An Optimal Control Approach to Sequential Machine Teaching

Laurent Lessard
University of Wisconsin–Madison

Xuezhou Zhang
University of Wisconsin–Madison

Xiaojin Zhu
University of Wisconsin–Madison

Abstract

Given a sequential learning algorithm and a target model, sequential machine teaching aims to find the shortest training sequence to drive the learning algorithm to the target model. We present the first principled way to find such shortest training sequences. Our key insight is to formulate sequential machine teaching as a time-optimal control problem. This allows us to solve sequential teaching by leveraging key theoretical and computational tools developed over the past 60 years in the optimal control community. Specifically, we study the Pontryagin Maximum Principle, which yields a necessary condition for optimality of a training sequence. We present analytic, structural, and numerical implications of this approach on a case study with a least-squares loss function and gradient descent learner. We compute and visualize the optimal training sequences for this problem, and find that they can vastly outperform the best available heuristics for generating training sequences.

1 INTRODUCTION

Machine teaching studies optimal control on machine learners (Zhu et al., 2018; Zhu, 2015). In controls language the plant is the learner, the state is the model estimate, and the input is the (not necessarily *i.i.d.*) training data. The controller wants to use the least number of training items—a concept known as the teaching dimension (Goldman and Kearns, 1995)—to force the learner to learn a target model. For example, in adversarial learning, an attacker may minimally poison the training data to force a learner to learn a nefarious model (Biggio et al., 2012; Mei and Zhu, 2015). Conversely, a defender may immunize the learner by injecting adversarial training examples into the training data (Goodfellow et al., 2014). In education systems, a teacher may optimize the training curriculum to enhance student (modeled as a learning algorithm) learning (Sen et al., 2018; Patil et al., 2014).

Machine teaching problems are either batch or sequential depending on the learner. The majority of prior work studied batch machine teaching, where the controller performs one-step control by giving the batch learner an input training *set*. Modern machine learning, however, extensively employs sequential learning algorithms. We thus study sequential machine teaching: what is the shortest training sequence to force a learner to go from an initial model $\mathbf{w}_0$ to some target model $\mathbf{w}_\star$? Formally, at time $t = 0, 1, \ldots$ the controller chooses input $(\mathbf{x}_t, y_t)$ from an input set $\mathcal{U}$. The learner then updates the model according to its learning algorithm. This forms a dynamical system f:

$$\mathbf{w}_{t+1} = f(\mathbf{w}_t, \mathbf{x}_t, y_t). \tag{1a}$$

The controller has full knowledge of $\mathbf{w}_0, \mathbf{w}_\star, f, \mathcal{U}$, and wants to minimize the terminal time T subject to $\mathbf{w}_T = \mathbf{w}_\star$.

As a concrete example, we focus on teaching a gradient descent learner of least squares:

$$f(\mathbf{w}_t, \mathbf{x}_t, y_t) = \mathbf{w}_t - \eta(\mathbf{w}_t^\mathsf{T}\mathbf{x}_t - y_t)\mathbf{x}_t \tag{1b}$$

with $\mathbf{w} \in \mathbb{R}^n$ and the input set $\|\mathbf{x}\| \le R_x, |y| \le R_y$. We caution the reader not to trivialize the problem: (1) is a *nonlinear* dynamical system due to the interaction between $\mathbf{w}_t$ and $\mathbf{x}_t$. A previous best attempt to solve this control problem by Liu et al. (2017) employs a greedy control policy, which at step t optimizes $\mathbf{x}_t, y_t$ to minimize the distance between $\mathbf{w}_{t+1}$ and $\mathbf{w}_\star$. One of our observations is that this greedy policy can be substantially suboptimal. Figure 1 shows three teaching problems and the number of steps T to arrive at $\mathbf{w}_\star$ using different methods. Our optimal control method formulated as Nonlinear Programming (NLP) found shorter teaching sequences compared to the greedy policy (lengths 151, 153, 259 for NLP vs 219, 241, 310 for GREEDY, respectively). This and other experiments are discussed in Section 4.

Proceedings of the 22[nd] International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89.

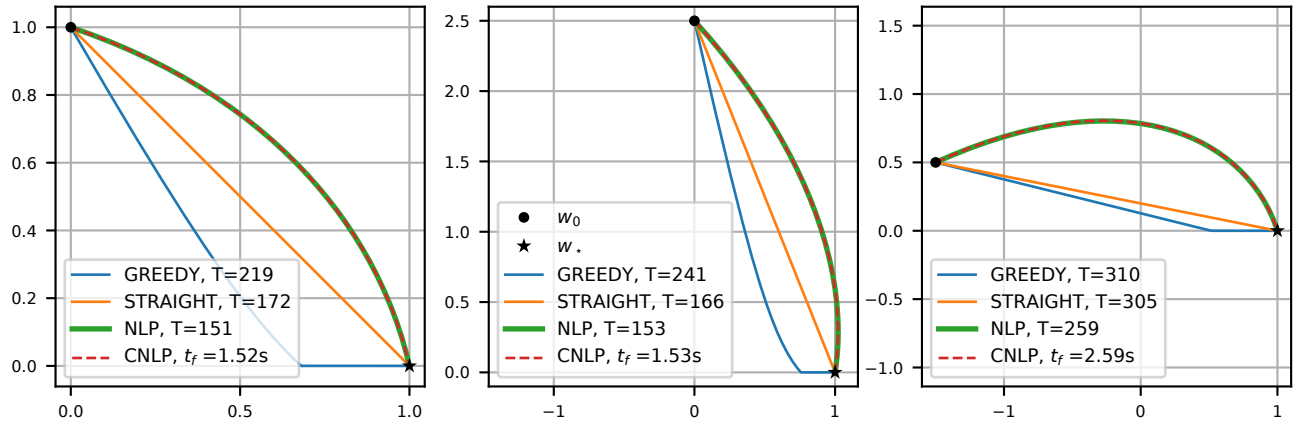

Figure 1: The shortest teaching trajectories found by different methods. All teaching tasks use the terminal point $\mathbf{w}_\star = (1, 0)$. The initial points used are $\mathbf{w}_0 = (0, 1)$ (left panel), $\mathbf{w}_0 = (0, 2.5)$ (middle panel), and $\mathbf{w}_0 = (-1.5, 0.5)$ (right panel). The learner is the least squares gradient descent algorithm (1) with $\eta = 0.01$ and $R_x = R_y = 1$. Total steps T to arrive at $\mathbf{w}_\star$ is indicated in the legends.

