agent may still benefit from modeling the individual transition steps of the underlying

MDP. Track ROBO has a more standard MDP formulation wherein the action is defined by the command sent to the abstract controller and the state space is given by the joint positions and velocities. The time-varying reward is given by the robot's accuracy in following the desired trajectory.

Classical online RL approaches learn the policy by iteratively interacting with the environment and improving the policy. These policy updates can be performed after each sampled transition or after a batch of samples has been collected. This trial-and-error learning approach allows the agent to collect additional data for optimizing the policy and thus account for the data distribution shift caused by the policy being learned. However, for the proposed task, the agent will need to use *offline* reinforcement learning. Offline reinforcement learning (ORL), also known as batch reinforcement learning [Levine et al., 2020, Lange et al., 2012]. ORL still solves the same MDP problem of standard RL. However, given the fixed training dataset, the agent does not need to be concerned about exploring the state-action space for collecting samples. Instead, some ORL methods need to assume that sufficient exploration has already been performed and that the provided training dataset covers a space of high-reward transitions that will allow the agent to learn a suitable policy.

ORL incorporates a number of different approaches [Levine et al., 2020]. For example, importance sampling could be used to compute policy gradients from the given samples to update the policy parameters [Precup et al., 2000], or model-based RL could be used to compute the transition distribution T and reward function R from the offline data and then used to learn the policy π [Oh et al., 2015, Deisenroth and Rasmussen, 2011]. A key challenge of ORL methods is to avoid shifting too far away from the data collection distribution. The agent may observe overfitting, and hence poor performance at test time, if the state-action distribution induced by the learned policy is in a region where few samples were originally collected. To address this challenge ORL approaches often incorporate policy constraints that constrain the new policy to be close to the data collection policy [Fujimoto et al., 2019], e.g., a limit on the KL divergence between the policies, or a policy penalty that is added to the cost function to deter deviating from the sample distribution.

2.2 Control theory

From the point of view of control theory, the posed task in tracks CHEM and ROBO (details will be added later) can be best described as control design for (partially) unknown dynamical systems. The ODEs in (6) and (11) are commonly called the *dynamical systems* or *plants* that are to be controlled. More specifically, the model represents a *continuous-time, state-space model*, where $Z(t)$ is the time-dependent state of the systems (in control often denoted by $x(t)$), $U(t)$ is the control input (in control, typically $u(t)$), and z is the initial state. The system (6) is generally nonlinear in the state, and linear in the input $U(t)$, while system (11) is nonlinear in both the state and the input. Both tasks assume a linear *observation* or *measurement model* of the form

$$y(t) = \mathbf{C}Z(t) + N(t), \tag{13}$$

where $N(t)$ is measurement noise. For Track CHEM, we have $\mathbf{C} = I$, in (7) where I is the identity matrix (i.e., all states are measured). For Track ROBO, we have $\mathbf{C} = [I, 0]$ (i.e., only some of the states are measured). All measurements are corrupted by additive noise $N(t)$. Because both measurements and inputs are vector-valued, these dynamical systems are also called *multi-input multi-output* (MIMO) systems.

Both tasks seek a functional map from measurements $y(t)$ to control inputs $u(t)$, which is the classical task of *control design.* In controls language, this map is called the *controller*, *control law*, or *control algorithm.* While Track ROBO does not make any restriction a-priori on the functional form of the controller, Track CHEM restricts control inputs to be of the form (9), which can be understood as a type of *impulse control.* The goal for the control design is to minimize the losses (10) and (12), respectively. In controls terminology, such losses are often called *control costs.* Formalizing the control objective as an optimization problem is commonly known as *optimal control* (see, e.g., Bertsekas [2000], Anderson and Moore [2007]). The objectives of both tasks are common control costs in the sense that they represent (weighted) sums of two terms: the first one being based on the state or measurement and evaluating the *tracking* or *control performance*, and the second one being a function of the control input, thus representing *control effort* or sometimes (a generalized form of) *energy.* Having quadratic functions of the states and inputs is the most common cost in controls.

