RL Perceptron: Generalization Dynamics of Policy Learning in High Dimensions

RL Perceptron: Generalization Dynamics of Policy Learning in High Dimensions

Nishil Patel[1,*] Sebastian Lee[1,2] Stefano Sarao Mannelli[5,6] Sebastian Goldt[3] and Andrew Saxe[1,4,*]

[1]*Gatsby Computational Neuroscience Unit, University College London, Gower Street, London, WC1E 6BT, United Kingdom*
[2]*Imperial College London, Exhibition Road, South Kensington, London SW7 2AZ, United Kingdom*
[3]*International School of Advanced Studies (SISSA), Via Bonomea, 265, 34136 Trieste, Italy*
[4]*Sainsbury Wellcome Centre, University College London, 25 Howland Street, London W1T 4JG, United Kingdom*
[5]*Data Science and AI, Computer Science and Engineering, Chalmers University of Technology and University of Gothenburg, Gothenburg, Sweden*
[6]*School of Computer Science and Applied Mathematics, University of the Witwatersrand, Johannesburg, South Africa*

(Received 5 March 2024; revised 20 December 2024; accepted 23 January 2025; published 13 May 2025)

Reinforcement learning (RL) algorithms have transformed many domains of machine learning. To tackle real-world problems, RL often relies on neural networks to learn policies directly from pixels or other high-dimensional sensory input. By contrast, many theories of RL have focused on discrete state spaces or worst-case analysis, and fundamental questions remain about the dynamics of policy learning in high-dimensional settings. Here, we propose a solvable high-dimensional RL model that can capture a variety of learning protocols, and we derive its typical policy learning dynamics as a set of closed-form ordinary differential equations. We obtain optimal schedules for the learning rates and task difficulty—analogous to annealing schemes and curricula during training in RL—and show that the model exhibits rich behavior, including delayed learning under sparse rewards, a variety of learning regimes depending on reward baselines, and a speed-accuracy trade-off driven by reward stringency. Experiments on variants of the Procgen game "Bossfight" and Arcade Learning Environment game "Pong" also show such a speed-accuracy trade-off in practice. Together, these results take a step toward closing the gap between theory and practice in high-dimensional RL.

DOI: 10.1103/PhysRevX.15.021051

Subject Areas: Computational Physics, Statistical Physics

I. INTRODUCTION

Thanks to algorithmic and engineering advancements, reinforcement learning (RL) methods have achieved superhuman performance in a variety of domains, for example, in playing complex games like Go [1,2]. Reinforcement learning involves an agent in an environment that takes actions based on the given state of an environment that it is exploring; for example, an agent playing chess must decide which action to take based on the state of the board. The map from states to actions is called a "policy." The overarching goal of the agent is to learn a policy that will allow them to maximize some kind of total reward, for example, a reward given for taking an opponent's piece in a game of chess.

In cases where the state and action spaces are discrete and small enough, policies can be represented by simple look-up tables. However, the curse of dimensionality limits this approach to low-dimensional problems. In today's applications, environments are complex, and policies are learned directly from high-dimensional inputs representing the state of the environment, using neural networks [3].

While comprehensive theoretical results exist for tabular RL, our theoretical grasp of RL for high-dimensional problems requiring neural networks to represent the policy remains limited, despite its practical success. The lack of a clear notion of similarity between discrete states further means that tabular methods do not address the core question of generalization: How are values and policies extended to unseen states and across seen states [4]? Consequently, much of this theoretical work is far from the current practice of RL, which increasingly relies on deep neural networks to approximate policies and other RL components like value functions. Moreover, while RL

*Contact author: ucabnp2@ucl.ac.uk; a.saxe@ucl.ac.uk

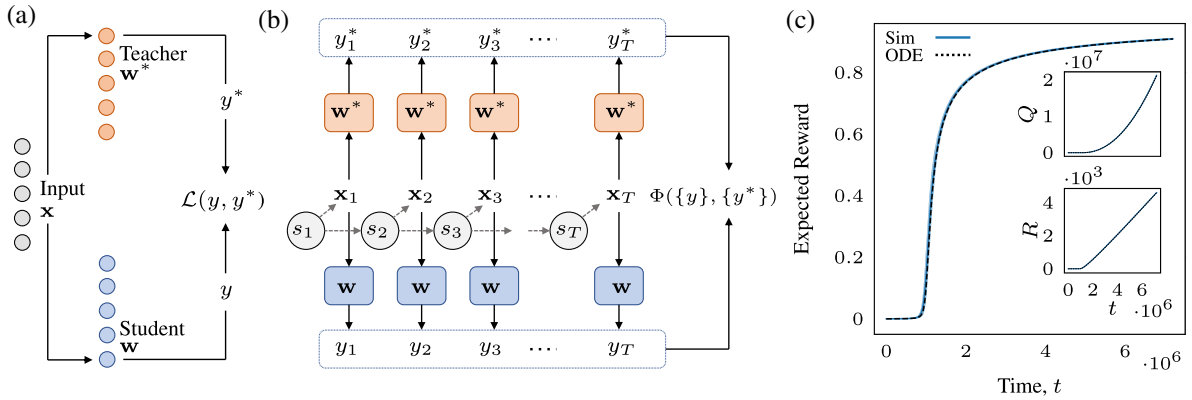

FIG. 1. RL-perceptron model for policy learning in high dimensions. (a) Classic teacher-student model for supervised learning, where a neural network, called the student, is trained on inputs x whose label y^* is given by another neural network, called the teacher. (b) RL setting, where the student moves through states s_t making a series of T choices given in response to inputs x_t. The RL perceptron is an extension of the teacher-student model, as we assume there is a "right" choice y_t on each time step given by a teacher network. The student receives a reward after T decisions according to a criterion Φ that depends on the choices made and the corresponding correct choices. (c) Example learning dynamics in the RL-perceptron model for a problem with $T = 12$ choices where the reward is given only if all the decisions are correct. The plot shows the expected reward of a student trained in the RL-perceptron setting in simulations (solid line) and for our theoretical results (dashed line) obtained from solving the dynamical equations (12) and (13). Finite-size simulations and theory show good agreement. We reduce the stochastic evolution of the high-dimensional student to the study of deterministic evolution of two scalar quantities, R and Q (more details are given in Sec. III A); their evolutions are shown in the inset. The parameters are as follows: $D = 900$, $\eta_1 = 1$, $\eta_2 = 0$, and $T = 12$.

theory has often addressed "worst-case" performance and convergence behavior, their *typical* behavior has received comparatively little attention (see further related work in Sec. I B).

Meanwhile, there is a long tradition of neural network theory that employs tools from statistical mechanics to analyze learning and generalization in high-dimensional settings with a focus on typical behaviors, as is usually the case in statistical mechanics. This theory was first developed in the context of supervised learning; see Refs. [5–9] for classical and recent reviews. More recently, this approach yielded new insights beyond vanilla supervised learning, for example, in curriculum learning [10], continual learning [11–13], few-shot learning [14], and transfer learning [15–17]. However, policy learning has not been analyzed using statistical mechanics yet—a gap we address here by studying the generalization dynamics of a simple neural network trained on a RL task.

Our goal is to develop a theory for the typical dynamics of policy learning. For example, we would like to explain how problem properties and algorithmic choices impact how quickly a model will learn or how effectively it will generalize. In order to achieve this goal, we work on an analysis of a perceptron-adapted online policy-learning update, which can be considered as an analog to the REINFORCE algorithm [18] (see Sec. II A for more details). REINFORCE is the simplest online "policy-gradient" algorithm. Policy-gradient methods, despite their simplicity, underpin much of modern reinforcement learning with deep neural networks [19–21]; consequently, an understanding of online policy learning dynamics is beneficial for transferable insights into more complex policy-gradient methods and as a starting point from which to analyze more complex methods. We further contrast our method with existing results in Sec. I B.

Our approach consists in considering a simple model of a RL problem which we approach through the teacher-student framework, allowing for exact solutions, i.e., the derivation of equations describing the typical learning dynamics *exactly*; we then analyze properties of the solution. In the classic teacher-student model of supervised learning [5,22], a neural network called the student is trained on inputs $\mathbf{x}$ whose labels y^* are specified by another neural network called the teacher [see Fig. 1(a)]. The goal of the student is to learn the function represented by the teacher from samples $(\mathbf{x}, y^*)$. This framework enables an exact analysis, characterizing the generalization error of learning algorithms over the entire learning trajectory. In many RL settings, agents face sequential decision-making tasks that require a series of intermediate choices to successfully complete an episode. We map this process into the RL perceptron, where the teacher can be viewed as specifying a "perfect policy network" that prescribes a reward signal used to train the student network representing the policy of the agent. This setup lends itself to an analysis that exactly describes the average-case dynamics over the entire learning trajectory.

A. Main results

In this work, we develop a teacher-student framework for a high-dimensional sequential policy learning task, the

RL perceptron, and derive asymptotically exact ordinary differential equations (ODEs) that describe the typical learning dynamics of policy-gradient RL agents in an online setting by building on classic work by Saad and Solla [23] (see Sec. III A).

Using these ODEs, we can characterize learning behavior in various scenarios: We investigate sparse delayed reward schemes and the impact of negative rewards (Sec. III B), derive optimal learning rate schedules and episode length curricula, and recover common annealing strategies (Sec. III C). We identify ranges of learning rates for which learning is "easy" and "hybrid-hard" (Sec. III D). We also identify a speed-accuracy trade-off driven by reward stringency (Sec. III E).

In Sec. IV, we demonstrate a similar speed-accuracy trade-off in simulations of high-dimensional policy learning from pixels using the Procgen environment "Bossfight" [24] and the Arcade Learning Environment (ALE) game "Pong" [25]. A link to the code and instructions for running all simulations and experiments is given in Appendix F.

B. Further related work

Sample complexity in RL.—An important line of work in the theory of RL focuses on the sample complexity and other learnability measures for specific classes of models such as tabular RL [26,27], state aggregation [28], various forms of Markov decision processes (MDPs) [29–34], reactive partially observable Markov decision processes (POMDPs) [35], and FLAMBE [36]. Here, we are instead concerned with the learning dynamics: How do reward rates, episode length, etc. influence the speed of learning and the final performance of the model.

Statistical learning theory for RL.—This theory aims at finding complexity measures analogous to the Rademacher complexity or Vapnik-Chervonenkis dimension from statistical learning theory for supervised learning [37,38]. Proposals include the Bellman rank [39], as well as the Eluder dimension [40] and its generalizations [41]. This approach focuses on worst-case analysis, which typically differs significantly from practice (at least in supervised learning [42]). Furthermore, complexity measures for RL are generally more suitable for value-based methods; policy-gradient methods have received less attention despite their prevalence in practice [43,44]. We focus instead on average-case dynamics of policy-gradient methods.

Dynamics of learning.—A series of recent papers considered the dynamics of temporal-difference learning and policy gradients in the limit of wide, two-layer neural networks [45–48]. These works focus on one of two "wide" limits. The first is the neural tangent kernel [49,50] or "lazy" regime [51], where the network behaves like an effective kernel machine and does not learn data-dependent features, which is key for efficient generalization in high dimensions. Lyle *et al.* [52] consider a framework to study value learning using the temporal-difference algorithm; their findings provide insights into quantities to track during learning to describe the dynamics of representations, but they still rely on input from the RL problem under consideration. In our setting, the success of the student crucially relies on learning the weight vector of the teacher, which is difficult for lazy methods [53–56]. The other wide regime is the mean-field limit of interacting particles [57–59], where learning dynamics are captured by a nonlinear partial differential equation. While this elegant description allows them to establish global convergence properties, the resulting equations are hard to solve in practice, and a further analysis is therefore difficult. The ODE description we derive here will instead allow us to describe a series of effects in the following sections. Similarly to our work, Bordelon *et al.* [60] give a typical-case analysis of the dynamics of temporal-difference learning, which learns the value function rather than the policy as we do here. We will come back to this point later. Finally, Rubin *et al.* [61] analyze the dynamics of the tempotron [62], a neuron model that learns spike timing–based decisions. Similarly to our work, they consider sparse rewards; however, beyond this similarity, the paper does not connect to RL, and their update rules are substantially different.

II. RL PERCEPTRON: SETUP AND LEARNING ALGORITHM

The RL perceptron considers a task, illustrated in Fig. 1(b), where an agent takes a sequence of choices over an episode (episodes are length T; i.e., a choice is made at every time step $t \in \{1, \ldots, T\}$). At each time step, the agent occupies some state s_t in the environment and receives some high-dimensional observation $\mathbf{x}_t \in \mathbb{R}^D$ conditioned on s_t with $t = 1, \ldots, T$. So far, we have considered the POMDP formalism (detailed in Sec. II B); we could equally consider the MDP formalism (also detailed in Sec. II B), where D could be considered the feature dimension. The student network determines the actions made at each t. We study the simplest possible student network, a perceptron with weight vector $\mathbf{w} \in \mathbb{R}^D$ that takes in observations $\mathbf{x}_t$ and outputs $y(\mathbf{x}_t) = \mathrm{sgn}(\mathbf{w}^\intercal \mathbf{x}_t/\sqrt{D})$. We interpret the outputs y_t as binary actions, for example, whether to go left or right in an environment. As the agent makes choices in response to the output of the student network with observations as inputs, the student is analogous to a policy network, where the deterministic policy $\pi_{\mathbf{w}}(a_t \| \mathbf{x}_t) = \frac{1}{2}(1 + a_t y_t)$ specifies the probability of taking action $a_t \in \{-1, 1\}$ given observation $\mathbf{x}_t$ (i.e., the action taken a_t is always equal to the action y_t specified by the student). In RL, a policy defines the learning agent's way of behaving at a given time. The teacher network $\mathbf{w}^* \in \mathbb{R}^D$ can be considered as a "perfect policy network" that specifies the "correct" decision $[y_t^* = \mathrm{sgn}(\mathbf{w}^{*\intercal} \mathbf{x}_t/\sqrt{D})]$ to be made at every step (the

policy that the student aims to emulate through the learning process). A significant simplification for analytical tractability is that the actions do not influence state transitions (i.e., the observations $\mathbf{x}_t$ are independent identically distributed). The crucial point is that the student does not have access to the correct choices but only receives feedback at time step t in the form of some (non-Markovian) reward $R_t(y_{1:t}, y^*_{1:t}, \Phi)$ that is conditioned on all the actions taken ($y_{1:t}$) and correct actions ($y^*_{1:t}$) up to time step t, and the condition for a reward, Φ. For instance, Φ could be the condition that the agent receives a reward on the Tth step only if all choices in an episode are correctly made; otherwise, a penalty is received—a learning signal that is considerably less informative than in supervised learning. In Sec. III D, we see that receiving penalties is not always beneficial. In the model, T plays the role of *difficulty*; a more complex task may be defined by the number of correct decisions needed in order to receive a reward. An alternative notion of difficulty would be to consider a planted version of the convex perceptron used in jamming [63]. However, this formulation would not allow some manipulations—e.g., dense rewards described in Sec. III B—that are standard practice in RL.

