Representation Dynamics and Feature Collapse

Clare Lyle
Representation Learning in RL Workshop
OxCSML

Credits

This talk is based on two papers

On the Effect of Auxiliary Tasks on Representation Dynamics

AISTATS 2021. CL*, Mark Rowland*, Georg Ostrovski, Will Dabney.

[Redacted]

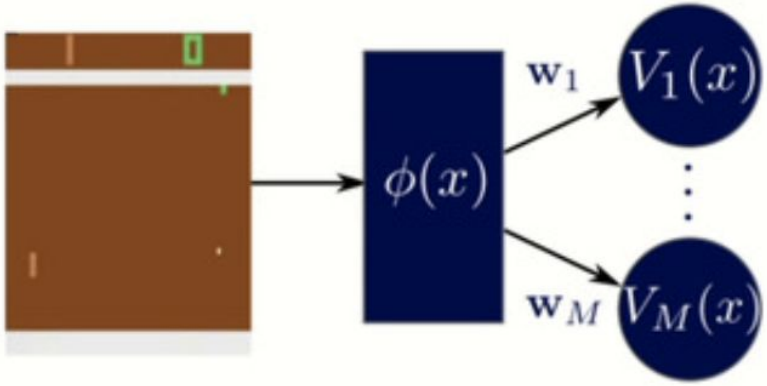
Under review. CL, Mark Rowland, Will Dabney.

Two Views on Representations

Feature space view

Representation = outputs of a layer of a neural network

- Usually the penultimate layer in RL
- Useful for contexts where the output of the “representation layer” will be used as input to another network later.



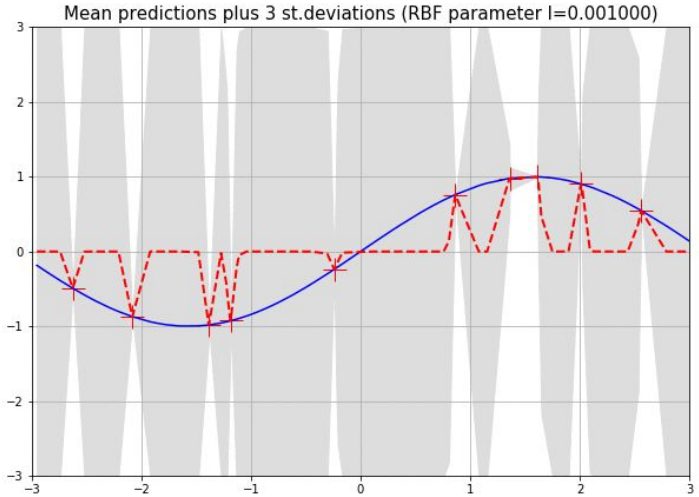
Generic formulation of auxiliary tasks: shared representation ϕ used to make many predictions.

Two Views on Representations

Update space view

Representation = how a network will respond to the data it sees in the future

- How does an update on one data point influence predictions on other data points?
- What types of target/input distributions is the network good at fitting?
- Useful for meta-learning or transfer learning, where network may be fine-tuned on a new task later.



Part I: Learning Dynamics in RL

Learning Dynamics

We're interested in studying how RL agents' **representations evolve over time**.

To do this, we study a continuous-time approximation of the *dynamics* followed by TD learning.

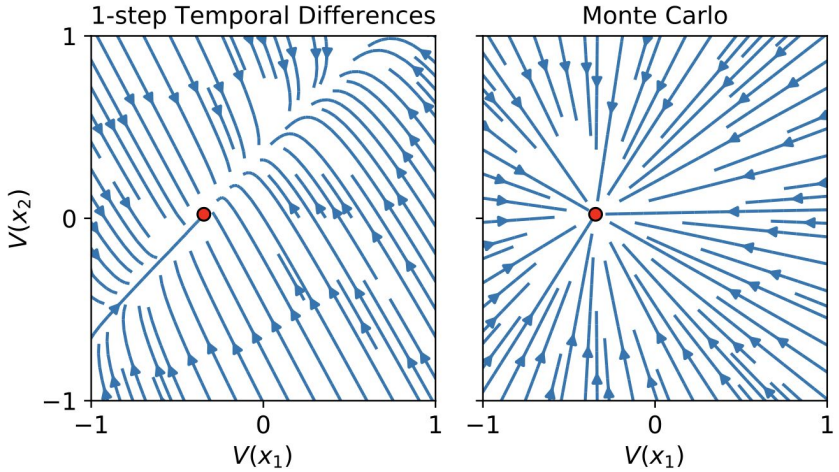
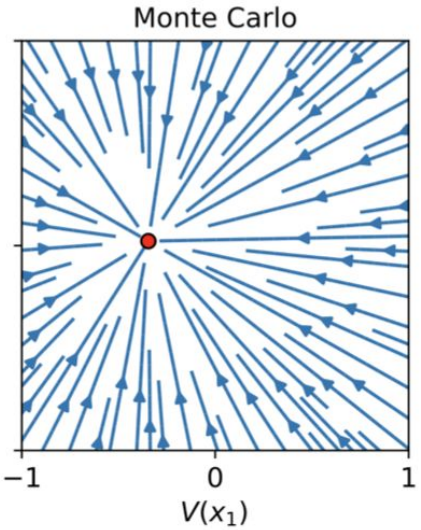


Figure 1: An example of qualitatively different value function dynamics for a two-state MDP for 1-step temporal difference learning and Monte Carlo learning, with fixed point V^π in red.