Control design is traditionally based on first principles models from laws of, for instance, chemistry (Track CHEM) or physics (Track ROBO). A key difficulty – and one core of this competition – is the lack of such a model of the dynamical systems (6). and (11). The task of estimating a model of a dynamical system from input-output data are known in controls under the term *system identification* (e.g., Ljung [1998]). Hence, the typical approach of control engineering for obtaining a controller for the task above would be:

1. Use problem insight to select a useful model structure (e.g., linear/nonlinear, auto-regressive, continuous-time/discrete-time, ...), often a parametric model structure;

2. Obtain a model of the dynamical system through system identification (e.g., fitting the parameters of the selected model structure);

3. Use the model in a control design method to obtain a control law (e.g., compute an optimal controller to minimize a quadratic cost function);

4. Test the controller on the system or in simulation; and

5. Possibly repeat the cycle if results are not satisfactory.

In slight deviation of the third item above, *model-predictive control* (e.g., Allgöwer and Zheng [2012], Borrelli et al. [2017]) – a popular modern control method – and similar optimization-based controller schemes, may not compute a control law explicitly, but use the model instead in an online optimization procedure. Identifying a system and obtaining a controller is a well-understood problem if one deals with linear dynamical systems. This is different for nonlinear systems. For nonlinear systems, many methods exist for specific problem settings, but they lack the generality of approaches for linear systems and there is a large space of problems which is still unsolved. For a more detailed introduction into system identification and controller design, we refer the reader to textbooks such as [Åström and Murray, 2021].

In recent years, researchers in control have been exploring various forms of incorporating data-based and machine learning approaches into control design, partly making the above pipeline more flexible. This emerging area between control and machine learning is known under the terms *learning-based control*, *learning control*, and *data-based control*. Refer to, for example, the series of *Learning-based Control* sessions [Trimpe et al., 2021] at CDC (the most important control conference) and the new conference on *Learning for Dynamics and Control* [Jadbabaie et al., 2021].

2.3 Causality

In classical statistics, a multivariate stochastic system is usually thought of as describing a single observational distribution. In contrast, a *causal* system describes a set of distributions that describe system behavior not only under passive observation, but also under *interventions* [Pearl, 2009, Spirtes et al., 2000]. That is, a causal model predicts how a system reacts under interventions. For dynamical systems, an intervention could fix a coordinate process at a certain value over an interval of time [see Peters et al., 2020, for an overview]. Usually, some type of *stability of mechanisms* is assumed that states that mechanisms that have not been intervened on remain the same as under the observational state of the system [Haavelmo, 1944, Aldrich, 1989]. The exact implementation of this principle depends on the type of system.

Consider again the predator-prey model introduced in Section 1.1. In the situation, described by this model, we can imagine intervening exogenously on the abundance of prey. If the prey and predators live in a controlled environment, we can imagine simply counting the number of prey animals (A) every day and adjusting it accordingly to keep it fixed. If the model is causal, one would often assume that reactions $A + B \xrightarrow{k_2} 2B$ and $B \xrightarrow{k_3} \emptyset$ remain the same after this intervention, i.e., that these mechanisms are stable under interventions on the reaction of the prey. We can then solve the new system of differential equations. In this sense a causal system does not only describe a single distribution of the observed data, but a set of distributions each corresponding to an intervention. In a stochastic system, there are usually many types of interventions that can be performed mathematically but in real-world applications only some of them may be realizable.

The tasks of Track CHEM and Track ROBO fit into the framework of causality. As an example, consider the system in Equation (6) from Track CHEM. The *observational regime* of the system corresponds to no intervention, in this case letting $U = 0$. The set of interventional regimes are parametrized by $\{Bu : u \in \mathcal{U}\}$ where $\mathcal{U}$ is a set of allowed control vectors. In this system, the intrinsic dynamics are the same regardless of which (if any) intervention is applied as they are described by the F function. That is, the mechanism described by this function is stable under interventions. In Track CHEM, each causal system is observed by the participant for 20 different (and known) interventions. The task is to use this information to compute interventions that will obtain a certain desired behavior when applied to the system.