1.1 Main Contributions

Our main contribution is to show how tools from optimal control theory may be brought to bear on the machine teaching problem. Specifically, we show that:

1. The Pontryagin optimality conditions reveal rich structural properties of the optimal teaching sequences. Specifically, we provide a structural characterization of solutions and show that, in the least-squares case (1), the optimal solution always lies in a 2D subspace. These results are detailed in Section 3.
2. Optimal teaching sequences can be vastly more efficient than what may be obtained via common heuristics. We present two optimal approaches: an exact method (NLP) and a continuous approximation (CNLP). Both agree when the stepsize η is small, but CNLP is more scalable because its runtime does not depend on the length of the training sequence. These results are shown in Section 4.

We begin with a survey of the relevant optimal control theory and algorithms literature in Section 2.

2 TIME-OPTIMAL CONTROL

To study the structure of optimal control we consider the continuous *gradient flow* approximation of gradient descent, which holds in the limit of diminishing step size, i.e. $\eta \to 0$. In this section, we present the corresponding canonical time-optimal control problem and summarize some of the key theoretical and computational tools in optimal control that address it. For a more detailed exposition on the theory, we refer the reader to modern references on the topic (Kirk, 2012; Liberzon, 2011; Athans and Falb, 2013).

This section is self-contained and we will use notation consistent with the control literature (x instead of $\mathbf{w}$, u instead of $(\mathbf{x}, y)$, t_f instead of T). We revert back to machine learning notation in section 3. Consider the following boundary value problem:

$$\dot{x} = f(x, u) \qquad \text{with } x(0) = x_0 \text{ and } x(t_f) = x_f. \quad (2)$$

The function $x : \mathbb{R}_+ \to \mathbb{R}^n$ is called the *state* and $u : \mathbb{R}_+ \to \mathcal{U}$ is called the *input*. Here, $\mathcal{U} \subseteq \mathbb{R}^m$ is a given constraint set that characterizes admissible inputs. The initial and terminal states x_0 and x_f are fixed, but the terminal time t_f is free. If an admissible u together with a state x satisfy the boundary value problem (2) for some choice of t_f, we call (x, u) a *trajectory* of the system. The objective in a time-optimal control problem is to find an *optimal trajectory*, which is a trajectory that has minimal t_f.

Established approaches for solving time-optimal control problems can be grouped in three broad categories: dynamic programming, indirect methods, and direct methods. We now summarize each approach.

2.1 Dynamic Programming

Consider the value function $V : \mathbb{R}^n \to \mathbb{R}_+$, where $V(x)$ is the minimum time required to reach x_f starting at the initial state x. The Hamilton–Jacobi–Bellman (HJB) equation gives necessary and sufficient conditions for optimality and takes the form:

$$\min_{\tilde{u} \in \mathcal{U}} \ \nabla V(x)^\mathsf{T} f(x, \tilde{u}) + 1 = 0 \qquad \text{for all } x \in \mathbb{R}^n \quad (3)$$

together with the boundary condition $V(x_f) = 0$. If the solution to this differential equation is $V_\star$, then the optimal input is given by the minimizer:

$$u(x) \in \arg\min_{\tilde{u} \in \mathcal{U}} \ \nabla V_\star(x)^\mathsf{T} f(x, \tilde{u}) \quad \text{for all } x \in \mathbb{R}^n \quad (4)$$

A nice feature of this solution is that the optimal input u depends on the current state x. In other words, HJB produces an optimal *feedback policy*.

Unfortunately, the HJB equation (3) is generally difficult to solve. Even if the minimization has a closed form solution, the resulting differential equation is often intractable. We remark that the optimal $V_\star$ may not be differentiable. For this reason, one looks for so-called *viscosity solutions*, as described by Liberzon (2011); Tonon et al. (2017) and references therein.

Numerical approaches for solving HJB include the fast-marching method (Tsitsiklis, 1995) and Lax–Friedrichs sweeping (Kao et al., 2004). The latter reference also contains a detailed survey of other numerical schemes.

2.2 Indirect Methods

Also known as "optimize then discretize", indirect approaches start with necessary conditions for optimality obtained via the Pontryagin Maximum Principle (PMP). The PMP may be stated and proved in several different ways, most notably using the Hamiltonian formalism from physics or using the calculus of variations. Here is a formal statement.

Theorem 2.1 (PMP)**.** *Consider the boundary value problem* (2) *where f and its Jacobian with respect to x are continuous on $\mathbb{R}^n \times \mathcal{U}$. Define the Hamiltonian $H : \mathbb{R}^n \times \mathbb{R}^n \times \mathcal{U} \to \mathbb{R}$ as $H(x,p,u) := p^\mathsf{T} f(x,u) + 1$. If $(x^\star, u^\star)$ is an optimal trajectory, then there exists some function $p^\star : \mathbb{R}_+ \to \mathbb{R}^n$ (called the "co-state") such that the following conditions hold.*

a) $x^\star$ and $p^\star$ satisfy the following system of differential equations for $t \in [0, t_f]$ with boundary conditions $x^\star(0) = x_0$ and $x^\star(t_f) = x_f$.

$$\dot{x}^\star(t) = \frac{\partial H}{\partial p}\big(x^\star(t), p^\star(t), u^\star(t)\big), \tag{5a}$$

$$\dot{p}^\star(t) = -\frac{\partial H}{\partial x}\big(x^\star(t), p^\star(t), u^\star(t)\big). \tag{5b}$$

b) For all $t \in [0, t_f]$, an optimal input $u^\star(t)$ satisfies:

$$u^\star(t) \in \operatorname*{arg\,min}_{\tilde{u} \in \mathcal{U}} \; H(x^\star(t), p^\star(t), \tilde{u}). \tag{6}$$

c) Zero Hamiltonian along optimal trajectories:

$$H(x^\star(t), p^\star(t), u^\star(t)) = 0 \quad \text{for all } t \in [0, t_f]. \tag{7}$$

In comparison to HJB, which needs to be solved for all $x \in \mathbb{R}^n$, the PMP only applies along optimal trajectories. Although the differential equations (5) may still be difficult to solve, they are simpler than the HJB equation and therefore tend to be more amenable to both analytical and numerical approaches. Solutions to HJB and PMP are related via $\nabla V^\star(x^\star(t)) = p^\star(t)$. PMP is only necessary for optimality, so solutions of (5)–(7) are not necessarily optimal. Moreover, PMP does not produce a feedback policy; it only produces optimal trajectory *candidates*. Nevertheless, PMP can provide useful insight, as we will explore in Section 3.

