Deep RL ||: Generalization

(what we talk about when we talk about deep RL)

Anna Harutyunyan

Thanks to: Will Dabney, Tom Schaul, Andre Barreto, Diana Borsa, Hado van Hasselt

Deep RL ||

- Advanced research in deep RL that'll matter in 5 years.

Deep RL ||

- Advanced research in deep RL that'll matter in 5 years.

DeepMind

Deep RL ||

- Advanced research in deep RL that'll matter in 5 years.

- We have no idea! Things are moving extremely quickly. It's exciting!
- Let's focus on the **problems**

Recap

$$v^{\pi}(x)$$

Recap

$$\mathbb{E}_\pi\Big[\sum_{t=0}^{\infty}\gamma^t R_{t+1}|X_0=x\Big]$$

$$v^\pi(x)$$

Recap

$$\mathbb{E}_\pi\Big[\sum_{t=0}^{\infty}\gamma^t R_{t+1}|X_0=x\Big]$$
$$=\mathbb{E}_\pi[R_1+\gamma[v^\pi(X_1)]$$

$$v^\pi(x)$$

Bellman

Recap

$$v^{\pi}(x)$$

Tabular RL

DeepMind

Recap

$$v^{\pi}(x)$$

$$v^{\pi}(x) \approx \phi(x)^{T}\theta$$

Tabular RL

RL with function approximation

Recap

$$v^{\pi}(x)$$

$$v^{\pi}(x) \approx \phi(x)^{T}\theta$$

$$\phi : \mathcal{X} \to \mathbb{R}^{K}$$

$$\theta \in \mathbb{R}^{K}$$

Tabular RL

RL with function approximation

Recap

$$v^{\pi}(x)$$

$$v^{\pi}(x) \approx \phi(x)^{T}\theta$$

$$\phi : \mathcal{X} \rightarrow \mathbb{R}^{K}$$

$$\theta \in \mathbb{R}^{K}$$

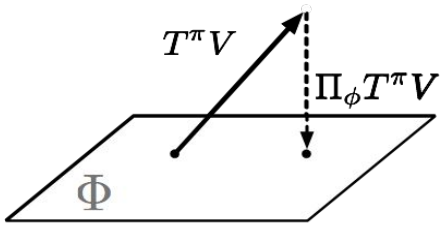
Tabular RL

RL with function approximation

Recap

Targets

Linear RL



$$v^\pi(x)$$

$$v^\pi(x) \approx \phi(x)^T \theta$$

$$\phi : \mathcal{X} \to \mathbb{R}^K$$

$$\theta \in \mathbb{R}^K$$

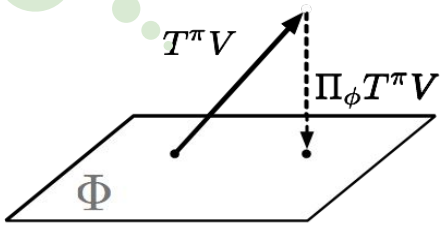
Tabular RL

RL with function approximation

$R + \gamma V'$

Targets

Linear RL



Recap

$$v^\pi(x)$$

$$v^\pi(x) \approx \phi(x)^T \theta$$

$$\phi : \mathcal{X} \to \mathbb{R}^K$$

$$\theta \in \mathbb{R}^K$$