It is common to assume that there is structural sparsity in a causal system. In terms of Equation (6), this would mean that only few coordinate processes enter the right-hand side of dX_j. This makes graphs useful as a means of visualizing a multivariate causal system. In such a graph, each node represents a coordinate process, Z_i. We include a *directed edge*, $Z_i \to Z_j$, $i \neq j$, if the right-hand side of $(dZ(t))_j = F(Z(t))_j$ is not constant in Z_i and in this case we say that Z_i is a *(causal) parent* of Z_j and that Z_j is a *child* of Z_i. If Z_k is a parent of Z_i and of Z_j, then we say that Z_k is a *confounder* of the processes Z_i and Z_j. Assuming that the causal mechanisms are stable, such a graph may also describe the sparsity structure of the system under interventions.

Often we will only be able to observe parts of a system and there may be unobserved coordinate processes that interact with the observed ones. Ignoring this may make it impossible to predict what will happen under interventions. As an example, consider again the predator-prey model. It may be that we believe the posed model to be accurate because we observe that more prey leads to more predators. However, imagine there is an unobserved prey species, C, which is actually required and sufficient for reproduction of both species A and B (that is, in reality, A is not the prey of B). In this case, our faulty model of reality would make us believe that intervening to increase the abundance of species A will increase the abundance of species B while this could actually deplete their common prey and have the opposite effect. In the case, species C would be a confounder of the relationship between species A and B.

References

J. Aldrich. Autonomy. *Oxford Economic Papers*, 41:15–34, 1989.

F. Allgöwer and A. Zheng. *Nonlinear model predictive control*, volume 26. Birkhäuser, 2012.

B. D. O. Anderson and J. B. Moore. *Optimal control: linear quadratic methods.* Courier Corporation, 2007.

K. J. Åström and R. M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers, Second Edition.* Princeton University Press, 2021.

D. Bell, J. Kay, and J. Malley. A non-parametric approach to non-linear causality testing. *Economics Letters*, 51(1):7–18, 1996.

D. P. Bertsekas. *Dynamic programming and optimal control: Vol. 1.* Athena scientific, 2000.

F. Borrelli, A. Bemporad, and M. Morari. *Predictive control for linear and hybrid systems.* Cambridge University Press, 2017.

Marc Peter Deisenroth and Carl Edward Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *ICML*, page 465–472, 2011.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2052–2062, 09–15 Jun 2019.

T. Haavelmo. The probability approach in econometrics. *Econometrica*, 12:S1–S115 (supplement), 1944.

A. Jadbabaie, J. Lygeros, G. Pappas, P. Parrilo, B. Recht, D. Scaramuzza, C. Tomlin, and M. Zeilinger. Annual Learning for Dynamics & Control Conference (L4DC). `https://l4dc.ethz.ch/`, 2021.

Sascha Lange, Thomas Gabel, and Martin Riedmiller. *Batch Reinforcement Learning*, pages 45–73. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-27645-3. doi: 10.1007/978-3-642-27645-3_2. URL `https://doi.org/10.1007/978-3-642-27645-3_2`.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *CoRR*, abs/2005.01643, 2020.

L. Ljung. *System Identification: Theory for the User.* Pearson Education, 1998.

A. J. Lotka. Contribution to the theory of periodic reactions. *The Journal of Physical Chemistry*, 14(3):271–274, 1909.

Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard Lewis, and Satinder Singh. Action-conditional video prediction using deep networks in atari games. In *Neurips*, page 2863–2871, 2015.

J. Pearl. *Causality: Models, Reasoning, and Inference.* Cambridge University Press, New York, USA, 2nd edition, 2009.

J. Peters, S. Bauer, and N. Pfister. Causal models for dynamical systems. *ArXiv e-prints (2001.06208)*, 2020.

Doina Precup, Richard S. Sutton, and Satinder P. Singh. Eligibility traces for off-policy policy evaluation. In *ICML*, page 759–766, 2000.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search.* MIT Press, 2nd edition, 2000.

R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, MA, 1998.

S. Trimpe, A.P. Schöllig, M. Zeilinger, and M.A. Müller. Invited session series on learning-based control. `https://idsc.ethz.ch/research-zeilinger/events/learning-control-session.html`, 2021.

P. Waage and C. M. Guldberg. Studier over affiniteten (in Danish). *Forhandlinger i Videnskabs-selskabet i Christiania*, pages 35–45, 1864.

D. J. Wilkinson. *Stochastic modelling for systems biology.* Chapman and Hall/CRC mathematical and computational biology series. Chapman & Hall/CRC, 2006.