If PMP cannot be solved analytically, a common numerical approach is the *shooting method*, where we guess $p^\star(0)$, propagate the equations (5)–(6) forward via numerical integration. Then $p^\star(0)$ is refined and the process is repeated until the trajectory reaches x_f.

2.3 Direct Methods

Also known as "discretize then optimize", a sparse nonlinear program is solved, where the variables are the state and input evaluated at a discrete set of timepoints. An example is *collocation methods*, which use different basis functions such as piecewise polynomials to interpolate the state between timepoints. For contemporary surveys of direct and indirect numerical approaches, see Rao (2009); Betts (2010).

If the dynamics are already discrete as in (1), we may directly formulate a nonlinear program. We refer to this approach as NLP. Alternatively, we can take the continuous limit and then discretize, which we call CNLP. We discuss the advantages and disadvantages of both approaches in Section 4.

3 TEACHING LEAST SQUARES: INSIGHT FROM PONTRYAGIN

In this section, we specialize time-optimal control to least squares. To recap, our goal is to find the minimum number of steps T such that there exists a control sequence $(\mathbf{x}_t, y_t)_{0:T-1}$ that drives the learner (1) with initial state $\mathbf{w}_0$ to the target state $\mathbf{w}_\star$. The constraint set is $\mathcal{U} = \{(\mathbf{x}, y) \mid \|\mathbf{x}\| \le R_x,\ |y| \le R_y\}$. This is an *nonlinear discrete-time time-optimal control problem*, for which no closed-form solution is available.

On the corresponding continuous-time control problem, applying Theorem 2.1 we obtain the following necessary conditions for optimality[1] for all $t \in [0, t_f]$.

$$\mathbf{w}(0) = \mathbf{w}_0, \quad \mathbf{w}(t_f) = \mathbf{w}_\star \tag{8a}$$

$$\dot{\mathbf{w}}(t) = \big(y(t) - \mathbf{w}(t)^\mathsf{T}\mathbf{x}(t)\big)\,\mathbf{x}(t) \tag{8b}$$

$$\dot{\mathbf{p}}(t) = \big(\mathbf{p}(t)^\mathsf{T}\mathbf{x}(t)\big)\,\mathbf{x}(t) \tag{8c}$$

$$\mathbf{x}(t), y(t) \in \operatorname*{arg\,min}_{\|\hat{\mathbf{x}}\| \le R_x,\, |\hat{y}| \le R_y} \big(\hat{y} - \mathbf{w}(t)^\mathsf{T}\hat{\mathbf{x}}\big)\big(\mathbf{p}(t)^\mathsf{T}\hat{\mathbf{x}}\big) \tag{8d}$$

$$0 = \big(y(t) - \mathbf{w}(t)^\mathsf{T}\mathbf{x}(t)\big)\big(\mathbf{p}(t)^\mathsf{T}\mathbf{x}(t)\big) + 1 \tag{8e}$$

[1] State, co-state, and input in Theorem 2.1 are (x, p, u), which is conventional controls notation. For this problem, we use $(\mathbf{w}, \mathbf{p}, (\mathbf{x}, y))$, which is machine learning notation.

We can simplify (8) by setting $y(t) = R_y$, as described in Proposition 3.1 below.

Proposition 3.1. *For any trajectory $(\mathbf{w}, \mathbf{p}, \mathbf{x}, y)$ satisfying (8), there exist another trajectory of the form $(\mathbf{w}, \mathbf{p}, \tilde{\mathbf{x}}, R_y)$. So we may set $y(t) = R_y$ without any loss of generality.*

Proof. Since (8d) is linear in $\hat{y}$, the optimal $\hat{y}$ occurs at a boundary and $\hat{y} = \pm R_y$. Changing the sign of $\hat{y}$ is equivalent to changing the sign of $\hat{\mathbf{x}}$, so we may assume without loss of generality that $\hat{y} = R_y$. These changes leave (8b)–(8c) and (8e) unchanged so $\mathbf{w}$ and $\mathbf{p}$ are unchanged as well. ∎

In fact, Proposition 3.1 holds if we consider trajectories of (1) as well. For a proof, see the appendix.

Applying Proposition 3.1, the conditions (8d) and (8e) may be combined to yield the following quadratically constrained quadratic program (QCQP) equation.

$$\min_{\|\mathbf{x}\| \le R_x} (R_y - \mathbf{w}^\mathsf{T}\mathbf{x})(\mathbf{p}^\mathsf{T}\mathbf{x}) = -1 \tag{9}$$

where we have omitted the explicit time specification (t) for clarity. Note that (9) constrains the possible tuples $(\mathbf{w}, \mathbf{p}, \mathbf{x})$ that can occur as part of an optimal trajectory. So in addition to solving the left-hand side to find $\mathbf{x}$, we must also ensure that it's equal to -1. We will now characterize the solutions of (9) by examining five distinct regimes of the solution space that depend on the relationship between $\mathbf{w}$ and $\mathbf{p}$ as well as which regime transitions are admissible.

Regime I (Origin): $\mathbf{w} = 0$ and $\mathbf{p} \neq 0$. This regime happens when the teaching trajectory pass through the origin. In this regime, one can obtain closed-form solutions. In particular, $\mathbf{x} = -\frac{R_x}{\|\mathbf{p}\|}\mathbf{p}$ and $\|\mathbf{p}\| = \frac{1}{R_x R_y}$. In this regime, both $\dot{\mathbf{w}}$ and $\dot{\mathbf{p}}$ are positively aligned with $\mathbf{p}$. Therefore, Regime I necessarily *transitions* from Regime II and into Regime III, given that it is not at the beginning or the end of the teaching trajectory.

Regime II (positive alignment): $\mathbf{w} = \alpha\mathbf{p}$ with $\mathbf{p} \neq 0$ and $\alpha > 0$. This regime happens when $\mathbf{w}$ and $\mathbf{p}$ are positively aligned. Again we have closed form solutions. In particular, $\mathbf{x}^\star = -\frac{R_x}{\|\mathbf{w}\|}\mathbf{w}$ and $\alpha = R_x\|\mathbf{w}\|(R_y + R_x\|\mathbf{w}\|)$. In this regime, both $\dot{\mathbf{w}}$ and $\dot{\mathbf{p}}$ are negatively aligned with $\mathbf{w}$, thus Regime II necessarily transitions into Regime I and can never transition from any other regimes.