Learning Dynamics

Warmup: fitting a known tabular value function by minimizing the l2 distance to the true value function.

$$\partial_t V_t = (I - \gamma P^\pi)^{-1} R^\pi - V_t$$

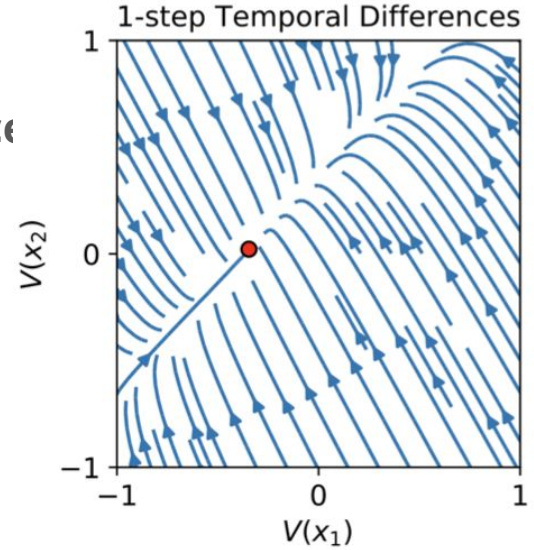


$$V_t = e^{-t}(V_0 - (I - \gamma P^\pi)^{-1} R^\pi) + (I - \gamma P^\pi)^{-1} R^\pi$$

Learning Dynamics

Warmup: fitting a value function using **temporal difference learning** (minimize l2 distance to TD target)

$$T^\pi V = R^\pi + \gamma P^\pi V$$



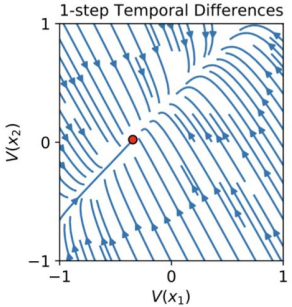
$$\partial_t V_t = -(I - \gamma P^\pi) V_t + R^\pi$$

$$V_t = \exp(-t(I - \gamma P^\pi))(V_0 - V^\pi) + V^\pi$$

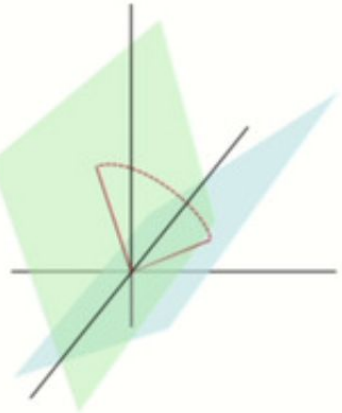
Convergence along eigenspaces of transition matrix

Proposition 3.4. Under Assumption 3.3, and $(V_t)_{t\geq 0}$ the solution to Equation (3), for almost every[1] initial condition V_0, we have

$$d(\langle V_t - V^\pi \rangle, \langle U_1 \rangle) \to 0\,.$$



To obtain our results, we quantify distances between subspaces, such as the **Grassmann distance.**



Proposition 3.5. Under Assumption 3.3, and $(V_t^{(k)})_{t\geq 0}$ the solution to Equation (3) for each $k = 1, \ldots, K$, for almost every initial condition $(V_0^{(k)})_{k=1}^K$, we have

$$d(\langle V_t^{(k)} - V^\pi \mid k \in [K] \rangle, \langle U_{1:K} \rangle) \to 0\,.$$

U_i is the eigenspace corresponding to the i-th largest eigenvalue

Learning dynamics with “linear” function approximation

Value function parameterization: $V = \Phi \mathbf{w}$ where we can learn *both* the features and the weight vector.

Idealized version of what a neural network update is “trying” to do to the features (if there were no interference between updates for different inputs).

$$\partial_t \Phi_t = \alpha(R^\pi + \gamma P^\pi \Phi_t \mathbf{w}_t - \Phi_t \mathbf{w}_t)\mathbf{w}_t^\top$$

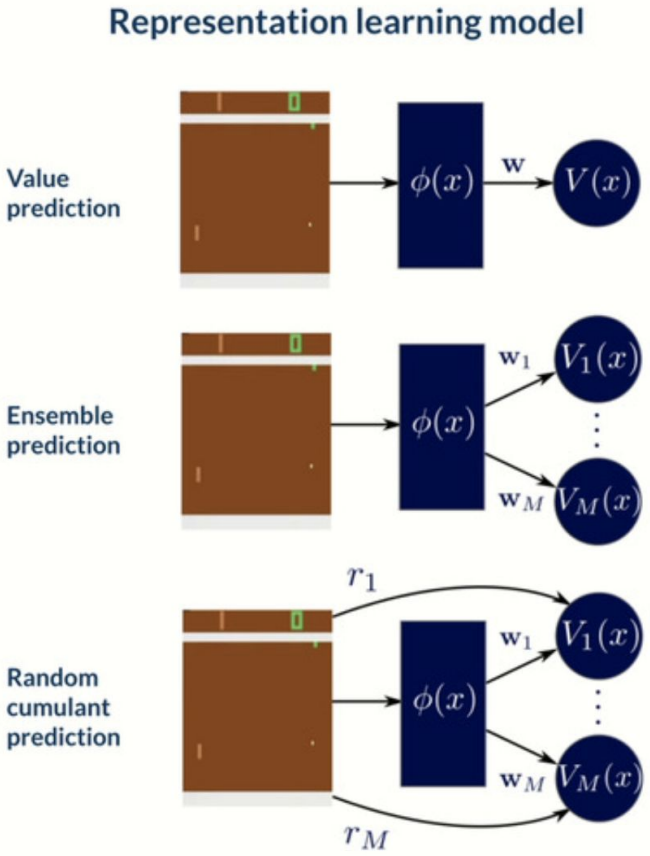
$$\partial_t \mathbf{w}_t = \beta \Phi_t^\top (R^\pi + \gamma P^\pi \Phi_t \mathbf{w}_t - \Phi_t \mathbf{w}_t)$$

