REPRESENTATIONAL ADVANTAGE OF DISTRIBUTIONAL REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Distributional reinforcement learning (dRL) agents, which learn the full distribution of returns, often outperform classical RL (cRL) agents and account for neural activity in mammalian reward-learning circuits. These findings suggest that dRL is a computationally advantageous learning strategy, although the mechanisms behind this advantage remain unclear. We develop an exact theory of learning under cRL and dRL objectives in a minimal model—a two-layer linear network—and characterize how the expectile-based dRL objective shapes state representations throughout learning. Unlike cRL, dRL learning is driven by higher-order moments of the reward distribution, which are captured by the input–output covariance matrix. As a result, dRL networks learn at least as fast as cRL networks, and differences in moments beyond the mean speed up learning further. Moments are learned sequentially, in an order set by the reward statistics, which also fully determine the geometry of the hidden representation. This representation supports faster transfer learning in dRL, whereas pre-training with the cRL objective can impede transfer. We validate our results in simulation in linear and non-linear networks. This work lays an analytic foundation for understanding the performance gains observed in more complex dRL models and tasks, and yields testable predictions for how biological circuits represent and acquire distributional information.

1 INTRODUCTION

Natural environments are characterized by significant uncertainty: an agent exploring an environment may receive rewards of variable size or with variable frequency. Efficient behavior in uncertain environments, then, requires more than knowing which actions pay off on average. Classical reinforcement learning (cRL) discards information about reward variability, learning only the expected rewards of state-action pairs from scalar reward prediction errors (RPEs) (Sutton & Barto, 2018). Distributional RL (dRL) instead learns a set of value predictors that together encode the entire reward distribution (Bellemare et al., 2023). Agents trained with distributional objectives learn faster and reach higher performance than their classical counterparts, even when policies used throughout training select actions based on expected rewards alone (Bellemare et al., 2017; Dabney et al., 2018b; Hessel et al., 2018; Wurman et al., 2022; Clark et al., 2025). This raises a question: why should it help to learn more than mean rewards, if only the mean is used to select actions?

A corresponding question arises in neuroscience. Midbrain dopamine neurons encode a signal that has long been interpreted as the cRL RPE (Schultz et al., 1997), and their response variability typically attributed to noise that would be averaged away downstream. However, recent work has shown that this variability is structured: individual neurons weigh positive and negative RPEs differently, in a way that is quantitatively consistent with dRL (Dabney et al., 2020; Sousa et al., 2025; Muller et al., 2024; Lowet et al., 2020; 2025). If the brain implements a distributional code, then what does this extra machinery buy it, and how does it shape downstream state representations and behavior?

Answering these related questions about artificial and biological intelligence requires a theory linking the distributional learning objective to the representations it produces, and to the dynamics of how they emerge. The prevailing explanation for the success of dRL is that the full return distribution provides a richer learning signal that acts like an auxiliary task, shaping better shared state representations (Bellemare et al., 2017). Consistent with this view, Lyle et al. (2019) showed that for categorical dRL in the tabular setting, and with linear function approximation over fixed features,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

dRL and cRL make equivalent predictions of mean rewards, so any advantage must arise from representation learning. However, this remains a hypothesis, as we lack a theory of which representations the distributional objective induces, how they are learned, and when they help performance.

This is largely because deep RL agents are too complex to analyze, and distributional representations in the brain are likewise hard to isolate. We take a step towards such a theory by studying the minimal model that learns a representation: a two-layer linear network, whose learning dynamics can be derived in closed form (Saxe et al., 2014; 2019; Braun et al., 2022), allowing us to characterize the advantages of dRL analytically.

Our contributions are: (i) an exact theory of learning under cRL and dRL objectives in linear networks expressed in terms of the moments of the reward distribution (Sec. 2-3); a derivation of how reward statistics (ii) set learning speed, showing that dRL learns at least as fast as cRL (Sec. 3) and how they (iii) determine the order in which reward distribution moments are learned and the geometry of the hidden representation (Sec. 4); and (iv) an analytic treatment of transfer learning which shows how dRL reuses its representation to aid transfer, and how the classical objective can conversely impede transfer (Sec. 5). We validate our main predictions in nonlinear networks throughout. Although our setting is deliberately minimal, our results provide a foundation for interpreting the behavior of more complex distributional agents, which we discuss in Sec. 6.

2 Background and theoretical setup

To gain a mechanistic understanding of how a distributional learning objective shapes neural representations and learning dynamics, we focus on the simplest setup that yields analytic solutions and learns a hidden representation: a two-layer linear neural network. We similarly test it on the simplest task with distributional structure: single-step tasks in which a two-layer linear network learns to predict the rewards r associated with different states z (without having to perform any actions). For such tasks, all quantities of interest, including the network weight dynamics and hidden representations, can be derived in closed form from the statistics of the task (Saxe et al., 2014). The mathematical insights gained here provide intuition for distributional learning performance in tasks with more complex (and therefore, analytically intractable) structure.

2.1 Classical and expectile distributional reinforcement learning

A standard approach in classical RL (cRL) is to learn expected rewards $V(z)$ of state z by the temporal difference learning (TD) update rule,

$$V(z) \leftarrow V(z) + \alpha\delta, \tag{1}$$

where $\alpha \in (0, 1)$ is the learning rate and δ is the reward prediction error (RPE) (Sutton, 1988). In the single-step tasks we consider here, $\delta = r - V(z)$, and we assume that the observed reward r is drawn stochastically from some state-dependent distribution $p(r|z)$. In this case, the update rule performs stochastic gradient descent on a mean squared error (MSE) loss (Sutton & Barto, 2018), such that $V(z)$ converges to the expected reward, $V(z) \to \mathbb{E}[r|z]$ (Fig. 1A).

Classical RL learns expected rewards, discards any additional information about the reward distributions, which is problematic when these become relevant for, e.g., action selection. Distributional RL (dRL), instead, focuses on learning reward distributions (Bellemare et al., 2023). In expectile dRL (Rowland et al., 2019; Dabney et al., 2020), this is achieved by a simple modification to Eq. (1),

$$V_i(z) \leftarrow V_i(z) + \begin{cases} \alpha_i^+ \cdot \delta_i & \text{if } \delta_i \geq 0, \\ \alpha_i^- \cdot \delta_i & \text{if } \delta_i < 0, \end{cases} \tag{2}$$

where each value predictor V_i has different learning rates, α_i^+ and α_i^-, for positive and negative RPEs, respectively. With this update rule, the value estimates $V_i(z)$ can be shown to converge to their respective expectiles e_i of $p(r|z)$ (Fig. 1B). These expectiles are statistics of the distribution that generalize the mean in the same way that quantiles generalize the median. The asymmetry parameter $\tau_i = \frac{\alpha_i^+}{\alpha_i^+ + \alpha_i^-} \in (0, 1)$ of each predictor determines which expectile is being learned (Fig. 1C). The expectile at $\tau = 0.5$ equals the mean of a distribution, and τ below or above 0.5

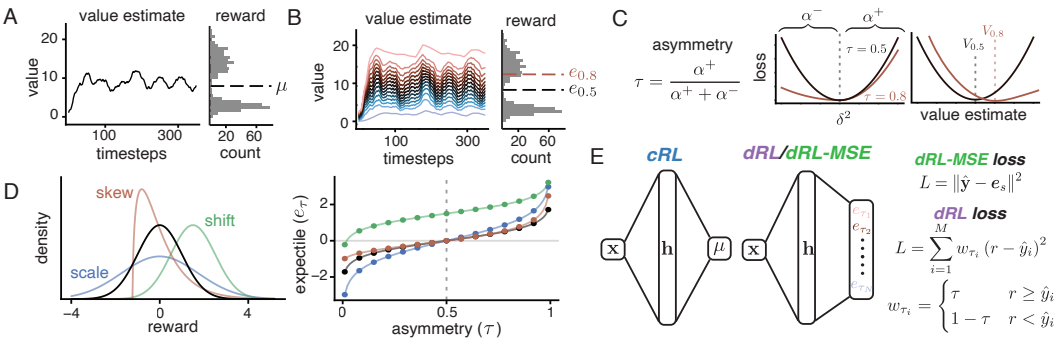

Figure 1: A & B. The value estimates of cRL (A) and dRL (B) converge to the mean reward and reward distribution expectiles, respectively. C. The asymmetry parameter τ shapes the expectile loss function and determines the associated value estimate (black: $\tau = 0.5$; red: $\tau = 0.8$). D. Unit Gaussian (black) under shift (green), scaling (blue) and skew (red) transformations (left) and corresponding expectiles (right). E. Graphical representation of the three networks (left) and distributional losses (right).

targets lower or higher values of the distribution. For expectile e_τ associated with parameter τ, Eq. (2) can be shown to perform stochastic gradient descent on the asymmetric MSE loss

$$L = \sum_i w_i(r, \hat{y}_i)(r - \hat{y}_i)^2 \qquad \text{with } w_i(r, \hat{y}_i) = \begin{cases} \tau_i & \text{if } r \geq \hat{y}_i, \\ 1 - \tau_i & \text{if } r < \hat{y}_i. \end{cases} \tag{3}$$

where the total loss is the sum of each expectile's asymmetric loss (Rowland et al., 2019; Dabney et al., 2020; Newey & Powell, 1987) (Fig. 1C). An "optimistic" expectile e_τ has larger τ and therefore weighs positive RPEs more strongly; the converse is true for "pessimistic" predictors.

Two other properties of expectiles matter for what follows. First, e_τ increases monotonically in τ, so a set of expectiles $\{e_{\tau_1}, \ldots, e_{\tau_M}\}$ with increasing $\tau_1 < \cdots < \tau_M$ tiles the reward distribution from pessimistic to optimistic (Bellini et al., 2014). Second, expectiles are equivariant under location-scale transformations of the distribution: if $r \mapsto ar + b$ with $a > 0$, then $e_\tau \mapsto ae_\tau + b$ (Fig. 1D, App. B.1). Consequently, for any distribution the vector $\boldsymbol{e}$ of M expectiles decomposes as

$$\boldsymbol{e} = \mu \mathbf{1} + \sigma \boldsymbol{\zeta}, \tag{4}$$

where μ and σ are the mean and standard deviation of the reward distribution, and $\boldsymbol{\zeta} \in \mathbb{R}^M$ is the expectile vector of the standardized reward distribution that is identical for all distributions in the same location-scale family (see B.2). In general, only certain reward families admit closed form expectiles (Daouia et al., 2024). Here, we restrict ourselves to discrete distributions to facilitate our derivations, but due to the properties of the expectile vectors, our results hold more broadly. Furthermore, unless stated otherwise, we assume a symmetric grid of asymmetries ($\tau_m = 1 - \tau_{M-m+1}$) that contain $\tau = 0.5$, such that cRL becomes a special case of dRL with a single expectile.

2.2 Linear network models and their learning objectives

We consider two-layer linear networks with the structure $\hat{\boldsymbol{y}} = \boldsymbol{W}_2\boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{b}_2$ with hidden representation $\boldsymbol{h} = \boldsymbol{W}_1\boldsymbol{x}$, and where $\boldsymbol{W}_1 \in \mathbb{R}^{N_h \times N_x}$, $\boldsymbol{W}_2 \in \mathbb{R}^{N_y \times N_h}$ and $\boldsymbol{b}_2 \in \mathbb{R}^{N_y}$. Results for networks with a ReLU hidden layer are given in the Appendix. During training, inputs $\boldsymbol{x}$ are drawn from a set of Z distributions $p(\boldsymbol{x}|z)$ where each state z maps to a reward distribution $p(r|z)$. Networks are trained to predict statistics of these reward distributions by observing inputs $\boldsymbol{x} \sim p(\boldsymbol{x}|z)$ and associated rewards $r \sim p(r|z)$.

All models share this network structure and differ only in their output targets and associated loss functions (Fig. 1E). The **cRL network** has a single output ($N_y = 1$) and is trained on the MSE loss $L = \frac{1}{2}(r - \hat{y})^2$ against sampled rewards; the per-state optimum is the reward distribution mean $\mu_z = \mathbb{E}[r|z]$. We also consider two distributional networks of the same size, but with different loss functions. Both have $N_y = M$ outputs with asymmetries $\tau_1 < \cdots < \tau_M$ which, for state z, learn the target expectile vector $\boldsymbol{e}_z$. The **dRL network** is trained to output these expectiles using

the aforementioned asymmetric MSE (expectile) loss function, Eq. (3). The **dRL-MSE network** is instead trained by the MSE objective $L = \frac{1}{2}\|\hat{\boldsymbol{y}} - \boldsymbol{e}_z\|^2$ directly applied to the oracle target expectile vector $\boldsymbol{e}_z$. We use this network as an analytically tractable proxy for our derivations. The two share the same global minimum, and hence the same final weights and hidden geometry (see C), but the dRL-MSE is not, strictly speaking, performing true distributional learning as it requires exact knowledge of the expectiles *a priori*. Contrasting the two also disentangles the roles of the loss and of the diversity of learning targets in learning.

2.3 Linear network learning dynamics

For linear networks without a bottleneck ($N_h \geq \min(N_x, N_y)$) and trained under full-batch gradient descent on an MSE loss, the closed-form expressions for weight dynamics can be derived exactly (Saxe et al., 2014). If inputs are whitened and centered ($\mathbb{E}[\boldsymbol{x}] = \boldsymbol{0}$ and $\boldsymbol{\Sigma}_{xx} = \boldsymbol{I}$) these dynamics are fully determined by the input-output covariance matrix and its singular value decomposition,

$$\boldsymbol{\Sigma}_{yx} = \mathbb{E}[\boldsymbol{y}\boldsymbol{x}^\top] = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^\top. \tag{5}$$

The network provably converges to $\boldsymbol{W}_2\boldsymbol{W}_1 \rightarrow \boldsymbol{\Sigma}_{yx}$. With zero-balanced initialization ($\boldsymbol{W}_1(0)\boldsymbol{W}_1(0)^\top = \boldsymbol{W}_2(0)^\top\boldsymbol{W}_2(0)$), the network converges to the task-defined "rich" solution with $\boldsymbol{W}_1 = \boldsymbol{R}\sqrt{\boldsymbol{S}}\,\boldsymbol{V}^\top$ and $\boldsymbol{W}_2 = \boldsymbol{U}\sqrt{\boldsymbol{S}}\boldsymbol{R}^\top$, up to an arbitrary shared rotation ($\boldsymbol{R} \in \mathbb{R}^{N_h \times N_h}$, Fig.1E) (Dominé et al., 2025), with hidden representation $\boldsymbol{h} = \boldsymbol{R}\sqrt{\boldsymbol{S}}\boldsymbol{V}^\top\boldsymbol{x}$. The network's learning dynamics can be decomposed into the singular modes $(\boldsymbol{u}_i, s_i, \boldsymbol{v}_i)$ with associated strengths $\sigma_i(t) = \boldsymbol{u}_i^\top\boldsymbol{W}_2(t)\boldsymbol{W}_1(t)\boldsymbol{v}_i$ whose dynamics follow a sigmoid that converges to $\sigma_i(t) \rightarrow s_i$ on a timescale $\propto \gamma/s_i$, so that modes are learned in order of their singular values (App. D). As in Saxe et al. (2014; 2019) we use the half-rise time $t_{i,1/2}$ (time at which σ_i reaches $s_i/2$) as our measure of learning speed throughout.

Past theories characterizing network learning dynamics assumed no output bias ($\boldsymbol{b}_2 = \boldsymbol{0}$). For the tasks we consider, we include a bias in order to capture reward distributions with non-zero means (see E.1). The bias exponentially approaches the output mean $\boldsymbol{\mu}_y$, and its dynamics are decoupled from those of $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ (see E.2). As a consequence, the bias absorbs the signal shared by all states and the weights encode any distributional information that distinguishes them. Finally, the network dynamics equations are only exact for an MSE loss. We therefore derive all results for the dRL-MSE surrogate, which shares the dRL network's fixed point. We show in App. F.3 that its dynamics qualitatively match those of the dRL network and verify every prediction in simulations of the dRL network itself.

3 How distributional information affects learning speed

To examine how distributional information affects cRL and dRL learning speed, we assume a simple task with two equiprobable states, $z \in \{1, 2\}$, and associated input and reward distributions, $p(\boldsymbol{x}|z)$ and $p(r|z)$ (Fig. 2A&B). For each of the states, the cRL network is trained to predict the expected reward ($y = \mu_z = \mathbb{E}[r|z]$), whereas the distributional networks are trained to predict the reward distribution expectiles, with either the dRL loss or the MSE loss, respectively ($\boldsymbol{y} = \boldsymbol{e}_z$). The inverse learning rate (γ) is shared across networks, and the dRL and dRL-MSE are initialized identically. Inputs are centered and whitened ($\boldsymbol{\Sigma}_{xx} = \boldsymbol{I}, \bar{\boldsymbol{x}}_1 = -\bar{\boldsymbol{x}}_2$ where $\bar{\boldsymbol{x}}_z = \mathbb{E}[\boldsymbol{x}|z]$) and the weights are initialized to be random and zero-balanced, such that the theory from Sec. 2.3 applies.

3.1 Only dRL "sees" distributional information

As described in Sec. 2.3, the learning speed of each network is determined by the singular values of the associated input-output covariance $\boldsymbol{\Sigma}_{yx}$. For the cRL network, the target output for each state z is the associated mean reward μ_z, resulting in

$$\boldsymbol{\Sigma}_{yx}^{\mathrm{cRL}} = \tfrac{1}{2}\,(\mu_1 - \mu_2)\,\bar{\boldsymbol{x}}_1^\top, \qquad s^{\mathrm{cRL}} = \tfrac{1}{2}\,|\mu_1 - \mu_2|\,\|\bar{\boldsymbol{x}}_1\|, \tag{6}$$

which relies on the inputs being centered and equiprobable ($\bar{\boldsymbol{x}}_2 = -\bar{\boldsymbol{x}}_1$; see F for derivation). For the distributional networks, the targets are the state expectile vectors $\boldsymbol{e}_z \in \mathbb{R}^M$, giving

$$\boldsymbol{\Sigma}_{yx}^{\mathrm{dRL}} = \tfrac{1}{2}\,(\boldsymbol{e}_1 - \boldsymbol{e}_2)\,\bar{\boldsymbol{x}}_1^\top, \qquad s^{\mathrm{dRL}} = \tfrac{1}{2}\,\|\boldsymbol{e}_1 - \boldsymbol{e}_2\|\,\|\bar{\boldsymbol{x}}_1\|. \tag{7}$$

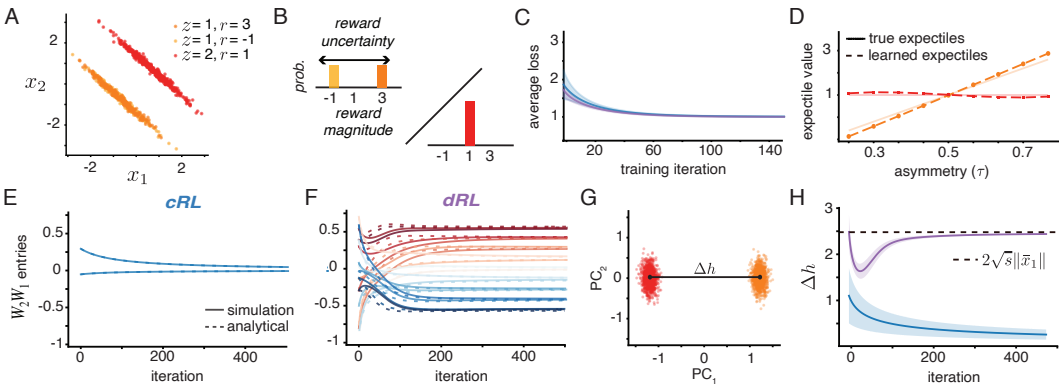

Figure 2: A. Whitened inputs for both states (orange: $z = 1$, red: $z = 2$). B. Reward distributions for $z = 1$ (left, stochastic) and $z = 2$ (right, deterministic). C. Average loss for cRL (blue) and $\tau = 0.5$ expectile for the dRL (purple) across 20 simulations (mean $\pm$ SD). D. True and learned expectiles for the dRL network for both states. E & F. Analytical (dashed) and simulated (solid) trajectories of each entry of the network weights ($\boldsymbol{W}_2\boldsymbol{W}_1$) for one simulation in the cRL (E) and dRL (F) networks. G. PCA of the hidden representation of the dRL network after convergence, with state separation Δh. H. Hidden representation separation of the dRL and cRL networks over learning, with the theoretical limit for the dRL in black (mean $\pm$ SD across 20 simulations).