Tabular RL

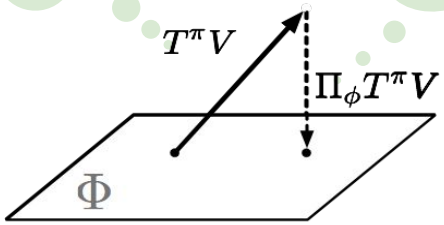
RL with function approximation

Targets

Linear RL

$R + \gamma V'$

Projection back



Recap

$$v^\pi(x)$$

$$v^\pi(x) \approx \phi(x)^T \theta$$

$$\phi : \mathcal{X} \to \mathbb{R}^K$$

$$\theta \in \mathbb{R}^K$$

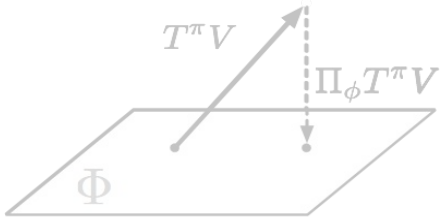
Tabular RL

RL with function approximation

Recap

Targets

Linear RL



$$v^\pi(x)$$

$$v^\pi(x) \approx \phi(x)^T \theta$$

$$\phi : \mathcal{X} \to \mathbb{R}^K$$

$$\theta \in \mathbb{R}^K$$

Tabular RL

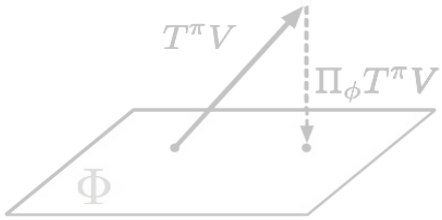
RL with function approximation

Representation learning

Recap

Targets

Linear RL



$$v^\pi(x)$$

$$v^\pi(x) \approx \phi(x)^T \theta$$

$$\phi : \mathcal{X} \to \mathbb{R}^K$$

$$\theta \in \mathbb{R}^K$$

Jointly, end-to-end

Tabular RL

RL with function approximation

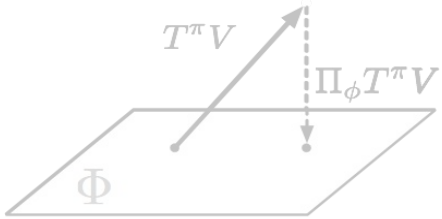
Deep RL

Representation learning

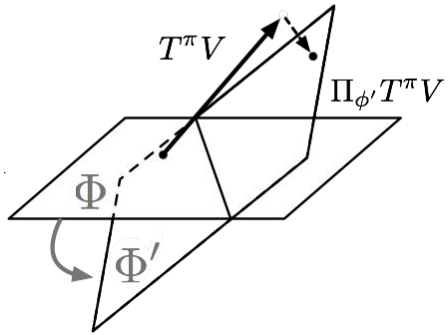
DeepMind

Recap

Linear RL



Deep RL



$$v^\pi(x)$$

Tabular RL

$$v^\pi(x) \approx \phi(x)^T \theta$$

RL with function approximation

$$\phi : \mathcal{X} \to \mathbb{R}^K$$

$$\theta \in \mathbb{R}^K$$

Jointly, end-to-end

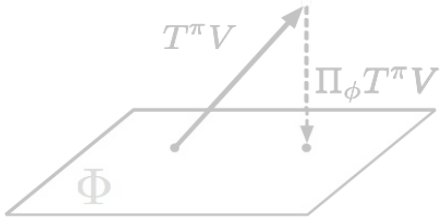
Deep RL

Representation learning

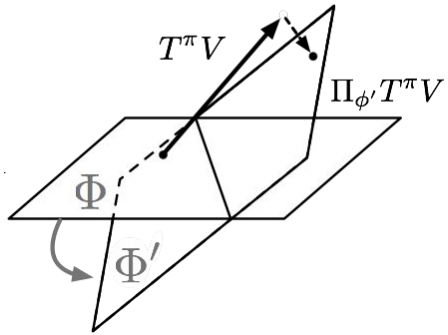
DeepMind

Recap

Linear RL



Deep RL



$$v^\pi(x)$$

Tabular RL

$$v^\pi(x) \approx \phi(x)^T \theta$$

RL with function approximation

$$\phi : \mathcal{X} \to \mathbb{R}^K$$

$$\theta \in \mathbb{R}^K$$

Jointly, end-to-end

Deep RL

Representation learning

Key challenge: Generalization

DeepMind

What is generalization in RL?

Usually in ML, this is generalizing to **unseen examples**

What is an unseen example in RL?