Regime III (negative alignment inside the origin-centered ball): $\mathbf{w} = -\alpha\mathbf{p}$ with $\mathbf{p} \neq 0$ and $\alpha > 0$ and $\|\mathbf{w}\| \le \frac{R_y}{2R_x}$. This regime happens when $\mathbf{w}$ and $\mathbf{p}$ are negatively aligned and $\mathbf{w}$ is inside the ball centered at the origin with radius $R = \frac{R_y}{2R_x}$. Again, closed form solutions exists: $\mathbf{x}^\star = \frac{R_x}{\|\mathbf{w}\|}\mathbf{w}$ and $\alpha = R\|\mathbf{w}\|(1 - R\|\mathbf{w}\|)$. Regime III necessarily transitions from Regime I and into Regime IV.

Regime IV (negative alignment out of the origin-centered ball): $\mathbf{w} = -\alpha\mathbf{p}$ with $\mathbf{p} \neq 0$ and $\alpha > 0$ and $\|\mathbf{w}\| > \frac{R_y}{2R_x}$. In this case, the solutions satisfies $\alpha = \frac{R_y^2}{4}$ so that $\mathbf{p}$ is uniquely determined by $\mathbf{w}$. However, the optimal $\mathbf{x}^\star$ is **not** unique. Any solution to $\mathbf{w}^\mathsf{T}\mathbf{x} = \frac{R_y}{2}$ with $\|\mathbf{x}\| \le R_x$ can be chosen. Regime IV can only transition from Regime III and cannot transition into any other regime. In other word, once the teaching trajectory enters Regime IV, it cannot escape. Another interesting property of Regime IV is that we know exactly how fast the norm of $\mathbf{w}$ is changing. In particular, knowing $\mathbf{w}^\mathsf{T}\mathbf{x} = \frac{R_y}{2}$, one can derive that $\frac{\mathrm{d}\|\mathbf{w}\|^2}{\mathrm{d}t} = \frac{R_y^2}{2}$. As a result, once the trajectory enters regime IV, we know exact how long it will take for the trajectory to reach $\mathbf{w}_\star$, if it is able to reach it.

Regime V (general positions): w and p are linearly independent. This case covers the remaining possibilities for the state and co-state variables. To characterize the solutions in this regime, we'll first introduce some new coordinates. Define $\{\hat{\mathbf{w}}, \hat{\mathbf{u}}\}$ to be the orthonormal basis for span$\{\mathbf{w}, \mathbf{p}\}$ such that $\mathbf{w} = \gamma\hat{\mathbf{w}}$ and $\mathbf{p} = \alpha\hat{\mathbf{w}} + \beta\hat{\mathbf{u}}$ for some $\alpha, \beta, \gamma \in \mathbb{R}$. Note that $\beta \neq 0$ because we assume $\mathbf{w}$ and $\mathbf{p}$ are assumed to be linearly independent in this regime. We can therefore express any input uniquely as $\mathbf{x} = w\hat{\mathbf{w}} + u\hat{\mathbf{u}} + z\hat{\mathbf{z}}$ where $\hat{\mathbf{z}}$ is an *out-of-plane* unit vector orthogonal to both $\hat{\mathbf{w}}$ and $\hat{\mathbf{u}}$, and $w, u, z \in \mathbb{R}$ are suitably chosen. Substituting these definitions, (9) becomes

$$\min_{w^2+u^2+z^2 \le R_x^2} \quad (R_y - \gamma w)(\alpha w + \beta u) = -1. \tag{10}$$

Now observe that the objective is linear in u and does not depend on z. The objective is linear in u because $\beta \neq 0$ and $(1 - \gamma w) \neq 0$ otherwise the entire objective would be zero. Since the feasible set is convex, the optimal u must occur at the boundary of the feasible set of variables w and u. Therefore, $z = 0$. This is profound, because it implies that in Regime V, the optimal solution necessarily lies on the 2D plane span$\{\mathbf{w}, \mathbf{p}\}$. In light of this fact, we can pick a more convenient parametrization. Let $w = R_x \cos\theta$ and $u = R_x \sin\theta$. Equation (10) becomes:

$$\min_{\theta} \quad R_x(R_y - \gamma R_x \cos\theta)(\alpha\cos\theta + \beta\sin\theta) = -1. \tag{11}$$

This objective function has at most four critical points, of which there is only one global minimum, and we can find it numerically. Last but not least, Regime V does not transition from or into any other Regime.

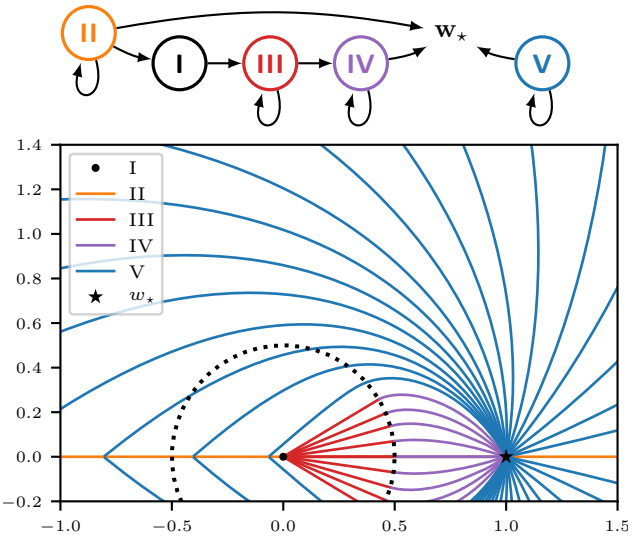

Figure 2: Optimal trajectories for $\mathbf{w}_\star = (1, 0)$ for different choices of $\mathbf{w}_0$. Trajectories are colored according to the regime to which they belong and the directed graph above shows all possible transitions. The optimal trajectories are symmetric about the x-axis. For implementation details, see Section 4.

Intrinsic low-dimensional structure of the optimal control solution. As is hinted in the analysis of Regime V, the optimal control $\mathbf{x}$ sometimes lies in the 2D subspace spanned by $\mathbf{w}$ and $\mathbf{p}$. In fact, this holds not only for Regime V but for the whole problem. In particular, we make the following observation.

Theorem 3.2. *There always exists a global optimal trajectory of* (8) *that lies in a 2D subspace of* $\mathbb{R}^n$.