Ensemble Prediction

Consider a simpler setting: M prediction heads that are updated slowly (or even fixed) relative to the features.

$$\partial_t \Phi_t^M = \alpha \sum_{m=1}^{M} (R^\pi + \gamma P^\pi \Phi_t^M \mathbf{w}_t^m - \Phi_t^M \mathbf{w}_t^m)(\mathbf{w}_t^m)^\top$$

$$\partial_t \mathbf{w}_t^m = \beta (\Phi_t^M)^\top (R^\pi + \gamma P^\pi \Phi_t^M \mathbf{w}_t^m - \Phi_t \mathbf{w}_t^m)$$



Ensemble Prediction Convergence

Theorem 4.1. For $M \in \mathbb{N}$, let $(\Phi_t^M)_{t\geq 0}$ be the solution to Equation (9), with each $\mathbf{w}_t^m$ for $m = 1, \ldots, M$ initialised independently from $N(0, \sigma_M^2)$, and fixed throughout training ($\beta = 0$). We consider two settings: first, where the learning rate α is scaled as $\frac{1}{M}$ and $\sigma_M^2 = 1$ for all M, and second where $\sigma_M^2 = \frac{1}{M}$ and the learning rate α is equal to 1. These two settings yield the following dynamics, respectively:

$$\lim_{M\to\infty} \partial_t \Phi_t^M \overset{P}{=} -(I - \gamma P^\pi)\Phi_t^M \quad \text{, and} \tag{11}$$

$$\lim_{M\to\infty} \partial_t \Phi_t^M \overset{D}{=} -(I - \gamma P^\pi)\Phi_t^M + R^\pi \epsilon^\top \,,\ \epsilon \sim \mathcal{N}(0, I)\,. \tag{12}$$

The corresponding limiting trajectories for a fixed initialisation $\Phi_0 \in \mathbb{R}^{\mathcal{X}\times K}$, are therefore given respectively by

$$\lim_{M\to\infty} \Phi_t^M \overset{P}{=} \exp(-t(I - \gamma P^\pi))\Phi_0 \quad \text{, and} \tag{13}$$

$$\lim_{M\to\infty} \Phi_t^M \overset{D}{=} \exp(-t(I - \gamma P^\pi))(\Phi_0 - (I - \gamma P^\pi)^{-1} R^\pi \varepsilon^\top) + (I - \gamma P^\pi)^{-1} R^\pi \varepsilon^\top \,,\ \epsilon \sim \mathcal{N}(0, I)\,. \tag{14}$$

Corollary 4.2. Under the feature flow (9) with $\mathbf{w}_t^m$ fixed at initialization for each $i = 1, \ldots, M$ and Assumption 3.3, for almost all initialisations Φ_0, we have when $R^\pi = 0$

$$d(\langle \Phi_t \rangle, \langle U_{1:K} \rangle) \to 0\,, \quad \text{as } t \to \infty.$$

Key insight.
Under the conditions of Theorem 4.1 and Corollary 4.2, the ensemble auxiliary tasks cause the agent's representation Φ to align with EBFs.

Principal eigenvectors of the transition function

Random cumulant prediction

Theorem 4.3. For fixed $M \in \mathbb{N}$, let the random rewards $(r^m)_{m=1}^M$ and weights $(\mathbf{w}^m)_{m=1}^M$ be as defined above, let $\alpha = 1$, and consider the representation dynamics in Equation (15), with weights fixed throughout training ($\beta = 0$). Let Σ denote the covariance matrix of the random cumulant distribution. Then

$$\lim_{M\to\infty} \sum_{m=1}^{M} r^m (\mathbf{w}^m)^\top \overset{D}{=} Z_\Sigma \sim \mathcal{N}(0, \Sigma), \text{ and}$$

$$\lim_{M\to\infty} \Phi_t^M \overset{D}{=} \exp(-t(I - \gamma P^\pi))(\Phi_0 - (I - \gamma P^\pi)^{-1} Z_\Sigma) + (I - \gamma P^\pi)^{-1} Z_\Sigma \,.$$

Tends to zero

Doesn't tend to zero

Key insight.
With random cumulant auxiliary tasks, under the assumptions of Theorem 4.3 and Corollary 4.4, the distribution of the limiting representation does not collapse, and is characterized by the RS-BFs of P^π, while the trajectory it follows to reach this subspace is determined by the EBFs of P^π.