In both cases, the covariance matrices are rank-one with "input" singular vector $\boldsymbol{v} = \frac{\bar{\boldsymbol{x}}_1}{\|\bar{\boldsymbol{x}}_1\|}$ where the input magnitude $\|\bar{\boldsymbol{x}}_1\|$ scales the singular value and learning speed. The two networks differ in the coupling of the input to the output mode, s, and the "output" direction, $\boldsymbol{u}$. The cRL output direction is the scalar $\text{sign}(\Delta\mu)$, and its learning speed is set by $|\Delta\mu|$, where $\Delta\mu = \mu_1 - \mu_2$. The dRL output direction is the expectile separation, $\Delta\boldsymbol{e} = \boldsymbol{e}_1 - \boldsymbol{e}_2$, which also sets the learning speed, and by Eq. (4) decomposes into

$$\Delta\boldsymbol{e} = \boldsymbol{e}_1 - \boldsymbol{e}_2 = \underbrace{\Delta\mu\mathbf{1}}_{\text{mean difference}} + \underbrace{(\sigma_1\boldsymbol{\zeta}_1 - \sigma_2\boldsymbol{\zeta}_2)}_{\text{variance/shape difference}} . \tag{8}$$

The first term is identical to what the cRL network learns to predict, replicated across M outputs. The second term is unique to distributional networks and is nonzero whenever the two reward distributions differ in variance or shape. Thus, learning speed in the cRL network only depends on the difference in mean rewards, $\Delta\mu$, whereas learning speed in the distributional networks is additionally determined by higher-order moments of the reward distribution.

To test these theoretical predictions, we simulated the learning dynamics of all three networks (cRL, dRL and dRL-MSE) on the two-state task, and compared the simulated weight and bias trajectories to the closed form predictions described in Sec. 2.3. We chose both reward distributions to have the same mean but different variances across states ($p(r = -1|z = 1) = p(r = 3|z = 1) = 1/2$ and $p(r = 1|z = 2) = 1$, such that $\mu_1 = \mu_2 = 1$, $\sigma_1 = 2$, and $\sigma_2 = 0$), resulting in $\Delta\mu = 0$, but a non-zero $\Delta\boldsymbol{e}$ (Fig. 2A,B). This task is diagnostic of a distributional agent: a cRL network learning average expected rewards sees no reason to distinguish between the states, since they encode the same average reward. The presence of distributional information leads the dRL network to learn faster than the cRL ($s^{\text{dRL}} > s^{\text{cRL}}$), despite both achieving the same final loss (Fig. 2C). Furthermore, the dRL correctly learns the reward distribution expectiles (Fig. 2D), with learning dynamics matching theoretical predictions (predictions for $\boldsymbol{W}_2\boldsymbol{W}_1$ follow (Braun et al., 2022), Fig. 2E,F ; see Apps. E.2 and F.2 for dRL-MSE and $\boldsymbol{b}_2$ dynamics). The same qualitative behavior holds in networks with a ReLU hidden layer (App. F.5). Another factor that boosts learning speed in dRL networks is the number of outputs, M, which increases the learning gradient's overall magnitude when compared to the single output of the cRL network. However, as we show in F.1, the dRL network always learns at least as fast as the cRL network even when controlling for output size (i.e., $s^{\text{dRL}} \geq s^{\text{cRL}}$ more generally).

3.2 EFFECTS OF MANIPULATIONS OF REWARD STOCHASTICITY

Single-moment differences in reward distributions. We first consider how differences in a single moment (mean, variance, or skew) between the two reward distributions impact the learning speed of the three networks. Our theory predicts that, with the other moments matched, a mean difference $\Delta\mu$ should linearly scale the singular value, and so the learning speed, of all three networks. Differences in spread $\Delta\sigma = \sigma_1 - \sigma_2$ or skew, $\Delta\kappa \neq 0$, in contrast, should have no impact on the cRL network, as it is "blind" to higher order moments. For the distributional networks, however, both changes manifest as increases in learning speed. For example, for a spread difference, $\Delta\boldsymbol{e} = \Delta\sigma\boldsymbol{\zeta}$ and $s^{\mathrm{dRL}} = \frac{1}{2}|\Delta\sigma|\|\boldsymbol{\zeta}\|\|\bar{\boldsymbol{x}}_1\|$. We confirm all of these predictions through simulations in App. G.

Global reward transformations. We next ask how linear scalings (a) or shifts (b) of reward magnitudes across states ($r \mapsto ar + b$) impact s^{dRL}. Such reward transformations cause equivalent linear transformations of the associated expectiles, making the predicted effects on learning straightforward to derive. A reward shift $r \mapsto r + b$ is absorbed by the bias and leaves $\boldsymbol{\Sigma}_{yx}$ (and therefore, learning speed and representational geometry) unchanged. A scaling $r \mapsto ar$ multiplies every singular value by a, increasing learning speed while preserving the representational geometry. We confirmed these predictions in simulations, where we find that the dRL exhibits slightly slower learning with positive shifts, as its network weights depend on the current bias (App. H).

Impact of the asymmetric expectile loss on learning speed. Learning speed is not only modulated by changes in the reward distributions' expectiles—the network's prediction targets—but also by the network's loss function. The dRL network uses the asymmetric loss Eq. (3), which weights each expectile's squared prediction error by either $w_\tau = \tau$ or $1 - \tau$, depending on the sign of the RPE. This causes a $S(\tau) = 1/(2\mathbb{E}[w_\tau])$ scaling for individual expectiles when compared to the dRL-MSE network, and an overall scaling of $S_{\mathrm{net}} = \|\Delta\boldsymbol{e}\|/\|2\boldsymbol{\Lambda}\,\Delta\boldsymbol{e}\|$ across all expectiles with $\boldsymbol{\Lambda} = \mathrm{diag}(\mathbb{E}[w_\tau])$ (under certain assumptions, see App. I.2). We show in App I that for sufficiently well-behaved symmetric distributions, $\mathbb{E}[w_\tau] < 1/2$ for $\tau \neq \frac{1}{2}$ always. This implies that $S_{\mathrm{net}} \geq 1$: the dRL network learns no faster than dRL-MSE (with equality for the two-point task of Fig. 2). Moreover, outer expectiles (τ close to 0 or 1) are learned more slowly than central ones. Since outer expectiles carry higher-moment information, dRL tends to learn lower-order moments first. More details and simulations to confirm our predictions are included in App. I.

4 HIDDEN REPRESENTATION STRUCTURE AND DIMENSIONALITY

The singular value decomposition of $\boldsymbol{\Sigma}_{yx}$ not only determines what is learned and how fast, but also the geometry of the hidden representation. For our two-state task (Fig. 2), the separation Δh between the hidden representations associated with either state grows along the direction defined by $\boldsymbol{v} \propto \bar{\boldsymbol{x}}_1$ to a final magnitude of $\Delta h \to 2\sqrt{s}\|\bar{\boldsymbol{x}}_1\|$ (see App. F.4). For the cRL network, our theory from Sec. 3.1 predicts that $s^{\mathrm{cRL}} = 0$ (as $\Delta\mu = 0$), such that $\sigma(t) \to 0, \Delta h \to 0$, making the hidden representation between states collapse. For distributional networks, in contrast, the hidden representation should keep both states separate in proportion to $\sqrt{\|\Delta\boldsymbol{e}\|}$, which we confirmed in simulation (Figs. 2G,H & 8A, B) and which persists in nonlinear networks (Fig. 8C,D).

Our two-state task restricts $\boldsymbol{\Sigma}_{yx}$ to be rank one, regardless of the network type. More generally, its rank is at most $\min(Z - 1, N_x, M, d)$, where centering removes one of the Z state dimensions, and $d = \dim\,\mathrm{span}\,\{1, \boldsymbol{\zeta}_1, \ldots, \boldsymbol{\zeta}_Z\}$ counts the mean and shape components of Eq. (8). With a single output ($M = 1$), the cRL network is limited to rank one. Only distributional networks with $M > 1$ can have $\boldsymbol{\Sigma}_{yx}$'s of higher rank, making higher dimensionality a hallmark of distributional coding.

Thus, we now ask how richer reward structures (i.e., higher-rank $\boldsymbol{\Sigma}_{yx}$) impact the hidden representations of distributional networks. To do so, we designed a task with five equiprobable states, each of which stochastically yields one of two equally likely reward magnitudes (Fig. 3A). We constructed these reward distributions in a way that makes their means and variances, μ_z and σ_z^2, completely uncorrelated across states, and yield expectiles $e_{z,\tau} = \mu_z + (2\tau - 1)\sigma_z$. As a result, the input-output covariance matrix factorizes into two rank-one components, $\boldsymbol{\Sigma}_{yx} = \frac{1}{Z}\mathbf{1}\Delta\boldsymbol{\mu}^\top\bar{\boldsymbol{X}} + \frac{1}{Z}\boldsymbol{\zeta}\Delta\boldsymbol{\sigma}^\top\bar{\boldsymbol{X}}$, where $\bar{\boldsymbol{X}}$ is the $Z \times Z$ five-state input matrix with input means vectors $\bar{\boldsymbol{x}}_z^\top$ as its rows, $\boldsymbol{\zeta}$ is the expectile shape vector with elements $2\tau - 1$, and $\Delta\boldsymbol{\mu}$ and $\Delta\boldsymbol{\sigma}$ are the vector of per-state reward distribution means and standard deviations respectively, measured relative to their across-state averages (see App. J). Here, the first component corresponds to the mean, with $\boldsymbol{u}_{\mathrm{mean}} \propto \mathbf{1}\Delta\boldsymbol{\mu}^\top\bar{\boldsymbol{X}}$

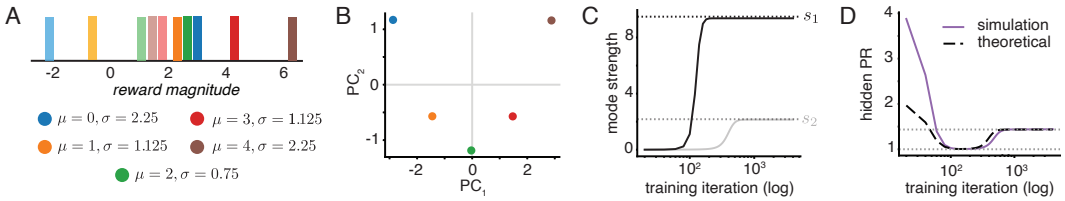

Figure 3: A. Reward distributions for all $Z = 5$ states. B. State centroids in the top two PCs of the hidden layer after training (dRL network). C. Strength of the mean (black) and spread (gray) modes over training; dotted lines are the predicted asymptotic values. D. Participation ratio of the hidden representation, with the PR computed from the measured mode strengths in C in a dashed black line.

and $s_{\text{mean}} = \sqrt{M/Z}\|\Delta\mu\|$, and the second component to the variance, with $\boldsymbol{u}_{\text{spread}} \propto \boldsymbol{\zeta}\Delta\boldsymbol{\sigma}^\top \bar{\boldsymbol{X}}$ and $s_{\text{spread}} = \sqrt{1/Z}\|\boldsymbol{\zeta}\|\|\Delta\sigma\|$. The relative strengths of these components, and thus the speed with which they are learned, is set by how much the reward distributions' moments vary across states. Choosing $\|\Delta\boldsymbol{\mu}\| > \|\Delta\boldsymbol{\sigma}\|$, we found that the trained network matches the predicted solution, for which the top principal component (PC) of the hidden representation aligns with $\Delta\boldsymbol{\mu}$ ($\text{corr}(\text{PC}_1, \Delta\boldsymbol{\mu}) = 1$) and orders the states by their mean, and the second PC aligns with $\Delta\boldsymbol{\sigma}$ ($\text{corr}(\text{PC}_2, \Delta\boldsymbol{\sigma}) = 1$) and orders states by their standard deviation (Fig. 3B). Because each mode is learned with a timescale $\propto 1/s_i$, the mean mode is learned before the variance mode (Fig. 3C). If we instead choose $\|\Delta\boldsymbol{\sigma}\| > \|\Delta\boldsymbol{\mu}\|$ the moments are learned in reverse order (Fig. 14B). Thus, which moment dominates learning is a property of the task rather than the learner. Finally, if we allow $\Delta\boldsymbol{\sigma}$ and $\Delta\boldsymbol{\mu}$ to correlate, the moments mix in the principal component, with the moment with the largest differences across states dominating PC_1 (Fig. 14C). The same geometry emerges in nonlinear networks, with slightly weaker alignment (Fig. 14D–F; App. J.3).

The rank of $\boldsymbol{\Sigma}_{yx}$ is reflected in the hidden layer's dimensionality, as measured by the participation ratio, $\text{PR} = (\sum_i s_i)^2 / \sum_i s_i^2$ (Gao et al., 2017). Because the hidden representation's covariance, $\mathbb{E}[\boldsymbol{h}\boldsymbol{h}^\top] = \boldsymbol{R}\boldsymbol{S}\boldsymbol{R}^\top$, shares the task's singular values, the asymptotic PR of the hidden representation can be directly computed from the task's reward distributions. For our five-state tasks, the singular value trajectory predicts a PR trajectory that is stepped across learning, which we confirmed in simulations (Fig. 3D). The measured PR starts near 4 due to random initialization, collapses towards 1 as the mean mode grows, and finally increases to its final value when the spread mode is learned.

When mean, variance, and skew vary orthogonally across states, they are likewise learned in order of how strongly they vary across states; if, after training, we change the task such that one moment (e.g., the mean) becomes equal across states, the network unlearns only that moment and preserves the others (App. J.2). Nonlinear networks show the same selective unlearning, although their principal axes mix the learned moments (App. J.3).

5 The hidden representation determines transfer dynamics

We now ask if richer distributional representations can support further learning, in particular by asking how they can support transfer learning to another task. We assume a two-phase task whose first phase is the task described in Fig. 2A,B. In the second phase the reward contingencies change to a deterministic reward of 3 for state $z = 1$, while state $z = 2$ continues to yield a reward of 1 (Fig. 4A). Across both phases, the input structure remains constant, and therefore the task covariance matrices share a common input direction, $\boldsymbol{v}_1 = \boldsymbol{v}_2 \propto \bar{\boldsymbol{x}}_1$. The second phase, then, involves a re-scaling of the mode strength, $\sigma(t)$, and learning a new output direction, $\boldsymbol{u}$.

As we demonstrated in simulations (Fig. 4B), the dRL network learns this change in reward contingencies more quickly than the cRL network. Intuitively, such a change is difficult for the cRL network, as the first phase imposes a collapse of state representations in the network's hidden layer (Sec. 4; Figs. 2H & 4C), but such state information is required for correct reward prediction in the task's second phase. Thus, the cRL has to re-build state representations in the hidden layer during phase two, which comes at a cost of slower transfer learning. The dRL network, in contrast, learned higher-order moments of the reward distributions in phase one, and so kept a reusable representation

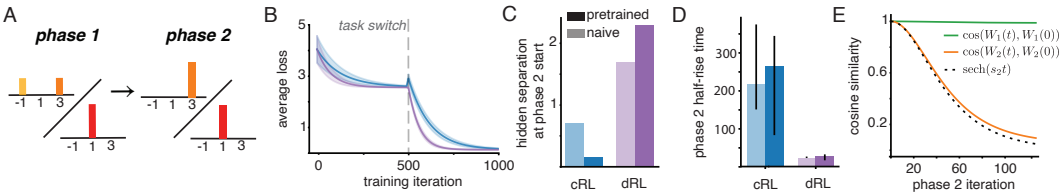

Figure 4: A. Two-phase task reward distributions. B. Average loss for cRL (blue) and dRL (purple) across 20 simulations (mean $\pm$ SD). C. Hidden representation separation of the cRL and dRL networks at the start of phase 2 for naive (light) and pretrained (dark) networks. D. Half-rise time over 20 simulations for the same (error bars: $\pm$ SD). E. Cosine similarity between $\boldsymbol{W}_1$ (green) or $\boldsymbol{W}_2$ (orange) at the start of phase two and thereafter. Theoretically predicted rotation for $\boldsymbol{W}_2$ in black.

of state identity (Figs. 2H & 4C). This allows it to learn the change in reward contingencies faster. The same holds for networks with a ReLU hidden layer (App. K.3, Fig. 17A–C).

We now confirm this intuition analytically. Starting with the cRL network, we have shown that Eq. (6) results in a phase one singular value of $s_1^{\mathrm{cRL}} \propto |\Delta\mu_1| = 0$. This implies that the associated network mode strength approaches $\sigma_1^{\mathrm{cRL}}(T_1) \to 0$ with increasing phase one training time T_1. As we show in App. K, this results in a half-rise time for the second task phase given by

$$t_{1/2}^{\mathrm{cRL}} = \underbrace{\frac{\gamma}{2s_2^{\mathrm{cRL}}} \ln \frac{s_2^{\mathrm{cRL}}}{\sigma_0}}_{\text{from scratch}} + \underbrace{\frac{\gamma}{2s_2^{\mathrm{cRL}}} \ln\left(1 + \frac{2\sigma_0 T_1}{\gamma}\right)}_{\text{pretraining penalty}}, \tag{9}$$

where $s_2^{\mathrm{cRL}} \propto |\Delta\mu_2| = 2$ is the singular value for the second task phase, and σ_0 is the initial mode strength before any training has occurred. The first term describes the half-rise time for a network without any pre-training on the first phase; the second term grows with T_1 and represents the penalty arising from collapsing state separability in the first task phase (Fig. 4D, blue). As $T_1 \to \infty$ this term grows without bound causing the cRL network to get stuck at a fixed point of the phase two dynamics, preventing any transfer learning from occurring.

The dRL network does not suffer from such a state collapse, and, as a consequence, avoids the associated slow-down when learning the second task phase. In each phase, the dRL network learns $s_j^{\mathrm{dRL}} \propto \|\Delta \boldsymbol{e}_j\|$ and "output" singular vector $\boldsymbol{u}_j^{\mathrm{dRL}} \propto \Delta \boldsymbol{e}_j$. The phase one expectiles are $\Delta e_{1,\tau} \propto 2\tau - 1$ and the phase two expectiles are $\Delta e_{2,\tau} \propto 1$. For a symmetric set of expectiles these vectors are orthogonal ($\boldsymbol{u}_1^{\mathrm{dRL}} \perp \boldsymbol{u}_2^{\mathrm{dRL}}$), so after pre-training the distributional network enters phase two with an output vector pointing in a direction orthogonal to the one required for the new task phase. We show in App. K that transfer learning can thus be reduced to a combination of rotation and scaling of the network weights on a plane defined by ($\boldsymbol{u}_1^{\mathrm{dRL}}, \boldsymbol{u}_2^{\mathrm{dRL}}$). The constant $\boldsymbol{v}_j^{\mathrm{dRL}}$ implies that the input weights $\boldsymbol{W}_1(t) = \sqrt{\sigma_2^{\mathrm{dRL}}(t)}\boldsymbol{v}_2^{\mathrm{dRL}\top}$ keep the same direction throughout phase two learning, and only change in magnitude through the mode strength pre-factor $\sigma_2^{\mathrm{dRL}}(t)$ (Fig. 4E, green). The readout weights $\boldsymbol{W}_2(t) = \sqrt{\sigma_2^{\mathrm{dRL}}(t)}\boldsymbol{u}(t)$ likewise undergo the same change in magnitude. However, they additionally rotate from $\boldsymbol{u}(0) = \boldsymbol{u}_1^{\mathrm{dRL}}$ (at rotation angle $\theta(0) = 0$) to $\boldsymbol{u}(\infty) = \boldsymbol{u}_2^{\mathrm{dRL}}$ (at $\theta(\infty) = \frac{\pi}{2}$) (Fig. 4E, orange). We show in App. K that this rotation trajectory satisfies $\sin\theta(t) = \tanh(s_2^{\mathrm{dRL}} t)$. We also show that the half-rise time is independent of T_1 and only depends on the first phase through the ratio $s_2^{\mathrm{dRL}}/s_1^{\mathrm{dRL}}$. In nonlinear networks the input weights are likewise preserved while the readout rotates, albeit more slowly than predicted (Fig. 17D).