The detailed proof can be found in the appendix. An immediate consequence of Theorem 3.2 is that if $\mathbf{w}_0$ and $\mathbf{w}_\star$ are linearly independent, we only need to consider trajectories that are confined to the subspace $\text{span}\{\mathbf{w}_0, \mathbf{w}_\star\}$. When $\mathbf{w}_0$ and $\mathbf{w}_\star$ are aligned, trajectories are still 2D, and any subspace containing $\mathbf{w}_0$ and $\mathbf{w}_\star$ is equivalent and arbitrary choice can be made.

This insight is extremely important because it enables us to restrict our attention to 2D trajectories even though the dimensionality of the original problem (n) may be huge. This allows us to not only obtain a more elegant and accurate solution in solving the necessary condition induced by PMP, but also to parametrize direct and indirect approaches (see Sections 2.2 and 2.3) to solve this intrinsically 2D problem more efficiently.

Multiplicity of Solution Candidates. The PMP conditions are only *necessary* for optimality. Therefore, the optimality conditions (8) need not have a unique solution. We illustrate this phenomenon in Figure 3. We used a shooting approach (Section 2.2) to propagate different choices of $\mathbf{p}^\star(0)$ forward in time. It turns out two choices lead to trajectories that end at $\mathbf{w}_\star$, and they do not have equal total times. So in general, PMP identifies *optimal trajectory candidates*, which can be thought of as local minima for this highly nonlinear optimization problem.

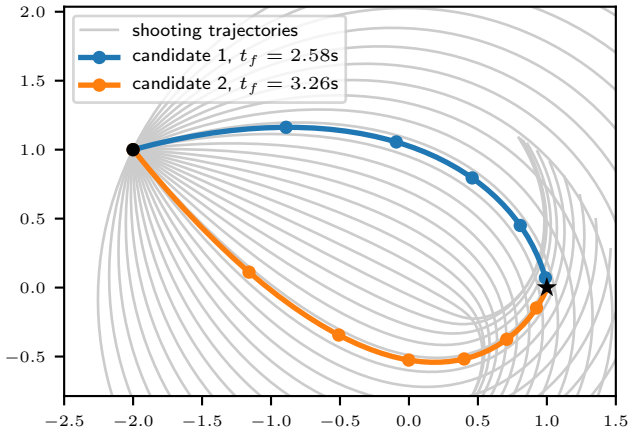

Figure 3: Trajectories found using a shooting approach (Section 2.2) with $\mathbf{w}_0 = (-2, 1)$ and $\mathbf{w}_\star = (1, 0)$. Gray curves show different shooting trajectories while the blue and orange curves show two trajectories that satisfy the necessary conditions for optimality (8). Markers show intervals of 0.5 seconds, which is roughly 50 steps when using a stepsize of $\eta = 0.01$.

4 NUMERICAL METHODS

While the PMP yields necessary conditions for time-optimal control as detailed in Section 3, there is no closed-form solution in general. We now present and discuss four numerical methods: CNLP and NLP are different implementations of time-optimal control, while GREEDY and STRAIGHT are heuristics.

CNLP: This approach solves the continuous gradient flow limit of the machine teaching problem using a direct approach (Section 2.3). Specifically, we used the NLOptControl package (Febbo, 2017), which is an implementation of the *hp*-pseudospectral method GPOPS-II (Patterson and Rao, 2014) written in the Julia programming language using the JuMP modeling language (Dunning et al., 2017) and the IPOPT interior-point solver (Wächter and Biegler, 2006). The main tuning parameters for this software are the integration scheme and the number of mesh points. We selected the trapezoidal integration rule with 100 mesh points for most simulations. We used CNLP to produce the trajectories in Figures 1 and 2.

NLP: A naïve approach to optimal control is to find the minimum T for which there is a feasible input sequence to drive the learner to $\mathbf{w}_\star$. Fixing T, the feasibility subproblem is a nonlinear program over $2T$ n-dimensional variables $\mathbf{x}_0, \ldots, \mathbf{x}_{T-1}$ and $\mathbf{w}_1, \ldots, \mathbf{w}_T$ constrained by learner dynamics. Recall $\mathbf{w}_0$ is given, and one can fix $y_t = R_y$ for all t by Proposition 3.1. For our learner (1), the feasibility problem is

$$\begin{aligned} \min_{\mathbf{w}_{1:T},\, \mathbf{x}_{0:T-1}} \quad & 0 \\ \text{s.t.} \quad & \mathbf{w}_T = \mathbf{w}_\star \\ & \mathbf{w}_{t+1} = \mathbf{w}_t - \eta(\mathbf{w}_t^\mathsf{T}\mathbf{x}_t - R_y)\mathbf{x}_t \\ & \|\mathbf{x}_t\| \le R_x, \quad \forall t = 0, \ldots, T-1. \end{aligned} \tag{12}$$

As in the CNLP case, we modeled and solved the subproblems (12) using JuMP and IPOPT. We also tried Knitro, a state-of-the-art commercial solver (Byrd et al., 2006), and it produced similar results. We stress that such feasibility problems are difficult; IPOPT and Knitro can handle moderately sized T. For our specific learner (1) there are 2D optimal control and state trajectories in $\text{span}\{\mathbf{w}_0, \mathbf{w}_\star\}$ as discussed in Section 3. Therefore, we reparameterized (12) to work in 2D.

On top of this, we run a binary search over positive integers to find the minimum T for which the subproblem (12) is feasible. Subject to solver numerical stability, the minimum T and its feasibility solution $\mathbf{x}_0, \ldots, \mathbf{x}_{T-1}$ is the time-optimal control. While NLP is conceptually simple and correct, it requires solving many subproblems with $2T$ variables and $2T$ constraints, making it less stable and scalable than CNLP.

GREEDY: We restate the greedy control policy initially proposed by Liu et al. (2017). It has the advantage of being computationally more efficient and readily applicable to different learning algorithms (i.e. dynamics). Specifically for the least squares learner (1) and given the current state $\mathbf{w}_t$, GREEDY solves the following optimization problem to determine the next teaching example $(\mathbf{x}_t, y_t)$:

$$\begin{aligned} \min_{(\mathbf{x}_t, y_t) \in \mathcal{U}} \quad & \|\mathbf{w}_{t+1} - \mathbf{w}_\star\|^2 \\ \text{s.t.} \quad & \mathbf{w}_{t+1} = \mathbf{w}_t - \eta(\mathbf{w}_t^\mathsf{T}\mathbf{x}_t - y_t)\mathbf{x}_t. \end{aligned} \tag{13}$$

The procedure repeats until $\mathbf{w}_{t+1} = \mathbf{w}_\star$. We used the MATLAB function `fmincon` to solve the above quadratic program iteratively. We point out that the optimization problem is not convex. Moreover, $\mathbf{w}_{t+1}$ does not necessarily point in the direction of $\mathbf{w}_\star$. This is evident in Figure 1 and Figure 6.