Other auxiliary tasks

Auxiliary task	Dynamics ($r = 0$)	Φ_∞ ($r = 0$)	Φ_∞ ($r \neq 0$)	Limit of $\langle \Phi_t - \Phi_\infty \rangle$
Ensemble	$-(I - \gamma P^\pi)\Phi_t$	0	$(I - \gamma P^\pi)^{-1} r \epsilon^\top$	EBFs of P^π
Random cumulants	$-(I - \gamma P^\pi)\Phi_t + Z_\Sigma$	$(I - \gamma P^\pi)^{-1} Z_\Sigma$	$(I - \gamma P^\pi)^{-1} Z_\Sigma$	EBFs of P^π
Additional policies	$-(I - \gamma P^{\overline{\pi}})\Phi_t$	0	$(I - \gamma P^{\overline{\pi}})^{-1} R^{\overline{\pi}} \epsilon^\top$	EBFs of $P^{\bar{\pi}}$
Multiple γs	$-(I - \overline{\gamma} P^\pi)\Phi_t$	0	$(I - \overline{\gamma} P^\pi)^{-1} R^\pi \epsilon^\top$	EBFs of P^π

What makes a “good” feature set?

EBFs and RSBFs pick up on “structure” in the environment.

But we also care about how well features let us predict the value of the *next* policy the agent will follow in the value improvement path.

These features seem to be better-than-random at generalizing under policy improvement!

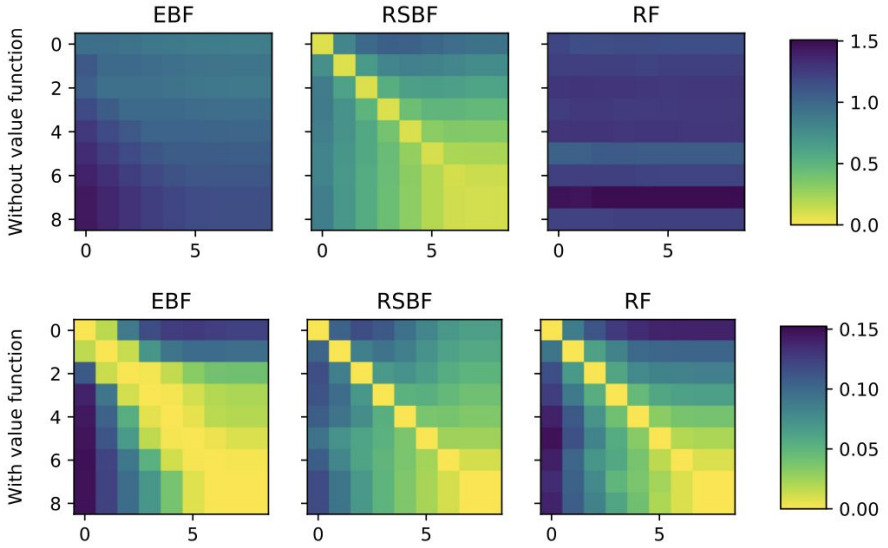


Figure 3: Transfer of EBFs, RSBFs, and RFs across the value-improvement path of a chain MDP, with and without the value function as an additional feature.

Part II: Feature Collapse and Sparsity

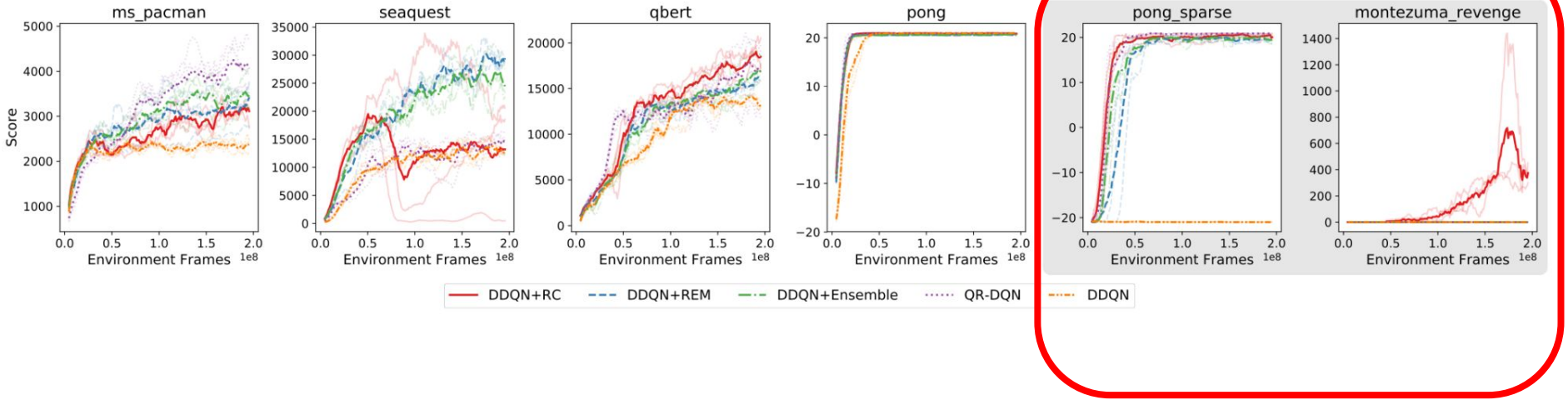
Motivation

Picking up on environment structure is nice, but not if your representation converges to the zero vector!

Previous results suggest that agents in sparse-reward environments may end up with features that are zeroed-out.

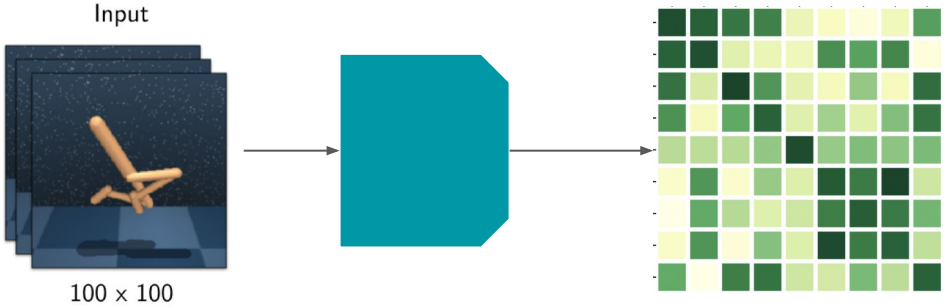
This makes it hard for the agent to update its policy if it sees a reward later.