What is an unseen example in RL?

$$\mathcal{M} = (\mathcal{X}, \mathcal{A}, p, r, \gamma)$$

What is an unseen example in RL?

$$\mathcal{M} = (\mathcal{X}, \mathcal{A}, p, r, \gamma)$$

$$v^{\pi} | \pi, \mathcal{M}$$

$$\pi^{*} | \mathcal{M}$$

What is an unseen example in RL?

$$\mathcal{M} = (\mathcal{X}, \mathcal{A}, p, r, \gamma)$$

$$v^\pi | \pi, \mathcal{M}$$

$$\pi^* | \mathcal{M}$$

Observations

- Auxiliary tasks
- Distributional RL

What is an unseen example in RL?

$$\mathcal{M} = (\mathcal{X}, \mathcal{A}, p, r, \gamma)$$

$$v^{\pi}|\pi, \mathcal{M}$$

$$\pi^{*}|\mathcal{M}$$

Observations
- Auxiliary tasks
- Distributional RL

Rewards
- Successor features

DeepMind

What is an unseen example in RL?

$$\mathcal{M} = (\mathcal{X}, \mathcal{A}, p, r, \gamma)$$

$$v^{\pi} | \pi, \mathcal{M}$$

$$\pi^{*} | \mathcal{M}$$

Observations
- Auxiliary tasks
- Distributional RL

Rewards
- Successor features

Dynamics
- Max entropy RL
- Empowerment

DeepMind

What is an unseen example in RL?