Therefore, as opposed to the cRL, the dRL transfer dynamics are independent of phase one pre-training. The learned hidden representation is fully re-used in phase two, with all task-relevant re-learning happening in the readout weights. Altogether this shows how distributional representations can be leveraged even in deterministic settings to increase performance. In other words, learning more than the mean can pay off, even when that information is not needed.

6 DISCUSSION

We have developed an exact theory of learning for classical and distributional RL objectives in two-layer linear networks, showing how the distributional objective produces rich, interpretable, and reusable representations. The theory rests on the input–output covariance $\mathbf{\Sigma}_{yx}$ which for distributional targets factorizes into components interpretable as moments of the reward distribution. This links the statistics of the environment to what is learned, how fast, and the geometry of the hidden representation. We showed that the dRL network always learns at least as fast as the cRL network; that it learns moments in an order set by how strongly they vary across states; and that distributional pre-training supports transfer, even to tasks without stochastic structure. Simulations confirm these results in linear networks, and their replication in non-linear networks suggests that they do not depend critically on linearity. In this minimal setting, the theory answers the question we set out to answer: what are the representational benefits of dRL? It also yields two counterintuitive results. First, higher order moments in the reward distributions speed up learning in distributional agents (by increasing s^{dRL}), defying intuition that it would take longer to learn richer signals. Second, information does not have to be used to be useful: even if variance and shape never enter action selection, they shape a representation of the state space that can aid later learning.

Relation to prior work. The success of dRL is commonly attributed to its additional prediction targets inducing a richer representation (Bellemare et al., 2017), an advantage that disappears in tabular settings (Lyle et al., 2019). Our results make this intuition precise and show that it arises from the interaction between the distributional objective and the statistics of the environment. We additionally build on theories of learning dynamics (Saxe et al., 2014; 2019; Advani et al., 2020) and of transfer and pre-training (Lampinen & Ganguli, 2019; Braun et al., 2022) in deep linear networks.

Limitations. Our theory is exact for linear networks trained with an MSE proxy for the expectile loss; we treat the expectile loss itself numerically and characterize its deviation from the proxy (App. I). Our task is also simplified compared to most dRL applications: single-step predictions with whitened inputs and no action selection. Extending the state space and incorporating policy learning are natural next steps. Our nonlinear results suggest that the qualitative insights will hold there, but it will be informative to see where they break down. Finally, we study expectile-based dRL, whereas other formulations use different targets and losses (Dabney et al., 2018a; Tano et al., 2020). Most results should also hold for quantile-based networks, but certain properties (decomposition into mean and shape components, invariance to transformations) may not apply more generally.

Predictions for neuroscience. If dopamine implements a distributional code (Dabney et al., 2020; Lowet et al., 2020), our theory makes predictions for state representations in its downstream targets, such as striatum and cortex, and for behavior. *First*, the dimensionality of state representations should follow reward statistics; introducing or removing moments should raise or lower the PR of the population, respectively. *Second*, because optimistic and pessimistic dopamine neurons are anatomically separated (Lowet et al., 2025), selectively silencing a subset of this population should selectively degrade variance and skew coding and affect downstream behavior. *Third*, because each neuron has a fixed asymmetry, predictions driven by neurons with extreme asymmetries ($\tau \to 0$ or $\to 1$) should converge more slowly for symmetric distributions, and faster for strongly skewed ones (App. I). *Fourth*, animals pre-trained on cues with matched mean rewards but different reward variability should learn a subsequent task in which the means differ faster than naive animals.

Predictions for deep RL. Our exact dynamics do not carry over directly to deeper nonlinear networks, but they suggest two predictions. *First*, because distributional transfer occurs through a rotation of the readout, fine-tuning should leave the encoder largely unchanged, and freezing it should cost little in terms of performance. *Second*, because the rank of $\mathbf{\Sigma}_{yx}$ is bounded by the number of independent shape components, adding expectiles beyond that number should speed up learning but bring no additional representational dimensions, giving a principled way to set M.

Shared predictions. Two predictions apply to both artificial and biological agents. *First*, reward distribution moments should become decodable in a task-determined order. Each moment is learned on a timescale inversely proportional to how much it varies across states, so a task in which variance differs strongly across states should give rise to an early variance-coding axis. This can be tested in cortical representations and in a network's final hidden layer. *Second*, global shifts and rescalings of reward should leave the geometry of the representation unchanged, which can be measured with representational similarity methods (Kriegeskorte, 2008; Kornblith et al., 2019; Giaffar et al., 2024).

AI USE STATEMENT

In this work, we used generative AI tools for implementing code, assisting in proving mathematical claims for the transfer learning section, and assisting with data analysis. We have not used generative AI tools for generating synthetic data sets, developing theoretical models or conceptual frameworks, formulating mathematical claims, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments or interpreting results, and the rest of the required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for creating or editing software code. We have reviewed all AI-assisted work - all LLM- generated code was verified and tested for correctness by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

All results in this paper are either derived analytically or obtained from simulations on synthetic tasks, and no external datasets are used. The theoretical setup and its assumptions are stated in Sec. 2 and App. A. Complete derivations are given in the appendix where relevant, and App. A lists all simulation parameters shared across experiments, including network architecture, initialization, optimization and the procedure for generating inputs and rewards. Task-specific parameters and any deviations from these defaults are given at the end of the corresponding appendix section. Code to reproduce all simulations and figures will be released upon publication.

REFERENCES

Madhu S. Advani, Andrew M. Saxe, and Haim Sompolinsky. High-dimensional dynamics of generalization error in neural networks. *Neural Networks*, 132:428–446, December 2020. ISSN 0893-6080. doi: 10.1016/j.neunet.2020.08.022. URL http://dx.doi.org/10.1016/j.neunet.2020.08.022.

Alexander B. Atanasov, Blake Bordelon, and Cengiz Pehlevan. Neural networks as kernel learners: The silent alignment effect. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=1NvflqAdoom.

Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 449–458. JMLR.org, 2017.

Marc G. Bellemare, Will Dabney, and Mark Rowland. *Distributional Reinforcement Learning*. The MIT Press, May 2023. ISBN 9780262374026. doi: 10.7551/mitpress/14207.001.0001. URL http://dx.doi.org/10.7551/mitpress/14207.001.0001.

Fabio Bellini, Bernhard Klar, Alfred Müller, and Emanuela Rosazza Gianin. Generalized quantiles as risk measures. *Insurance: Mathematics and Economics*, 54:41–48, January 2014. ISSN 0167-6687. doi: 10.1016/j.insmatheco.2013.10.015. URL http://dx.doi.org/10.1016/j.insmatheco.2013.10.015.

Lukas Braun, Clémentine C. J. Dominé, James Fitzgerald, and Andrew M. Saxe. Exact learning dynamics of deep linear networks with prior knowledge. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/2b3bb2c95195130977a51b3bb251c40a-Abstract-Conference.html.

Lukas Braun, Erin Grant, and Andrew M. Saxe. Not all solutions are created equal: An analytical dissociation of functional and representational similarity in deep linear neural networks. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine*

Learning Research. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/braun25a.html.

Tyler Clark, Mark Towers, Christine Evers, and Jonathon Hare. Beyond the rainbow: High performance deep reinforcement learning on a desktop PC. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 11064–11091. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/clark25a.html.

Will Dabney, Georg Ostrovski, David Silver, and Rémi Munos. Implicit quantile networks for distributional reinforcement learning. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1104–1113. PMLR, 2018a. URL http://proceedings.mlr.press/v80/dabney18a.html.

Will Dabney, Mark Rowland, Marc G. Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In Sheila A. McIlraith and Kilian Q. Weinberger (eds.), *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 2892–2901. AAAI Press, 2018b. doi: 10.1609/AAAI.V32I1.11791. URL https://doi.org/10.1609/aaai.v32i1.11791.

Will Dabney, Zeb Kurth-Nelson, Naoshige Uchida, Clara Kwon Starkweather, Demis Hassabis, Rémi Munos, and Matthew Botvinick. A distributional code for value in dopamine-based reinforcement learning. *Nature*, 577(7792):671–675, January 2020. ISSN 1476-4687. doi: 10.1038/s41586-019-1924-6. URL http://dx.doi.org/10.1038/s41586-019-1924-6.

Abdelaati Daouia, Gilles Stupfler, and Antoine Usseglio-Carleve. An expectile computation cookbook. *Statistics and Computing*, 34(3), March 2024. ISSN 1573-1375. doi: 10.1007/s11222-024-10403-x. URL http://dx.doi.org/10.1007/s11222-024-10403-x.

Clémentine Carla Juliette Dominé, Nicolas Anguita, Alexandra Maria Proca, Lukas Braun, Daniel Kunin, Pedro A. M. Mediano, and Andrew M. Saxe. From lazy to rich: Exact learning dynamics in deep linear networks. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=ZXaocmXc6d.

Peiran Gao, Eric M. Trautmann, Byron M. Yu, Gopal Santhanam, Stephen I. Ryu, Krishna V. Shenoy, and Surya Ganguli. A theory of multineuronal dimensionality, dynamics and measurement. *bioRxiv*, 2017. URL https://api.semanticscholar.org/CorpusID:19938440.

Hamza Giaffar, Camille Rullán Buxó, and Mikio Aoi. The effective number of shared dimensions between paired datasets. In Sanjoy Dasgupta, Stephan Mandt, and Yingzhen Li (eds.), *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pp. 4249–4257. PMLR, 02–04 May 2024. URL https://proceedings.mlr.press/v238/giaffar24a.html.

Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Gheshlaghi Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In Sheila A. McIlraith and Kilian Q. Weinberger (eds.), *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 3215–3222. AAAI Press, 2018. doi: 10.1609/AAAI.V32I1.11796. URL https://doi.org/10.1609/aaai.v32i1.11796.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey E. Hinton. Similarity of neural network representations revisited. In Kamalika Chaudhuri and Ruslan Salakhutdinov

(eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3519–3529. PMLR, 2019. URL http://proceedings.mlr.press/v97/kornblith19a.html.

Nikolaus Kriegeskorte. Representational similarity analysis – connecting the branches of systems neuroscience. *Frontiers in Systems Neuroscience*, 2008. ISSN 1662-5137. doi: 10.3389/neuro.06.004.2008. URL http://dx.doi.org/10.3389/neuro.06.004.2008.

Andrew K. Lampinen and Surya Ganguli. An analytic theory of generalization dynamics and transfer learning in deep linear networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=ryfMLoCqtQ.

Thomas Laurent and James von Brecht. Deep linear networks with arbitrary loss: All local minima are global. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2902–2907. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/laurent18a.html.

Adam S. Lowet, Qiao Zheng, Sara Matias, Jan Drugowitsch, and Naoshige Uchida. Distributional reinforcement learning in the brain. *Trends in Neurosciences*, 43(12):980–997, December 2020. ISSN 0166-2236. doi: 10.1016/j.tins.2020.09.004. URL http://dx.doi.org/10.1016/j.tins.2020.09.004.

Adam S. Lowet, Qiao Zheng, Melissa Meng, Sara Matias, Jan Drugowitsch, and Naoshige Uchida. An opponent striatal circuit for distributional reinforcement learning. *Nature*, 639 (8055):717–726, February 2025. ISSN 1476-4687. doi: 10.1038/s41586-024-08488-5. URL http://dx.doi.org/10.1038/s41586-024-08488-5.

Clare Lyle, Marc G. Bellemare, and Pablo Samuel Castro. A comparative analysis of expected and distributional reinforcement learning. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pp. 4504–4511. AAAI Press, 2019. doi: 10.1609/AAAI.V33I01.33014504. URL https://doi.org/10.1609/aaai.v33i01.33014504.

Timothy H. Muller, James L. Butler, Sebastijan Veselic, Bruno Miranda, Joni D. Wallis, Peter Dayan, Timothy E. J. Behrens, Zeb Kurth-Nelson, and Steven W. Kennerley. Distributional reinforcement learning in prefrontal cortex. *Nature Neuroscience*, 27(3):403–408, January 2024. ISSN 1546-1726. doi: 10.1038/s41593-023-01535-w. URL http://dx.doi.org/10.1038/s41593-023-01535-w.

Whitney K. Newey and James L. Powell. Asymmetric least squares estimation and testing. *Econometrica*, 55(4):819, July 1987. ISSN 0012-9682. doi: 10.2307/1911031. URL http://dx.doi.org/10.2307/1911031.

Mark Rowland, Robert Dadashi, Saurabh Kumar, Rémi Munos, Marc G. Bellemare, and Will Dabney. Statistics and samples in distributional reinforcement learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5528–5536. PMLR, 2019. URL http://proceedings.mlr.press/v97/rowland19a.html.

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In Yoshua Bengio and Yann LeCun (eds.), *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6120.

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. A mathematical theory of semantic development in deep neural networks. *Proceedings of the National Academy of Sciences*, 116 (23):11537–11546, May 2019. ISSN 1091-6490. doi: 10.1073/pnas.1820226116. URL http://dx.doi.org/10.1073/pnas.1820226116.

Wolfram Schultz, Peter Dayan, and P. Read Montague. A neural substrate of prediction and reward. *Science*, 275(5306):1593–1599, March 1997. ISSN 1095-9203. doi: 10.1126/science.275.5306.1593. URL http://dx.doi.org/10.1126/science.275.5306.1593.

Margarida Sousa, Pawel Bujalski, Bruno F. Cruz, Kenway Louie, Daniel C. McNamee, and Joseph J. Paton. A multidimensional distributional map of future reward in dopamine neurons. *Nature*, 642(8068):691–699, June 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09089-6. URL http://dx.doi.org/10.1038/s41586-025-09089-6.

Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, August 1988. ISSN 1573-0565. doi: 10.1007/bf00115009. URL http://dx.doi.org/10.1007/BF00115009.

Richard S Sutton and Andrew G Barto. *Reinforcement Learning*. Adaptive Computation and Machine Learning series. Bradford Books, Cambridge, MA, 2 edition, November 2018.

Pablo Tano, Peter Dayan, and Alexandre Pouget. A local temporal difference code for distributional reinforcement learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 13662–13673. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/9dd16e049becf4d5087c90a83fea403b-Paper.pdf.

Peter R. Wurman, Samuel Barrett, Kenta Kawamoto, James MacGlashan, Kaushik Subramanian, Thomas J. Walsh, Roberto Capobianco, Alisa Devlic, Franziska Eckert, Florian Fuchs, Leilani Gilpin, Piyush Khandelwal, Varun Kompella, HaoChih Lin, Patrick MacAlpine, Declan Oller, Takuma Seno, Craig Sherstan, Michael D. Thomure, Houmehr Aghabozorgi, Leon Barrett, Rory Douglas, Dion Whitehead, Peter Dürr, Peter Stone, Michael Spranger, and Hiroaki Kitano. Outracing champion gran turismo drivers with deep reinforcement learning. *Nature*, 602 (7896):223–228, February 2022. ISSN 1476-4687. doi: 10.1038/s41586-021-04357-7. URL http://dx.doi.org/10.1038/s41586-021-04357-7.

APPENDIX

A SIMULATION PARAMETERS

Here we report simulation parameters that are shared across all reported experiments. We note in the relevant appendix sections where these parameters change and provide simulation details not reported in the main text.

Network. Every simulation used a two-layer ($\hat{\boldsymbol{y}} = \boldsymbol{W}_2 f(\boldsymbol{W}_1 \boldsymbol{x}) + \boldsymbol{b}_2$) network with one layer of 16 hidden units. Simulations reported in the main text used linear networks ($f(\boldsymbol{x}) = \boldsymbol{x}$), while the non-linear cases reported in the appendix used a ReLU activation function ($f(\boldsymbol{x}) = \max(0, \boldsymbol{x})$).

We did not include a bias term for the hidden layer, and the output bias $\boldsymbol{b}_2$ was initialized to zero. The initial weights for $\boldsymbol{W}_1, \boldsymbol{W}_2$ were i.i.d. Gaussian with a standard deviation of $\sigma = 0.35$ (or $\sigma = 0.01$ when we note that the weights were initialized to be small, as in Figs. 3 & 4). We enforced zero-balance by taking the SVD of the randomly initialized $\boldsymbol{W}_2\boldsymbol{W}_1 = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^\top$ and subsequently set $\boldsymbol{W}_1 = \boldsymbol{R}\sqrt{\boldsymbol{S}}\boldsymbol{V}^\top$ and $\boldsymbol{W}_2 = \boldsymbol{U}\sqrt{\boldsymbol{S}}\boldsymbol{R}^\top$, where $\boldsymbol{R}$ was a random orthogonal 16×16 matrix. This preserved the singular vectors of the random product, while enforcing the zero-balance condition ($\boldsymbol{W}_2^\top \boldsymbol{W}_2 = \boldsymbol{W}_1 \boldsymbol{W}_1^\top$). The two distributional networks were initialized with identical weights and the classical network got its own balanced initialization. The expectile vector has M = 9 channels with τ spaced evenly on $[0.1, 0.9]$

Optimization. All networks were trained by full-batch gradient-descent with no momentum and no weight decay and a learning rate of $\eta = 0.02$. Training ran for 500 epochs (with 500 epochs per phase for Fig. 4). One epoch was a full batch iteration, and weights, hidden activations and the loss were stored for subsequent analysis at every iteration. When reporting the loss, we report the loss on the $\tau = 0.5$ expectile for the dRL, since that reduces to the cRL loss exactly. Unless otherwise noted, results were averaged over independent simulations (with a fresh draw of initial weights and data). Shaded bands are $\pm$1SD across simulations.

Task structure. The non-whitened input vector $\tilde{\boldsymbol{x}}$ associated with each state z was drawn from a two-dimensional Gaussian with state-dependent mean and an isotropic covariance of $\sigma_x^2 \boldsymbol{I}$. Input for tasks with $Z > 2$ use one-hot centroids in $\mathbb{R}^Z$ with Gaussian noise with variance σ_x. All reward distributions were discrete, with a small number of state-dependent point masses. On every trial, a sample $\tilde{\boldsymbol{x}}$ was drawn from the state-dependent Gaussian distribution and a reward is drawn from the associated state's reward distribution. Rewards are assigned in exact proportion and the training set contains 500 samples per state, unless otherwise stated. An independent validation set of the same size is generated in the same way and is used for monitoring the hidden representation. The pooled training inputs are centered and whitened before training. This makes the input covariance the identity matrix, which is the setting where the learning dynamics of the linear network are exactly solvable (Saxe et al., 2014; Braun et al., 2022).

Analysis definitions. Mode strength is defined as $\boldsymbol{u}_i^\top (\boldsymbol{W}_2\boldsymbol{W}_1)\boldsymbol{v}_i$ for the ith singular value pair. The half-rise time is computed as the first iteration at which the mode strength crosses half of its asymptotic value. The participation ratio for the network is taken on the hidden activation covariance, pooled over both states' validation sets. Hidden state separation is likewise calculated on the hidden activations. The cosine similarity is computed between flattened weight matrices and averaged over simulations.

Figure 1. Here we did not use a network, instead running the simulations directly on the temporal difference update rules (Eqs. (1) and (2)). Fig. 1A,B delivered rewards drawn from a bimodal mixture (half from $\mathcal{N}(3, 1)$ and half from $\mathcal{N}(15, 9)$). Both TD update rules were run on the same sequence of 500 sampled rewards, with $V(0) = 0$; the first 350 steps are shown. The distributional simulation had 15 channels with τ spaced evenly on $[0.01, 0.99]$. Traces are smoothed with a causal running mean of 20 steps.

Fig. 1D uses four distributions: $\mathcal{N}(0, 1), \mathcal{N}(0, 3), \mathcal{N}(1.5, 1)$, and a gamma distribution with shape parameter 1.5. The gamma distribution was standardized to zero mean and unit variance so that it matched the first two moments of the standard normal, and differed only in skew. Their expectiles were found by gradient descent on the expectile loss.