STRAIGHT: We describe an intuitive control policy: at each step, move $\mathbf{w}$ in straight line toward $\mathbf{w}_\star$ as far as possible subject to the constraint $\mathcal{U}$. This policy is less greedy than GREEDY because it may not reduce $\|\mathbf{w}_{t+1} - \mathbf{w}_\star\|^2$ as much at each step. The per-step optimization in $\mathbf{x}$ is a 1D line search:

$$\begin{aligned} \min_{a, y_t \in \mathbb{R}} \quad & \|\mathbf{w}_{t+1} - \mathbf{w}_\star\|^2 \\ \text{s.t.} \quad & \mathbf{x}_t = a(\mathbf{w}_\star - \mathbf{w}_t)/\|\mathbf{w}_\star - \mathbf{w}_t\| \\ & (\mathbf{x}_t, y_t) \in \mathcal{U} \\ & \mathbf{w}_{t+1} = \mathbf{w}_t - \eta(\mathbf{w}_t^\mathsf{T}\mathbf{x}_t - y_t)\mathbf{x}_t. \end{aligned} \tag{14}$$

The line search (14) can be solved in closed-form. In particular, one can obtain that

$$a = \begin{cases} \min\{R_x, \frac{R_y\|\mathbf{w}_\star - \mathbf{w}\|}{2(\mathbf{w}_\star - \mathbf{w})^\mathsf{T}\mathbf{w}}\}, & \text{if } (\mathbf{w}_\star - \mathbf{w})^\mathsf{T}\mathbf{w} > 0 \\ R_x, & \text{otherwise.} \end{cases}$$

4.1 Comparison of Methods

We ran a number of experiments to study the behavior of these numerical methods. In all experiments, the learner is gradient descent on least squares (1), and the control constraint set is $\|\mathbf{x}\| \le 1, |y| \le 1$. Our first observation is that CNLP has a number of advantages:

1. CNLP's continuous optimal state trajectory matches NLP's discrete state trajectories, especially on learners with small η. This is expected, since the continuous optimal control problem is obtained asymptotically from the discrete one as $\eta \to 0$. Figure 4 shows the teaching task $\mathbf{w}_0 = (1, 0) \Rightarrow \mathbf{w}_* = (1, 0)$. Here we compare CNLP with NLP's optimal state trajectories on four gradient descent learners with different η values. The NLP optimal teaching sequences vary drastically in length T, but their state trajectories quickly overlap with CNLP's optimal trajectory.

2. CNLP is quick to compute, while NLP runtime grows as the learner's η decreases. Table 1 presents the wall clock time. With a small η, the optimal control takes more steps (larger T). Consequently, NLP must solve a nonlinear program with more variables and constraints. In contrast, CNLP's runtime does not depend on η.

3. CNLP can be used to approximately compute the "teaching dimension", i.e. the minimum number of sequential teaching steps T for the discrete problem. Recall CNLP produces an optimal terminal time t_f. When the learner's η is small, the discrete "teaching dimension" T is related by $T \approx t_f/\eta$. This is also supported by Table 1.

That said, it is not trivial to extract a discrete control sequence from CNLP's continuous control function. This hinders CNLP's utility as an optimal teacher.

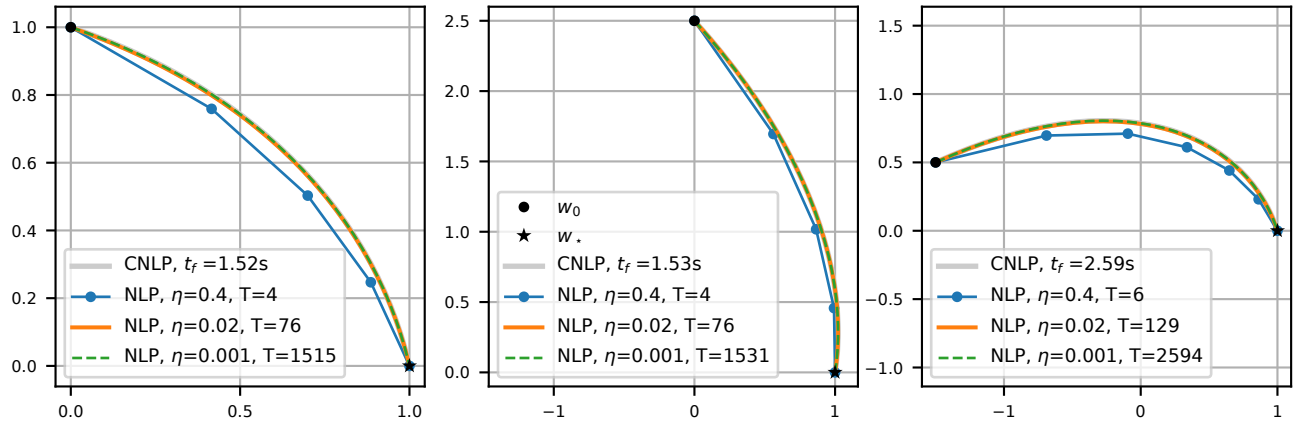

Figure 4: Comparison of CNLP vs NLP. All teaching tasks use the terminal point $\mathbf{w}_\star = (1, 0)$. The initial points used are $\mathbf{w}_0 = (0, 1)$ (left panel), $\mathbf{w}_0 = (0, 2.5)$ (middle panel), and $\mathbf{w}_0 = (-1.5, 0.5)$ (right panel). We observe that the NLP trajectories on learners with smaller η's quickly converges to the CNLP trajectory.

Table 1: Teaching sequence length and wall clock time comparison. NLP teaches three learners with different η's. Target is always $\mathbf{w}_\star = (1, 0)$. All experiments were performed on a conventional laptop.