$$\mathcal{M} = (\mathcal{X}, \mathcal{A}, p, r, \gamma)$$

$$v^{\pi} | \pi, \mathcal{M}$$

$$\pi^{*} | \mathcal{M}$$

Observations
- Auxiliary tasks
- Distributional RL

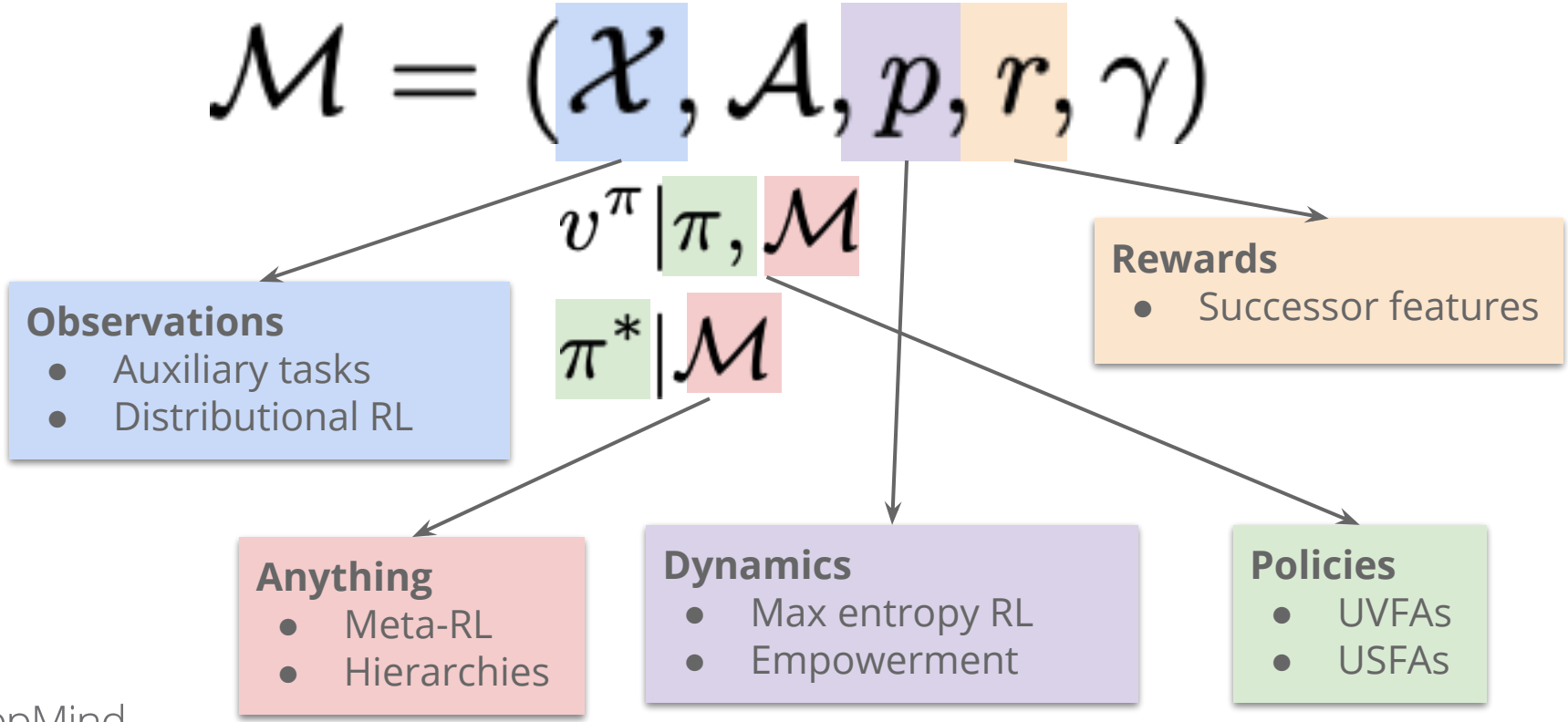
Rewards
- Successor features

Dynamics
- Max entropy RL
- Empowerment

Policies
- UVFAs
- USFAs

DeepMind

What is an unseen example in RL?



DeepMind

What is an unseen example in RL?

Time

$$\mathcal{M} = (\mathcal{X}, \mathcal{A}, p, r, \gamma)$$

$$v^{\pi} | \pi, \mathcal{M}$$

$$\pi^{*} | \mathcal{M}$$

Observations
- Auxiliary tasks
- Distributional RL

Rewards
- Successor features

Anything
- Meta-RL
- Hierarchies

Dynamics
- Max entropy RL
- Empowerment

Policies
- UVFAs
- USFAs

DeepMind

What is an unseen example in RL?

Time

$$\mathcal{M} = (\mathcal{X}, \mathcal{A}, p, r, \gamma)$$

$$v^{\pi} | \pi, \mathcal{M}$$

$$\pi^{*} | \mathcal{M}$$

Observations
- Auxiliary tasks
- Distributional RL

Rewards
- Successor features

Anything
- Meta-RL
- Hierarchies

Dynamics
- Max entropy RL
- Empowerment

Policies
- UVFAs
- USFAs

DeepMind

This lecture

This lecture

1. Generalizing within a task
    - Auxiliary tasks
    - Distributional RL

This lecture

1. Generalizing within a task
    - Auxiliary tasks
    - Distributional RL
2. Generalizing across tasks
    - Successor Features
    - Universal Value Function Approximators
    - Universal Successor Feature Approximators

This lecture

1. Generalizing within a task
    - Auxiliary tasks
    - Distributional RL
2. Generalizing across tasks
    - Successor Features
    - Universal Value Function Approximators
    - Universal Successor Feature Approximators

This lecture

1. Generalizing within a task
   - Auxiliary tasks
   - Distributional RL
2. Generalizing across tasks
   - Successor Features
   -
   -

Goals:

- Understand dimensions of generalization in deep RL
- Give intuition into some key problems / solution families
- *Focus on principles over details and empirical performance*

1) Generalizing within a task

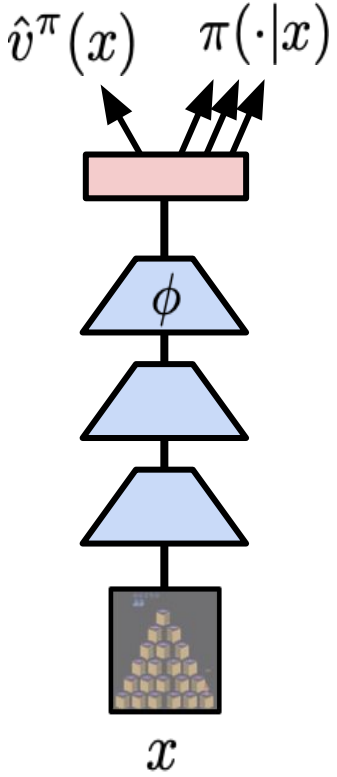
Contents

- Generalizing to new observations
- Auxiliary tasks
- Distributional RL
- Value improvement path

Generalizing to new observations

What determines generalization to unseen observations?