B EXPECTILE BACKGROUND AND DERIVATIONS

B.1 LOCATION SCALE INVARIANCE OF EXPECTILES

Assume some fixed asymmetry parameter $\tau \in (0, 1)$ and a scalar reward r drawn from $p(r)$. The associated expectile loss is given by

$$L_\tau(y) = \mathbb{E}[w_\tau(r, y)(r - y)^2], \qquad \text{with } w_\tau(r, y) = \begin{cases} \tau & \text{if } r \geq y, \\ 1 - \tau & \text{if } r < y. \end{cases} \tag{10}$$

This loss has a unique minimizer (Newey & Powell, 1987), which is by definition the expectile e_τ:

$$\mathbb{E}[w_\tau(r, e_\tau)(r - e_\tau)] = 0. \tag{11}$$

We are interested in showing that expectiles are equivariant to location-scale transforms of the reward. We define $r' = ar + b$, and restrict ourselves to $a > 0$. To show equivariance of expectiles, we need to show that $e'_\tau = ae_\tau + b$ is the unique solution to $\mathbb{E}[w_\tau(r', e'_\tau)(r' - e'_\tau)] = 0$. For the first term in this expectation, we know that $w_\tau(r', e'_\tau) = w_\tau(r, e_\tau)$, since for all $a > 0$, $r' > e'_\tau \iff ar + b > ae_\tau + b \iff r > e_\tau$. For second term, $r' - e'_\tau = a(r - e_\tau)$. Therefore,

$$\mathbb{E}[w_\tau(r', e'_\tau)(r' - e'_\tau)] = a\mathbb{E}[w_\tau(r, e_\tau)(r - e_\tau)] = 0$$

which means that e'_τ is a solution to the equation. By uniqueness, this must be the expectile e'_τ, so $e'_\tau = ae_\tau + b$.

B.2 DECOMPOSING EXPECTILES INTO MEAN AND SHAPE

Using the result above, we can write any reward r as a function of a reward drawn from the standardized distribution $p(\bar{r})$ with $\mathbb{E}[\bar{r}] = 0, \text{var}(\bar{r}) = 1$, such that $r = \mu + \sigma\bar{r}$. Applying this to the expectiles e_τ, then for each expectile at every $\boldsymbol{\tau} = \{\tau_M\}$, $e_\tau(r) = \mu\mathbf{1} + \sigma\boldsymbol{\zeta}$ where we have defined $\boldsymbol{\zeta} = \{\bar{e}_\tau\} \in \mathbb{R}^M$ as the expectiles of the standardized distribution, or the "shape vector". This shape vector is unique to the distributional location-scale family and depends only on the standardized distribution $p(\bar{r})$. This means that every expectile vector in the same family shares a $\boldsymbol{\zeta}$, where $\boldsymbol{\zeta}$ can be calculated analytically for certain distributions and evaluated numerically for all others.

C dRL AND dRL-MSE CONVERGE TO THE SAME SOLUTION

According to Laurent & von Brecht (2018), so long as the loss function is convex and differentiable and the network is not bottlenecked, then no linear neural network has sub-optimal minimizers. This implies that any local minimum is global (Theorem 3 in the Laurent & von Brecht (2018)). Additionally, any minimum of the loss corresponds to the global optimum of the convex single-layer optimization problem. This result makes no assumptions on the structure of the data or targets.

The distributional loss is convex (Newey & Powell, 1987). The proof of Theorem 3 also requires there to be a single-valued derivative at the local optimum, making differentiability of the loss function essential. To show that this result also applies to the asymmetric MSE underlying dRL, Eq. (10), we show that this loss function is differentiable at any point. For any asymmetry τ, the minimum is at $y = e_\tau$. We consider the loss for a single such parameter and associated expectile, but the result also holds for a set of expectiles, as the sum of a finite series of differentiable functions is itself differentiable.

For $y < e_\tau$ and $y > e_\tau$, the expectile loss is equivalent to the MSE scaled by $1-\tau$ and τ, respectively. Therefore, it is differentiable in these regions. The loss is also differentiable at the minimum, $y = e_\tau$, as it is continuous and the first derivative exists. We can show continuity because as $y \to e_{\tau,+}$ and $y \to e_{\tau,-}$, the quadratic term in the loss goes to 0 and the scaling term becomes irrelevant. It is differentiable because taking the limit as $h \to 0$ from both sides of the function

$$\frac{f(y + h) - f(y)}{h}.$$

Here $f(y) = L_\tau(y)$ and we evaluate at the minimum, $y = e_\tau$, in which case the expression simplifies to

$$\left.\frac{L_\tau(y+h) - L_\tau(y)}{h}\right|_{y=e_\tau} = \mathbb{E}\left[w_\tau(r, y+h)\right] h$$

The limit as $h \to 0_+$ is $\lim_{h\to 0_+}(1-\tau)h = 0$. Likewise, from the left side, $\lim_{h\to 0_-}\tau h = 0$. Therefore, the expectile loss is differentiable and we can apply the results of Laurent & von Brecht (2018) to neural networks trained with expectile losses.

If the expectile map is exactly representable by the network, then the expectile loss optimum outputs $\boldsymbol{e}_z$ exactly. Since both the dRL and the dRL-MSE networks share the same minimum ($\boldsymbol{y} = \boldsymbol{e}_z$), they will achieve the same solution. The balance condition in turn implies that the dRL network weights converge to $\boldsymbol{W}_1 \propto \sqrt{\boldsymbol{S}}\boldsymbol{V}^\top$ and $\boldsymbol{W}_2 \propto \boldsymbol{U}\sqrt{\boldsymbol{S}}$, which is unique up to a rotation.

D Theoretical analysis of linear networks

As shown by (Saxe et al., 2014; 2019), the network's learning dynamics can be decomposed into the singular modes $(\boldsymbol{u}_i, s_i, \boldsymbol{v}_i)$ with associated strengths $\sigma_i(t) = \boldsymbol{u}_i^\top \boldsymbol{W}_2(t)\boldsymbol{W}_1(t)\boldsymbol{v}_i$. The dynamics of each mode strength satisfies $\gamma\dot{\sigma}_i = 2\sigma_i(s_i - \sigma_i)$, with solution

$$\sigma_i(t) = \frac{s_i e^{2s_i t\gamma}}{e^{2s_i t\gamma} - 1 + s_i/\sigma_i(0)}, \qquad t_{i,1/2} \approx \frac{\gamma}{2s_i}\ln\frac{s_i}{\sigma_i(0)} \tag{12}$$

where γ is the inverse learning rate. Thus, mode strength dynamics follow a sigmoid that converges on a timescale $\propto \gamma/s_i$, so that modes are learned in order of their singular values. As in Saxe et al. (2014; 2019) we use the half-rise time $t_{i,1/2}$ (time to reach $s_i/2$) as our measure of learning speed throughout. Equation (12) is exact when initial weights are aligned with the singular basis of $\boldsymbol{\Sigma}_{yx}$, and approximate otherwise, as the network aligns to the task basis early in training (Atanasov et al., 2022; Saxe et al., 2019).

E Weight and bias trajectories

E.1 Loss floor without bias term

Without a bias term (or, in our case, if $\boldsymbol{b}_2 = \boldsymbol{0}$) and with centered inputs ($\mathbb{E}[\boldsymbol{x}] = \boldsymbol{0}$), the network output is zero on average, $\mathbb{E}[\hat{\boldsymbol{y}}] = \boldsymbol{W}_2\boldsymbol{W}_1\mathbb{E}[\boldsymbol{x}] = \boldsymbol{0}$. This implies that any task with a nonzero-mean target , $\mathbb{E}[\boldsymbol{y}] \neq \boldsymbol{0}$, cannot be learned exactly, and the loss will never go to zero. Expanding the MSE loss and using $\mathbb{E}\left[\|\boldsymbol{y}\|^2\right] = \|\boldsymbol{\mu}_y\|^2 + \operatorname{tr}\operatorname{Cov}(\boldsymbol{y}) = \|\boldsymbol{\mu}_y\|^2 + \mathbb{E}\big[\|\boldsymbol{y} - \boldsymbol{\mu}_y\|^2\big]$,

$$\begin{aligned}
L &= \tfrac{1}{2}\mathbb{E}\left[\|\boldsymbol{y} - \hat{\boldsymbol{y}}\|^2\right] && (13)\\
&= \tfrac{1}{2}\left(\mathbb{E}\left[\|\boldsymbol{y}\|^2\right] - 2\,\mathbb{E}\left[\boldsymbol{y}^\top\hat{\boldsymbol{y}}\right] + \mathbb{E}\left[\|\hat{\boldsymbol{y}}\|^2\right]\right) && (14)\\
&= \tfrac{1}{2}\|\boldsymbol{\mu}_y\|^2 + \tfrac{1}{2}\mathbb{E}\left[\|\boldsymbol{y} - \boldsymbol{\mu}_y\|^2\right] - \mathbb{E}\left[(\boldsymbol{y} - \boldsymbol{\mu}_y)^\top\hat{\boldsymbol{y}}\right] + \tfrac{1}{2}\mathbb{E}\left[\|\hat{\boldsymbol{y}}\|^2\right] && (15)\\
&= \tfrac{1}{2}\|\boldsymbol{\mu}_y\|^2 + \tfrac{1}{2}\mathbb{E}\left[\|(\boldsymbol{y} - \boldsymbol{\mu}_y) - \hat{\boldsymbol{y}}\|^2\right] \;\geq\; \tfrac{1}{2}\|\boldsymbol{\mu}_y\|^2, && (16)
\end{aligned}$$

where the third line uses $\mathbb{E}\left[\boldsymbol{y}^\top\hat{\boldsymbol{y}}\right] = \mathbb{E}\left[(\boldsymbol{y} - \boldsymbol{\mu}_y)^\top\hat{\boldsymbol{y}}\right] + \boldsymbol{\mu}_y^\top\mathbb{E}[\hat{\boldsymbol{y}}] = \mathbb{E}\left[(\boldsymbol{y} - \boldsymbol{\mu}_y)^\top\hat{\boldsymbol{y}}\right]$, since $\mathbb{E}[\hat{\boldsymbol{y}}] = \boldsymbol{0}$. The second term in the last line is the only part the network can reduce. The network without a bias therefore cannot achieve a loss below $\frac{1}{2}\|\boldsymbol{\mu}_y\|^2$.

Fig. 5 shows this empirically. For these simulations, we trained two dRL-MSE networks on the same training set from identical initial weights, where one had a bias and one did not. The task was the same as the one described in Sec. 3, Fig. 2. All network parameters are left the same, as well as training parameters except the learning rate is faster (here, $\eta = 0.1$) and the number of epochs (here, 300). The results shown are for one simulation and the reported loss is over all M channels of the dRL-MSE. The network's readout at the final iteration (Fig. 5A) demonstrates that the bias-free network cannot fit the target expectiles. It reaches a minimum loss of $\frac{1}{2}\|\boldsymbol{\mu}_y\|^2$ (Fig. 5B). The network with a bias, on the other hand, is able to encode the correct expectiles and achieve zero loss.

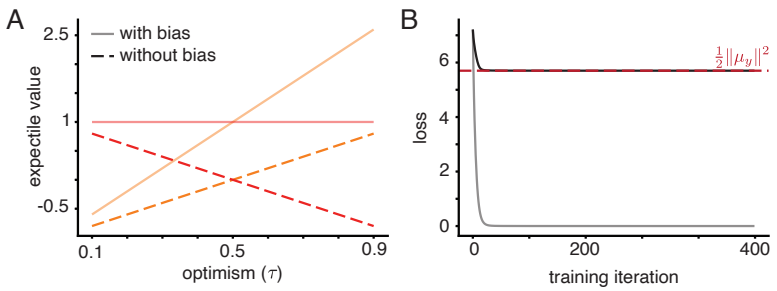

Figure 5: A. Without a bias, the network is unable to learn the correct expectiles, here illustrated for the expectiles of the two-state task shown in Fig. 2. B. With a bias, the network is able to achieve zero loss (gray), whereas without a bias the network's minimum loss is $\frac{1}{2}\|\boldsymbol{\mu}_y\|^2$ (black).

E.2 Network with a bias

Because the task cannot be solved without a bias term, we re-derive the network optima and dynamics equations with a bias term included. The network is

$$\hat{\boldsymbol{y}} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}_2 = \boldsymbol{W}_2\boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{b}_2.$$

We assume batch training with N training examples $\{(\boldsymbol{x}_n, \boldsymbol{y}_n)\}_{n=1}^N$ with non-central moments

$$\boldsymbol{\mu}_x = \frac{1}{N}\sum_n \boldsymbol{x}_n \approx \mathbb{E}[\boldsymbol{x}], \qquad \boldsymbol{\Sigma}_{xx} = \frac{1}{N}\sum_n \boldsymbol{x}_n\boldsymbol{x}_n^\top, \tag{17}$$

$$\boldsymbol{\mu}_y = \frac{1}{N}\sum_n \boldsymbol{y}_n \approx \mathbb{E}[\boldsymbol{y}], \qquad \boldsymbol{\Sigma}_{yx} = \frac{1}{N}\sum_n \boldsymbol{y}_n\boldsymbol{x}_n^\top. \tag{18}$$

Assuming centered, whitened inputs implies $\boldsymbol{\mu}_x = \boldsymbol{0}$ and $\boldsymbol{\Sigma}_{xx} = \boldsymbol{I}$, such that non-centered input-output moment becomes for large N equal to the input-output covariance matrix, $\boldsymbol{\Sigma}_{yx} \to \mathrm{cov}(\boldsymbol{y}, \boldsymbol{x})$.

Fixed point. Starting from the MSE loss,

$$L = \frac{1}{2N}\sum_n \|\boldsymbol{y}_n - (\boldsymbol{W}_2\boldsymbol{W}_1\boldsymbol{x}_n + \boldsymbol{b}_2)\|^2, \tag{19}$$

The residual is $\boldsymbol{\varepsilon}_n = \boldsymbol{y}_n - \boldsymbol{W}_2\boldsymbol{W}_1\boldsymbol{x}_n - \boldsymbol{b}_2$, and taking derivatives with respect to the parameters gives

$$\frac{\partial L}{\partial \boldsymbol{W}} = -\frac{1}{N}\sum_n \boldsymbol{\varepsilon}_n\boldsymbol{x}_n^\top, \qquad \frac{\partial L}{\partial \boldsymbol{b}_2} = -\frac{1}{N}\sum_n \boldsymbol{\varepsilon}_n, \tag{20}$$

where we have used $\boldsymbol{W} = \boldsymbol{W}_2\boldsymbol{W}_1$. Setting both to zero results in

$$\boldsymbol{W}\boldsymbol{\Sigma}_{xx} = \boldsymbol{\Sigma}_{yx} - \boldsymbol{b}_2\boldsymbol{\mu}_x^\top, \qquad \boldsymbol{b}_2 = \boldsymbol{\mu}_y - \boldsymbol{W}\boldsymbol{\mu}_x. \tag{21}$$

Substituting the optimal $\boldsymbol{b}_2$ into the equation for $\boldsymbol{W}$ finally yields

$$\boldsymbol{W}\boldsymbol{\Sigma}_{xx} = \boldsymbol{\Sigma}_{yx} - \boldsymbol{\mu}_y\boldsymbol{\mu}_x^\top + \boldsymbol{W}\boldsymbol{\mu}_x\boldsymbol{\mu}_x^\top \quad \Rightarrow \quad \boldsymbol{W}\tilde{\boldsymbol{\Sigma}}_{xx} = \tilde{\boldsymbol{\Sigma}}_{yx} \quad \Rightarrow \quad \boldsymbol{W} = \tilde{\boldsymbol{\Sigma}}_{yx}(\tilde{\boldsymbol{\Sigma}}_{xx})^\dagger, \tag{22}$$

where $\tilde{\boldsymbol{\Sigma}}$ denotes a centered covariance matrix ($\tilde{\boldsymbol{\Sigma}}_{yx} = \boldsymbol{\Sigma}_{yx} - \boldsymbol{\mu}_y\boldsymbol{\mu}_x^\top$, $\tilde{\boldsymbol{\Sigma}}_{xx} = \boldsymbol{\Sigma}_{xx} - \boldsymbol{\mu}_x\boldsymbol{\mu}_x^\top$). This is exactly the bias-free solution, but on the centered covariance matrices. Therefore, the optimal weights are determined by the SVD of the centered covariance matrices (the MWNS of Braun et al. (2025)). Intuitively, this means the bias absorbs the mean $\boldsymbol{\mu}_y$ that $\boldsymbol{W}_2\boldsymbol{W}_1$ cannot represent.

Learning dynamics. If we assume that $\Sigma_{xx} = I$ and full-batch updates with a small learning rate, the parameter updates become

$$\gamma\dot{\boldsymbol{b}}_2 = -\frac{\partial L}{\partial \boldsymbol{b}_2} = \frac{1}{N}\sum_n \boldsymbol{\varepsilon}_n = \boldsymbol{\mu}_y - \boldsymbol{b}_2, \tag{23}$$

$$\gamma\dot{\boldsymbol{W}}_2 = -\frac{\partial L}{\partial \boldsymbol{W}_2} = (\boldsymbol{\Sigma}_{yx} - \boldsymbol{W}_2\boldsymbol{W}_1)\,\boldsymbol{W}_1^\top, \tag{24}$$

$$\gamma\dot{\boldsymbol{W}}_1 = -\frac{\partial L}{\partial \boldsymbol{W}_1} = \boldsymbol{W}_2^\top(\boldsymbol{\Sigma}_{yx} - \boldsymbol{W}_2\boldsymbol{W}_1), \tag{25}$$

where γ is the inverse learning rate. Since $\boldsymbol{\mu}_x = \mathbf{0}$, $\boldsymbol{b}_2$ fully decouples from $\boldsymbol{W}_2$ and $\boldsymbol{W}_1$, making the bias and the weights evolve independently. The differential equation for $\boldsymbol{b}_2$ can be solved exactly,

$$\gamma \dot{\boldsymbol{b}}_2 = \boldsymbol{\mu}_y - \boldsymbol{b}_2 \quad \Rightarrow \quad \boldsymbol{b}_2(t) = \boldsymbol{\mu}_y + (\boldsymbol{b}_2(0) - \boldsymbol{\mu}_y) e^{-t/\gamma}, \tag{26}$$

which describes an exponential relaxation to $\boldsymbol{\mu}_y$ on a timescale γ. This timescale is independent of the task singular values s_i, while the dynamics of $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ remains governed by the bias-free dynamics from Eq. (12) (since $\boldsymbol{\mu}_x = \mathbf{0}$ makes $\boldsymbol{\Sigma}_{yx} = \tilde{\boldsymbol{\Sigma}}_{yx}$). Likewise, balance $\boldsymbol{W}_2^\top \boldsymbol{W}_2 - \boldsymbol{W}_1 \boldsymbol{W}_1^\top$ is conserved throughout, unaffected by the bias. The network parameters therefore decompose into separate objects: $\boldsymbol{W}_2 \boldsymbol{W}_1$, which carries the differential structure across states, and $\boldsymbol{b}_2$, which contains signal shared by all states.

We note that a bias on the hidden layer, $\boldsymbol{b}_1$, also admits an exact treatment, but couples the dynamics of $\boldsymbol{W}_1$ and $\boldsymbol{b}_1$. With biases on both layers the solution becomes initialization-dependent, as $\boldsymbol{\mu}_y$ is split between $\boldsymbol{W}_2 \boldsymbol{b}_1$ and $\boldsymbol{b}_2$. Therefore, we restrict ourselves to networks with biases only at the final layer.