To train the network (learn a policy), we consider a weight update where the student evolves after the μth episode as

$$\mathbf{w}^{(\mu+1)} = \mathbf{w}^{(\mu)} + \frac{\eta}{T\sqrt{D}} \sum_{t=1}^{T} y_t^{(\mu)} \mathbf{x}_t^{(\mu)} G_t^{(\mu)}, \tag{1}$$

where $G_t = \sum_{t'=t}^{T} \gamma^{t'-t} R_{t'}$ is the total discounted reward from time t, $\gamma \in (0, 1]$ is the discount factor, η is the learning rate, and the superscript μ denotes variables from the μth time step in the algorithm. (N.B. algorithm time and episode time are different and are, respectively, denoted by μ and t). The motivation for this update comes as a modification of a Hebbian update, which occurs after an episode that can contain multiple observations and actions, and is weighted by G_t (which also can depend on previous observations and actions). We comment on the connection of this update to policy-gradient methods in the next section. For the analysis and simulations in Sec. III, we consider $\gamma = 1$ and largely restrict ourselves to receiving sparse rewards, i.e., where a reward or penalty is only received at the end of an episode upon successful or unsuccessful completion of the episode; in this setting, the total discounted reward may be written as

$$G_t = r_1 \mathbb{I}(\Phi(y_{1:T}, y^*_{1:T})) - r_2(1 - \mathbb{I}(\Phi(y_{1:T}, y^*_{1:T}))) \quad \forall\, t, \tag{2}$$

where r_1 is the reward, r_2 is the penalty, $\mathbb{I}$ is an indicator function, and Φ is the Boolean criterion that determines whether the episode was completed successfully—for instance, $\mathbb{I}(\Phi) = \prod_t^T \theta(y_t y_t^*)$ (where θ is the step function) if the student has to get every decision right in order to receive a reward. The reward or penalty can be amalgamated into η: We define $\eta_1 = \eta r_1$ and $\eta_2 = \eta r_2$, essentially "positive" and "negative" learning rates, and we use these rates instead of r_1 and r_2 in the remaining text.

Note that in the case of $T = 1$, $\eta_1 = 0$, $\eta_2 > 0$, and $\mathbb{I}(\Phi) = \theta(-yy^*)$ (i.e., the learning rule updates the weight only if the student is incorrect on a given sample), we recover the famous perceptron learning rule of supervised learning [64].

A. Connection to REINFORCE policy gradient

The update in Eq. (1) can be related to the REINFORCE policy-gradient algorithm [18]. REINFORCE is a Monte Carlo policy-gradient method. Policy-gradient methods aim to optimize parametrized policies with respect to the return J (total expected reward over an episode). In the case of REINFORCE, the return is estimated from episode samples—i.e., single episodes are sampled by acting under a current policy, and these sampled episodes are used to update said policy. In an arbitrary environment where an agent occupies states s_t and may take actions a_t by acting under some policy π parametrized by θ, the policy gradient is given by

$$\nabla_\theta J = \left\langle \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) G_t \right\rangle, \tag{3}$$

and the REINFORCE update of θ at the μth time step in the algorithm for the μth sampled episode is hence given by

$$\theta^{(\mu+1)} = \theta^{(\mu)} + \eta \sum_{t=0}^{T-1} \nabla_\theta \log \pi_{\theta^\mu}\left(a_t^{(\mu)}|s_t^{(\mu)}\right) G_t^{(\mu)}. \tag{4}$$

In the RL-perceptron setup, the true policy is deterministic and given by $\pi_{\mathbf{w}}(a_t|\mathbf{x}_t) = \frac{1}{2}(1 + a_t \mathrm{sgn}(\mathbf{w}^\intercal \mathbf{x}_t/\sqrt{D}))$. In this case, Eq. (4) becomes

$$\mathbf{w}^{(\mu+1)} = \mathbf{w}^{(\mu)} + \frac{\eta}{\sqrt{D}} \sum_{t=0}^{T-1} y_t^{(\mu)} \mathbf{x}_t^{(\mu)} G_t^{(\mu)} \delta\left(\mathbf{w}^{(\mu)\intercal} \mathbf{x}_t^{(\mu)}\right) \tag{5}$$

where $\delta(\cdot)$ is the Dirac delta. We have inserted observations $\mathbf{x}_t$ in place of states s_t, and we can replace a_t by y_t due to the deterministic policy. This update is not tractable, but we can see the structural similarity to Eq. (1), which can be fully recovered by replacing the gradient of $\mathrm{sgn}(\mathbf{w}^\intercal \mathbf{x})$ with the linearized gradient $(\nabla_{\mathbf{w}} \mathbf{w}^\intercal \mathbf{x})$ and rescaling by T. We would like to comment on policy gradients, in general, so we verify that this linearization does not change the qualitative behavior of the REINFORCE update in Appendix B—where we plot the learning dynamics for a variety of reward settings when training under the exact REINFORCE policy-gradient update and acting under a logistic policy. Figures 9(a), 9(c), and 9(d) can be compared

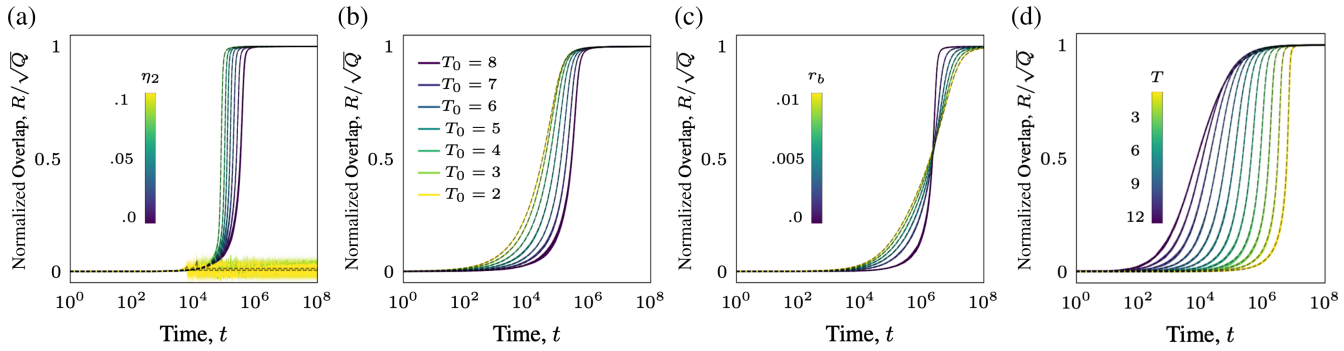

FIG. 2. ODEs accurately describing diverse learning protocols. We show the evolution of the normalized student-teacher overlap ρ for the numerical solution of the ODEs (dashed lines) and simulation (colored lines) in three reward protocols. All students receive a reward of η_1 for getting all decisions in an episode correct, and additional results are as follows: In panel (a), a penalty η_2 (i.e., negative reward) is received if the agent does not survive until the end of an episode. Panel (b) shows that an additional reward of 0.2 is received at time step T_0 if the agent survives beyond T_0 time steps. In panel (c), an additional reward r_b is received at time step t for every correct decision y_t made in an episode. In panel (d), the episode length T is varied. The parameters are as follows: $D = 900$, $T = 11$, and $\eta_1 = 1$.

to Figs. 2(a)–2(c), respectively, and Fig. 9(b) can be compared to Fig. 5 for verification of consistent qualitative behavior.

This connection of the RL-perceptron update to policy gradients is analogous to the connection of the classic perceptron update to gradient descent; the perceptron update rule in classic supervised learning is equal to the linearized version of the single sample gradient descent update for binary classification with a perceptron using L_2 loss.

B. Connection to Markov decision processes

Formally, most RL problems can be described as a MDP or a POMDP; see Appendix A. Therefore, a lot of theoretical work on reinforcement learning has been formulated in this framework [65]. To make it easier to connect the RL perceptron with this literature, in Appendix A, we show that the RL perceptron can be formulated as either an MDP or POMDP with non-Markovian rewards. In a nutshell, the observations $\mathbf{x_t}$ in the implementation described can be thought of as high-dimensional states s_t (MDP) or as the noisy high-dimensional observations of underlying low-dimensional latent states s_t (POMDP). Each interpretation naturally leads to different extensions. The MDP framework is able to incorporate kernelized high-dimensional feature maps of the underlying state, and the POMDP framework is more amenable to tractable calculations of expectations for trajectories involving state and action-dependent state transitions. We do not explore the connection further, as we are primarily interested in the dynamics of learning.

III. THEORETICAL RESULTS

The RL perceptron enables an analytical investigation of average dynamics through the identification and characterization of a few relevant order parameters, as explained in Sec. III A. This approach significantly simplifies the problem by transitioning from a high-dimensional to a low-dimensional framework. Moreover, it offers adaptability for characterizing and comparing various learning protocols, as detailed in Sec. III B. On a practical level, the derived equations allow for the determination of optimal learning rate annealing strategies to maximize expected rewards, and the use of a curriculum protocol enhances training efficiency (Sec. III C). At a fundamental level, studying the low-dimensional equations provides valuable insights into the nature of the problem. First, in Sec. III D, we observe that the presence of negative rewards can result in suboptimal fixed points and a counterintuitive slowing down of dynamics near the emergence of such suboptimal fixed points. Second, in Sec. III E, we demonstrate that several protocols aimed at expediting the initial learning phase actually lead to poorer long-term performance.

A. Set of ODEs exactly captures learning dynamics of RL perceptron

The goal of the student during training is to emulate the teacher as closely as possible or, in other words, have a small number of disagreements with the teacher, $y(\mathbf{x}) \neq y^*(\mathbf{x})$. The generalization error is given by the average number of disagreements,

$$\epsilon_g \equiv \frac{1}{2}\langle (y - y^*)^2 \rangle = \frac{1}{2}\left(1 - \left\langle \mathrm{sgn}\left(\frac{\mathbf{w}^* \cdot \mathbf{x}}{\sqrt{D}}\right)\mathrm{sgn}\left(\frac{\mathbf{w} \cdot \mathbf{x}}{\sqrt{D}}\right)\right\rangle\right) \tag{6}$$

$$= \frac{1}{2}(1 - \langle \mathrm{sgn}(\nu)\mathrm{sgn}(\lambda)\rangle), \tag{7}$$

where the average $\langle\cdot\rangle$ is taken over the inputs $\mathbf{x}$, and we have introduced the scalar preactivations for the student and the teacher, $\lambda \equiv \mathbf{w}\cdot\mathbf{x}/\sqrt{D}$ and $\nu \equiv \mathbf{w}^*\cdot\mathbf{x}/\sqrt{D}$, respectively. We can therefore transform the high-dimensional average over the inputs $\mathbf{x}$ into a low-dimensional average over the preactivations (λ,ν). By specifying a distribution over observations, $\mathbf{x}_t \sim \mathcal{N}(\mathbf{0}, \mathbb{1}_D)$, the average in Eq. (6) can be carried out by noting that the tuple (λ,ν) follows a jointly Gaussian distribution with means $\langle\lambda\rangle = \langle\nu\rangle = 0$ and covariances

$$Q \equiv \langle\lambda^2\rangle = \frac{\mathbf{w}\cdot\mathbf{w}}{D}, \qquad R \equiv \langle\lambda\nu\rangle = \frac{\mathbf{w}\cdot\mathbf{w}^*}{D},$$
$$\text{and} \quad S \equiv \langle\nu^2\rangle = \frac{\mathbf{w}^*\cdot\mathbf{w}^*}{D}. \tag{8}$$

These covariances, or order parameters as they are known in statistical physics, have a simple interpretation. The overlap S is simply the length of the weight vector of the teacher; for simplicity of equations, we choose $S = 1$. Likewise, the overlap Q gives the length of the student weight vector; however, this quantity will vary during training. For example, when starting from small initial weights, Q will be small, and it will grow throughout training. Lastly, the alignment R quantifies the correlation between the student and the teacher weight vector. At the beginning of training, $R \approx 0$, as both the teacher and the initial condition of the student are drawn at random. As the student starts learning, the overlap R increases. Evaluating the Gaussian average in Eq. (6) shows that the generalization error is then a function of the normalized overlap $\rho = R/\sqrt{Q}$, and is given by

$$\epsilon_g = \frac{1}{\pi}\arccos\left(\frac{R}{\sqrt{Q}}\right). \tag{9}$$

The crucial point is that the description of the high-dimensional learning problem has been reduced from D parameters to two time-evolving quantities, Q and R, which are self-averaging in the $D \to \infty$ limit. We now discuss their dynamics.

The dynamics of order parameters.—At any given point during training, the value of the order parameters determines the test error via Eq. (9). How do the order parameters evolve during training with the stochastic update rule in Eq. (1)? We follow the approach of Saad and Solla [23], Kinzel and Ruján [66], and Biehl and Schwarze [67] to derive a set of dynamical equations that describe the dynamics of the student in the thermodynamic limit where the input dimension goes to infinity. The general ODEs derived in Appendix C are given below:

$$\frac{dR}{d\alpha} = \frac{\eta}{T}\left\langle \sum_{t=1}^{T} \mathrm{sgn}(\lambda_t)\nu_t G_t \right\rangle, \tag{10}$$

$$\frac{dQ}{d\alpha} = \frac{2\eta}{T}\left\langle \sum_{t=1}^{T} \mathrm{sgn}(\lambda_t)\lambda_t G_t \right\rangle + \frac{\eta^2}{T^2}\left\langle \sum_{t=1}^{T} G_t^2 \right\rangle, \tag{11}$$

where α serves as a continuous time variable in the limit $D \to \infty$ (not to be confused with t, which counts episode steps). In this way, the stochastic evolution of the student in high dimensions has been mapped to the deterministic evolution of two order parameters in a continuous time description.