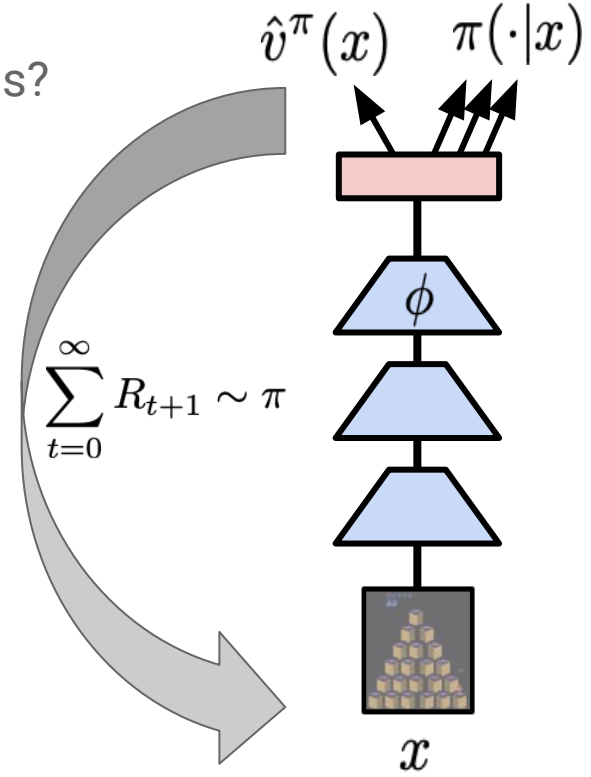
- The **architecture** gives an inductive bias
  - Convolutions like symmetries, invariances, etc
- The **loss** determines what is learned



Generalizing to new observations

What determines generalization to unseen observations?

- The **architecture** gives an inductive bias
  - Convolutions like symmetries, invariances, etc
- The **loss** determines what is learned
- Here loss = the RL objective

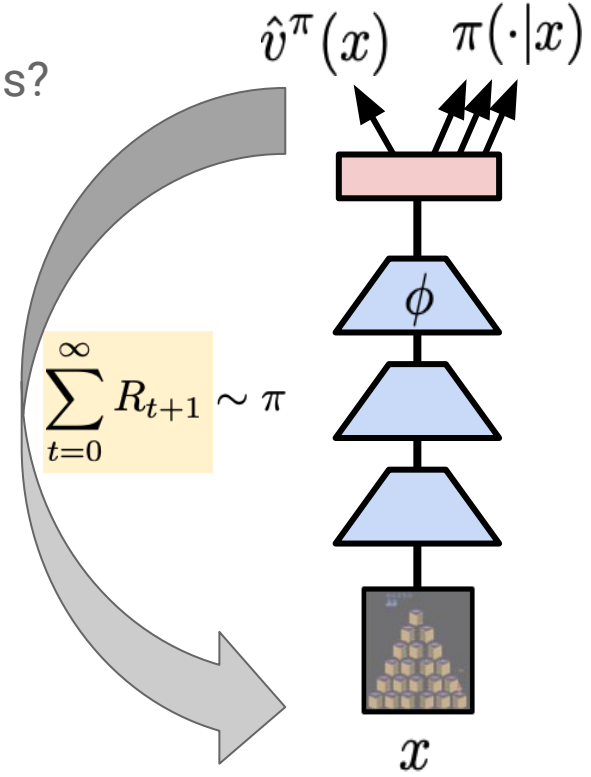


DeepMind

Generalizing to new observations

What determines generalization to unseen observations?