F Learning speed and hidden representations for a two-state task

We consider samples n drawn from one of two states $z \in \{1, 2\}$, with input $\boldsymbol{x}_n \sim p(\boldsymbol{x}_n|z) \in \mathbb{R}^{N_x}$ and prediction targets $\boldsymbol{y}_n \sim P(y_n|z) \in \mathbb{R}^{N_y}$ that are either mean rewards $\mu_z = \mathbb{E}[r|z]$ for the cRL network, or expectiles $\boldsymbol{e}_z$ of state-dependent reward distributions $p(r|z)$ for the dRL network. Inputs are centered such that $\frac{1}{N}\sum_n \boldsymbol{x}_n = \mathbf{0}$. Each state z has n_z training samples (where $n_1 + n_2 = N$) and a state-conditioned input and output mean, $\bar{\boldsymbol{x}}_z$ and $\bar{\boldsymbol{y}}_z$. The covariance matrix $\boldsymbol{\Sigma}_{yx}$ is given by

$$\boldsymbol{\Sigma}_{yx} = \frac{1}{N}\sum_n \boldsymbol{y}_n \boldsymbol{x}_n^\top = \sum_z \frac{n_z}{N} \left(\frac{1}{n_z} \sum_{n \in z} \boldsymbol{y}_n \boldsymbol{x}_n^\top \right) = \sum_z \frac{n_z}{N} \bar{\boldsymbol{y}}_z \bar{\boldsymbol{x}}_z^\top. \tag{27}$$

Centered inputs imply that $\sum_n \boldsymbol{x}_n = n_1 \bar{\boldsymbol{x}}_1 + n_2 \bar{\boldsymbol{x}}_2 = \mathbf{0}$. With equiprobable states, $n_1 = n_2 = N/2$, this implies that the two state means are additive inverses $\bar{\boldsymbol{x}}_2 = -\bar{\boldsymbol{x}}_1$. Substituting this into Eq. (27) results in

$$\boldsymbol{\Sigma}_{yx} = \tfrac{1}{2}\, \bar{\boldsymbol{y}}_1\, \bar{\boldsymbol{x}}_1^\top + \tfrac{1}{2}\, \bar{\boldsymbol{y}}_2\, (-\bar{\boldsymbol{x}}_1)^\top = \tfrac{1}{2}\, (\bar{\boldsymbol{y}}_1 - \bar{\boldsymbol{y}}_2)\, \bar{\boldsymbol{x}}_1^\top. \tag{28}$$

As this is an outer product of two vectors, the rank of $\boldsymbol{\Sigma}_{yx}$ is one, with singular value decomposition

$$s = \tfrac{1}{2} \|\bar{\boldsymbol{y}}_1 - \bar{\boldsymbol{y}}_2\|\, \|\bar{\boldsymbol{x}}_1\|, \qquad \boldsymbol{u} \propto \bar{\boldsymbol{y}}_1 - \bar{\boldsymbol{y}}_2, \qquad \boldsymbol{v} \propto \bar{\boldsymbol{x}}_1. \tag{29}$$

If the two states occur with different probabilities, the result changes by a prefactor. With centered inputs, non-equiprobable states yield $n_1 \bar{\boldsymbol{x}}_1 = -n_2 \bar{\boldsymbol{x}}_2$ and $\boldsymbol{\Sigma}_{yx} = \frac{n_1}{N}(\bar{\boldsymbol{y}}_1 - \bar{\boldsymbol{y}}_2)\bar{\boldsymbol{x}}_1^\top$, with rank and singular value decomposition unaffected.

In our two-state task, the cRL network aims to predict the mean reward, $\bar{y}_z = \mu_z$, so $\boldsymbol{\Sigma}_{yx}^{\mathrm{cRL}} = \frac{1}{2}(\mu_1 - \mu_2)\, \bar{\boldsymbol{x}}_1^\top$ and $s^{\mathrm{cRL}} = \frac{1}{2}\, |\mu_1 - \mu_2|\, \|\bar{\boldsymbol{x}}_1\|$. The dRL network has a target vector $\bar{\boldsymbol{y}}_z = \boldsymbol{e}_z \in \mathbb{R}^M$, resulting in $\boldsymbol{\Sigma}_{yx}^{\mathrm{dRL}} = \frac{1}{2}(\boldsymbol{e}_1 - \boldsymbol{e}_2)\, \bar{\boldsymbol{x}}_1^\top$ and $s^{\mathrm{dRL}} = \frac{1}{2}\, \|\boldsymbol{e}_1 - \boldsymbol{e}_2\|\, \|\bar{\boldsymbol{x}}_1\|$.

F.1 Classical vs distributional singular values and learning speed

As discussed in the main text, we can decompose the expectile vector for each state z into $\boldsymbol{e}_z = \mu_z \mathbf{1} + \sigma_z \boldsymbol{\zeta}_z$. For what follows we assume both states to be equally likely, and for the expectile vector to be symmetric around the mean, $\tau = \frac{1}{2}$. Then, for the cRL network, $s^{\mathrm{cRL}} = \frac{1}{2}|\Delta\mu| \|\bar{\boldsymbol{x}}_1\|$, where $\Delta\mu = \mu_1 - \mu_2$. For the dRL network, $s^{\mathrm{dRL}} = \frac{1}{2}\|\Delta \boldsymbol{e}\| \|\bar{\boldsymbol{x}}_1\|$, where $\Delta \boldsymbol{e} = \Delta\mu \mathbf{1} + \Delta\boldsymbol{\eta}$, with $\Delta\boldsymbol{\eta} = \sigma_1 \boldsymbol{\zeta}_1 - \sigma_2 \boldsymbol{\zeta}_2$. For the mean expectile with $\tau = \frac{1}{2}$, $\Delta e_\tau = \Delta\mu$. Since the norm of a vector is at least the absolute value of any of its components,

$$\|\Delta \boldsymbol{e}\| \geq |\Delta\mu| \quad \Rightarrow \quad s^{\mathrm{dRL}} \geq s^{\mathrm{cRL}}, \tag{30}$$

with equality only if $M = 1$, for any two reward distributions.

Pure mean task. If the reward distributions associated with the two states share the same scale and shape, we have $\Delta\boldsymbol{\eta} = \mathbf{0}$ such that $\Delta \boldsymbol{e} = \Delta\mu \mathbf{1}$, and

$$s^{\mathrm{dRL}} = \tfrac{1}{2}\|\Delta\mu \mathbf{1}\| \|\bar{\boldsymbol{x}}_1\| = \sqrt{M} s^{\mathrm{cRL}} \tag{31}$$

In this case, the M dRL network target outputs are M repeats of the same scalar $\Delta\mu$. Thus, the speed gain of the dRL network over cRL network arises purely from the larger number of outputs, and not the presence of any information about the reward distributions. For distributional information to introduce any learning speed gains beyond that, we require $\Delta\boldsymbol{\eta} \neq \mathbf{0}$, i.e., either a difference in distribution variance, shape, or both, across the two states.

General case. Expanding $\|\Delta\boldsymbol{e}\|^2$, we find

$$\|\Delta\boldsymbol{e}\|^2 = M\Delta\mu^2 + 2\Delta\mu\mathbf{1}^\top\Delta\boldsymbol{\eta} + \|\Delta\boldsymbol{\eta}\|^2. \tag{32}$$

If $p(r|z)$ is symmetric for both $z \in \{1, 2\}$, and the asymmetry parameter grid, $\tau_1, \ldots, \tau_M$, is symmetric around $\tau = \frac{1}{2}$, then $\mathbf{1}^\top\boldsymbol{\zeta}_z = 0$ and the cross term $\mathbf{1}^\top\Delta\boldsymbol{\eta}$ vanishes. This gives

$$\frac{s^{\mathrm{dRL}}}{s^{\mathrm{cRL}}} = \sqrt{M + \frac{\|\Delta\boldsymbol{\eta}\|^2}{\Delta\mu^2}} \geq \sqrt{M}. \tag{33}$$

which shows that differences in scale and shape of the two reward distributions can boost the learning rate beyond that induced by the increased number of outputs. If $\mathbf{1}^\top\boldsymbol{\zeta}_z \neq 0$ (which happens when the grid is not symmetric around $\tau = \frac{1}{2}$, or for skewed distributions on symmetric grids), the cross term $\mathbf{1}^\top\Delta\boldsymbol{\eta}$ can become negative, which might cause this ratio to drop below $\sqrt{M}$, but never below 1 .

F.2 Convergence to theoretical results (learning trajectory)

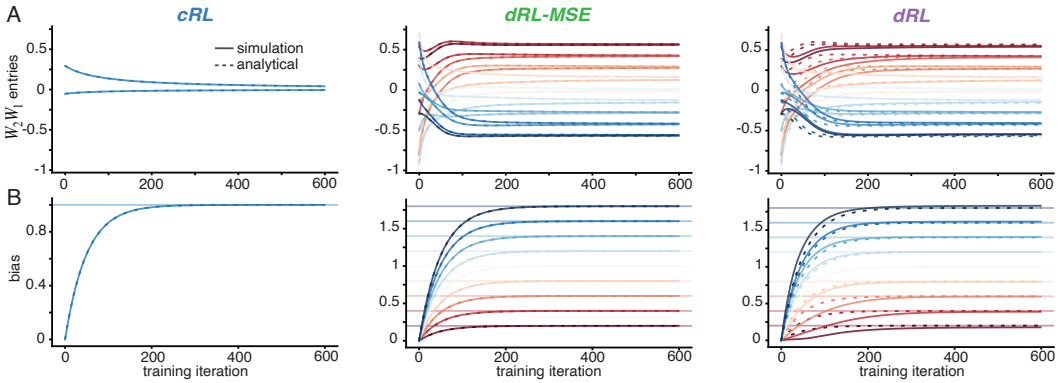

Figure 6: A & B. Analytical (dashed) and simulation (solid) trajectories of each entry of the network weights ($\boldsymbol{W}_2\boldsymbol{W}_1$; A) and output bias ($\boldsymbol{b}_2$, B) after random weight initialization of the cRL (left), dRL-MSE (center) and dRL (right) networks. The horizontal lines in B show the theoretical asymptotic values of each entry in b_2.

F.3 dRL and dRL-MSE similarity

We know the endpoint is the same for both networks (see C), so now we ask if the learning trajectories are similar to each other. Our simulations of the two-state task (Fig. 2) show that the dRL network weights remain closely aligned to those of the dRL-MSE network throughout training (cosine similarity ≈ 1), but not those of the cRL network (Fig.7A). Despite matching weights, the dRL network learns more slowly than the dRL-MSE network (Fig. 7B). This speed difference is not due to the different loss functions, as these predict similar learning speeds for the equiprobable reward distributions used in this task ($S_{\mathrm{net}} = 1$, see App. I). Instead, it arises through differences in learning the bias term $\boldsymbol{b}_2$. This is because the effective gradient of the dRL network loss depends on where the rewards sit with respect to the network's current predictions, which is determined by the current bias $\boldsymbol{b}_2(t)$. Despite this bias-dependent difference, the overall shape of the learning curves closely matches between the dRL and dRL-MSE networks, as seen by the agreement of the re-scaled dRL trajectory with the dRL-MSE trajectory. (Fig.7C).

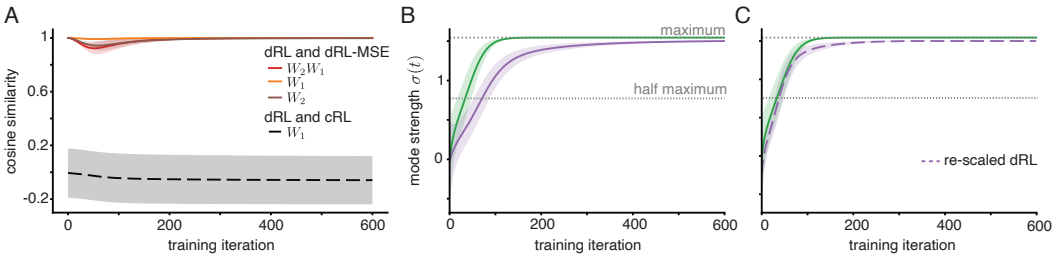

Figure 7: A. Cosine similarity between the dRL and the dRL-MSE network weights (red), input weights (orange) and output weights (brown) across 20 simulations (mean ± SD). As a control, we also show the cosine similarity between the dRL and cRL input weights (black). B. Mode strengths over learning for the dRL (purple) and dRL-MSE (green) networks, showing the half-maximum (used to calculate the half-rise time) and maximum values. C. Same as B but with the time axis compressed by a factor of 2 ($t \mapsto t/2$) (empirically determined), to show that the dRL learning trajectory shape matches that of the dRL-MSE network, with the slowdown arising from the reference-dependence effect due to $\boldsymbol{b}_2$

F.4 Hidden layer separability

The hidden representation of the two-layer neural network is $\boldsymbol{h} = \boldsymbol{W}_1\boldsymbol{x}$. We are interested in computing how well this hidden representation separates across the two states after training, as quantified by $\Delta h = \|\bar{\boldsymbol{h}}_1 - \bar{\boldsymbol{h}}_2\| = \|\boldsymbol{W}_1\bar{\boldsymbol{x}}_1 - \boldsymbol{W}_1\bar{\boldsymbol{x}}_2\|$. Using $\boldsymbol{W}_1 = \sqrt{s}\boldsymbol{v}^\top$ after training, this separation measure turns into

$$\Delta h = \|\bar{\boldsymbol{h}}_1 - \bar{\boldsymbol{h}}_2\| = \|\sqrt{s}\boldsymbol{v}^\top\bar{\boldsymbol{x}}_1 - \sqrt{s}\boldsymbol{v}^\top\bar{\boldsymbol{x}}_2\| = 2\sqrt{s}\frac{\|\bar{\boldsymbol{x}}_1\|^2}{\|\bar{\boldsymbol{x}}_1\|} = 2\sqrt{s}\|\bar{\boldsymbol{x}}_1\| \tag{34}$$

where we used $\bar{\boldsymbol{x}}_2 = -\bar{\boldsymbol{x}}_1$ and $\boldsymbol{v} = \bar{\boldsymbol{x}}_1/\|\bar{\boldsymbol{x}}_1\|$. Likewise, we can get the trajectory of the separation $\Delta h(t)$ from the trajectory of $s(t)$. This shows that the hidden state separability after training is proportional to s and implies that hidden states remain separable for the dRL and dRL-MSE networks, but collapse onto a non-separable representation for the cRL network. We confirm this in simulation in Fig. 8A, B and show in App. F.5 that the same dissociation holds for ReLU networks (Fig. 8C,D).

F.5 Validation in non-linear networks

To test whether our two-state results depend on linearity, we repeated the simulations of Sec. 3 with a ReLU nonlinearity on the hidden layer ($\hat{\boldsymbol{y}} = \boldsymbol{W}_2 f(\boldsymbol{W}_1\boldsymbol{x}) + \boldsymbol{b}_2$, $f(x) = \max(0, x)$), keeping all other parameters as in App. A. The ReLU dRL network converges to the same loss as its linear counterpart on a comparable timescale (Fig. 9A). The central qualitative prediction also carries over: the hidden representations of the two states collapse in the ReLU cRL network but remain separated in both distributional networks (Fig. 8C), with Δh decaying for cRL and rising to a plateau for dRL and dRL-MSE (Fig. 8D).

The nonlinear networks deviate quantitatively from the linear theory in three ways. First, the asymptotic separation lies below the linear prediction $2\sqrt{s}\|\bar{\boldsymbol{x}}_1\|$ (Fig. 8D, dashed line). Second, the hidden representation is higher dimensional (Fig. 9B vs. C). We find that this qualitative observation holds across our other simulations. Third, the timescale for the collapse of the state representations in the cRL is much slower. In general we also find that the hidden representation of the non-linear networks takes longer to converge to steady state; we expect, therefore, that over a longer time horizon the state separability in the cRL will go to zero.

G Relative reward changes across states

To assess how different moments (mean μ, variance σ^2, and skew κ) of the reward distributions impact the learning speed of cRL and dRL, we used two-point reward distributions $p(r|z) = p_z\delta(r - a_z) + (1 - p_z)\delta(r - b_z)$ with state-dependent distribution parameters p_z, a_z, and b_z. For each task we held two of (μ, σ, κ) fixed across states while varying the third. To test the impact of different

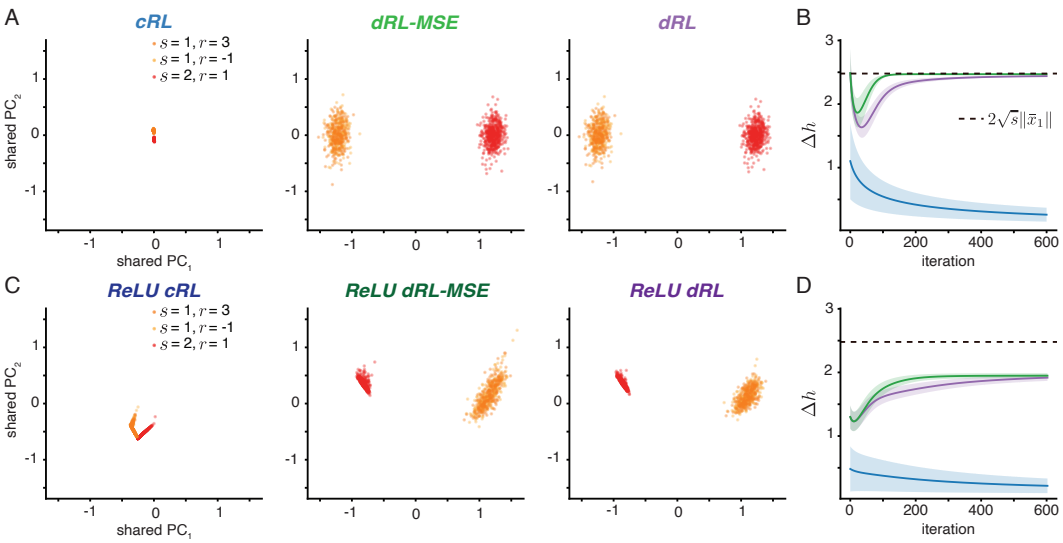

Figure 8: A. PCA of the hidden shared subspace for the cRL (left), dRL-MSE (center) and dRL (right) networks after having learned the two-state task from Fig. 2 (orange: $z = 1$; red: $z = 2$). B. Separability Δh of the three networks over learning (mean $\pm$ SD across 20 simulations). Theoretical maximum separability for the dRL network shown in black. C, D are the same as the first row, but for networks with a ReLU nonlinearity.

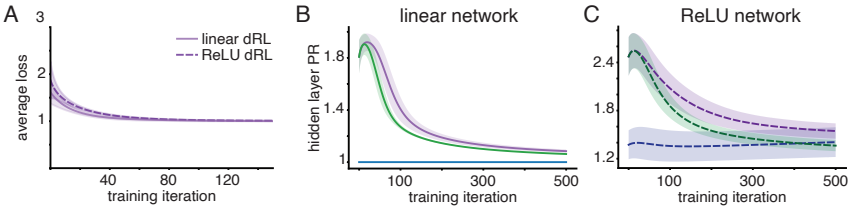

Figure 9: A. Average loss over 20 simulations (mean $\pm$ SD) of the task described in Sec. 3 for the linear (light purple) and nonlinear (dark purple) dRL network. B. & C. Participation ratio of the hidden layer for the linear (B) and non-linear (C) cRL (blue), dRL (purple) and dRL-MSE (green) networks.

means, both states had symmetric reward distributions ($p_1 = p_2 = \frac{1}{2}$) with matched σ's but different μ's. To test the impact of different variances, we again used symmetric reward distributions with the same μ but different σ's. To test the impact of skew, we used a symmetric reward distribution for $z = 1$, and a skewed one for $z = 2$ ($p_2 \neq \frac{1}{2}$). In all cases, we adjusted the distribution's (p_z, a_z, b_z) parameters to have them feature the desired moments (μ, σ, κ). For the dRL-MSE network we computed expectile targets exactly using the closed-form two-point expectile equation, and each output was initialized to the zero-balanced regime so the half-rise time applies with no alignment transient. We trained the cRL, dRL, and dRL-MSE networks on identical batches, and for each tested moment determined both the final singular value s (Fig.10A) as well as the half-rise time (Fig.10B) for multiple differences of the tested moment across the two states. As the figures illustrate, we found a close match between simulations and the closed-form predictions of our theory.

Simulation details. The weights are initialized to be small ($\sigma = 0.1$), otherwise all parameters are kept standard. For the mean difference sweep, both states are symmetric with $\sigma_z = 1.5$ and with means $\mu_z \pm \Delta\mu/2$, where $\Delta\mu \in \{0.5, 1, 1.5, 2.5\}$. For the variance sweep, state $z = 2$ is kept fixed with $\mu_2 = 1, \sigma_2 = 1.5$. State $z = 1$ has $\mu_1 = 1$ and $\sigma_1 \in \{1.8, 2, 3, 4\}$. For the skew sweep, $z = 2$ is the symmetric baseline ($\mu_2 = 1, \sigma_2 = 1.5$). State $z = 1$ has the same mean and variance, but with asymmetric probabilities: $p(\text{high}) \in \{0.7, 0.8, 0.88, 0.94\}$.