We give explicit dynamics for a variety of learning protocols (different protocols are encapsulated by the functional form of the discounted reward, G_t). Because of the length of these expressions, we report the explicit ODEs of the dynamics in Appendix C. In Secs. III C and III D, we devote our analysis to the reward condition where the agent must survive until the end of an episode to receive a reward and receives a penalty otherwise [this is described by the G_t given in Eq. (2) and with $\mathbb{I}(\Phi) = \prod_t^T \theta(y_t y_t^*)$]; as such, we explicitly state the ODEs for the order parameters below:

$$\frac{dR}{d\alpha} = \frac{\eta_1+\eta_2}{\sqrt{2\pi}}\left(1+\frac{R}{\sqrt{Q}}\right)P^{T-1} - \eta_2 R\sqrt{\frac{2}{\pi Q}}, \tag{12}$$

$$\frac{dQ}{d\alpha} = (\eta_1+\eta_2)\sqrt{\frac{2Q}{\pi}}\left(1+\frac{R}{\sqrt{Q}}\right)P^{T-1} - 2\eta_2\sqrt{\frac{2Q}{\pi}} + \frac{(\eta_1^2-\eta_2^2)}{T}P^T + \frac{\eta_2^2}{T}, \tag{13}$$

where $P = 1 - \cos^{-1}(R/\sqrt{Q})/\pi$ is the probability of a single correct decision. While our derivations of the equations follow heuristics from statistical physics, we anticipate that their asymptotic correctness in the limit $D \to \infty$ can be established rigorously using the techniques of Goldt *et al.* [68], Veiga *et al.* [69], and Arnaboldi *et al.* [70]. We illustrate the accuracy of these equations already in finite dimensions ($D = 900$) in Fig. 1(c), where we show the expected reward, as well as the overlaps R and Q, of a student as measured during a simulation and from integration of the dynamical equations (solid and dotted lines, respectively).

The derivation of the dynamical equations that govern the learning dynamics of the RL perceptron is our first main result. Equipped with this tool, we now analyze several phenomena exhibited by the RL perceptron through a detailed study of these equations.

B. Learning protocols

The RL perceptron allows for the characterization of different RL protocols by adapting the reward condition Φ. We consider the following three settings:

Vanilla.—The dynamics in the case without a penalty, and where the survival of the entire episode is required for a

reward [$G_t = \prod_t^T \theta(y_t y_t^*)$], is shown in Fig. 2(d). Rewards are sparsest in this protocol; as a result, we observe a characteristic initial plateau in the expected reward followed by a rapid jump. The length of this plateau increases with T, consistent with the notion that sparser rewards slow learning [71]. Plateaus during learning, which arise from saddle points in the loss landscape, have also been studied for (deep) neural networks in the supervised setting [23,72], but they do not arise in the supervised perceptron. Hence, the RL setting can qualitatively change the learning trajectory.

Penalty.—The initial plateau can be reduced by providing a penalty or negative reward ($\eta_2 > 0$) when the student fails in the task. This change provides weight updates much earlier in training and thus accelerates the escape from the plateau. The dynamics under this protocol are shown in Fig. 2(a). It is clear that the penalty provides an initial speedup in learning, as expected if the agent were unaligned and more likely to commit an error. However, a high penalty can create additional suboptimal fixed points in the dynamics, leading to a low asymptotic performance as seen in Fig. 2(a) (more details are given in Sec. III D). In the simulations, finite-size effects occasionally permit escape from the suboptimal fixed point and jumps to the optimal one, leading to a high variance in the results. The general form of the discounted reward G_t in this case is given by Eq. (2).

Subtask.—The model is also able to capture the dynamics of more complicated protocols: Figure 2(b) shows learning under the protocol where a smaller subreward r_b is received if the agent survives beyond a shorter duration $T_0 < T$ in addition to the final reward received for survival until time step T; i.e., some reward is still received even if the agent does not survive for the entire episode. In this case, $G_t = r_b \mathbb{I}(t \le T_0) \prod_{t'=1}^{T_0} \theta(y_{t'} y_{t'}^*) + \prod_{t'=1}^{T} \theta(y_{t'} y_{t'}^*)$.

Dense.—The model can capture scenarios in which rewards are densely received throughout an episode, which is reflected by the learning protocol where the agent receives a small reward r_b for every correct decision made in an episode and a reward of 1 at time T if the entire episode is successfully completed; i.e., like the previous method, some reward is still received even if the agent does not survive for the entire episode, and these dynamics are captured in Fig. 2(c). In this case, the discounted reward is $G_t = \sum_{t'=1}^{T} r_b \theta(y_{t'} y_{t'}^*) + \prod_{t'=t}^{T} \theta(y_{t'} y_{t'}^*)$.

C. Optimal hyperparameter schedules

Hyperparameter schedules are crucial for successful training of RL agents. In our setup, the two most important hyperparameters are the learning rates and the episode length. In the RL perceptron, we can derive optimal schedules for both hyperparameters. For simplicity, here we report the results in the spherical case, where the length of the student vector is fixed at $\sqrt{D}$ (we discuss the unconstrained case in Appendix D); then, $Q(\alpha) = 1$ at all times, and we only need to track the teacher-student overlap $\rho = R/\sqrt{Q}$, which quantifies the generalization performance of the agent. Keeping the choice $\mathbb{I}(\Phi) = \prod_{t=1}^{T} \theta(y_t y_t^*)$, we find that the optimal schedules over episodes for T and η can then be found by maximizing the change in overlap at each update, i.e., setting $\partial/\partial T(d\rho/d\alpha)$ and $\partial/\partial\eta(d\rho/d\alpha)$ to zero, respectively. After some calculations, we find the optimal schedules below:

$$T_{\mathrm{opt}} = \left\lfloor \frac{\sqrt{\pi}}{2} \frac{\eta\rho P}{(1-\rho^2)\sqrt{2Q}} \left[1 + \sqrt{1 - \frac{\sqrt{2Q}}{\eta\rho} \frac{4(1-\rho^2)}{\sqrt{\pi} P \ln(P)}} \right] \right\rfloor$$

$$\text{and} \quad \eta_{\mathrm{opt}} = \sqrt{\frac{Q}{2\pi}} \frac{T(1-\rho^2)}{\rho P}, \tag{14}$$

where $\lfloor \cdot \rfloor$ indicates the floor function. Figure 3(a) shows the evolution of ρ under the optimal episode length

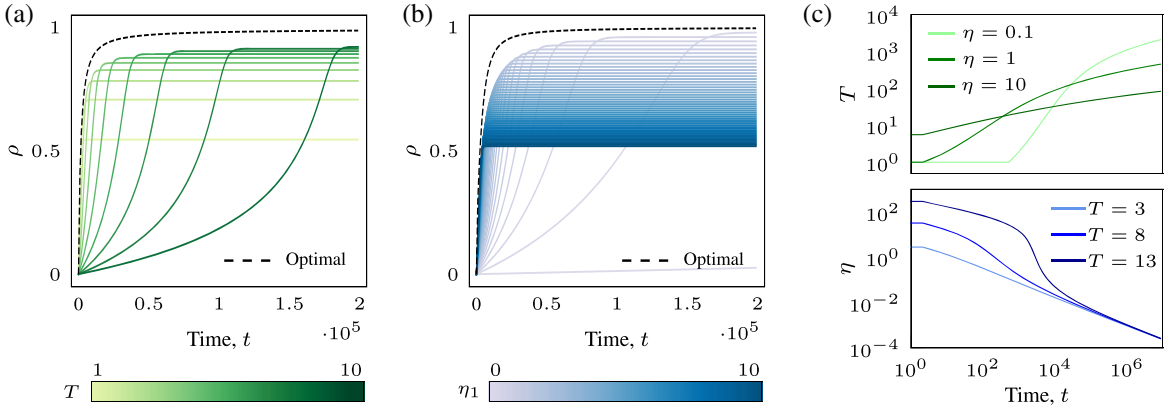

FIG. 3. Optimal schedules for episode length T and learning rate η. (a) Evolution of the normalized overlap under optimal episode length scheduling (dashed line) and various constant episode lengths (green lines). (b) Evolution of the normalized overlap under optimal learning rate scheduling (dashed line) and various constant learning rates (blue lines). (c) Evolution of optimal T (green lines) and η (blue lines) over learning. The parameters are as follows: $D = 900$, $Q = 1$, $\eta_2 = 0$, (a) $\eta_1 = 1$, and (b) $T = 8$.

schedule (dashed line) compared to other constant episode lengths (green lines). Similarly, Fig. 3(b) shows the evolution of ρ under the optimal learning rate schedule (dashed line) compared to other constant learning rates (blue lines). The functional forms of T_{opt} and η_{opt} over time are shown in Fig. 3(c).

Our analysis shows that a polynomial increase in the episode length gives the optimal performance in the RL perceptron [see Fig. 3(c) (top)]; increasing T in the RL perceptron is akin to increasing the task difficulty, and the polynomial scheduling of T_{opt} specifies a curriculum. Curricula of increasing task difficulty are commonly used in RL to give convergence speedups and learn problems that otherwise would be too difficult to learn *ab initio* [73]. Analogously, the fluctuations can be reduced by annealing the learning rate and averaging over a larger number of samples. Akin to work in the RL literature studying adaptive step sizes [74,75], we find that annealing the learning rate during training is beneficial for greater speed and generalization performance. For the RL perceptron, a polynomial decay in the learning rate gives optimal performance, as shown in Fig. 3(c) (bottom), consistent with work in the parallel area of high-dimensional nonconvex optimization problems [76] and stochastic approximation algorithms in RL [77].

D. Phase space

With a nonzero penalty (η_2), the generalization performance of the agent can enter different regimes of learning. This case is most clearly exemplified in the spherical case, where the number of fixed points of the ODE governing the dynamics of the overlap exist in distinct phases determined by the combination of reward and penalty. For the simplest case ($\mathbb{I}(\Phi) = \prod_t^T (y_t y_t^*)$), these phases are shown in Fig. 4. Figure 4(a) shows the fixed points achievable over a range of penalties for a fixed $\eta_1 = 1$ (obtained from a numerical solution of the ODE in ρ). There are two distinct regions: (1) easy, where there is a unique fixed point and the algorithm naturally converges to this optimal ρ_{fix} from a random initialization; (2) a hybrid-hard region (given the analogy with results from inference problems [78]), where there are two stable (one good and one bad) fixed points and one unstable fixed point, and either stable point is achievable depending on the initialization of the student (orange). The hybrid-hard region separates two easy regions with very distinct performance levels. In this region, the algorithm with high probability converges to ρ_{fix} with the worst performance level. These two regions are visualized in (η_1, η_2) space in Fig. 4(b) for an episode length of $T = 13$. The topology of these regions is also governed by the episode length, with a sufficiently small T reducing the area of the hybrid-hard phase to zero, meaning there is always one stable fixed point that may not necessarily give a "good" generalization. Figure 4(c) shows the phase plot for $T = 8$, where the orange (hybrid-hard) area has shrunk, which corresponds to the s-shaped curve in Fig. 4(a) becoming flatter (closer to monotonic). Details of the construction of Fig. 4 are given in Appendix E. These regimes of learnability are not a peculiarity specific to the spherical case; indeed, we observe different regimes in the learning dynamics in the setting with unrestricted Q, which we report in Appendix D. We also show that using a logistic policy with the exact REINFORCE update indeed results in the RL perceptron not being able to learn above some threshold of learning rates [see Figure 9(a)].

These phases show that, at a fixed η_1, increasing η_2 will eventually lead to a first-order phase transition, and the speed benefits gained from a nonzero η_2 will be nullified due to the transition into the hybrid-hard phase. In fact, when taking η_2 close to the transition point, instead of speeding up learning, there is a critical slowing down, which we report in Sec. III F. A common problem with

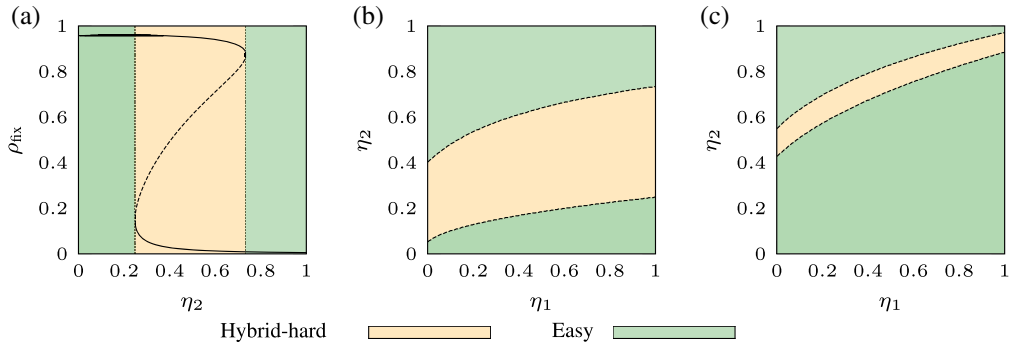

FIG. 4. Phase plots of learnability. We show the case where all decisions in an episode of length T must be correct. (a) Fixed points of ρ for $T = 13$ and $\eta_1 = 1$. The dashed portion of the line denotes where the fixed points are unstable. (b) Phase plot showing regions of hardness for $T = 13$. (c) Phase plot showing regions of hardness for $T = 8$. Green regions represent the easy phase, where, with probability 1, the algorithm naturally converges to the optimal ρ_{fix} from random initialization. The orange region indicates the hybrid-hard phase, where, with high probability, the algorithm converges to the suboptimal ρ_{fix} from random initialization. The parameters are $D = 900$ and $Q = 1$.