	NLP			CNLP
$\mathbf{w}_0$	$\eta = 0.4$	0.02	0.001	
$(0, 1)$	$T = 3$	75	1499	$t_f = 1.52$s
	0.013s	0.14s	59.37s	4.1s
$(0, 2.5)$	$T = 5$	76	1519	$t_f = 1.53$s
	0.008s	0.11s	53.28s	2.37s
$(-1.5, 0.5)$	$T = 6$	128	2570	$t_f = 2.59$s
	0.012s	0.63s	310.08s	2.11s

Table 2: Comparison of teaching sequence length T. We fixed $\eta = 0.01$ in all cases.

$\mathbf{w}_0$	$\mathbf{w}_\star$	NLP	STRAIGHT	GREEDY
$(0, 1)$	$(2, 0)$	148	161	233
$(0, 2)$	$(4, 0)$	221	330	721
$(0, 4)$	$(8, 0)$	292	867	2667
$(0, 8)$	$(16, 0)$	346	2849	10581

Our second observation is that NLP, being the discrete-time optimal control, produces shorter teaching sequences than GREEDY or STRAIGHT. This is not surprising, and we have already presented three teaching tasks in Figure 1 where NLP has the smallest T. In fact, there exist teaching tasks on which GREEDY and STRAIGHT can perform arbitrarily worse than the optimal teaching sequence found by NLP. A case study is presented in Table 2. In this set of experiments, we set $\mathbf{w}_0 = (a, 0)$ and $\mathbf{w}_\star = (0, 2a)$. As a increases, the ratio of teaching sequence length between STRAIGHT and NLP and between GREEDY and NLP grow at an exponential rate.

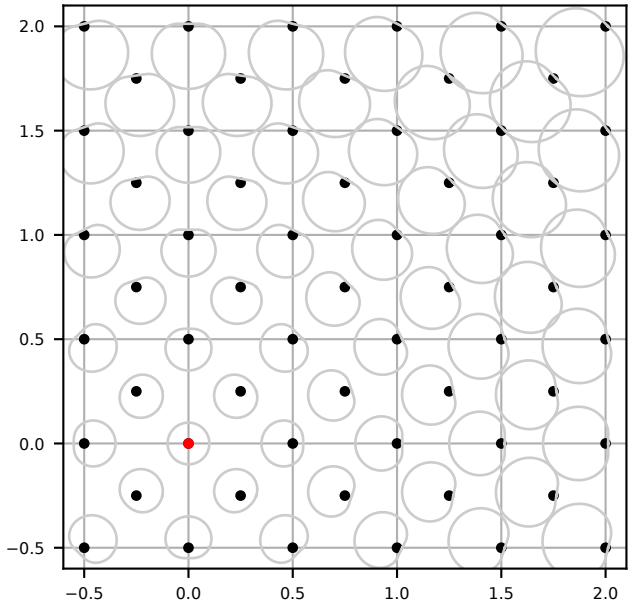

Figure 5: Points reachable in one step of gradient descent (with $\eta = 0.1$) on a least-squares objective starting from each of the black dots. There is circular symmetry about the origin (red dot).

We now dig deeper and present an intuitive explanation of why GREEDY requires more teaching steps than NLP. The fundamental issue is the nonlinearity of the learner dynamics (1) in $\mathbf{x}$. For any $\mathbf{w}$ let us define the one-step reachable set $\{\mathbf{w} - \eta(\mathbf{w}^\mathsf{T}\mathbf{x} - y)\mathbf{x} \mid (\mathbf{x}, y) \in \mathcal{U}\}$. Figure 5 shows a sample of such reachable sets. The key observation is that the starting $\mathbf{w}$ is quite close to the boundary of most reachable sets. In other words, there is often a compressed direction—from $\mathbf{w}$ to the closest boundary of $\mathcal{U}$—along which $\mathbf{w}$ makes minimal progress. The GREEDY scheme falls victim to this phenomenon.

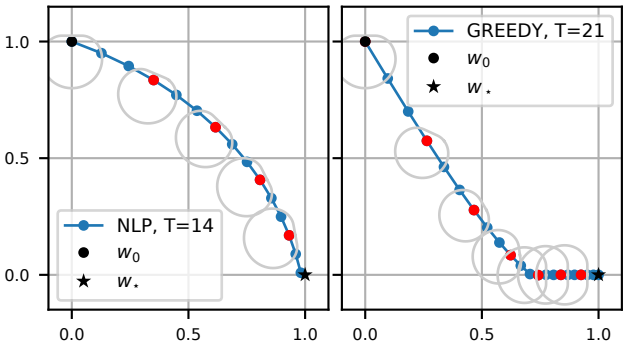

Figure 6: Reachable sets along the trajectory of NLP (left panel) and GREEDY (right panel). To minimize clutter, we only show every 3rd reachable set. For this simulation, we used $\eta = 0.1$. The greedy approach makes fast progress initially, but slows down later on.

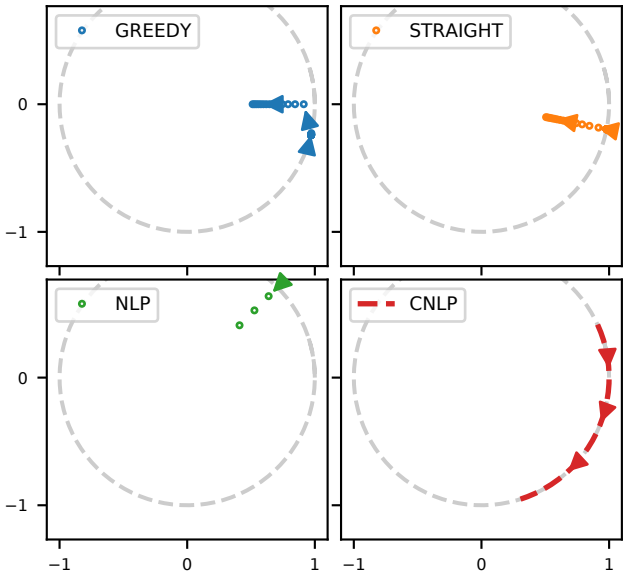

Figure 7: Trajectories of the input sequence $\{\mathbf{x}_t\}$ for GREEDY, STRAIGHT, and NLP methods and the corresponding $\mathbf{x}(t)$ for CNLP. The teaching task is $\mathbf{w}_0 = (-1.5, 0.5)$, $\mathbf{w}_\star = (1, 0)$, and $\eta = 0.01$. Markers show every 10 steps. Input constraint is $\|\mathbf{x}\| \leq 1$.