How do different auxiliary tasks fare in Atari?



Measuring feature collapse in neural networks

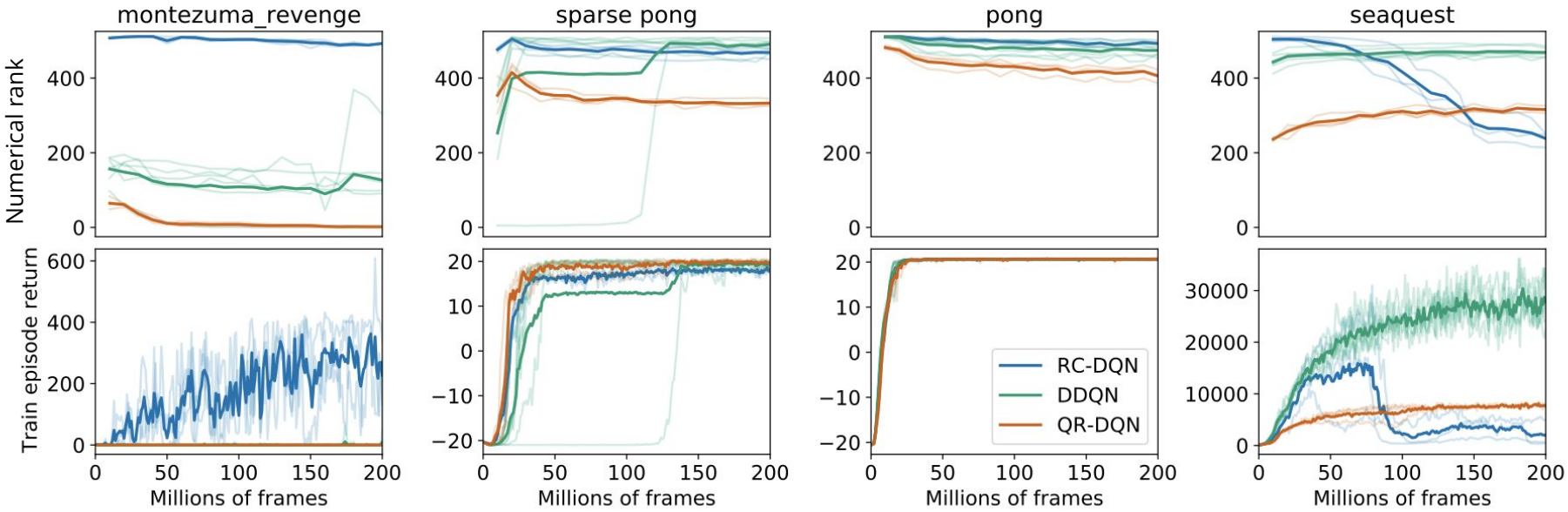
Estimate “numerical rank” of feature embeddings by looking at how many non-negligible singular values the matrix of features for *n* randomly sampled states has.



Compute SVD of this matrix, count nontrivial singular values

$$\hat{\rho}_n(\phi, \mathbf{X}, \epsilon) = |\{\sigma \in \mathbf{SVD}\left(\frac{1}{\sqrt{n}}\phi(\mathbf{X})\right) | \sigma > \varepsilon\}|$$

What's happening to the agents' features?



From feature collapse to capacity loss

Capacity in Feature Space vs Output Space

- Feature rank gives us a sense of how much of the last layer the network is using at a given moment
- It doesn't tell us whether the network could use more of the last layer given a sufficiently informative reward signal.
- What if we instead measured **how well the network can fit new targets starting from its current parameter values**?

$$\mathcal{C}(\mathcal{N}, \mathcal{O}, \mathcal{D}) = \mathbb{E}_{f \sim P_{\mathcal{F}}}[\mathbb{E}_{x \sim P_X}[(g_{\theta'}(x) - f(x))^2]] \quad \textit{where } \theta' = \mathcal{O}(\theta_0, P_X, f)$$

Capacity of a network N using optimization algorithm O on data-generating distribution D

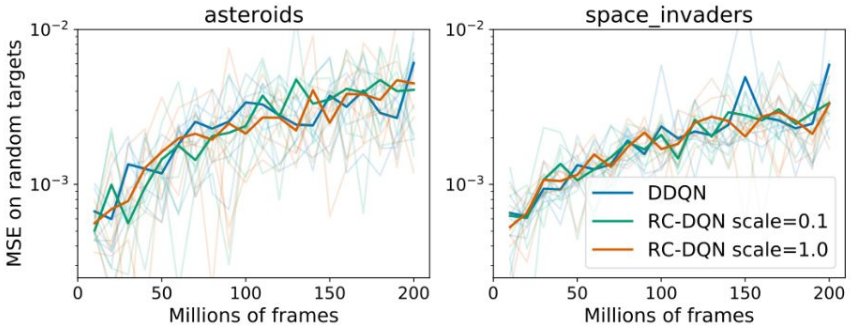
MSE after running optimizer O on targets f starting at parameters θ_0