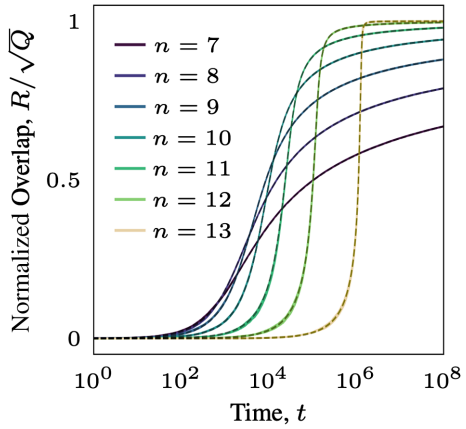

FIG. 5. Speed-accuracy trade-off. We show the evolution of the normalized overlap between the student and teacher for the simulation (solid lines) and the ODE solution (dashed lines) for the case where n or more decisions in an episode of $T = 13$ are required to be correct for an update with $\eta_2 = 0$. More stringent reward conditions slow learning but can improve performance. The parameters are as follows: $D = 900$, $\eta_1 = 1$.

REINFORCE is that high-variance gradient estimates lead to bad performances [79,80]. The reward (η_1) and punishment (η_2) magnitude alters the variance of the updates, and we show that the interplay between the reward, penalty, and reward condition, as well as their effect on performance, can be probed within our model. This framework opens the possibility for studying phase transitions between learning regimes [81].

E. Speed-accuracy trade-off

Figure 5 shows the evolution of normalized overlap $\rho = R/\sqrt{Q}$ between the student and teacher obtained from simulations and from solving the ODEs in the case where n or more decisions must be correctly made in an episode of length $T = 13$ in order to receive a reward (with $\eta_2 = 0$). We observe a speed-accuracy trade-off, where decreasing n increases the initial speed of learning but leads to worse asymptotic performance; this trade-off alleviates the initial plateau in learning seen previously in Fig. 2(d) at the cost of good generalization. In essence, a lax reward function is probabilistically more achievable early in learning; however, it rewards some fraction of incorrect decisions, leading to lower asymptotic accuracy. By contrast, a stringent reward function slows learning but eventually produces a highly aligned student. For a given MDP, it is known that arbitrary shaping applied to the reward function will change the optimal policy (reduce asymptotic performance) [82]. Empirically, reward shaping has been shown to speed up learning and help overcome difficult exploration problems [83]. Reconciling these results with the phenomena observed in our setting is an interesting avenue for future work.

F. Critical slowing down

With the addition of a penalty term, we observe an initial speedup in learning, as shown in Fig. 6. Toward the end of learning, however, we observe a critical slowing down, and we see how, in many instances, a nonzero η_2 can instead cause an overall slowing down of learning. This observation is most easily seen in the spherical case for the rule where all decisions in an episode of length T must be correct to receive a reward: Figure 6(a) shows the time taken for the student to converge to the fixed point starting from an initial $\rho = 0$ for $T = 13$ and $\eta_1 = 1$. We observe that increasing η_2 (up to η_{crit}, at which point the algorithm enters the hybrid-hard phase detailed in Sec. III D) increases the time taken to reach the fixed point, which is similarly seen for $T = 20$ in Fig. 6(b). This slowing is not present over the entire range of η_2; it is true that, for small values of η_2, there is actually a small speedup in reaching

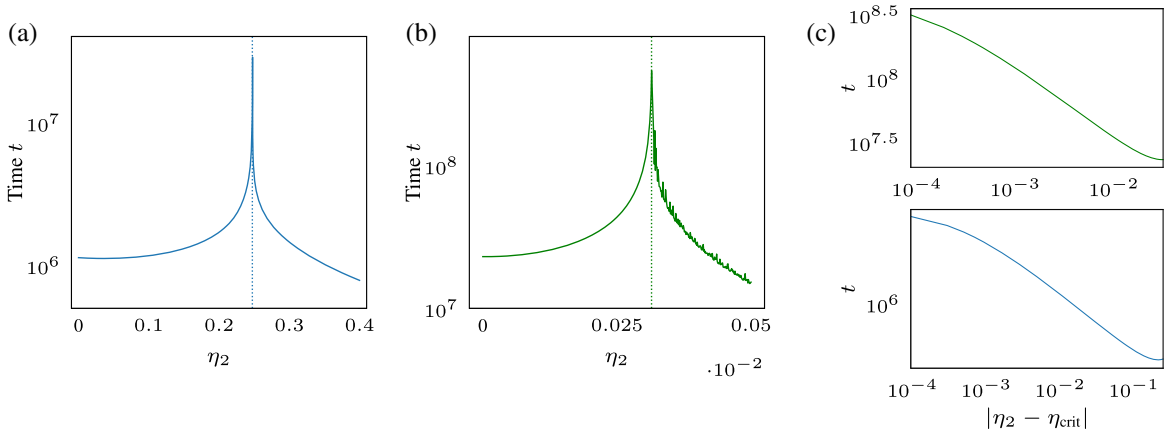

FIG. 6. Critical slowing down for $\eta_2 \neq 0$. (a) Time for convergence to the fixed point for $T = 13$. (b) Time for convergence to the fixed point for $T = 20$. (c) Time for convergence plotted against distance of η_2 away from the critical penalty for $T = 13$ (bottom) and $T = 20$ (top). All plots are for the spherical case where the agent must get every decision correct in order to receive a reward of $\eta_1 = 1$, and the agent receives a penalty of η_2 otherwise. The parameters are as follows: $D = 900$, $Q = 1$.

the fixed point, showing that the criticality severely reduces the range of η_2 that improves convergence speed. We plot the distance of η_2 away from the critical penalty value ($|\eta_2 - \eta_{\text{crit}}|$) against time for convergence in Fig. 6(c), for $T = 20$ (top) and $T = 13$ (bottom). We observe a polynomial scaling of the convergence time with distance away from criticality.

IV. EXPERIMENTS

To verify that our theoretical framework captures qualitative features of more general settings, we train agents from pixels on the Procgen [24] game Bossfight. To remain close to our theoretical setting, we consider a modified version of the game where the agent cannot defeat the enemy and only wins if it survives for a given duration T. On each time step, the agent has the binary choice of moving left or right and aims to dodge incoming projectiles. We give the agent h lives, where the agent loses a life if struck by a projectile and continues an episode if it has lives remaining. This reward structure reflects the sparse reward setup from our theory and is analogous to requiring n out of T decisions to be correct within an episode. We further add asteroids at the left and right boundaries of the playing field that destroy the agent on contact, such that the agent cannot hide in the corners [see the screenshots of the game in Fig. 7(c)]. Observations, shown in the top panel of Fig. 7(c), are centered on the agent and downsampled to size 35×64 with three color channels, yielding a 6720 dimensional input. The pixels corresponding to the agent are set to zero since these otherwise act as near-constant bias inputs not present in our model [bottom panel of Fig. 7(c)]. The agent is endowed with a shallow policy network with a logistic output unit that indicates the probability of left or right action. The weights of the policy network are trained using the exact REINFORCE policy-gradient update (with additional entropy regularization to steer away from an early deterministic policy).

To study the speed-accuracy trade-off, we train agents with different numbers of lives. As seen in Fig. 7(a), we observe a clear speed-accuracy trade-off mediated by agent health, consistent with our theoretical findings (cf. Fig. 5). Figure 7(b) shows the final policy weights for agents trained with $h = 1$ and $h = 4$. When compared to the game screenshots in Fig. 7(c), we see an interpretable structure, roughly split into thirds vertically: The weights in the top third detect the position of the boss and, in the center, the agent beneath it, which causes projectiles to arrive vertically rather than obliquely, making them easier to dodge. The weights in the middle third dodge projectiles. Finally, the weights in the bottom third avoid asteroids near the agent. Notably, the agent trained in the more stringent reward condition ($h = 1$) places greater weight on dodging projectiles [seen from the bolder colors in Fig. 7(b)], showing the qualitative impact of rewards on learned policy. Hence, similar qualitative phenomena as in our theoretical model can arise in more general settings.

For a test of the generality of our conclusions, we train agents from pixels on the ALE [25] game Pong (using the exact REINFORCE update with a deep nonlinear network). The notion of lives (or requiring n or more correct decisions in an episode for a reward) is essentially a way to control the difficulty of a task, whereby higher n (fewer lives) is a more stringent condition, i.e., a more difficult task. We examine a corresponding setup in Pong, where task difficulty is varied in order to study the dynamics of

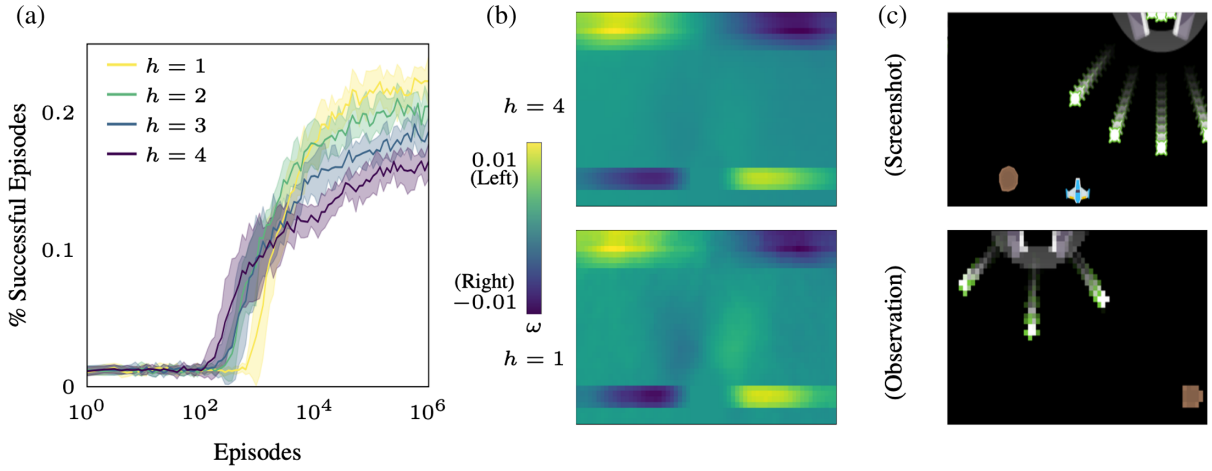

FIG. 7. Empirical speed-accuracy trade-off in Bossfight. (a) Generalization performance over training a perceptron policy network with the REINFORCE algorithm, measured on evaluation episodes with $h = 1$ lives. Agents trained in stringent conditions ($h = 1$) learn slowly but eventually outperform agents trained in lax conditions ($h = 4$), an instance of the speed-accuracy trade-off. Shaded regions indicate SEM over ten repetitions. (b) Policy network weights (ω) for an agent with (top) $h = 4$ lives and (bottom) $h = 1$ life. For simplicity, one color channel (red) is shown. Training with fewer lives increases the weight placed on dodging projectiles (see text). (c) Top panel: example screenshot of bossfight. Bottom panel: example observation that the policy network sees. The parameters are as follows: $T = 100$, $\eta_1 = 2e - 3$, and $\eta_2 = 0$.

generalization performance of agents. The Pong task difficulty is varied by changing the episode length T to which the agent must survive in order to receive a reward. Intuitively, longer T poses a more difficult task as an agent is required to survive for longer. On each time step, the agent has a binary choice of moving left or right and aims to return the ball. If the ball manages to get past the agent, the episode ends without a reward; if the agent survives until the end of the episode, it receives a reward. The decisions of the agent are sampled from the logistic output of a deep policy network, consisting of two convolutional layers, two fully connected layers, and ReLU nonlinearities with a sigmoidal output. Pong is deterministic, so in order to avoid memorization, we introduce stochasticity by employing two approaches: (1) frameskips [84], where actions are taken a random number of times; (2) random initialization, where a randomly selected, pretrained agent is run for a random number of time steps in order to progress the game into a "random" initialization state. The weights of the policy network are trained using the exact REINFORCE policy-gradient update (with additional entropy regularization) by running 20 agents in parallel. Figure 8 shows a clear trend in the speed of learning with more stringent reward conditions taking longer to learn. It is clear from Fig. 8 that asymptotic accuracy has not yet been reached within this time frame; unfortunately, because of computational and time constraints, it is not possible to train for longer. However, we can see the separation in the generalization accuracy as agents trained to survive on episodes of $T = 70$ begin to survive for longer at an increasing rate compared to agents trained on episodes of $T = 50$, which in turn begin to survive for longer at an increasing rate compared to agents trained on episodes of $T = 30$. Agents trained on episodes of $T = 90$, however, are not seen to reach generalization performance which surpasses that of $T = 70$ agents, which is likely due to not training for long enough; it can be seen that agents trained on $T = 90$ have not reached a regime where they are consistently receiving a reward yet (and, therefore, updates are still infrequent). It would require more training steps for $T = 90$ agents to more frequently receive a reward and update weights, at which point we would expect an overtake in generalization performance compared to the other agents. The inset in Fig. 8 shows the same data as the main plot but replotted against the number of total steps (steps of the agent in the game) instead of the number of episodes. The trend remains, with an exaggerated speed difference but a tighter difference in generalization performance between the different agents. In practice, choosing whether to measure speed differences in the number of episodes versus the number of steps will be application dependent. For instance, suppose the bottleneck in time cost is in taking a single step; then, step number is the appropriate measure. However, if the bottleneck in time cost is the episode itself, e.g., due to having to return a robot to the initial position, then episode number will be the appropriate measure. For a more detailed experimental setup, see Appendix F.