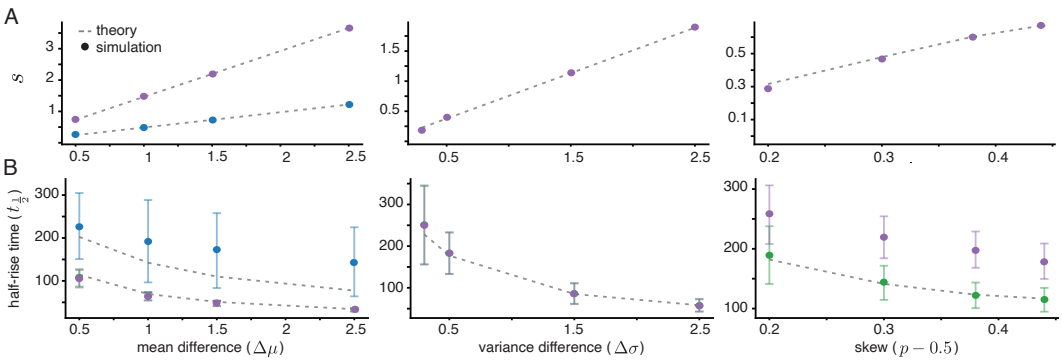

Figure 10: A. Empirically calculated singular values s for changes in mean ($\Delta\mu$, left), variance ($\Delta\sigma$, center) and skew ($\Delta\kappa$, controlled through p_z, right) for the dRL network (purple) and dRL-MSE network (green; below purple dots and so not visible), with all other moments matching across both states. The cRL network singular value (blue) is shown only for $\Delta\mu$, as s^{cRL} is unaffected by changes in other moments. B. Empirically measured half-rise time ($t_{1/2}$) for the simulations. Theoretical predictions shown in gray, and error bars are across 20 simulations (mean $\pm$ SD).

H THE IMPACT OF REWARD LOCATION AND SCALE TRANSFORMS

As we have shown in App. B.1, a linear transformation of every reward by $r \mapsto ar + b$ with $a > 0$ leads to a corresponding linear transformation $\boldsymbol{e}_z \mapsto a\boldsymbol{e}_z + b\mathbf{1}$ of the expectile vector of each state. The dRL-MSE network uses these expectile vectors as output targets, $\boldsymbol{y} = \boldsymbol{e}$, such that its learning dynamics depend on these expectile targets only through $\boldsymbol{\Sigma}_{yx} = \mathbb{E}[\boldsymbol{y}\boldsymbol{x}^\top] = \mathbb{E}[\boldsymbol{e}\boldsymbol{x}^\top]$.

Reward shifts. With centered inputs ($\mathbb{E}[\boldsymbol{x}] = 0$), a constant added to $\boldsymbol{y}$ has no impact on $\boldsymbol{\Sigma}_{yx}$,

$$\mathbb{E}[(\boldsymbol{y} + b\mathbf{1})\boldsymbol{x}^\top] = \mathbb{E}[\boldsymbol{y}\boldsymbol{x}^\top] + b\mathbf{1}\mathbb{E}[\boldsymbol{x}]^\top = \boldsymbol{\Sigma}_{yx}. \tag{35}$$

Since $\boldsymbol{\Sigma}_{yx}$ is unaffected, so is $\boldsymbol{W}_2\boldsymbol{W}_1$, its singular vectors $\boldsymbol{u}$ and $\boldsymbol{v}$, its singular value s, and the network's hidden representation. The reward shift only affects the bias, which relaxes to the new mean, $\boldsymbol{b}_2 \to \bar{\boldsymbol{y}} + b\mathbf{1}$.

Scaling rewards. A re-scaling of rewards by $a > 0$ scales the input-output covariance matrix $\boldsymbol{\Sigma}_{yx}$:

$$\mathbb{E}[a\boldsymbol{y}\boldsymbol{x}^\top] = a\mathbb{E}[\boldsymbol{y}\boldsymbol{x}^\top] = a\boldsymbol{\Sigma}_{yx}, \tag{36}$$

which can be rewritten as $a\boldsymbol{\Sigma}_{yx} = \boldsymbol{U}(a\boldsymbol{S})\boldsymbol{V}^\top$: the singular vectors remain unaffected, but every singular value is multiplied by a ($s \mapsto as$). Since the half-rise time decreases with s, scaling rewards up speeds learning approximately by a factor of a, while the learned weights and hidden representation keeps the same geometry.

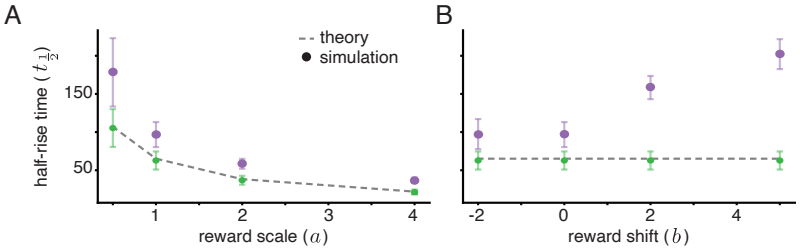

Figure 11: A & B. Half-rise time of the dRL (purple) and dRL-MSE (green) networks under reward scaling (A) and reward shifts (B). Error bars are across 20 simulations (mean $\pm$ SD)

In simulation we found that the dRL-MSE network matched our predictions (Fig.11). Furthermore, the dRL network qualitatively matched the scale predictions (Fig.11A), but was not invariant to reward shifts, instead featuring slower learning with increasing b (Fig.11B). We can trace the origin of this slowdown to the reference-dependence of the asymmetric expectile loss, which weighs errors

by τ or $1-\tau$ according to the sign of the prediction error. The effective gradient thus depends on where the rewards sit with respect to the overall network signal, which is determined by network's current bias $\boldsymbol{b}_2(t)$.

Simulation details. The weights are initialized to be small ($\sigma = 0.1$), otherwise all parameters are kept standard. The base task is the same as in Sec. 3. Rewards are mapped to $ar + b$ with $a \in \{0.5, 1, 2, 4\}, b = 0$ for the scale sweep and $b \in \{-2, 0, 2, 5\}, a = 1$ for the shift sweep.

I LEARNING SPEED DIFFERENCES IN DRL VS. DRL-MSE NETWORKS

Here, we show that, due to its use of the asymmetric expectile loss function, the dRL network can learn at a different speed than the dRL-MSE network, and that this speed difference depends on the shape of the reward distribution and the learned expectiles.

I.1 LEARNING SPEED FOR LEARNING INDIVIDUAL EXPECTILES

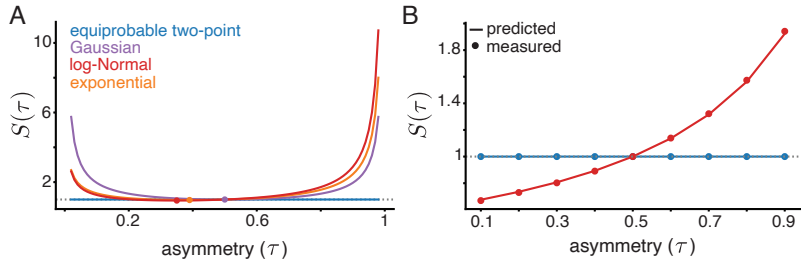

Figure 12: A. $S(\tau)$ for two symmetric reward distributions (equiprobable two-point, blue and Gaussian, purple) and two non-symmetric reward distributions (log-Normal, red, and exponential, orange). The log-Normal and exponential distributions have minimum values at $S \approx 0.97$ and 0.94, respectively, indicating a relative speedup of the dRL network over the dRL-MSE network. B. $S(\tau)$ for different asymmetries τ for a two-point reward distribution with $p_z = 0.5$ (blue; equiprobable) and $p_z = 0.2$ (red; skewed).

Consider a single output with asymmetry τ that aims to learn expectile e_τ of the reward distribution $p(r)$. Recall that, for network output y, the dRL loss Eq. (10) is

$$L_{\tau,\mathrm{dRL}}(y) = \mathbb{E}[w_\tau(r,y)(r-y)^2], \qquad w_\tau(r,y) = \begin{cases} \tau & \text{if } r \geq y, \\ 1-\tau & \text{if } r < y, \end{cases} \tag{37}$$

and the corresponding dRL-MSE loss is $L_{\tau,\mathrm{dRL-MSE}}(y) = \frac{1}{2}(e_\tau - y)^2$. The loss function gradients are

$$L'_{\tau,\mathrm{dRL}} = -2\mathbb{E}[w(r,y)(r-y)], \qquad L'_{\tau,\mathrm{dRL-MSE}} = -(e_\tau - y). \tag{38}$$

The expectile e_τ is the minimizer of both $L_{\tau,\mathrm{dRL}}$ and $L_{\tau,\mathrm{dRL-MSE}}$, that is the solution of both $\mathbb{E}[w_\tau(r,e_\tau)(r-e_\tau)] = 0$ and $e_\tau - y = 0$, resulting in $e_\tau = \mathbb{E}[w_\tau r]/\mathbb{E}[w_\tau]$. Near this fixed point,

$$L'_{\tau,\mathrm{dRL}} = -2\mathbb{E}[w_\tau]\left(\frac{\mathbb{E}[w_\tau r]}{\mathbb{E}[w_\tau]} - y\right) = 2\mathbb{E}[w_\tau]L'_{\tau,\mathrm{dRL-MSE}}, \tag{39}$$

meaning that the dRL network learns the e_τ expectile at a speed $2\mathbb{E}[w_\tau]$ times that of the dRL-MSE network. We define this per-expectile speed difference by

$$S(\tau) = \frac{L'_{\tau,\mathrm{dRL-MSE}}}{L'_{\tau,\mathrm{dRL}}} = \frac{1}{2\mathbb{E}[w_\tau]}, \qquad \mathbb{E}[w_\tau] = \tau p(r \geq e_\tau) + (1-\tau)p(r < e_\tau), \tag{40}$$

where the scaling $S(\tau) > 1$ implies that the dRL network learns more slowly than the dRL-MSE network.

For a two-point reward distribution given by $p(r|z) = p_z\delta(r-a_z) + (1-p_z)\delta(r-b_z)$ (assuming $a_z \geq b_z$) used in Fig. 2 and App. G, we have $S(\tau) = 1/[2(\tau p_z + (1-\tau)(1-p_z))]$ exactly.

If the two rewards are equiprobable, that is $p_z = \frac{1}{2}$, this speed difference reduces to $S(\tau) = 1$. The same holds for $\tau = 0.5$. In other words, for two-point reward distributions for which either $p_z = \frac{1}{2}$ or $\tau = 0.5$, the dRL network learns at the same speed as the dRL-MSE.

A slowdown for symmetric reward distributions. Assume a reward distribution $p(r)$ that has half of its probability mass above and below its mean, that is $P(r \geq \mathbb{E}[r]) = P(r < E[r]) = \frac{1}{2}$. This is the case, for example, if this reward distribution is symmetric around its mean and has no additional point mass at its median. Furthermore, it is easy to show that $\mathbb{E}[w_\tau(r, y)] = \frac{1}{2}$ for $\tau = \frac{1}{2}$, irrespective of the reward distribution. Since we know that e_τ is increasing in τ, such that $P(r \geq e_\tau)$ will be decreasing in τ, that is $\partial_\tau P(r \geq e_\tau) \leq 0$, then

$$\partial_\tau E[w_\tau(r, y)] = -1 + 2P(r \geq e_\tau) + (2\tau - 1)\partial_\tau P(r \geq e_\tau) = \begin{cases} \geq 0 & \text{if } \tau < 1/2, \\ \leq 0 & \text{if } \tau > 1/2, \end{cases} \tag{41}$$

where we have used $2P(r \geq e_\tau) \geq 1$ (as $P(r \geq e_{0.5}) = 1/2$ by the above assumption) and $(2\tau - 1)\partial_\tau P(r \geq e_\tau) \geq 0$ for $\tau < 1/2$, and $2P(r \geq e_\tau) \leq 1$ and $(2\tau - 1)\partial_\tau P(r \geq e_\tau) \leq 0$ for $\tau > 1/2$. This implies that $\mathbb{E}[w_\tau(r, y)]$ is increasing below $\tau = 1/2$ and decreasing above it, such that it peaks at $\tau = 1/2$. As a consequence $E[w_\tau(r, y)] \leq 1/2$ and $S(\tau) \geq 1$, always, which implies that, for reward distributions with $P(r \geq \mathbb{E}[r]) = \frac{1}{2}$ (for example, the aforementioned symmetric distribution), the learning speed of the dRL network is slower than that of the dRL-MSE network, with larger slowdowns for asymmetries τ closer to 0 or 1 (see Fig. 12A, Gaussian and equiprobable distributions).

A potential speedup for skewed reward distributions. For skewed reward distributions, mean and median might not match. As a result, some expectiles may be learned faster by the dRL than the dRL-MSE network. This can be seen in Fig. 12A, where the log-Normal and exponential distributions have a minimum at $S(\tau) = 0.97$ and 0.94, respectively, representing a relative speedup. The speedup effect can be seen more clearly in a two-point skewed distribution; for example, $p_z = 0.2$ and $\tau = 0.2$ results in $S(0.2) = \frac{1}{2(0.68)} \approx 0.74$, which implies faster learning for the dRL network. In general, every expectile with $\tau > \frac{1}{2}$ is learned more slowly than the dRL-MSE network if $p_z < \frac{1}{2}$ (and every expectile with $\tau < \frac{1}{2}$ is learned faster), as we show in Fig. 12B.

The last two paragraphs together imply that we can only guarantee slower learning of the dRL network for reward distributions with matching mean and median, in which case the slowdown is larger for τ's further away from $\tau = 1/2$. For example, as $\tau \to 1$, $\mathbb{E}[w_\tau] \to P(r > e_\tau)$, that is, the probability that the reward is larger than the expectile. This probability decreases as the expectile increases, especially for distributions with light tails. Therefore, $S(\tau)$ increases with τ. A similar argument establishes an increasing $S(\tau)$ for decreasing $\tau < 1/2$. For asymmetric distributions, in contrast, lower expectiles can be learned faster if most of the probability mass is below the mean, and vice versa.

In general, the per-expectile slowdown in learning with the dRL loss is highest for small or large τ, with the intuition that learning is slow in instances in which w_τ is either mostly τ or $1 - \tau$, which is the case at the tails of the reward distribution.

One caveat to the arguments in this section is that $S(\tau)$ is defined under the assumption that $L'_{\tau,\mathrm{dRL}}$ is close to zero, implying that learning is almost complete. Away from this minimum, the learning speed is proportional to $\mathbb{E}[w_\tau(\cdot, y)]$ at the *current* y. Furthermore, the argument relies on the loss gradient for individual expectiles, whereas our networks mix gradients across different expectiles. Therefore the actual observed speed during training can differ.

Simulation details. The simulations in Fig. 12 were done empirically, where we calculated the expectiles via gradient descent. For Fig. 12A, the distributions shown were the equiprobable distribution from Sec. 3, standard Gaussian, exponential with rate 1, and log-normal with shape parameter 1. The asymmetries τ were placed on a uniform 100-point grid between $(0.01, 0.99)$. Fig. 12B was calculated using a single scalar predictor per channel τ on a two-point distribution with rewards $\{-1, 3\}$ and $p(\mathrm{high}) = p$. The scaling $S(\tau)$ was calculated via the ratio of the half-rise time of the dRL to dRL-MSE. The expectiles were calculated using gradient descent on the expectile or the dRL-MSE loss, with step size 0.01 and 20,000 steps.

I.2 Learning speed across all learned expectiles

A learning slowdown across all expectiles also implies a slowdown across the whole network. In the full network, learning speed is determined by the gradient magnitude across all target expectiles. If $\Delta \boldsymbol{e} = \boldsymbol{e}_1 - \boldsymbol{e}_2$ denotes the difference in target expectile vectors for the two-state task, and $\boldsymbol{\Lambda} = \mathrm{diag}(\mathbb{E}[w_{\tau_1}], \ldots, \mathbb{E}[w_{\tau_M}])$ is the diagonal matrix of per-expectile scaling factors, then the dRL loss gradient magnitude is $\|2\boldsymbol{\Lambda}\Delta\boldsymbol{e}\|$. The dRL-MSE loss gradient magnitude lacks the $\boldsymbol{\Lambda}$ pre-factor and so is simply $\|\Delta\boldsymbol{e}\|$. The overall scaling factor is

$$S_{\mathrm{net}} = \frac{\|\Delta \boldsymbol{e}\|}{\|2\boldsymbol{\Lambda}\,\Delta \boldsymbol{e}\|} = \left(\frac{\sum_\tau \Delta e_\tau^2}{\sum_\tau 4\mathbb{E}[w_\tau]^2\,\Delta e_\tau^2} \right)^{1/2}, \tag{42}$$

where we have assumed the same $\boldsymbol{\Lambda}$ across states, and have assumed that the w_τ's are independent across τ's. Taking the correlations across w_τ's into account turns the above expression into an upper bound. For state-dependent $\boldsymbol{\Lambda}$, the expression would depend on $\boldsymbol{\Lambda}_1\boldsymbol{e}_1 - \boldsymbol{\Lambda}_2\boldsymbol{e}_2$ rather than $\boldsymbol{\Lambda}\Delta\boldsymbol{e}$. Depending on the reward distributions and the associated expectile structure, S_{net} can be smaller or larger than one. Indeed, in Fig. 13A we show a scenario where the dRL network learns more slowly than the dRL-MSE network, and in Fig. 13B we show one where the dRL network learns faster.

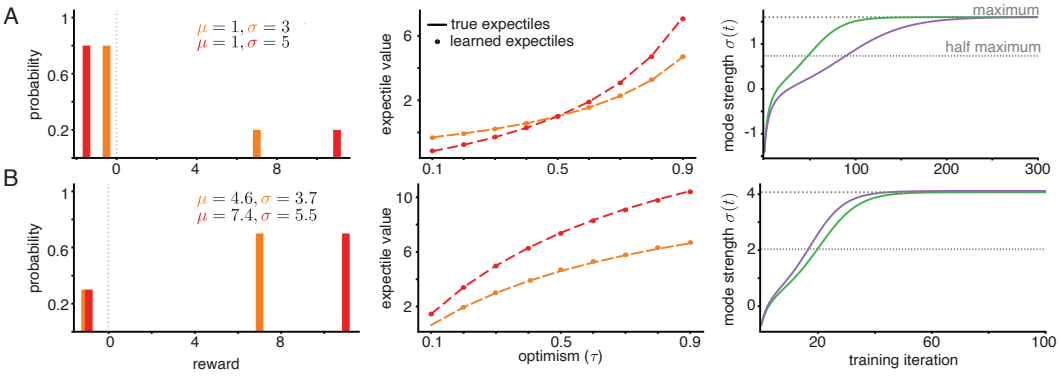

Figure 13: A & B. A two-state task ($z = 1$ in orange, $z = 2$ in red) where the dRL network learns slower (A) and faster (B) than the dRL-MSE. Panels show the two-point reward distributions associated with each state (left), dRL expectiles after learning (center) and mode strength trajectories (right) for one simulation of task in the dRL (purple) and dRL-MSE (green) networks.

Simulation details. The simulations shown above use a small weight initialization ($\sigma = 0.01$); otherwise all other details are kept as standard. For the slowdown simulation (Fig. 13A), both states had $p(\mathrm{high}) = 0.2$ and $\mu_z = 1$; $\sigma_1 = 3$ and $\sigma_2 = 5$. For the speedup simulation (Fig. 13B), state $z = 1$ had rewards at $r_1 = \{-1, 7\}$ and state $z = 2$ at $r_2 = \{-1, 11\}$, both with $p(\mathrm{high}) = 0.7$.