Figure 6 compares NLP and GREEDY on a teaching task chosen to have short teaching sequences in order to minimize clutter. GREEDY starts by eagerly descending a slope and indeed this quickly brings it closer to $\mathbf{w}_\star$. Unfortunately, it also arrived at the x-axis. For $\mathbf{w}$ on the x-axis, the compressed direction is horizontally outward. Therefore, subsequent GREEDY moves are relatively short, leading to a large number of steps to reach $\mathbf{w}_\star$. Interestingly, STRAIGHT is often better than GREEDY because it also avoids the x-axis compressed direction for general $\mathbf{w}_0$.

We illustrate the optimal inputs in Figure 7, which compares $\{\mathbf{x}_t\}$ produced by STRAIGHT, GREEDY, and NLP and the $\mathbf{x}(t)$ produced by CNLP. The heuristic approaches eventually take smaller-magnitude steps as they approach $\mathbf{w}_\star$ while NLP and CNLP maintain a maximal input norm the whole way.

5 CONCLUDING REMARKS

Techniques from optimal control are under-utilized in machine teaching, yet they have the power to provide better quality solutions as well as useful insight into their structure.

As seen in Section 3, optimal trajectories for the least squares learner are fundamentally 2D. Moreover, there is a taxonomy of regimes that dictates their behavior. We also saw in Section 4 that the continuous CNLP solver can provide a good approximation to the true discrete trajectory when η is small. CNLP is also more scalable than simply solving the discrete NLP directly because NLP becomes computationally intractable as T gets large (or η gets small), whereas the runtime of CNLP is independent of η.

A drawback of both NLP and CNLP is that they produce *trajectories* rather than *policies*. In practice, using an open-loop teaching sequence $(\mathbf{x}_t, y_t)$ will not yield the $\mathbf{w}_t$ we expect due to the accumulation of small numerical errors as we iterate. In order to find a control policy, which is a map from state $\mathbf{w}_t$ to input $(\mathbf{x}_t, y_t)$, we discussed the possibility of solving HJB (Section 2.1) which is computationally expensive.

An alternative to solving HJB is to pre-compute the desired trajectory via CNLP and then use *model-predictive control* (MPC) to find a policy that tracks the reference trajectory as closely as possible. Such an approach is used in Liniger et al. (2015), for example, to design controllers for autonomous race cars, and would be an interesting avenue of future work for the machine teaching problem.

Finally, this paper presents only a glimpse at what is possible using optimal control. For example, the PMP is not restricted to merely solving time-optimal control problems. It is possible to analyze problems with state- and input-dependent running costs, state and input pointwise or integral constraints, conditional constraints, and even problems where the goal is to reach a target *set* rather than a target point.

Acknowledgements

We thank Ross Boczar and Max Simchoitz for helpful discussions. This work is supported by NSF 1837132, 1545481, 1704117, 1623605, 1561512, the MADLab AF

Center of Excellence FA9550-18-1-0166, and the University of Wisconsin.

References

M. Athans and P. L. Falb. *Optimal control: An introduction to the theory and its applications*. Courier Corporation, 2013.

J. T. Betts. *Practical methods for optimal control and estimation using nonlinear programming*, volume 19. SIAM, 2010.

B. Biggio, B. Nelson, and P. Laskov. Poisoning attacks against support vector machines. In J. Langford and J. Pineau, editors, *29th Int'l Conf. on Machine Learning (ICML)*. Omnipress, Omnipress, 2012.

R. H. Byrd, J. Nocedal, and R. A. Waltz. Knitro: An integrated package for nonlinear optimization. In *Large Scale Nonlinear Optimization, 35–59, 2006*, pages 35–59. Springer Verlag, 2006.

I. Dunning, J. Huchette, and M. Lubin. Jump: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017. doi: 10.1137/15M1020575.

H. Febbo. Nloptcontrol.jl, 2017. URL : `https://github.com/JuliaMPC/NLOptControl.jl`.

S. Goldman and M. Kearns. On the complexity of teaching. *Journal of Computer and Systems Sciences*, 50(1):20–31, 1995.

I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and Harnessing Adversarial Examples. *ArXiv e-prints*, Dec. 2014.

C. Y. Kao, S. Osher, and J. Qian. Lax–Friedrichs sweeping scheme for static Hamilton–Jacobi equations. *Journal of Computational Physics*, 196(1): 367–391, 2004.

D. E. Kirk. *Optimal control theory: An introduction*. Courier Corporation, 2012.

D. Liberzon. *Calculus of variations and optimal control theory: A concise introduction*. Princeton University Press, 2011.

A. Liniger, A. Domahidi, and M. Morari. Optimization-based autonomous racing of 1:43 scale RC cars. *Optimal Control Applications and Methods*, 36(5):628–647, 2015.

W. Liu, B. Dai, A. Humayun, C. Tay, C. Yu, L. B. Smith, J. M. Rehg, and L. Song. Iterative machine teaching. In *International Conference on Machine Learning*, pages 2149–2158, 2017.

S. Mei and X. Zhu. Using machine teaching to identify optimal training-set attacks on machine learners. In *The Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

K. Patil, X. Zhu, L. Kopec, and B. Love. Optimal teaching for limited-capacity human learners. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

M. A. Patterson and A. V. Rao. GPOPS-II: A MATLAB software for solving multiple-phase optimal control problems using hp-adaptive Gaussian quadrature collocation methods and sparse nonlinear programming. *ACM Transactions on Mathematical Software (TOMS)*, 41(1):1, 2014.

A. V. Rao. A survey of numerical methods for optimal control. *Advances in the Astronautical Sciences*, 135 (1):497–528, 2009.

A. Sen, P. Patel, M. A. Rau, B. Mason, R. Nowak, T. T. Rogers, and X. Zhu. Machine beats human at sequencing visuals for perceptual-fluency practice. In *Educational Data Mining*, 2018.

D. Tonon, M. S. Aronna, and D. Kalise. *Optimal control: Novel directions and applications*, volume 2180. Springer, 2017.

J. N. Tsitsiklis. Efficient algorithms for globally optimal trajectories. *IEEE Transactions on Automatic Control*, 40(9):1528–1538, 1995.

A. Wächter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

X. Zhu. Machine teaching: an inverse problem to machine learning and an approach toward optimal education. In *The Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI "Blue Sky" Senior Member Presentation Track)*, 2015.

X. Zhu, A. Singla, S. Zilles, and A. N. Rafferty. An Overview of Machine Teaching. *ArXiv e-prints*, Jan. 2018. https://arxiv.org/abs/1801.05927.