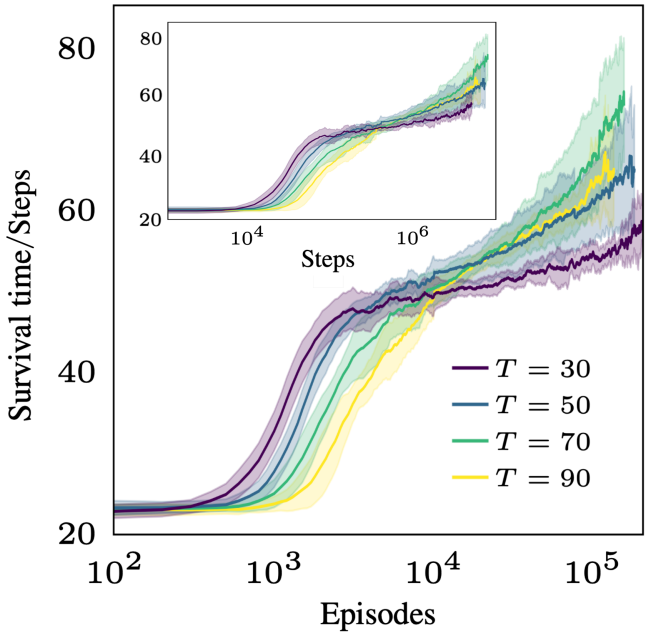

FIG. 8. Empirical speed-accuracy trade-off in Pong. The mean survival time over the course of training for agents required to survive up to completion of an episode of length T, in order to receive a reward. Inset: same data replotted for the total number of training steps taken in the game. The parameters are $\eta_1 = 2e - 3$ and $\eta_2 = 0$.

V. CONCLUDING PERSPECTIVES

The RL perceptron provides a framework to investigate high-dimensional policy-gradient learning in RL for a range of plausible sparse reward structures. We derived closed ODEs that capture the average-case learning dynamics in high-dimensional settings. The reduction of the high-dimensional learning dynamics to a low-dimensional set of differential equations permits a precise, quantitative analysis of learning behaviors: computing optimal hyperparameter schedules or tracing out phase diagrams of learnability. Our framework offers a starting point to explore additional settings that are closer to many real-world RL scenarios, such as those with conditional next states. Furthermore, the RL perceptron offers a means to study common training practices, including curricula, and more advanced algorithms, like actor-critic methods. We hope to extract more analytical insights from the ODEs, particularly on how initialization and learning rates influence an agent's learning regime. Our findings emphasize the intricate interplay of task, reward, architecture, and algorithm in modern RL systems.

We designed the RL perceptron to be the simplest possible model of reinforcement learning that lends itself to an analytical treatment, so we limited the model to binary action spaces, environmental states sampled from standard

Gaussian distributions, and simple shallow networks to learn the policy. We did not consider state transitions that are conditioned on previous action(s). While this simple model already showed a rich behavior, it is indeed possible to extend our model to problems with more states, more realistic inputs, and more decisions using universality results from statistical physics [85–90]. These extensions would make it possible to define the notion of "value" on states or actions, meaning there is the potential to incorporate value-based RL algorithms or algorithms that combine policy and value-based methods, which is a plan for future works. It would also be possible to extend to higher-dimensional action spaces instead of binary ones by considering work that finds learning curves for the multiclass perceptron [91], which would, again, widen the possibilities of RL agents we can consider and also the number of suitable environments we can test against.

We note the similarity of our sparse reward model to the policy learning framework used to fine-tune large language models (LLMs) for reasoning. A fruitful method [92] has been to train models on sample "chains of thought," where a reward is received only for a correct final result. The application of the RL perceptron to gain insight into learning of reasoning abilities in LLMs could prove useful.

Policy learning is a key aspect of modern RL, but real-world applications require learning of both policies and value functions, which evaluate policies by assigning an expected reward to each (state, action) pair. Recently, Bordelon *et al.* [60] gave an analysis of the typical-case dynamics of the temporal difference algorithm to learn value functions using tools from statistical physics. The ultimate goal of a theory of reinforcement learning combining policy learning and the learning of value functions remains elusive due to the nontrivial interactions between the two processes; thus, we are presented with an exciting challenge for further research.

ACKNOWLEDGMENTS

We would like to thank Roberta Raileanu and Tim Rocktäschel for useful discussions. This work was supported by a Sir Henry Dale Fellowship from the Wellcome Trust and Royal Society (216386/Z/19/Z) to A. S., the Sainsbury Wellcome Centre Core Grant from Wellcome (219627/Z/19/Z), and the Gatsby Charitable Foundation (GAT3755). S. G. acknowledges funding from the European Research Council under the European Union's Horizon 2020 research and innovation programme, Grant Agreement No. 101166056, and co-funding from Next Generation EU, in the context of the National Recovery and Resilience Plan, Investment PE1—Project FAIR "Future Artificial Intelligence Research," co-financed by the Next Generation EU. S. S. M. was supported by the Wallenberg AI, Autonomous Systems, and Software Program (WASP).

DATA AVAILABILITY

The data that support the findings of this article are openly available [93].

APPENDIX A: POMDP FORM

The RL perceptron with its update rule, Eq. (1), can be grounded in the (partially observable) MDP formalism where, at every time step t, the agent occupies some state s_t in the environment and receives an observation $\mathbf{x}_t$ conditioned on s_t. An action y_t is then taken by sampling from the policy $\pi(y_t|\mathbf{x}_t)$ (parametrized by the student $\mathbf{w}$), and the agent receives a reward accordingly. In the MDP framework, the observations $x_t \sim N(\mathbf{0}, \mathbb{1}_D)$ themselves are considered states. In the POMDP framework, the observations are noisy representations of some low-dimensional latent states: The state s_t of the environment can be one of two states, s_+, s_-, and $\mathbf{x}_t \sim P(\cdot|s_t)$ is a high-dimensional sample representative of the underlying state, with $P(\cdot|s_\pm) = \mathcal{N}_\pm(\cdot|\mathbf{w}^*)$. Here, $\mathcal{N}_+(\cdot|\mathbf{w}^*)$ is the $N(\mathbf{0}, \mathbb{1}_D)$ distribution but with zero-probability mass everywhere except in the half-space whose normal is parallel to $\mathbf{w}^*$; $\mathcal{N}_-(\cdot|\mathbf{w}^*)$ is correspondingly nonzero in the half-space with a normal that is antiparallel to $\mathbf{w}^*$ [$N(\mathbf{0}, \mathbb{1}_D)$ has been partitioned into two]. The next state s_{t+1} is sampled with probability $P(s_{t+1}|s_t) \equiv P(s_{t+1}) = 1/2$ independently from the decision made by the student in previous steps. At the end of an episode, after all decisions have been made, we update the agent as in Eq. (1). The POMDP and MDP frameworks are both equivalent, and they lead to the same dynamics, as shown in Appendix B.

APPENDIX B: EXACT REINFORCE SIMULATIONS

Simulations using the exact REINFORCE update were performed to show the validity of the RL-perceptron update in capturing the qualitative behavior of REINFORCE in our setting. We follow the standard RL-perceptron setup detailed in Sec. II, but in order to be able to take gradients, actions are sampled instead from a logistic policy $\pi_{\mathbf{w}}(a_t|\mathbf{x}_t) = [1 + \exp(-\Lambda a_t(\mathbf{w}^\intercal \mathbf{x}_t/\sqrt{D}))]^{-1}$ for large Λ. In the large Λ limit, the deterministic policy $\pi_{\mathbf{w}}(a_t|\mathbf{x}_t) = \frac{1}{2}(1 + a_t \mathrm{sgn}(\mathbf{w}^\intercal \mathbf{x}_t/\sqrt{D}))$ is well approximated. Instead of the RL-perceptron update, Eq. (1), we compute the true policy gradient and perform the REINFORCE update. Figure 9 shows the evolution of the normalized overlap between the teacher and student for four various learning scenarios. Figure 9(a) shows the case where a negative reward is received if not all decisions in an episode are correctly made. It is consistent with Fig. 2(a), where an increase in the size of the negative reward gives an initial speedup in learning before learning fails altogether (which is also consistent with the results of Sec. III D and Appendix D, where, above some threshold η_2, learning fails). Figure 9(b) shows the reward scheme where n or

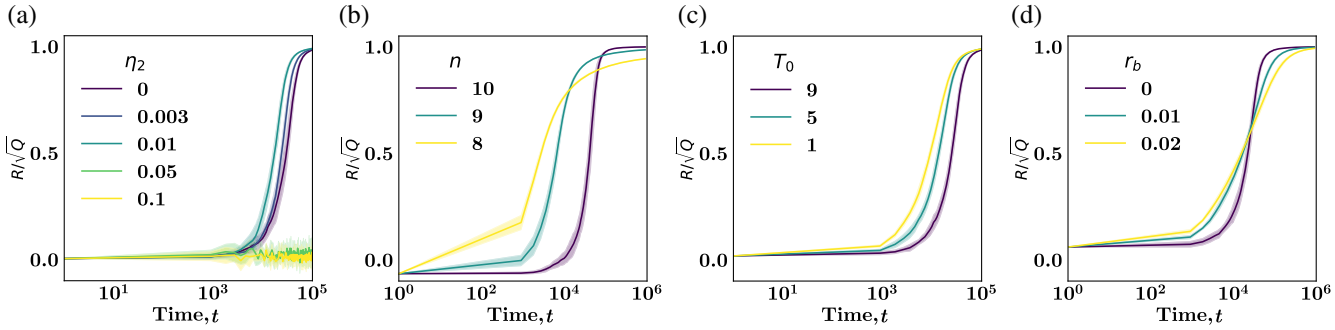

FIG. 9. Qualitatively consistent exact REINFORCE update. We show the evolution of the normalized student-teacher overlap ρ using REINFORCE for simulations with actions sampled from a logistic policy with growth parameter Λ in four reward protocols with episode length T and reward η_1. (a) All decisions required to be correct in an episode for a reward, with a negative reward η_2 otherwise. (b) n or more decisions required to be correct for a reward. (c) Additional reward of 0.2 received at time step T_0 if the agent survives beyond T_0 time steps. (d) Additional reward r_b received at time step t for every correct decision y_t made in an episode. The corresponding figures to compare for the RL-perceptron updates are as follows: Panels (a), (c), and (d) correspond to Figs. 2(a), 2(b), and 2(c), respectively; panel (b) corresponds to Fig. 5. The parameters are as follows: $\Lambda = 10000$, $T = 10$, $\eta_1 = 1$, and $D = 900$.

more decisions in an episode are required to be correct for a reward; here, there is a speed-accuracy trade-off consistent with Fig. 5. Figures 9(c) and 9(d) show the subtask and dense reward protocols (Sec. II), respectively. The shapes and ordering of the curves show good agreement with Figs. 2(b) and 2(c).

APPENDIX C: DERIVATIONS

Thermodynamic limit.—In going from the stochastic evolution of the state vector $\mathbf{w}$ to the deterministic dynamics of the order parameters, we must take the thermodynamic limit. For the ODE involving R, we must take the inner product of Eq. (1) with $\mathbf{w}^*$:

$$\mathbf{w}^{(\mu+1)} = \mathbf{w}^{(\mu)} + \frac{\eta}{T\sqrt{D}} \sum_{t=1}^{T} y_t^{(\mu)} \mathbf{x}_t^{(\mu)} G_t^{(\mu)}, \tag{C1}$$

$$DR^{(\mu+1)} = DR^{(\mu)} + \frac{\eta}{T\sqrt{D}} \sum_{t=1}^{T} y_t^{(\mu)} \mathbf{w}^{*\intercal} \mathbf{x}_t^{(\mu)} G_t^{(\mu)}. \tag{C2}$$

From this point, for ease of notation, any variable(s) with a subscript μ refers to said variable(s) in the μth episode. We subtract DR^{μ} from Eq. (C2) and sum over l episodes; the lhs is a telescopic sum, and Eq. (C2) becomes

$$\frac{D(R^{\mu+l} - R^{\mu})}{l} = \frac{\eta}{T\sqrt{D}} \frac{1}{l} \sum_{i=0}^{l-1} \left(\sum_{t=1}^{T} y_t \mathbf{w}^{*\intercal} \mathbf{x}_t G_t \right)^{\mu+i}, \tag{C3}$$

$$\frac{dR}{d\alpha} = \frac{\eta}{T} \left\langle \sum_{t=1}^{T} y_t \frac{\mathbf{w}^{*\intercal} \mathbf{x}_t}{\sqrt{D}} G_t \right\rangle \tag{C4}$$

$$= \frac{\eta}{T} \left\langle \sum_{t=1}^{T} \mathrm{sgn}(\lambda_t) \nu_t G_t \right\rangle. \tag{C5}$$

We go from Eq. (C3) to Eq. (C4) by taking the limits $D \to \infty$, $l \to \infty$ and $l/D = d\alpha \to 0$. The rhs of Eq. (C3) is the sum of a large number of random variables, and by the central limit theorem, it is self-averaging in the thermodynamic limit (under the assumption of weak correlations between episodes). Consequently, the lhs is self-averaging, and we go from Eq. (C4) to Eq. (C5) by considering the aligning fields defined in Sec. III A. A similar procedure can be followed for order parameter Q, but we instead take the square of Eq. (1) and go to the limit described, obtaining

$$\begin{aligned} DQ^{\mu+1} &= DQ^{\mu} + \frac{2\eta}{\sqrt{D}} \left(\frac{1}{T} \sum_{t=1}^{T} y_t \mathbf{w}^{\intercal} \mathbf{x}_t G_t \right)^{\mu} \\ &\quad + \frac{\eta^2}{D} \left(\frac{1}{T^2} \sum_{t,t'=1}^{T} y_t y_{t'} \mathbf{x}_t^{\intercal} \mathbf{x}_{t'} G_t G_{t'} \right)^{\mu}, \\ \frac{D(Q^{\mu+l} - Q^{\mu})}{l} &= \frac{2\eta}{T} \frac{1}{l} \sum_{i=0}^{l-1} \left(\sum_{t=1}^{T} y_t \frac{\mathbf{w}^{\intercal} \mathbf{x}_t}{\sqrt{D}} G_t \right)^{\mu+i} \\ &\quad + \frac{\eta^2}{T^2} \frac{1}{l} \sum_{i=0}^{l-1} \left(\sum_{t,t'=1}^{T} y_t y_{t'} \frac{\mathbf{x}_t^{\intercal} \mathbf{x}_{t'}}{D} G_t G_{t'} \right)^{\mu+i}, \end{aligned} \tag{C6}$$

$$\begin{aligned} \frac{dQ}{d\alpha} &= \frac{2\eta}{T} \left\langle \sum_{t=1}^{T} \mathrm{sgn}(\lambda_t) \lambda_t G_t \right\rangle \\ &\quad + \frac{\eta^2}{T^2} \left\langle \sum_{t=1}^{T} G_t^2 \right\rangle + \mathcal{O}\left(\frac{1}{D}\right). \end{aligned} \tag{C7}$$