F sampled from some pre-defined target distribution (e.g. Bellman targets, outputs of different random network initialization, hash function)

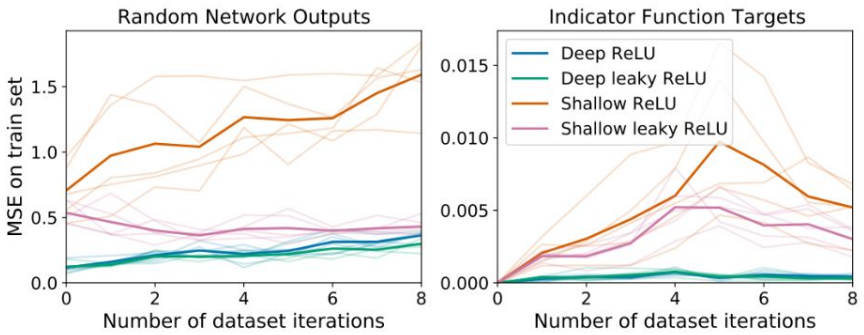
Target-fitting capacity under non-stationary targets

Neural networks tend to get worse at fitting new targets after they've been made to fit several different target functions.

RL



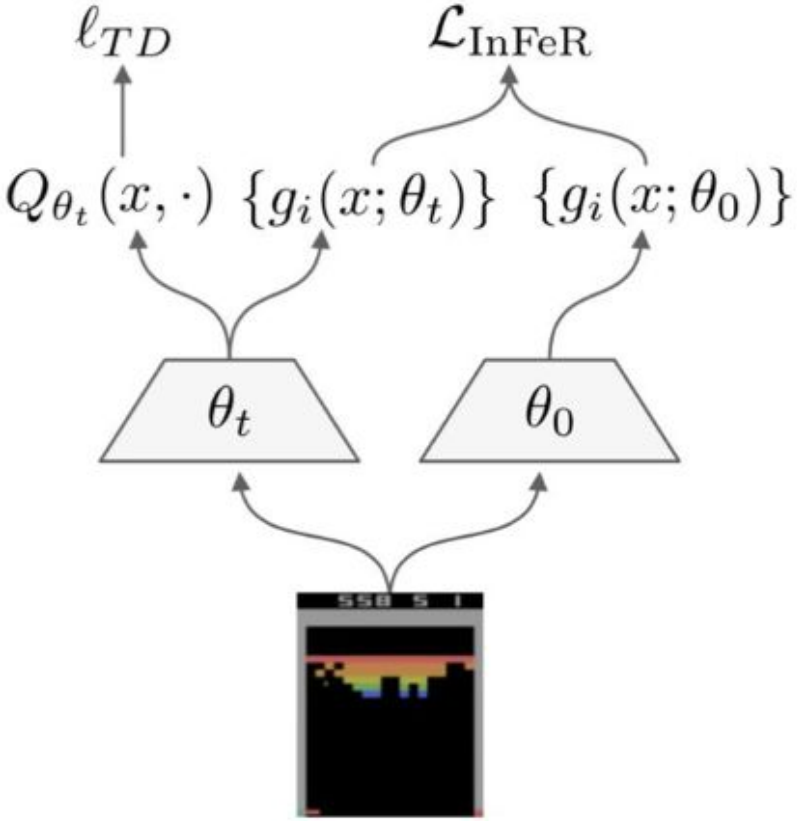
Supervised learning with different target function classes on MNIST inputs



Preserving Representation Capacity

Freshly initialized networks are really good at fitting lots of different types of targets.

Idea: regularize the network so that it maintains its ability to fit functions it could predict at initialization.



Regularization Objective

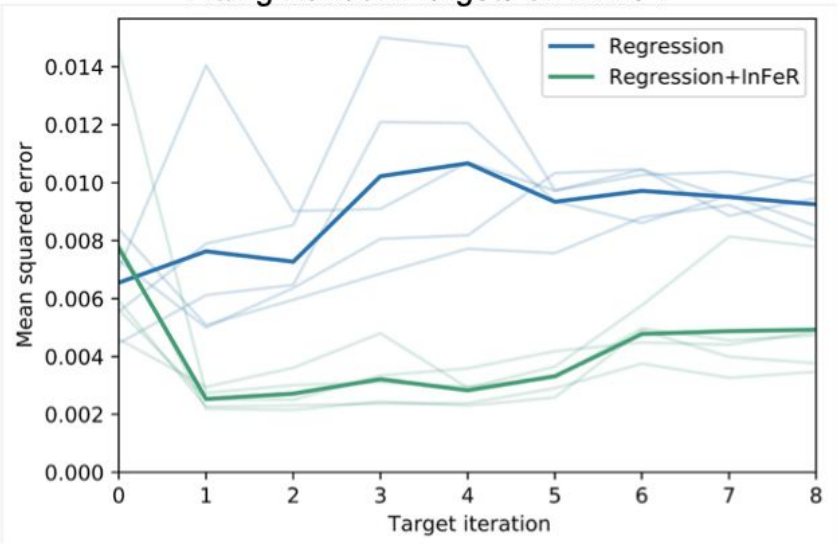
$$\mathcal{L}_{\text{InFeR}}(\theta, \theta_0; \mathcal{B}, \beta) = \mathbb{E}_{x \sim \mathcal{B}} \left[\sum_{i=1}^{k} (g_i(x; \theta) - \beta g_i(x; \theta_0))^2 \right]$$