J Hidden representations for two-point reward distributions

As hidden representations share the singular values of the input-output covariance matrix, let us derive the singular value decomposition of this covariance matrix. For this we assume a task with Z states, each of which has a two-point reward distribution with two equally likely rewards of different magnitudes. For each state z, these magnitudes are fully determined by the distribution's mean, $\mathbb{E}[r|z] = \mu_z$, and its variance, $\mathrm{var}(r|z) = \sigma_z^2$, leading to two point masses at $\mu_z + \sigma_z$ and $\mu_z - \sigma_z$. For such reward distributions, the τ-expectile satisfies $\tau\mathbb{E}[(r - e_{z,\tau})_+] = (1-\tau)\mathbb{E}[(e_{z,\tau} - r)_+]$, which results in

$$e_{z,\tau} = \mu_z + (2\tau - 1)\sigma_z, \tag{43}$$

and an associated expectile shape vector $\boldsymbol{\zeta} := 2\boldsymbol{\tau} - \mathbf{1}$, where $\boldsymbol{\tau}$ is the vector of considered asymmetries. As always, we consider a $\boldsymbol{\tau}$ which is symmetric and contains $\tau = 0.5$. As a result, the expectile vector for state z is given by $\boldsymbol{e}_z = \mu_z\mathbf{1} + \sigma_z\boldsymbol{\zeta}$. Using $\boldsymbol{y}_z = \boldsymbol{e}_z$ then results in the input-output covariance matrix,

$$\boldsymbol{\Sigma}_{yx} = \mathbb{E}[\boldsymbol{y}\boldsymbol{x}^\top] = \frac{1}{Z}\sum_z \boldsymbol{e}_z\bar{\boldsymbol{x}}_z^\top = \frac{1}{Z}\sum_z (\mu_z\mathbf{1} + \sigma_z\boldsymbol{\zeta})\,\bar{\boldsymbol{x}}_z^\top = \frac{1}{Z}\mathbf{1}\boldsymbol{\mu}^\top\bar{\boldsymbol{X}} + \frac{1}{Z}\boldsymbol{\zeta}\boldsymbol{\sigma}^\top\bar{\boldsymbol{X}}, \tag{44}$$

where $\bar{\boldsymbol{x}}_z$ is the mean input for state z, $\bar{\boldsymbol{X}} \in \mathbb{R}^{Z\times Z}$ is the matrix that contains these mean inputs as rows, and $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are the vectors containing the reward distribution means and standard deviations across all states, respectively. Because the inputs are centered across states, $\sum_z \bar{\boldsymbol{x}}_z = \mathbf{1}^\top \bar{\boldsymbol{X}} = \mathbf{0}$. This allows us to use $\boldsymbol{\mu}^\top \bar{\boldsymbol{X}} = \Delta\boldsymbol{\mu}^\top \bar{\boldsymbol{X}}$ and $\boldsymbol{\sigma}^\top \bar{\boldsymbol{X}} = \Delta\boldsymbol{\sigma}^\top \bar{\boldsymbol{X}}$, where $\Delta\boldsymbol{\mu}$ and $\Delta\boldsymbol{\sigma}$ are the reward moment deviations from their averages across states, that is $\Delta\boldsymbol{\mu} = \boldsymbol{\mu} - \mathbb{E}[\mu_z]\mathbf{1}$ and $\Delta\boldsymbol{\sigma} = \boldsymbol{\sigma} - \mathbb{E}[\sigma_z]\mathbf{1}$. In other words, the reward components that are shared across states are absorbed by the bias $\boldsymbol{b}_2$. Therefore

$$\boldsymbol{\Sigma}_{yx} = \tfrac{1}{Z}\mathbf{1}\Delta\boldsymbol{\mu}^\top \bar{\boldsymbol{X}} + \tfrac{1}{Z}\boldsymbol{\zeta}\Delta\boldsymbol{\sigma}^\top \bar{\boldsymbol{X}}. \tag{45}$$

This covariance is a sum of two rank-one terms, and because the right and left parts of each term are orthogonal (since, for symmetric expectile grids $\mathbf{1} \perp \boldsymbol{\zeta}$ and for this task $\Delta\boldsymbol{\mu} \perp \Delta\boldsymbol{\sigma}$ by design), the covariance matrix is rank two with the singular value decomposition

$$\boldsymbol{u}_{\text{mean}} = \frac{1}{\sqrt{M}}\mathbf{1}, \qquad \boldsymbol{v}_{\text{mean}} = \frac{\bar{\boldsymbol{X}}^\top \Delta\boldsymbol{\mu}}{\sqrt{Z}\|\Delta\boldsymbol{\mu}\|}, \qquad s_{\text{mean}} = \frac{1}{Z}\sqrt{M}\sqrt{Z}\|\Delta\boldsymbol{\mu}\| = \sqrt{\tfrac{M}{Z}}\|\Delta\boldsymbol{\mu}\|, \tag{46}$$

$$\boldsymbol{u}_{\text{spread}} = \frac{\boldsymbol{\zeta}}{\|\boldsymbol{\zeta}\|}, \qquad \boldsymbol{v}_{\text{spread}} = \frac{\bar{\boldsymbol{X}}^\top \Delta\boldsymbol{\sigma}}{\sqrt{Z}\|\Delta\boldsymbol{\sigma}\|}, \qquad s_{\text{spread}} = \frac{1}{Z}\|\boldsymbol{\zeta}\|\sqrt{Z}\|\Delta\boldsymbol{\sigma}\| = \frac{\|\boldsymbol{\zeta}\|\|\Delta\boldsymbol{\sigma}\|}{\sqrt{Z}}. \tag{47}$$

J.1 Hidden representations in five-state tasks

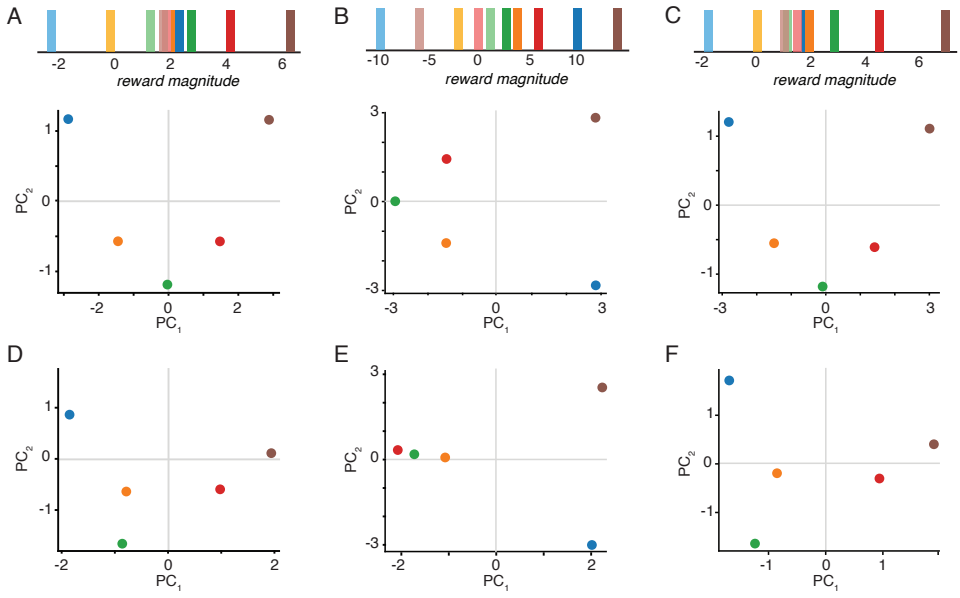

Figure 14: A.-C. Reward distributions (top) and state centroids in the top two hidden PCs (bottom) for the trained dRL network for the mean dominates task (A), spread dominates task (B) and mixed moments task (C) with exact values in Table 1 (linear) and Table 2 (nonlinear). D.-F. are the same as the first row but for networks with a ReLU nonlinearity.

The hidden representations for all five-state tasks were obtained in simulations that share the same setup. Each of the $Z = 5$ states yields one of two equally likely rewards, $\mu_z \pm \sigma_z$, so the expectiles are exactly $e_{z,\tau} = \mu_z + (2\tau - 1)\sigma_z$ and the centered target matrix is $\mathbf{1}\Delta\boldsymbol{\mu}^\top + \boldsymbol{\zeta}\Delta\boldsymbol{\sigma}^\top$. Inputs are one-hot state centroids with additive isotropic Gaussian noise ($\sigma_x^2 = 0.01$), whitened. Training and initialization parameters are kept standard, except the number of training iterations, which was raised to 2000, and the random weights initialization scale, which was lowered to $\sigma = 0.01$ to avoid any re-alignment transients. Hidden PCs are computed from the class-mean hidden activity at the end of training, and we plot results for the dRL network.

Table 1 lists the reward distributions, theoretical and measured singular values, and PC alignments for three reward distribution regimes. Reward means are $\mu_z = 0, 1, 2, 3, 4$ throughout; only their standard deviations change across regimes. In the mean-dominates and spread-dominates regimes, $\Delta\boldsymbol{\sigma}$ was chosen orthogonal to $\Delta\boldsymbol{\mu}$. In the spread-dominates regime, the deviations $\Delta\sigma$ are scaled

by a factor of 6, so $s_{\text{spread}} > s_{\text{mean}}$ and the roles of the two PCs swap (Fig. 14A vs. Fig. 14B). In the mixed regime, we changed $\Delta\boldsymbol{\sigma}$ by adding $0.3\Delta\boldsymbol{\mu}$, such that $\text{corr}(\Delta\boldsymbol{\mu}, \Delta\boldsymbol{\sigma}) = 0.56$. The two rank-one terms are then no longer orthogonal, so the singular vectors of the targets mix them: PC_1 is still the mean axis but also correlates with $\Delta\boldsymbol{\sigma}$ ($\text{corr} = 0.59$), and PC_2 partially correlates with the spread ($\text{corr} = 0.81$) (Fig. 14C).

	Mean dominates	Spread dominates	Mixed moments
State 1: (μ_1, σ_1)	(0, 2.25)	(0, 9.75)	(0, 1.75)
State 2: (μ_2, σ_2)	(1, 1.125)	(1, 3.00)	(1, 0.925)
State 3: (μ_3, σ_3)	(2, 0.75)	(2, 0.75)	(2, 0.85)
State 4: (μ_4, σ_4)	(3, 1.125)	(3, 3.00)	(3, 1.525)
State 5: (μ_5, σ_5)	(4, 2.25)	(4, 9.75)	(4, 2.95)
$s_{\text{mean}},\ s_{\text{spread}}$ (theory)	9.49, 2.17	9.49, 13.04	9.49, 2.62
$s_1,\ s_2$ (SVD of targets)	9.49, 2.17	13.04, 9.49	9.61, 2.15
$\text{corr}(\boldsymbol{\mu}, \boldsymbol{\sigma})$	0.00	0.00	0.56
$\text{corr}(\text{PC}_1, \boldsymbol{\mu})$, $\text{corr}(\text{PC}_1, \boldsymbol{\sigma})$	1.00, 0.00	0.00, 1.00	1.00, 0.59
$\text{corr}(\text{PC}_2, \boldsymbol{\mu})$, $\text{corr}(\text{PC}_2, \boldsymbol{\sigma})$	0.00, 1.00	1.00, 0.00	−0.04, 0.81

Table 1: Reward distributions (two-point, $\mu \pm \sigma$), predicted and measured mode strengths s_{mean} and s_{spread} and hidden layer PCA alignment for the three reward distribution regimes. PCs are from the hidden layer of the dRL network at the end of training.

	Mean dominates	Spread dominates	Mixed moments
$\text{corr}(\text{PC}_1, \boldsymbol{\mu})$, $\text{corr}(\text{PC}_1, \boldsymbol{\sigma})$	0.96, 0.22	0.05, 0.96	0.92, 0.78
$\text{corr}(\text{PC}_2, \boldsymbol{\mu})$, $\text{corr}(\text{PC}_2, \boldsymbol{\sigma})$	−0.25, 0.92	0.91, −0.12	−0.36, 0.52

Table 2: Hidden layer PCA alignment for the three reward distribution regimes for the nonlinear (ReLU) network. PCs are from the hidden layer of the dRL network at the end of training. Compare to Table 1 for the linear network.

J.2 Hidden representations and learning in a three moment task

Next we tested a slightly richer task paradigm, where we varied three moments (mean, variance, and skew) of the reward distributions. The input to the network came from one of $Z = 4$ states. Inputs were one-hot vectors with additive isotropic Gaussian noise ($\sigma_x^2 = 0.01$), whitened. Training and initialization parameters are kept standard, except the number of training iterations, which was raised to 2000, and the random weights initialization scale, which was lowered to $\sigma = 0.01$ to avoid any re-alignment transients.

We divided the task into two phases: a first phase where the states had reward distributions with unique means, variances and skews, and a second phase where the states all had reward distributions with the same mean, while keeping their variances and skews the same as in the first phase. In both cases, rewards were drawn from skewed two-point distributions.

The reward distributions in the first task phase were chosen such that the mean, variance and skew were uncorrelated across all states (Fig. 15A, Table 3). This resulted in a $\boldsymbol{\Sigma}_{yx}^{(1)}$ of rank 3, whose three modes correspond to the mean, variance and skew of the reward distributions. After learning, the network organized the four states in a three-dimensional subspace whose principal axes align with these three modes, with the state coordinates along the top three hidden PCs matching the moment vectors ($\text{corr}(\text{PC}_1^{(1)}, \Delta\boldsymbol{\mu}) = 1$, $\text{corr}(\text{PC}_2^{(1)}, \Delta\boldsymbol{\sigma}) = 1$, $\text{corr}(\text{PC}_3^{(1)}, \Delta\boldsymbol{\kappa}) = 1$; Fig. 15B). The chosen reward distributions varied the most in mean, then variance, then skew, such that $s_{\text{mean}} > s_{\text{variance}} > s_{\text{skew}}$ and the moments were learned in this order (Fig. 15C).

Simulation details. The network is initialized with small weights ($\sigma = 0.01$), has an input dimension of 4 ($Z = 4$), and is run for 2000 epochs. Otherwise all other simulation parameters are kept standard.

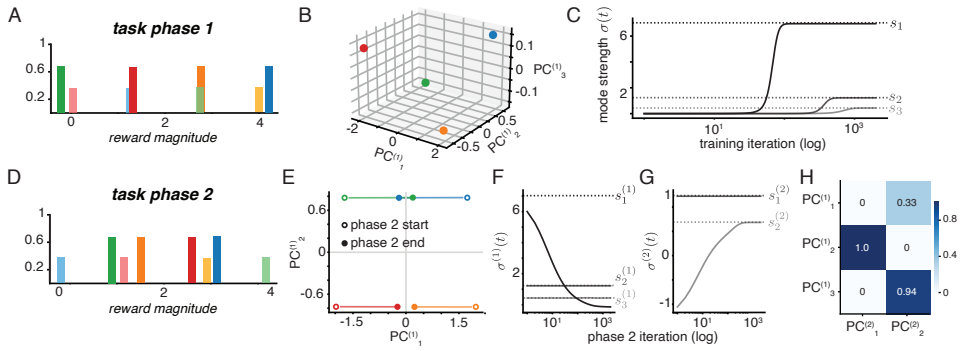

Figure 15: A. Reward distributions for the first phase of the three moments task, with each state's distribution shown in a different color (see Table 3 for reward distribution moments). B. Hidden representation of the four states after learning the first task phase. The PCs align with the three modes of $\mathbf{\Sigma}^{(1)}_{yx}$ (mean, spread, skew). C. Mode strength growth over learning in phase one. D. Reward distributions for the second task phase, where the moments are kept the same as phase one but the distributions now have a common $\mu_z = 2$. E. Projection of the state centroids onto the phase one subspace before phase two training (empty dots) and after phase 2 training (filled dots). F. Phase one mode strengths ($\sigma^1_i(t) = (\boldsymbol{u}^{(1)}_i)^\top W_2 W_1(t)(\boldsymbol{v}^{(1)}_i)$) during training of the second task phase. G. Same as F, but for phase two mode strengths ($\sigma^2_i(t) = (\boldsymbol{u}^{(2)}_i)^\top W_2 W_1(t)(\boldsymbol{v}^{(2)}_i)$). H. Cosine similarities of the phase one and phase two PCs. All results shown here are for a single run of the dRL network.

State z	μ_z	σ_z	p_z
1	3.2	1.4	0.65
2	3.2	0.6	0.35
3	0.8	1.4	0.35
4	0.8	0.6	0.65

Table 3: Per-state reward distribution moments in phase one of the three moments task.

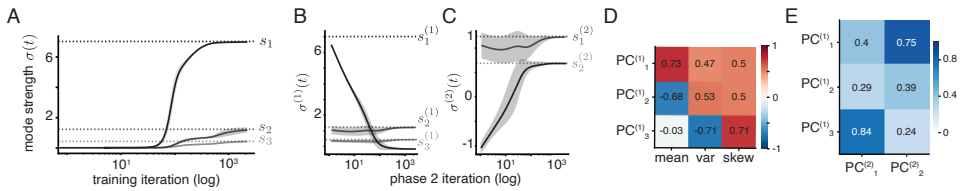

Figure 16: Results from the same task as Fig. 15, but for networks with a ReLU nonlinearity. A. Mode strength growth over learning in phase one. B. Phase one mode strengths during training of the second task phase. C. Same as B, but for phase two mode strengths. D. Cosine similarities of the phase one PCs and the mean, spread, and skew vectors. E. Cosine similarities of the phase one and phase two PCs. Results shown here are for 20 runs of the nonlinear dRL network (mean ± SD).

In the second task phase we switched the reward distributions to share the same mean ($\mu_z = 2$) but kept the variances and skews the same. Inputs were likewise unchanged, and we continued training the network that was pretrained on the first task phase. The new $\mathbf{\Sigma}_{yx}^{(2)}$ is rank 2, with its first mode $(\boldsymbol{u}_1^{(2)}, \boldsymbol{v}_1^{(2)})$ corresponding to variance and its second mode $(\boldsymbol{u}_2^{(2)}, \boldsymbol{v}_2^{(2)})$ to skew. After training on phase 2, the hidden representation of the four states collapsed along $\text{PC}_1^{(1)}$ (the mean axis) (Fig. 15E). Indeed, the phase one mean mode decayed over training on phase two, whereas the phase one variance and skew modes were unchanged (Fig. 15F). The phase two spread mode was identical to the phase one spread mode ($\text{corr}(\boldsymbol{u}_2^{(1)}, \boldsymbol{u}_1^{(2)}) = 1$), whereas the new skew mode overlapped partially with the old mean mode ($\text{corr}(\boldsymbol{u}_1^{(1)}, \boldsymbol{u}_2^{(2)}) = 0.73$) and the old skew mode ($\text{corr}(\boldsymbol{u}_3^{(1)}, \boldsymbol{u}_2^{(2)}) = 0.68$). This was reflected in the growth of the new modes over phase two training: the new skew mode starts below zero (reflecting the component along $\boldsymbol{u}_1^{(1)}$), and grew to its final value. This suggests that the network was able to selectively preserve task-relevant dimensions. Calculating the cosine similarities of the PCs across phases confirms this picture: the variance and skew coding PCs are conserved across tasks (Fig. 15H). Altogether, we find that when a feature is removed from a task, the distributional network unlearns only that feature. The directions that encode the remaining relevant features are reused in place.

J.3 Validation in nonlinear networks

We repeated the same five-state and three-moment simulations described above with a ReLU hidden layer, keeping all other parameters unchanged.

Five state tasks. The ReLU dRL network keeps the PC alignment with the moments of the distribution: when the mean dominates, PC_1 aligns with $\Delta\boldsymbol{\mu}$ and PC_2 with $\Delta\boldsymbol{\sigma}$. When the spread dominates, the two PCs swap (Table 2). However, this alignment is slightly weaker than in the linear network, with small cross-correlations to the other moments. In the mixed-moments regime, the ReLU network mixes the moments more strongly than the linear one: for example, PC_1 correlates more with $\Delta\boldsymbol{\sigma}$ (0.78 vs. 0.59). The relative positioning of the states in the hidden layer also changes: while states are ordered roughly according to their moments, the ordering is less clear than for the linear network (Fig. 14).

Three-moment task. The ReLU network learns the three phase one modes one after another, in the same order as the linear network (Fig. 15 vs. Fig. 16A). When the means become equal in phase two, the phase one mean mode decays while the spread and skew modes persist (Fig. 16B), and the phase-two modes grow to their asymptotic values (Fig. 16C), as in the linear network. However, individual phase one PCs no longer align with single moments (Fig. 16D). Their strongest correlations are still the same as those in the linear network, but the PCs now mix moments. As a result, phase one and phase two PCs correspond less cleanly than in the linear network (Fig. 16E vs. Fig. 15H).