We have now obtained a general set of ODEs describing the learning dynamics. Note that G_t is general and will depend on the environment-specific condition for a reward. For the case of a sparse reward received at the end of an episode, we find $(G_t = r_1 \mathbb{I}(\Phi(y_{1:T}, y^*_{1:T})) - r_2(1 - \mathbb{I}(\Phi(y_{1:T}, y^*_{1:T}))))$. Then, Eqs. (C5) and (C7) become

$$\frac{dR}{d\alpha} = \frac{\eta_1+\eta_2}{T}\left\langle \sum_{t=1}^{T} \nu_t \mathrm{sgn}(\lambda_t)\mathbb{I}(\Phi)\right\rangle - \eta_2\langle \nu \mathrm{sgn}(\lambda)\rangle, \quad \text{(C8)}$$

$$\begin{aligned}\frac{dQ}{d\alpha} &= \frac{2(\eta_1+\eta_2)}{T}\left\langle \sum_{t=1}^{T} \lambda_t \mathrm{sgn}(\lambda_t)\mathbb{I}(\Phi)\right\rangle - 2\eta_2\langle \lambda \mathrm{sgn}(\lambda)\rangle \\ &+ \frac{\eta_1^2-\eta_2^2}{T^2 D}\left\langle \sum_{t,t'=1}^{T} y_t y_{t'} \mathbf{x}_t^\intercal \mathbf{x}_{t'}\mathbb{I}(\Phi)\right\rangle \\ &+ \frac{\eta_2^2}{T^2 D}\left\langle \sum_{t,t'=1}^{T} y_t y_{t'} \mathbf{x}_t^\intercal \mathbf{x}_{t'}\right\rangle. \end{aligned} \quad \text{(C9)}$$

Computing averages.—Next, we compute the expectations. Recalling the definitions

$$\nu = \frac{\mathbf{w}^{*\intercal}\mathbf{x}}{\sqrt{D}} \quad \text{and} \quad \lambda = \frac{\mathbf{w}^\intercal \mathbf{x}}{\sqrt{D}}, \quad \text{(C10)}$$

which are sums of N independent terms (and by the central limit theorem, they obey a Gaussian distribution), we note

$$\langle \nu \rangle = \langle \lambda \rangle = 0, \quad \text{(C11)}$$

$$\langle \nu^2 \rangle = 1, \qquad \langle \lambda^2 \rangle = Q, \quad \text{(C12)}$$

$$\langle \nu\lambda \rangle = \frac{\mathbf{w}^{*\intercal}\mathbf{w}}{D} = R. \quad \text{(C13)}$$

All expectations in the ODEs can be expressed in terms of the constituent expectations given below (which are trivially computed by considering the Gaussianity of the above variables):

$$\begin{aligned}&\langle \nu \mathrm{sgn}(\lambda)\rangle = \sqrt{\frac{2}{\pi}}\frac{R}{\sqrt{Q}}, \qquad \langle \lambda \mathrm{sgn}(\lambda)\rangle = \sqrt{\frac{2Q}{\pi}}, \\ &\langle \nu \mathrm{sgn}(\nu)\rangle = \sqrt{\frac{2}{\pi}}, \qquad \langle \lambda \mathrm{sgn}(\nu)\rangle = \sqrt{\frac{2}{\pi}}R, \end{aligned} \quad \text{(C14)}$$

$$\begin{aligned}&\frac{1}{D}\left\langle \sum_{t,t'=1}^{T} y_t y_{t'} \mathbf{x}_t^\intercal \mathbf{x}_{t'}\right\rangle \\ &= \frac{1}{D}\left\langle \left(\sum_{t=1}^{T} \mathbf{x}_t^\intercal \mathbf{x}_t + 2\sum_{t=2}^{T}\sum_{t'=1}^{t-1} y_t y_{t'} \mathbf{x}_t^\intercal \mathbf{x}_{t'}\right)\right\rangle \\ &= T + \mathcal{O}(1/D). \end{aligned} \quad \text{(C15)}$$

The terms involving Φ will, in general, consist of expectations containing step functions $\theta(x)$ (1 for $x > 0$, 0 otherwise), specifically $\theta(\nu\lambda)$ (1 if the student decision agrees with the teacher, 0 otherwise) and $\theta(-\nu\lambda)$ (1 if the student decision disagrees with the teacher, 0 otherwise). When we encounter these terms, they can be greatly simplified by considering the following equivalences:

$$\mathrm{sgn}(\lambda)\theta(\nu\lambda) = \frac{1}{2}(\mathrm{sgn}(\lambda) + \mathrm{sgn}(\nu)), \quad \text{(C16)}$$

$$\text{and} \quad \mathrm{sgn}(\lambda)\theta(-\nu\lambda) = \frac{1}{2}(\mathrm{sgn}(\lambda) - \mathrm{sgn}(\nu)). \quad \text{(C17)}$$

We show, as an example, the case where Φ is the condition to get all decisions correct in an episode, $\mathbb{I}(\Phi) = \prod_{t=1}^{T}\theta(\nu_t\lambda_t)$, where $\theta(x)$ is the step function (1 for $x > 0$, 0 otherwise). The first term in Eq. (C8) can be given as

$$\begin{aligned}&\left\langle \frac{1}{T}\sum_{t=1}^{T} \nu_t \mathrm{sgn}(\lambda_t)\mathbb{I}(\Phi)\right\rangle \\ &\to \left\langle \frac{1}{T}\sum_{t=1}^{T} \nu_t \mathrm{sgn}(\lambda_t)\prod_{s=1}^{T}\theta(\nu_s\lambda_s)\right\rangle \quad \text{(C18)} \\ &= \langle \nu_t \mathrm{sgn}(\lambda_t)\theta(\nu_t\lambda_t)\rangle\left\langle \prod_{s\neq t}^{T}\theta(\nu_s\lambda_s)\right\rangle \quad \text{(C19)} \\ &= \frac{1}{2}\langle \nu_t(\mathrm{sgn}(\lambda_t) + \mathrm{sgn}(\nu_t))\rangle P^{T-1} \quad \text{(C20)} \\ &= \frac{1}{\sqrt{2\pi}}\left(1 + \frac{R}{\sqrt{Q}}\right)P^{T-1}, \quad \text{(C21)} \end{aligned}$$

where P is the probability of making a single correct decision, and it can be calculated by considering that an incorrect decision is made if $\mathbf{x}$ lies in the hypersectors defined by the intersection of $\mathcal{N}_\pm(\cdot|\mathbf{w}^*)$ and $\mathcal{N}_\pm(\cdot|\mathbf{w})$; the angle ϵ subtended by these hypersectors is equal to the angle between $\mathbf{w}^*$ and $\mathbf{w}$,

$$P = \left(1 - \frac{\epsilon}{\pi}\right) = \left(1 - \frac{1}{\pi}\cos^{-1}\left(\frac{R}{\sqrt{Q}}\right)\right). \quad \text{(C22)}$$

Similarly, the first term in Eq. (C9) can be given as

$$\begin{aligned}&\left\langle \frac{2}{T}\sum_{t=1}^{T} \lambda_t \mathrm{sgn}(\lambda_t)\prod_{s=1}^{T}\theta(\nu_s\lambda_s)\right\rangle \\ &= \langle \lambda_t(\mathrm{sgn}(\lambda_t) + \mathrm{sgn}(\nu_t))\rangle P^{T-1} \\ &= \sqrt{\frac{2Q}{\pi}}\left(1 + \frac{R}{\sqrt{Q}}\right)P^{T-1}. \end{aligned} \quad \text{(C23)}$$

The cross terms in Eq. (C9) can also be computed:

$$\begin{aligned}&\frac{1}{D}\left\langle \sum_{t,t'=1}^{T} y_t y_{t'} \mathbf{x}_t^\intercal \mathbf{x}_{t'}\prod_{s=1}^{T}\theta(\nu_s\lambda_s)\right\rangle \\ &= \frac{1}{D}\left\langle \left(\sum_{t=1}^{T} \mathbf{x}_t^\intercal \mathbf{x}_t + 2\sum_{t=2}^{T}\sum_{t'=1}^{t-1} y_t y_{t'} \mathbf{x}_t^\intercal \mathbf{x}_{t'}\right)\prod_{s=1}^{T}\theta(\nu_s\lambda_s)\right\rangle \\ &= TP^T + \mathcal{O}(1/D), \end{aligned} \quad \text{(C24)}$$

where the second term can be neglected in the high-dimensional limit. Substituting these computed averages

into Eqs. (C8) and (C9), the ODEs for the order parameters can be written:

$$\frac{dR}{d\alpha} = \frac{\eta_1+\eta_2}{\sqrt{2\pi}}\left(1+\frac{R}{\sqrt{Q}}\right)P^{T-1} - \eta_2 R\sqrt{\frac{2}{\pi Q}}, \tag{C25}$$

$$\frac{dQ}{d\alpha} = (\eta_1+\eta_2)\sqrt{\frac{2Q}{\pi}}\left(1+\frac{R}{\sqrt{Q}}\right)P^{T-1} - 2\eta_2\sqrt{\frac{2Q}{\pi}} + \frac{(\eta_1^2-\eta_2^2)}{T}P^T + \frac{\eta_2^2}{T}. \tag{C26}$$

Equivalence of POMDP formulation.—The ODEs governing the dynamics of the order parameters in the previous section can be equivalently calculated under the formulation involving the underlying states $\{s_+, s_-\}$ defined in Sec. II. The underlying system can take a multitude of trajectories (τ) in state space; there are 2^T trajectories in total (as the system can be in two possible states at each time step), and expectations must now include the averaging over all possible trajectories. All expectations will now be of the following form, where the dot $(\cdot)$ denotes some arbitrary term to be averaged over:

$$\langle\cdot\rangle = \sum_\tau \mathrm{P}(\tau)\langle\cdot|\tau\rangle. \tag{C27}$$

By considering the symmetry of the Gaussian and "half-Gaussian" ($\mathcal{N}_\pm$) distributions, all expectations in Eq. (C14) are seen to be identical regardless of whether expectations are taken with respect to the full Gaussian or the half-Gaussian distributions, i.e.,

$$\langle\cdot\rangle_{\mathcal{N}} = \langle\cdot\rangle_{\mathcal{N}_+} = \langle\cdot\rangle_{\mathcal{N}_-}. \tag{C28}$$

This finding implies that all expectations are independent of the trajectory of the underlying system; hence, averaging over all trajectories leaves all expectations unchanged. This approach also allows the extension to arbitrary transition probabilities between the underlying states $\{s_+, s_-\}$.

Other reward structures.—The expectations can be calculated in other conditions of Φ by considering combinatorial arguments. We state the ODEs for two reward conditions.

n or more.—We consider the case where Φ is the requirement of getting n or more decisions in an episode of length T correct. We give the ODEs below for the case of $\eta_2 = 0$:

$$\frac{dR}{d\alpha} = \frac{\eta_1}{T\sqrt{2\pi}}\sum_{i=n}^{T}\binom{T}{i}\left[i\left(1+\frac{R}{\sqrt{Q}}\right)(1-P) - (T-i)\left(1-\frac{R}{\sqrt{Q}}\right)P\right]P^{i-1}(1-P)^{T-i-1}, \tag{C29}$$

$$\frac{dQ}{d\alpha} = \frac{\eta_1}{T}\sqrt{\frac{2Q}{\pi}}\sum_{i=n}^{T}\binom{T}{i}\left[i\left(1+\frac{R}{\sqrt{Q}}\right)(1-P) - (T-i)\left(1-\frac{R}{\sqrt{Q}}\right)P\right]P^{i-1}(1-P)^{T-i-1} + \frac{\eta_1^2}{T}\sum_{i=n}^{T}\binom{T}{i}P^i(1-P)^{T-i}. \tag{C30}$$

Breadcrumb trails.—We also consider the case where a reward of size η_1 is received if all decisions in an episode are correct, in addition to a smaller reward of size β for each individual decision correctly made in an episode:

$$\frac{dR}{d\alpha} = \frac{1}{\sqrt{2\pi}}\left[\left(1+\frac{R}{\sqrt{Q}}\right)(\eta_1 P^{T-1}+\beta) + \beta(T-1)\frac{R}{\sqrt{Q}}P\right],$$

$$\frac{dQ}{d\alpha} = \sqrt{\frac{2Q}{\pi}}\left[\left(1+\frac{R}{\sqrt{Q}}\right)(\eta_1 P^{T-1}+\beta) + 2\beta(T-1)P\right] + (\eta_1^2 + (T+1)\eta_1\beta)\frac{P^T}{T} + \beta^2(T+1)\left(\frac{1}{2}+\frac{1}{3}(T-1)P\right)\frac{P}{T}. \tag{C31}$$

APPENDIX D: UNCONSTRAINED SIMULATIONS

Optimal scheduling.—The optimal schedules for the learning rate and episode length (Sec. III C) also hold in the unconstrained case [where $Q(\alpha)$ is not restricted to the surface of a sphere] because the parameters were derived from the general requirement of extremizing the update of ρ from any point in the (ρ, Q) plane. The evolution of T_{opt} and η_{opt} over time (while following their respective scheduling) is shown in Fig. 10. In the unconstrained case, the magnitude of the student grows quadratically; an increase Q acts as a decrease in the effective learning rate. Hence, contrary to the spherical case, a decaying learning rate is not optimal, and optimal T grows much more slowly, as shown in Fig. 10(b). The plots for T_{opt} do not show a clear trend, and they require further investigation. The evolution of $\eta_{\mathrm{opt}}/\sqrt{Q}$ is plotted in Fig. 9(a); this value is the effective learning rate, and we observe a polynomial decay in the value as with the spherical case presented in Sec. III C.