Auxiliary head outputs using online parameters

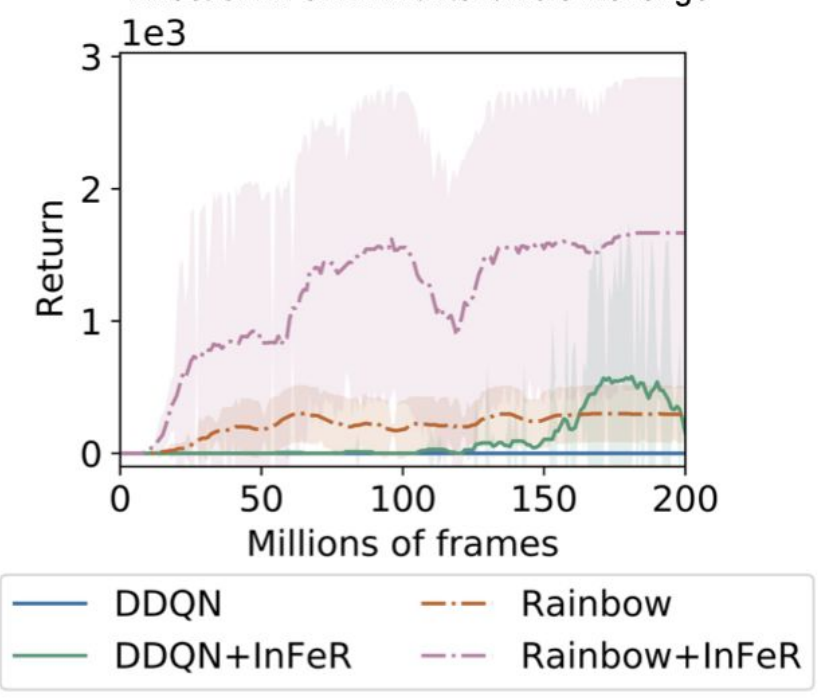
Auxiliary head outputs at initialization

What effect does this have on networks?

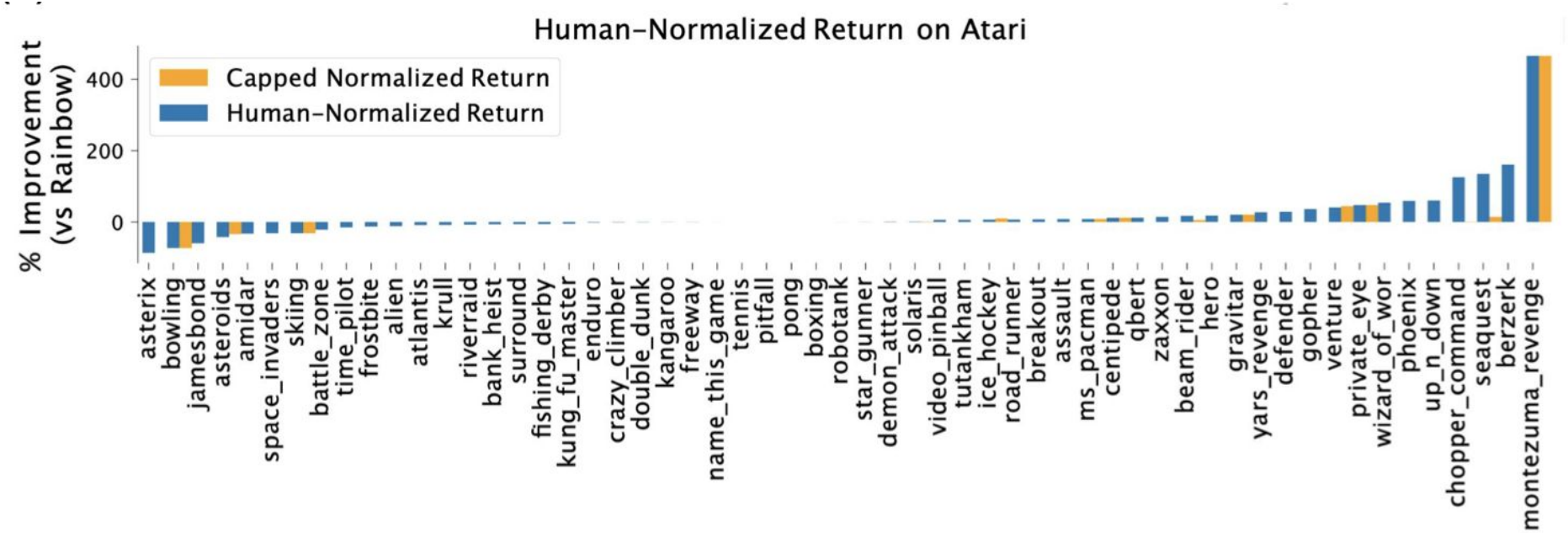
(b)



(c)



Have we overfit to sparse-reward tasks?

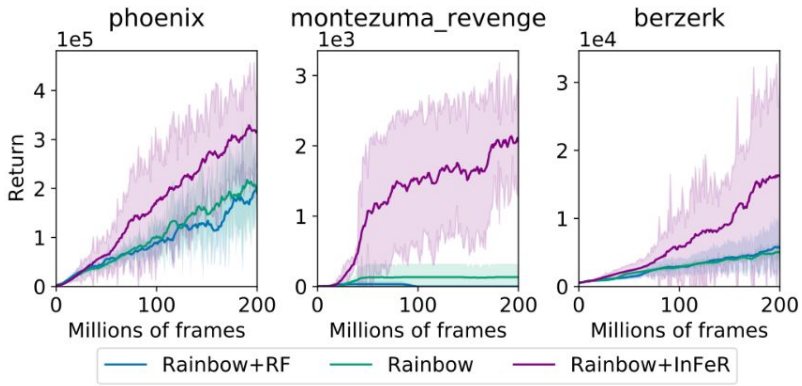


How does InFeR work?