Taken together, the five state and three moment tasks show that nonlinearity preserves which moments are represented, the order in which they are learned and which are unlearned, but it does not necessarily preserve the one-to-one mapping between moments and principal axes.

K THE DYNAMICS OF TRANSFER LEARNING

To study the dynamics of transfer learning, we assume a task with two phases. The first phase matches the two-state task described in Fig. 2, and the second phase changes the reward distribution associated with state $z = 1$ to yield a reward of 3, while keeping the reward for state $z = 2$ fixed at 1 (Fig. 4A).

Our aim is to analyze how the "pretraining" in the first task phase affects learning in phase two, for both cRL and dRL networks. For both, learning dynamics are controlled by the phase- and network-specific input-output covariance matrix $\mathbf{\Sigma}_{yx}$. To determine this covariance matrix, we note that $\Delta\mu_1 = 0$ in the first task phase, and $\Delta\mu_2 = 2$ in the second task phase. Furthermore, the reward distribution expectile differences are $\Delta e_{1,\tau} = (4\tau - 1) - 1 = 4\tau - 2$ in the first task phase, and change to $\Delta e_{2,\tau} = 3 - 1 = 2$ in the second task phase. From App. F we know that the covariance matrices have singular values

$$s_1^{\mathrm{cRL}} = 0, \qquad s_2^{\mathrm{cRL}} = \|\bar{\boldsymbol{x}}_1\| \tag{48}$$

$$s_1^{\mathrm{dRL}} = \frac{1}{2}\|\Delta \boldsymbol{e}_1\|\|\bar{\boldsymbol{x}}_1\|, \qquad s_2^{dRL} = \frac{1}{2}\|\Delta \boldsymbol{e}_2\|\|\bar{\boldsymbol{x}}_1\|, \tag{49}$$

and singular vectors

$$\boldsymbol{v}_1^{\mathrm{cRL}} = \boldsymbol{v}_2^{\mathrm{cRL}} \propto \bar{\boldsymbol{x}}_1, \qquad u_1^{\mathrm{cRL}} \propto 0, \qquad u_2^{\mathrm{cRL}} \propto 1, \tag{50}$$

$$\boldsymbol{v}_1^{\mathrm{dRL}} = \boldsymbol{v}_2^{\mathrm{dRL}} \propto \bar{\boldsymbol{x}}_1, \qquad \boldsymbol{u}_1^{\mathrm{dRL}} \propto \Delta \boldsymbol{e}_1, \qquad \boldsymbol{u}_2^{\mathrm{dRL}} \propto \Delta \boldsymbol{e}_2. \tag{51}$$

Thus, both networks share the same "input" singular vectors $\boldsymbol{v}_j$, but singular values s_j and "output" singular vectors $\boldsymbol{u}_j$ differ across both networks and task phases. For the cRL network the singular value is zero for the first phase, and only non-zero for the second. For the dRL network, the singular values are non-zero for both phases, and, for a grid of asymmetries that is symmetric around $\tau = \frac{1}{2}$, the expectile differences and "output" singular vectors are orthogonal, that is $\Delta \boldsymbol{e}_1 \perp \Delta \boldsymbol{e}_2$ and $\boldsymbol{u}_1^{\mathrm{dRL}} \perp \boldsymbol{u}_2^{\mathrm{dRL}}$.

K.1 TRANSFER LEARNING DYNAMICS IN THE CRL NETWORK

In general, the mode strength σ associated with singular value s obeys $\gamma\dot{\sigma} = 2\sigma(s - \sigma)$, where γ is the inverse learning rate (Saxe et al., 2014, Eq. (10)). For first phase of the cRL network we have $s_1^{\mathrm{cRL}} = 0$, such that the mode dynamics simplifies to $\gamma\dot{\sigma}_1^{\mathrm{cRL}} = -2{\sigma_1^{\mathrm{cRL}}}^2$. This gives

$$\sigma_1^{\mathrm{cRL}}(t) = \frac{\sigma_0}{1 + \frac{2\sigma_0 t}{\gamma}}, \tag{52}$$

where $\sigma_0 = \sigma_1^{\mathrm{cRL}}(0)$ is the initial mode strength. This implies that the mode decays towards the fixed point $\sigma_1^{\mathrm{cRL}}(t) \to 0$ with $t \to \infty$. Thus, if T_1 is the training time of phase one, the mode magnitude at the end of phase one will be

$$\sigma_1^{\mathrm{cRL}}(T_1) = \frac{\sigma_0}{1 + \frac{2\sigma_0 T_1}{\gamma}}, \tag{53}$$

which, for sufficiently large T_1, will be very small. Therefore, the cRL network enters the second phase with a very small mode strength ($\sigma_1^{\mathrm{cRL}}(T_1) \approx 0$). For small initial weights, the mode in phase two can be shown to have a half-rise time of approximately (Saxe et al., 2014)

$$t_{1/2}^{\mathrm{cRL}} = \frac{\gamma}{2s_2^{\mathrm{cRL}}} \ln \frac{s_2^{\mathrm{cRL}}}{\sigma_0}, \tag{54}$$

where σ_0 here is the mode strength when entering the second task phase. Substituting $\sigma_1^{\mathrm{cRL}}(T_1)$ for σ_0 and re-expressing it through Eq. (53), we find

$$t_{1/2}^{\mathrm{cRL}} = \frac{\gamma}{2s_2^{\mathrm{cRL}}} \ln \frac{s_2^{\mathrm{cRL}}}{\sigma_1^{\mathrm{cRL}}(T_1)} = \frac{\gamma}{2s_2^{\mathrm{cRL}}} \ln \frac{s_2^{\mathrm{cRL}}}{\sigma_0} + \frac{\gamma}{2s_2^{\mathrm{cRL}}} \ln\left(1 + \frac{2\sigma_0 T_1}{\gamma}\right), \tag{55}$$

where σ_0 now denotes the initial mode strength before any learning has occurred. The first term in this expression is the standard half-rise time from any small random initialization. The second term

is the term that quantifies the negative impact of pretraining the network in the first phase. Because pretraining on the $s_1^{\mathrm{cRL}} = 0$ phase shrinks network's mode strength, this negative impact grows as phase one training time $T_1 \to \infty$. For infinitely long phase one training times, $\sigma_1^{\mathrm{cRL}} \to 0$, and since $\sigma_2^{\mathrm{cRL}} = 0$ is a trivial (but unstable) fixed point of the dynamics $\gamma \dot{\sigma}_2^{\mathrm{cRL}} = 2\sigma_2^{\mathrm{cRL}} \left(s_2^{\mathrm{cRL}} - \sigma_2^{\mathrm{cRL}}\right)$ in the second phase, the half-rise time in phase two would be infinite. In other words, a network that learns phase one perfectly would be unable to learn phase two.

K.2 Transfer learning dynamics in the dRL network

To model the transfer learning dynamics in the dRL network, we define $t = 0$ as the time at which the task transitions from phase one to phase two, that is, at which training of phase one has ended. For every time t throughout phase two training, we can write

$$\boldsymbol{W}_2 \boldsymbol{W}_1(t) = \sigma(t)\, \boldsymbol{u}(t)\, \boldsymbol{v}^\top, \tag{56}$$

where $\boldsymbol{u}(t)$ is the unit "output" singular vector, $\sigma(t)$ is the mode strength, and $\boldsymbol{v}$ is the constant input singular vector. Even though all of these quantities are specific to the dRL network, we will omit the $\cdot^{\mathrm{dRL}}$ superscript going forward for clarity.

Our aim is to compute the $(\sigma(t), \boldsymbol{u}(t))$ trajectory throughout phase two training. To do so, note that the network has approximately reached $(\sigma(0), \boldsymbol{u}(0)) = (s_1, \boldsymbol{u}_1)$ at the end of phase one, and is expected to reach $(\sigma(\infty), \boldsymbol{u}(\infty)) = (s_2, \boldsymbol{u}_2)$ at the end of phase two. In terms of the network weights, we know that $\boldsymbol{W}_1 = \sqrt{s_j}\boldsymbol{v}_j^\top$ after phase one ($j = 1$) and phase two ($j = 2$). Since both phases share the same $\boldsymbol{v}$ and only differ in $s_1 \neq s_2$, the input weights will undergo rescaling with no change in direction during phase two learning ($\cos(\boldsymbol{W}_1(0), \boldsymbol{W}_1(t)) = 1$). Over learning, therefore,

$$\boldsymbol{W}_1(t) = \sqrt{\sigma(t)}\boldsymbol{v}^\top. \tag{57}$$

For the readout weights, the network starts with direction $\boldsymbol{u}_1$ and ends with direction $\boldsymbol{u}_2$, where we have previously observed that $\boldsymbol{u}_1 \perp \boldsymbol{u}_2$. Therefore $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ form the orthogonal basis of a two-dimensional plane. The gradient of $\boldsymbol{W}_2$ is a linear combination of $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ at every time point so that $\boldsymbol{W}_2(t)$ never leaves that plane. Therefore we can define

$$\boldsymbol{W}_2(t) = \alpha(t)\boldsymbol{u}_1 + \beta(t)\boldsymbol{u}_2, \tag{58}$$

where $\alpha(t)$ and $\beta(t)$ define the strengths of the old and new output directions, respectively, or a set of coordinates on the $(\boldsymbol{u}_1, \boldsymbol{u}_2)$ plane. At $t = 0$, $(\alpha(0), \beta(0)) = (\sqrt{s_1}, 0)$ and at the end of phase two, $(\alpha(\infty), \beta(\infty)) = (0, \sqrt{s_2})$.

Since the weights are balanced at initialization, they remain balanced throughout, enforcing $\|\boldsymbol{W}_2(t)\|^2 = \|\boldsymbol{W}_1(t)\|^2 = \sigma(t)$ and the norm of the weight can be written as

$$\sigma(t) = \|\boldsymbol{W}_2(t)\|^2 = \alpha(t)^2 + \beta(t)^2 + 2\alpha(t)\beta(t)\, \boldsymbol{u}_1^\top \boldsymbol{u}_2 = \alpha(t)^2 + \beta(t)^2 \tag{59}$$

where we have used the fact that $\boldsymbol{u}_2 \perp \boldsymbol{u}_1$ to eliminate the cross term. For brevity we define $r(t) = \sqrt{\sigma(t)} = \|\boldsymbol{W}_1(t)\| = \|\boldsymbol{W}_2(t)\|$, so that $\boldsymbol{W}_1(t) = r(t)\boldsymbol{v}^\top$ and $r(t)^2 = \alpha(t)^2 + \beta(t)^2$.

We can now re-write Eq. (56) in terms of these coordinates, resulting in

$$\boldsymbol{W}_2 \boldsymbol{W}_1(t) = r(t)(\alpha(t)\boldsymbol{u}_1 + \beta(t)\boldsymbol{u}_2)\boldsymbol{v}^\top \tag{60}$$

and plug it into the gradient equations for the second phase,

$$\gamma \dot{\boldsymbol{W}}_2(t) = \left(\boldsymbol{\Sigma}_{yx}^{(2)} - \boldsymbol{W}_2 \boldsymbol{W}_1(t)\right) \boldsymbol{W}_1(t)^\top \tag{61}$$

$$\gamma \dot{\boldsymbol{W}}_1(t) = \boldsymbol{W}_2(t)^\top \left(\boldsymbol{\Sigma}_{yx}^{(2)} - \boldsymbol{W}_2 \boldsymbol{W}_1(t)\right). \tag{62}$$

where, for simplicity, we assume $\gamma = 1$ from here on. Using the coordinate re-expression for $\boldsymbol{W}_2 \boldsymbol{W}_1(t)$ causes the shared error term in these gradient equations to become

$$\boldsymbol{\Sigma}_{yx}^{(2)} - \boldsymbol{W}_2 \boldsymbol{W}_1(t) = ((s_2 - r(t)\beta(t))\boldsymbol{u}_2 - r(t)\alpha(t)\boldsymbol{u}_1)\, \boldsymbol{v}^\top. \tag{63}$$

Multiplying the error by $\boldsymbol{W}_1(t)^\top$ and $\boldsymbol{W}_2(t)^\top$, the gradient equations can be written as

$$\dot{\boldsymbol{W}}_1(t) = (\alpha(t)\boldsymbol{u}_1 + \beta(t)\boldsymbol{u}_2)^\top ((s_2 - r(t)\beta(t))\boldsymbol{u}_2 - r(t)\alpha(t)\boldsymbol{u}_1)\, \boldsymbol{v}^\top \tag{64}$$

$$\dot{\boldsymbol{W}}_2(t) = ((s_2 - r(t)\beta(t))\boldsymbol{u}_2 - r(t)\alpha(t)\boldsymbol{u}_1)\, \boldsymbol{v}^\top r(t)\boldsymbol{v} \tag{65}$$

Setting this equal to

$$\dot{\boldsymbol{W}}_1(t) = \dot{r}(t)\boldsymbol{v}^\top \qquad \dot{\boldsymbol{W}}_2(t) = \dot{\alpha}(t)\boldsymbol{u}_1 + \dot{\beta}(t)\boldsymbol{u}_2, \tag{66}$$

and separating them into orthogonal components, we find the dynamics of α, β, and r to be given by

$$\dot{\alpha}(t) = -r^2(t)\alpha(t) \tag{67}$$

$$\dot{\beta}(t) = s_2 r(t) - r^2(t)\beta(t) \tag{68}$$

$$\dot{r}(t) = -r(t)(\alpha^2(t) + \beta^2(t)) + \beta(t)s_2 \tag{69}$$

These three equations describe how the mode grows (through $\dot{r}(t)$) and how the output vector rotates and grows (through $\dot{\alpha}(t), \dot{\beta}(t)$).

To separate the rotation and growth phases of learning, we re-map these expressions into polar coordinates by defining

$$\alpha(t) = r(t)\cos(\theta(t)), \qquad \beta(t) = r(t)\sin(\theta(t)), \qquad \sigma(t) = r(t)^2, \tag{70}$$

where the last one follows from $\sigma(t) = \alpha(t)^2 + \beta(t)^2$. As r and θ are time-dependent quantities, we can write

$$\dot{\alpha}(t) = \dot{r}\cos(\theta) - r\sin(\theta)\dot{\theta} \qquad \dot{\beta}(t) = \dot{r}\sin(\theta) + r\cos(\theta)\dot{\theta} \qquad \dot{\sigma}(t) = 2r\dot{r} \tag{71}$$

(omitting dependence of θ, r on t for clarity). Equating with Eq. (67), we find

$$\dot{\theta}(t) = s_2\cos\theta(t) \qquad \dot{r}(t) = r(t)(s_2\sin\theta(t) - r(t)^2). \tag{72}$$

Finally, solving for θ gives us

$$\cos\theta(t) = \mathrm{sech}(s_2 t), \qquad \sin\theta(t) = \tanh(s_2 t) \tag{73}$$

assuming $\theta(0) = 0$. Equation (56) can then be rewritten in terms of this rotation and scaling,

$$\boldsymbol{W}_2\boldsymbol{W}_1(t) = \sigma(t)\,\boldsymbol{u}(t)\,\boldsymbol{v}^\top, \qquad \boldsymbol{u}(t) = \cos\theta(t)\,\boldsymbol{u}_1 + \sin\theta(t)\,\boldsymbol{u}_2, \qquad \sigma(t) = r(t)^2 \tag{74}$$

From this equation we can see that, throughout transfer learning, the network stays rank one and its output singular vector rotates continuously from $\boldsymbol{u}_1$ to $\boldsymbol{u}_2$ with a magnitude proportional to the mode strength. Substituting $\sin\theta(t) = \tanh(s_2 t)$ into $\dot{r}(t) = r(s_2\sin\theta(t) - r(t)^2)$ gives the full equation for the mode dynamics in terms of s_1, s_2 and t,

$$r(t)^2 = \frac{\cosh^2(s_2 t)}{\frac{1}{s_1} + t + \frac{\sinh(s_2 t)\cosh(s_2 t)}{s_2}}, \tag{75}$$

which satisfies the endpoint conditions, $r(0)^2 = s_1$ and $r(\infty)^2 = s_2$. From these equations, we can show that $r(t)$, and therefore the mode strength, undergoes an initial shrinking phase before growing to the final value, as $\dot{r}(0) < 0$. We also know that $\cos(\boldsymbol{W}_1(t), \boldsymbol{W}_1(0)) = 1$, $\cos(\boldsymbol{W}_2(t), \boldsymbol{W}_2(0)) = \mathrm{sech}(s_2 t)$ and $\|\boldsymbol{W}_2(t)\|^2 = \|\boldsymbol{W}_1(t)\|^2 = \sigma(t)$ throughout learning, such that the hidden representation is only re-scaled but otherwise remains unchanged, while the network output rotates to read this representation into the new, task-relevant space.

To find an expression for the half-rise time in phase two, we observe that the mode strength in this phase can be calculated as $\sigma_2(t) = \boldsymbol{u}_2^\top \boldsymbol{W}_2\boldsymbol{W}_1(t)\boldsymbol{v}$. Substituting this into Eq. (74) results in

$$\sigma_2(t) = r(t)^2\sin(\theta(t)) \tag{76}$$

Substituting $\sin\theta(t) = \tanh(s_2 t)$, we can rewrite the phase two mode strength dynamics in terms of s_1, s_2 and t,

$$\sigma_2(t) = r(t)^2\tanh(s_2 t). \tag{77}$$

This allows us to find the half-rise time by setting $\sigma_2(t_{1/2}) = s_2/2$, resulting in the transcendental condition

$$\tfrac{1}{2}\sinh(2s_2 t_{1/2}) \;=\; s_2\, t_{1/2} + \frac{s_2}{s_1}, \tag{78}$$

which must be solved numerically for a given s_1, s_2. Unlike the half-rise time for the cRL network, the dRL half-rise time is independent of pretraining length T_1, and depends on the phase one task structure only through the s_2/s_1 term.

Simulation details. The networks are initialized to small initial weights ($\sigma = 0.01$) and trained for 500 epochs per phase (1000 total for pretrained nets, 500 for naive nets). For each simulation, the pretrained networks are trained first on the phase one task then directly on the phase two task. The naive networks start from identical initial weights and are trained on phase two only.

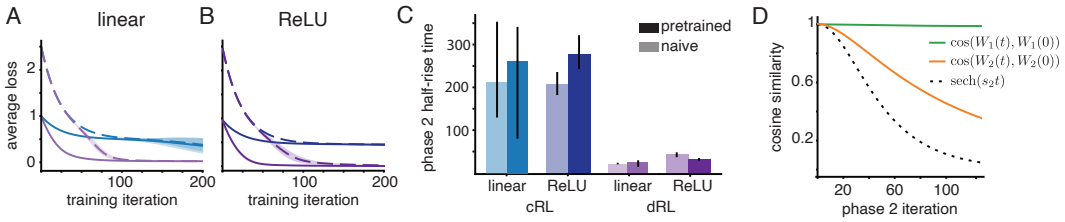

Figure 17: A. & B. Average loss during phase two for pretrained (solid) and naive (dashed) cRL (blue) and dRL (purple) networks with linear (A) and ReLU (B) hidden layers (mean $\pm$ SD across 20 simulations). C. Phase-two half-rise time for pretrained (dark) and naive (light) networks (mean $\pm$ SD). D. Cosine similarity between $\boldsymbol{W}_1(t)$ and $\boldsymbol{W}_1(0)$ (green) and between $\boldsymbol{W}_2(t)$ and $\boldsymbol{W}_2(0)$ (orange) during phase two for the ReLU dRL network. Black dotted line is the linear-theory prediction $\mathrm{sech}(s_2 t)$.

K.3 Validation in nonlinear networks

We repeated the transfer experiment of Sec. 5 with a ReLU hidden layer. The ReLU networks learn more slowly than their linear counterparts overall, but reproduce the qualitative pattern of the linear networks (Fig. 17A,B). Pretraining on phase one speeds up phase two learning in the nonlinear dRL network as well, and in both the dRL and cRL networks the nonlinear variant learns more slowly. The pretrained ReLU dRL network also reuses its hidden representation: the input weights $\boldsymbol{W}_1$ keep their phase-one direction throughout phase two, while the readout $\boldsymbol{W}_2$ rotates away from it (Fig. 17D). The rotation is slower than the linear prediction and the linear network's dynamics (Fig. 4) but has the same qualitative form, so we conclude that the transfer mechanism of App. K.2 extends beyond the linear setting.