Phases.—The phases observed in Fig. 4 are not an artifact of the spherical case. When $Q(\alpha)$ is not constrained, we also observe regimes where a "bad" fixed point of ρ may be attained. Figure 11 shows flow diagrams in the (ρ, Q) plane for various parameter instantiations in the case where a reward of $\eta_1 = 1$ is received if all decisions in an episode of length $T = 8$ are correctly made and a penalty of η_2 otherwise. Figure 11(a) is the flow diagram for $\eta_2 = 0$;

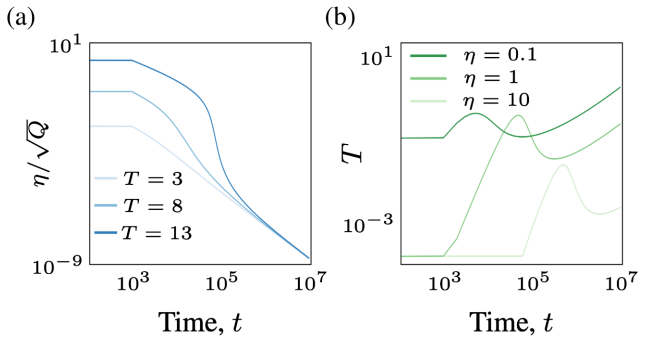

FIG. 10. Optimal schedules for unconstrained students. Panels (a) and (b) show the evolution of optimal η and T, respectively, over learning, while following the specified optimal schedule, over a range of rewards and episode lengths. The parameters are $D = 900$, $\eta_2 = 0$, (a) $T = 8$, and (b) $\eta = 1$.

in this regime, the agent can always perfectly align with the teacher from any initialization (the student flows to $\rho = 1$ at $Q = \infty$). This case is analogous to the student being in the easy phase in the lower green region of the plot in Fig. 4(b), as with probability 1, the algorithm naturally converges to the optimal $\rho = 1$. Figure 11(b) shows the flow for $\eta_2 = 0.05$. In this regime, we observe the flow to some suboptimal ρ at $Q = \infty$, which is analogous to the student being in the easy phase in the top of the plot in Fig. 4(b), as with probability 1, the algorithm converges to a value of ρ from any initialization. However, this value of ρ is suboptimal. Figure 11(c) shows the flow for $\eta = 0.045$. We see that, depending on the initial ρ, the agent will flow to one of two fixed points in ρ at $Q = \infty$; this case is analogous to the agent being in the hybrid-hard phase in Fig. 4(b), where, with high probability, the agent converges to the worse ρ. The "good easy phase," characterizing the behavior seen in Fig. 11(a), is indicated by the green region in Fig. 11(d).

APPENDIX E: PHASE PORTRAIT CONSTRUCTION

In the spherical case [constant $Q(\alpha) = Q$] with $\mathbb{I}(\Phi) = \prod_t^T \theta(y_t y_t^*)$, the ODE governing the evolution of normalized overlap ρ is

$$\frac{d\rho}{d\alpha} = \frac{\eta_1 + \eta_2}{\sqrt{2\pi Q}}\left(1 - \frac{1}{\pi}\cos^{-1}(\rho)\right)^{T-1}\left[1-\rho^2 - \frac{\eta_1 - \eta_2}{T}\sqrt{\frac{\pi}{2}}\frac{\rho}{\sqrt{Q}}\left(1 - \frac{1}{\pi}\cos^{-1}(\rho)\right)\right] - \frac{\eta_2^2}{2TQ}\rho. \quad \text{(E1)}$$

The fixed points of this equation for $Q = 1$ were found by numerically solving for $(d\rho/d\alpha) = 0$. We observe that there are always one or three fixed points. From observing the sign of $d\rho/d\alpha$ on either side of the fixed points, we see that, if there is one fixed point, it is stable; if there are three fixed points, the outermost points are stable, but the innermost fixed point, sandwiched between the first and third points, is unstable. We then construct the phase plots in Fig. 4 by sweeping over (η_1, η_2) values and counting the number of fixed points—the yellow hybrid-hard region then corresponds to a region with three fixed points in Eq. (E1).

APPENDIX F: ADDITIONAL EXPERIMENTAL DETAILS

Instructions for running simulations and experiments can be found in [93]. In both the Bossfight and Pong experiments, the loss was augmented to have an entropy regularization term weighted by $\beta = 0.01$ to prevent early convergence to a deterministic policy. Note that β is multiplied by a factor of 0.995 after each episode to gradually decrease the regularization-term contribution.

Bossfight.—The policy network was optimized with Adam, with a learning rate of 2×10^{-3}. We discounted

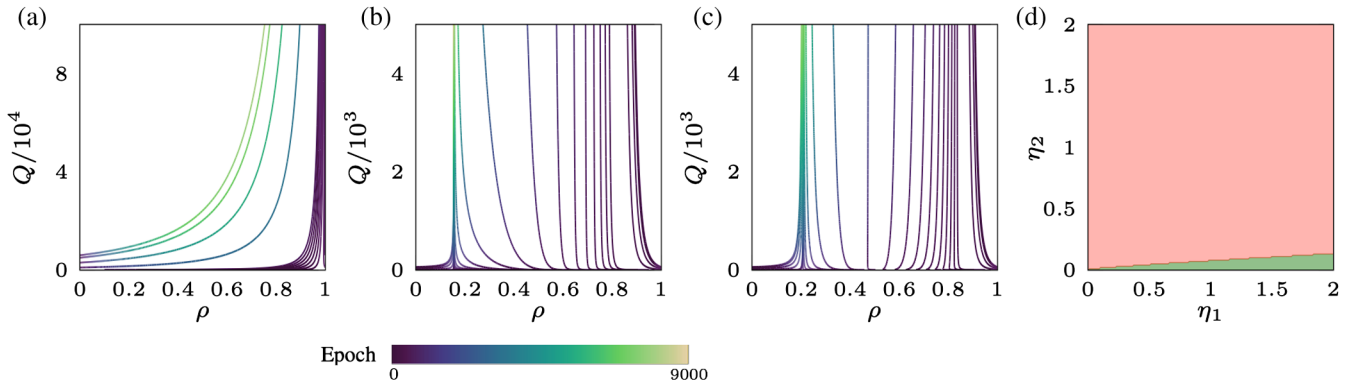

FIG. 11. Unconstrained flow and phase plots for increasing size of negative reward. We show the flow in the (ρ, Q) plane (flow goes in the direction of increasing Q) for the case where all decisions in an episode are required to be correct for a reward of $\eta_1 = 1$; otherwise, there are penalties of (a) $\eta_2 = 0$, (b) 0.05, and (c) 0.045. (d) Phase plot showing the region where learning failed (red) and succeeded (green) over the (η_1, η_2) plane, for the same learning rule. The parameters are initialized from $\rho = 0$ and $Q = 1$.

rewards with a discount factor of $\gamma = 0.999$. The calculated returns (G_t) over an episode were normalized by centering around the episode mean and dividing by the episode standard deviation (a standard practice for stability). Ten parallel environments (all using the same policy network for actions) were used to collect trajectories for a mean update. Generalization performance was calculated at preset intervals by taking the mean of 1000 environments running in parallel (all acting on the same policy at the same time point in training). Learning curves for ten separately trained agents were obtained in order to calculate the mean and standard deviation, as plotted in Fig. 7(a).

Pong.—For enhanced stochasticity to prevent memorization of the game, random initialization was implemented. Here, before each episode, one of ten pretrained agents were randomly chosen; then, the environment was run for a random number of time steps (randomly sampled between 10 and 55 inclusive) while acting under the policies of the pretrained agents in order to provide random initializations. We also implemented frameskip, where every action taken would be taken a random number of times (sampled between 1 and 5 inclusive). The policy network was optimized with Adam, with a learning rate of 2×10^{-3}. We discounted rewards with a discount factor of $\gamma = 1$ (i.e., no discount). The returns (G_t) over an episode were normalized by centering around the episode mean and dividing by the episode standard deviation. Sixteen parallel environments (all using the same policy network for actions) were used to collect trajectories for a mean update. Generalization performance was calculated at preset intervals by taking the mean performance of eight environments running in parallel (all acting on the same policy at the same time point in training), then repeating this 20 times and taking the mean. Learning curves for 20 separately trained agents were obtained in order to calculate the mean and standard deviation, which was plotted in Fig. 8.

[1] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, *Mastering the game of Go with deep neural networks and tree search*, Nature (London) **529**, 484 (2016).

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, *Human-level control through deep reinforcement learning*, Nature (London) **518**, 529 (2015).

[3] K. Yu, K. Jin, and X. Deng, *Review of deep reinforcement learning*, in *2022 IEEE 5th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, Vol. 5 (2022), pp. 41–48.

[4] R. Kirk, A. Zhang, E. Grefenstette, and T. Rocktäschel, *A survey of zero-shot generalisation in deep reinforcement learning*, J. Artif. Intell. Res. **76**, 201 (2023).

[5] H. S. Seung, H. Sompolinsky, and N. Tishby, *Statistical mechanics of learning from examples*, Phys. Rev. A **45**, 6056 (1992).

[6] A. Engel and C. Van den Broeck, *Statistical Mechanics of Learning* (Cambridge University Press, Cambridge, England, 2001).

[7] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová, *Machine learning and the physical sciences*, Rev. Mod. Phys. **91**, 045002 (2019).

[8] Y. Bahri, J. Kadmon, J. Pennington, S. S. Schoenholz, J. Sohl-Dickstein, and S. Ganguli, *Statistical mechanics of deep learning*, Annu. Rev. Condens. Matter Phys. **11**, 501 (2020).

[9] M. Gabrié, S. Ganguli, C. Lucibello, and R. Zecchina, *Neural networks: From the perceptron to deep nets*, arXiv:2304.06636.

[10] L. Saglietti, S. S. Mannelli, and A. Saxe, *An analytical theory of curriculum learning in teacher–student networks*, J. Stat. Mech. (2022) 114014.

[11] H. Asanuma, S. Takagi, Y. Nagano, Y. Yoshida, Y. Igarashi, and M. Okada, *Statistical mechanical analysis of catastrophic forgetting in continual learning with teacher and student networks*, J. Phys. Soc. Jpn. **90**, 104001 (2021).

[12] S. Lee, S. Goldt, and A. M. Saxe, *Continual learning in the teacher-student setup: Impact of task similarity*, in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, virtual event*, in Proceedings of Machine Learning Research, Vol. 139, edited by M. Meila and T. Zhang, pp. 6109–6119.

[13] S. Lee, S. S. Mannelli, C. Clopath, S. Goldt, and A. Saxe, *Maslow's hammer for catastrophic forgetting: Node re-use vs node activation*, arXiv:2205.09029.

[14] B. Sorscher, S. Ganguli, and H. Sompolinsky, *Neural representational geometry underlies few-shot concept learning*, Proc. Natl. Acad. Sci. U.S.A. **119**, e2200800119 (2022).

[15] A. K. Lampinen and S. Ganguli, *An analytic theory of generalization dynamics and transfer learning in deep linear networks*, in *7th International Conference on Learning Representations, ICLR, New Orleans*, 2019.

[16] O. Dhifallah and Y. M. Lu, *Phase transitions in transfer learning for high-dimensional perceptrons*, Entropy **23**, 400 (2021).

[17] F. Gerace, L. Saglietti, S. S. Mannelli, A. Saxe, and L. Zdeborová, *Probing transfer learning with a model of synthetic correlated datasets*, Mach. Learn. **3**, 015030 (2022).

[18] R. S. Sutton, D. Mcallester, S. Singh, and Y. Mansour, *Policy gradient methods for reinforcement learning with function approximation*, in *Advances in Neural Information Processing Systems*, Vol. 12 (MIT Press, Cambridge, MA, 2000), pp. 1057–1063.

[19] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, *Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor*, arXiv:1801.01290.

[20] G. Barth-Maron, M. W. Hoffman, D. Budden, W. Dabney, D. Horgan, D. TB, A. Muldal, N. Heess, and T. Lillicrap, *Distributional policy gradients*, in *International Conference on Learning Representations* (2018).

[21] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, *Proximal policy optimization algorithms*, arXiv:1707.06347.

[22] E. Gardner and B. Derrida, *Three unfinished works on the optimal storage capacity of networks*, J. Phys. A **22**, 1983 (1989).

[23] D. Saad and S. A. Solla, *On-line learning in soft committee machines*, Phys. Rev. E **52**, 4225 (1995).

[24] K. Cobbe, C. Hesse, J. Hilton, and J. Schulman, *Leveraging procedural generation to benchmark reinforcement learning*, in *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, virtual event*, Proceedings of Machine Learning Research, Vol. 119, pp. 2048–2056.

[25] M. C. Machado, M. G. Bellemare, E. Talvitie, J. Veness, M. J. Hausknecht, and M. Bowling, *Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents*, J. Artif. Intell. Res. **61**, 523 (2018).

[26] M. G. Azar, I. Osband, and R. Munos, *Minimax regret bounds for reinforcement learning*, in *Proceedings of the 34th International Conference on Machine Learning, ICML, Sydney, NSW, Australia, 2017*, Proceedings of Machine Learning Research, Vol. 70, edited by D. Precup and Y. W. Teh, pp. 263–272.

[27] Z. Zhang, Y. Zhou, and X. Ji, *Almost optimal model-free reinforcement learning via reference-advantage decomposition*, in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, virtual*, edited by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin.

[28] S. Dong, B. Van Roy, and Z. Zhou, *Provably efficient reinforcement learning with aggregated states*, arXiv:1912.06366.

[29] C. Jin, Z. Yang, Z. Wang, and M. I. Jordan, *Provably efficient reinforcement learning with linear function approximation*, in *Conference on Learning Theory, COLT 2020, Graz, Austria, virtual event, Proceedings of Machine Learning Research*, Vol. 125, edited by J. D. Abernethy and S. Agarwal, pp. 2137–2143.

[30] L. Yang and M. Wang, *Sample-optimal parametric q-learning using linearly additive features*, in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, Long Beach, California, USA*, Proceedings of Machine Learning Research, Vol. 97, edited by K. Chaudhuri and R. Salakhutdinov, pp. 6995–7004.