Hypothesis 1: the random subspace of features that InFeR preserves is useful for value prediction.

Experiment: concatenate a subset of the network's randomly initialized features to the penultimate layer outputs and let the final linear layer use these for prediction.

Result: this doesn't produce the same effect as InFeR on performance. ❌



Conclusions

- TD updates implicitly encourage agents' representations to pick up on the structure of the environment transition function.
- However, they can also lead to feature collapse when the agent doesn't receive any reward signal.
- The non-stationary prediction tasks agents fit over the course of training have an adverse effect on their ability to fit future value functions.
- We present a simple approach that mitigates both of these effects, and leads to significant performance improvements in sparse-reward environments.

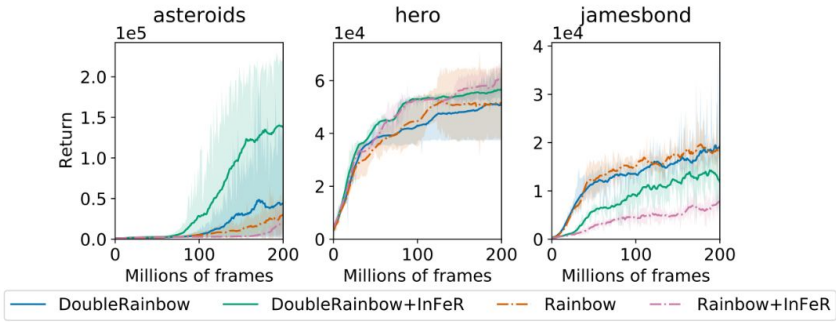
Thanks!

How does InFeR work?

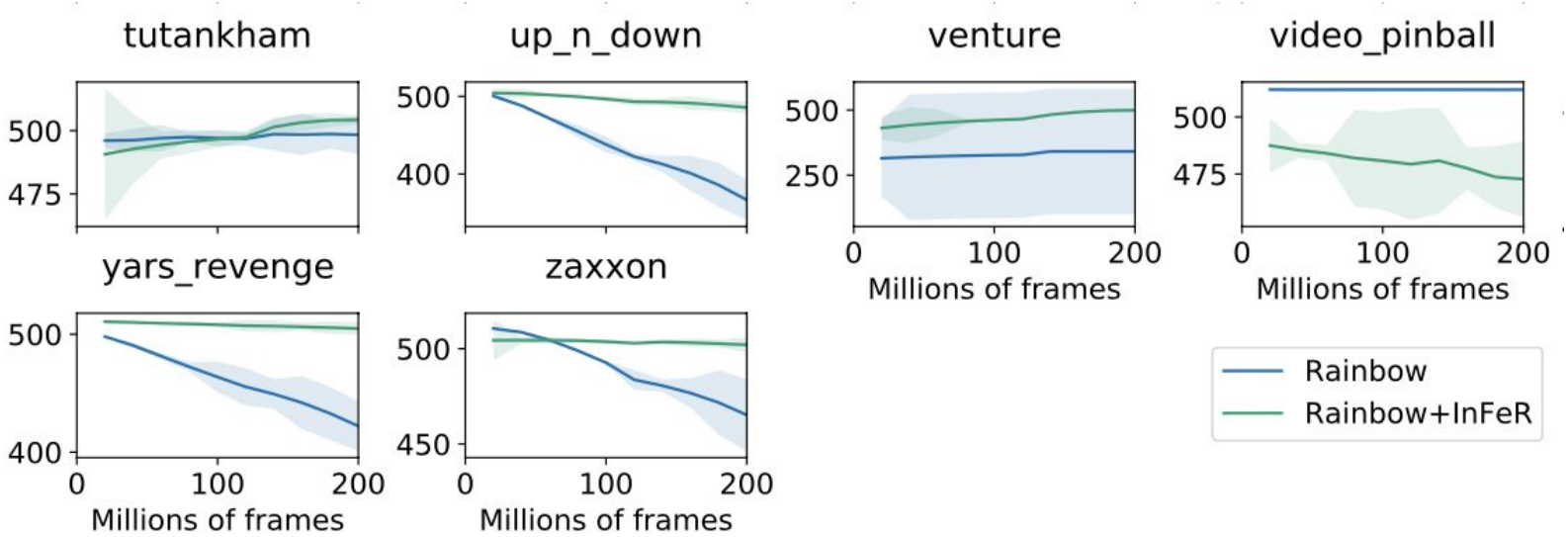
Hypothesis 2: InFeR **constrains how much the network can let its representation drift from its initial value**. In environments where InFeR hurts performance, making significant changes to the representation is necessary for learning progress.

Experiment: increase the width of the penultimate layer, preserve the number of heads to **give the representation more degrees of freedom** during training.

Result: this significantly reduces or eliminates the performance gap in most environments where InFeR hurts performance over Rainbow. ✅



InFeR and Numerical Rank



Feature trajectories

The dynamics under this parameterization aren't linear, so we can't construct a general closed form for the value function in this case.

Still, empirically the eigendecomposition of the transition matrix looks like it plays an important role!