[31] A. Modi, N. Jiang, A. Tewari, and S. P. Singh, *Sample complexity of reinforcement learning using linearly combined model ensembles*, in *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, Palermo, Sicily, Italy, virtual event*, Proceedings of Machine Learning Research, Vol. 108, edited by S. Chiappa and R. Calandra, pp. 2010–2020.

[32] A. Ayoub, Z. Jia, C. Szepesvári, M. Wang, and L. Yang, *Model-based reinforcement learning with value-targeted regression*, in *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, virtual event*, Proceedings of Machine Learning Research, Vol. 119, pp. 463–474.

[33] S. S. Du, A. Krishnamurthy, N. Jiang, A. Agarwal, M. Dudík, and J. Langford, *Provably efficient RL with rich observations via latent state decoding*, in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, Long Beach, California, USA*, Proceedings of Machine Learning Research, Vol. 97, edited by K. Chaudhuri and R. Salakhutdinov, pp. 1665–1674.

[34] X. Zhang, Y. Song, M. Uehara, M. Wang, A. Agarwal, and W. Sun, *Efficient reinforcement learning in block mdps: A model-free representation learning approach*, in *International Conference on Machine Learning, ICML 2022, Baltimore, Maryland, USA*, Proceedings of Machine Learning Research, Vol. 162, edited by K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvári, G. Niu, and S. Sabato, pp. 26517–26547.

[35] A. Krishnamurthy, A. Agarwal, and J. Langford, *PAC reinforcement learning with rich observations*, in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, Barcelona, Spain*, edited by D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, pp. 1840–1848.

[36] A. Agarwal, S. M. Kakade, A. Krishnamurthy, and W. Sun, *FLAMBE: Structural complexity and representation learning of low rank MDPs*, in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, virtual event*, edited by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin.

[37] P. L. Bartlett and S. Mendelson, *Rademacher and Gaussian complexities: Risk bounds and structural results*, in *Computational Learning Theory, 14th Annual Conference on Computational Learning Theory, COLT 2001 and 5th European Conference on Computational Learning Theory, EuroCOLT 2001, Amsterdam, The Netherlands, 2001*, Lecture Notes in Computer Science, Vol. 2111, edited by D. P. Helmbold and R. C. Williamson (Springer, New York, 2001), pp. 224–240.

[38] V. N. Vapnik and A. Y. Chervonenkis, *On the uniform convergence of relative frequencies of events to their probabilities, Measures of Complexity: Festschrift for Alexey Chervonenkis* (Springer Cham, Switzerland, 2015), p. 11.

[39] N. Jiang, A. Krishnamurthy, A. Agarwal, J. Langford, and R. E. Schapire, *Contextual decision processes with low bellman rank are pac-learnable*, in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia*, Proceedings of Machine Learning Research, Vol. 70, edited by D. Precup and Y. W. Teh, pp. 1704–1713.

[40] D. Russo and B. V. Roy, *Eluder dimension and the sample complexity of optimistic exploration*, in *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013, Lake Tahoe, Nevada, United States*, edited by C. J. C. Burges, L. Bottou, Z. Ghahramani, and K. Q. Weinberger, pp. 2256–2264.

[41] C. Jin, Q. Liu, and S. Miryoosefi, *Bellman Eluder dimension: New rich classes of RL problems, and sample-efficient algorithms*, in *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021,*

virtual, edited by M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, pp. 13406–13418.

[42] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, *Understanding deep learning (still) requires rethinking generalization*, Commun. ACM **64**, 107 (2021).

[43] J. Bhandari and D. Russo, *Global optimality guarantees for policy gradient methods*, arXiv:1906.01786.

[44] A. Agarwal, S. M. Kakade, J. D. Lee, and G. Mahajan, *On the theory of policy gradient methods: Optimality, approximation, and distribution shift*, J. Mach. Learn. Res. **22**, 98 (2021).

[45] Q. Cai, Z. Yang, J. D. Lee, and Z. Wang, *Neural temporal-difference learning converges to global optima*, in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, Vancouver, BC, Canada*, edited by H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, pp. 11312–11322.

[46] Y. Zhang, Q. Cai, Z. Yang, Y. Chen, and Z. Wang, *Can temporal-difference and q-learning learn representation? A mean-field theory*, in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, virtual event*, edited by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin.

[47] A. Agazzi and J. Lu, *Global optimality of softmax policy gradient with single hidden layer neural networks in the mean-field regime*, in *9th International Conference on Learning Representations, ICLR 2021, Austria, virtual event*.

[48] A. Agazzi and J. Lu, *Temporal-difference learning with nonlinear function approximation: Lazy training and mean field regimes*, in *Proceedings of the 2nd Mathematical and Scientific Machine Learning Conference*, Proceedings of Machine Learning Research, Vol. 145, edited by J. Bruna, J. Hesthaven, and L. Zdeborova, pp. 37–74.

[49] A. Jacot, C. Hongler, and F. Gabriel, *Neural tangent kernel: Convergence and generalization in neural networks*, in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, Montréal, Canada*, edited by S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, pp. 8580–8589.

[50] S. S. Du, X. Zhai, B. Póczos, and A. Singh, *Gradient descent provably optimizes over-parameterized neural networks*, in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA*.

[51] L. Chizat, E. Oyallon, and F. R. Bach, *On lazy training in differentiable programming*, in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, Vancouver, BC, Canada*, edited by H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, pp. 2933–2943.

[52] C. Lyle, M. Rowland, W. Dabney, M. Kwiatkowska, and Y. Gal, *Learning dynamics and generalization in deep reinforcement learning*, in *Proceedings of the 39th International Conference on Machine Learning, Proceedings of Machine Learning Research*, edited by K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato (PMLR, 2022), Vol. 162, pp. 14560–14581.

[53] B. Ghorbani, S. Mei, T. Misiakiewicz, and A. Montanari, *Limitations of lazy training of two-layers neural network*, in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, Vancouver, BC, Canada*, edited by H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, pp. 9108–9118.

[54] B. Ghorbani, S. Mei, T. Misiakiewicz, and A. Montanari, *When do neural networks outperform kernel methods?*, in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, virtual event*, edited by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin.

[55] L. Chizat and F. R. Bach, *Implicit bias of gradient descent for wide two-layer neural networks trained with the logistic loss*, in *Conference on Learning Theory, COLT 2020, Graz, Austria, virtual event*, Proceedings of Machine Learning Research, Vol. 125, edited by J. D. Abernethy and S. Agarwal, pp. 1305–1338.

[56] M. Refinetti, S. Goldt, F. Krzakala, and L. Zdeborová, *Classifying high-dimensional Gaussian mixtures: Where kernel methods fail and neural networks succeed*, in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, virtual event*, Proceedings of Machine Learning Research, Vol. 139, edited by M. Meila and T. Zhang, pp. 8936–8947.

[57] S. Mei, A. Montanari, and P.-M. Nguyen, *A mean field view of the landscape of two-layer neural networks*, Proc. Natl. Acad. Sci. U.S.A. **115**, E7665 (2018).

[58] L. Chizat and F. R. Bach, *On the global convergence of gradient descent for over-parameterized models using optimal transport*, in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, Montréal, Canada*, edited by S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, pp. 3040–3050.

[59] G. M. Rotskoff and E. Vanden-Eijnden, *Parameters as interacting particles: Long time convergence and asymptotic error scaling of neural networks*, in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, Montréal, Canada*, edited by S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, pp. 7146–7155.

[60] B. Bordelon, P. Masset, H. Kuo, and C. Pehlevan, *Loss dynamics of temporal difference reinforcement learning*, in Thirty-seventh Conference on Neural Information Processing Systems (2023).

[61] R. Rubin, R. Monasson, and H. Sompolinsky, *Theory of spike timing-based neural classifiers*, Phys. Rev. Lett. **105**, 218102 (2010).

[62] R. Gütig and H. Sompolinsky, *The tempotron: A neuron that learns spike timing–based decisions*, Nat. Neurosci. **9**, 420 (2006).

[63] S. Franz and G. Parisi, *The simplest model of jamming*, J. Phys. A **49**, 145001 (2016).

[64] F. Rosenblatt, *Principles of Neurodynamics* (Spartan, New York, 1962).

[65] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction* (A Bradford Book, Cambridge, MA, 2018).
[66] W. Kinzel and P. Ruján, *Improving a network generalization ability by selecting examples*, Europhys. Lett. **13**, 473 (1990).
[67] M. Biehl and H. Schwarze, *Learning by on-line gradient descent*, J. Phys. A **28**, 643 (1995).
[68] S. Goldt, M. Advani, A. M. Saxe, F. Krzakala, and L. Zdeborová, *Dynamics of stochastic gradient descent for two-layer neural networks in the teacher-student setup*, in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, Vancouver, BC, Canada*, edited by H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, pp. 6979–6989.
[69] R. Veiga, L. STEPHAN, B. Loureiro, F. Krzakala, and L. Zdeborova, *Phase diagram of stochastic gradient descent in high-dimensional two-layer neural networks*, in *Advances in Neural Information Processing Systems*, edited by A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho (2022).
[70] L. Arnaboldi, L. Stephan, F. Krzakala, and B. Loureiro, *From high-dimensional & mean-field dynamics to dimensionless ODEs: A unifying approach to SGD in two-layers networks*, arXiv:2302.05882.
[71] G. Vasan, Y. Wang, F. Shahriar, J. Bergstra, M. Jagersand, and A. R. Mahmood, *Revisiting sparse rewards for goal-reaching reinforcement learning*, arXiv:2407.00324.
[72] Y. N. Dauphin, R. Pascanu, Ç. Gülçehre, K. Cho, S. Ganguli, and Y. Bengio, *Identifying and attacking the saddle point problem in high-dimensional non-convex optimization*, in *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, Montreal, Quebec, Canada*, edited by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, pp. 2933–2941.
[73] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone, *Curriculum learning for reinforcement learning domains: A framework and survey*, J. Mach. Learn. Res. **21**, 181 (2020).
[74] W. Dabney and A. Barto, *Adaptive step-sizes for reinforcement learning*, AAAI **26**, 872 (2021).
[75] M. Pirotta, M. Restelli, and L. Bascetta, *Adaptive step-size for policy gradient methods*, in *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013, Lake Tahoe, Nevada, United States*, edited by C. J. C. Burges, L. Bottou, Z. Ghahramani, and K. Q. Weinberger, pp. 1394–1402.
[76] S. d'Ascoli, M. Refinetti, and G. Biroli, *Optimal learning rate schedules in high-dimensional non-convex optimization problems*, arXiv:2202.04509.
[77] G. Dalal, B. Szörényi, G. Thoppe, and S. Mannor, *Concentration bounds for two timescale stochastic approximation with applications to reinforcement learning*, arXiv:1703.05376.
[78] F. Ricci-Tersenghi, G. Semerjian, and L. Zdeborová, *Typology of phase transitions in Bayesian inference problems*, Phys. Rev. E **99**, 042109 (2019).
[79] P. Marbach and J. N. Tsitsiklis, *Approximate gradient methods in policy-space optimization of Markov reward processes*, Discrete Event Dynamic Systems: Theory and Applications;Designs, Codes and Cryptography **13**, 111 (2003).
[80] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel, *High-dimensional continuous control using generalized advantage estimation*, in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, 2016, Conference Track Proceedings*, edited by Y. Bengio and Y. LeCun.
[81] D. Gamarnik, C. Moore, and L. Zdeborová, *Disordered systems insights on computational hardness*, J. Stat. Mech. (2022) 114015.
[82] A. Y. Ng, D. Harada, and S. Russell, *Policy invariance under reward transformations: Theory and application to reward shaping*, in *Proceedings of the Sixteenth International Conference on Machine Learning (ICML 1999)* (Morgan Kaufmann, San Francisco, 1999), pp. 278–287.
[83] V. Gullapalli and A. G. Barto, *Shaping as a method for accelerating reinforcement learning*, in *Proceedings of the 1992 IEEE International Symposium on Intelligent Control, Glasgow*, (IEEE, 1992), pp. 554–559, 10.1109/ISIC.1992.225046.
[84] S. Kalyanakrishnan, S. Aravindan, V. Bagdawat, V. Bhatt, H. Goka, A. Gupta, K. Krishna, and V. Piratla, *An analysis of frame-skipping in reinforcement learning*, arXiv:2102.03718.
[85] S. Mei and A. Montanari, *The generalization error of random features regression: Precise asymptotics and the double descent curve*, Commun. Pure Appl. Math. **75**, 667 (2022).
[86] S. Goldt, M. Mézard, F. Krzakala, and L. Zdeborová, *Modeling the influence of data structure on learning in neural networks: The hidden manifold model*, Phys. Rev. X **10**, 041044 (2020).
[87] S. Goldt, B. Loureiro, G. Reeves, F. Krzakala, M. Mézard, and L. Zdeborová, *The Gaussian equivalence of generative models for learning with shallow neural networks*, in *Mathematical and Scientific Machine Learning* (2022), pp. 426–471.
[88] F. Gerace, B. Loureiro, F. Krzakala, M. Mézard, and L. Zdeborová, *Generalisation error in learning with random features and the hidden manifold model*, in *International Conference on Machine Learning* (2020), pp. 3452–3462.
[89] H. Hu and Y. M. Lu, *Universality laws for high-dimensional learning with random features*, IEEE Trans. Inf. Theory **69**, 1932 (2022).
[90] Y. Dandi, L. Stephan, F. Krzakala, B. Loureiro, and L. Zdeborová, *Universality laws for Gaussian mixtures in generalized linear models*, arXiv:2302.08933.
[91] E. Cornacchia, F. Mignacco, R. Veiga, C. Gerbelot, B. Loureiro, and L. Zdeborová, *Learning curves for the multi-class teacher–student perceptron*, Mach. Learn. **4**, 015019 (2023).
[92] DeepSeek-AI, D. Guo *et al.*, *DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning*, arXiv:2501.12948.
[93] https://github.com/nishp99/RL_Perceptron.