- The **architecture** gives an inductive bias
  - Convolutions like symmetries, invariances, etc
- The **loss** determines what is learned
- Here loss = the RL objective
  - **Issue 1:** often uninformative (sparse rewards)

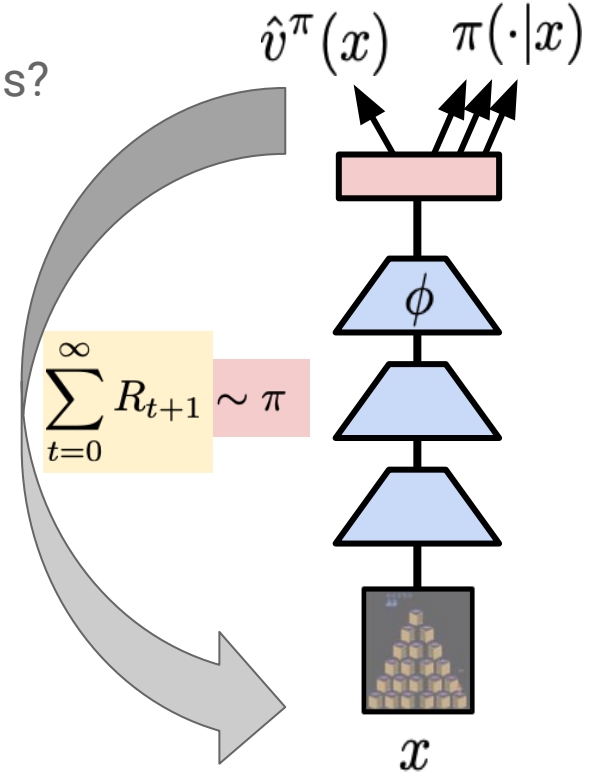


DeepMind

Generalizing to new observations

What determines generalization to unseen observations?

- The **architecture** gives an inductive bias
  - Convolutions like symmetries, invariances, etc
- The **loss** determines what is learned
- Here loss = the RL objective
  - **Issue 1:** often uninformative (sparse rewards)
  - **Issue 2:** always non-stationary (changing policies)

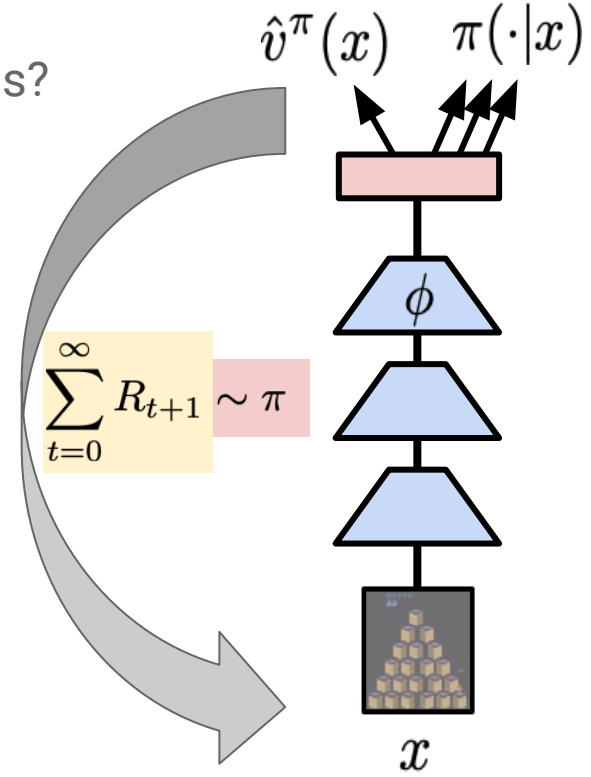


DeepMind

Generalizing to new observations

What determines generalization to unseen observations?

- The **architecture** gives an inductive bias
  - Convolutions like symmetries, invariances, etc
- The **loss** determines what is learned
- Here loss = the RL objective
  - **Issue 1:** often uninformative (sparse rewards)
  - **Issue 2:** always non-stationary (changing policies)



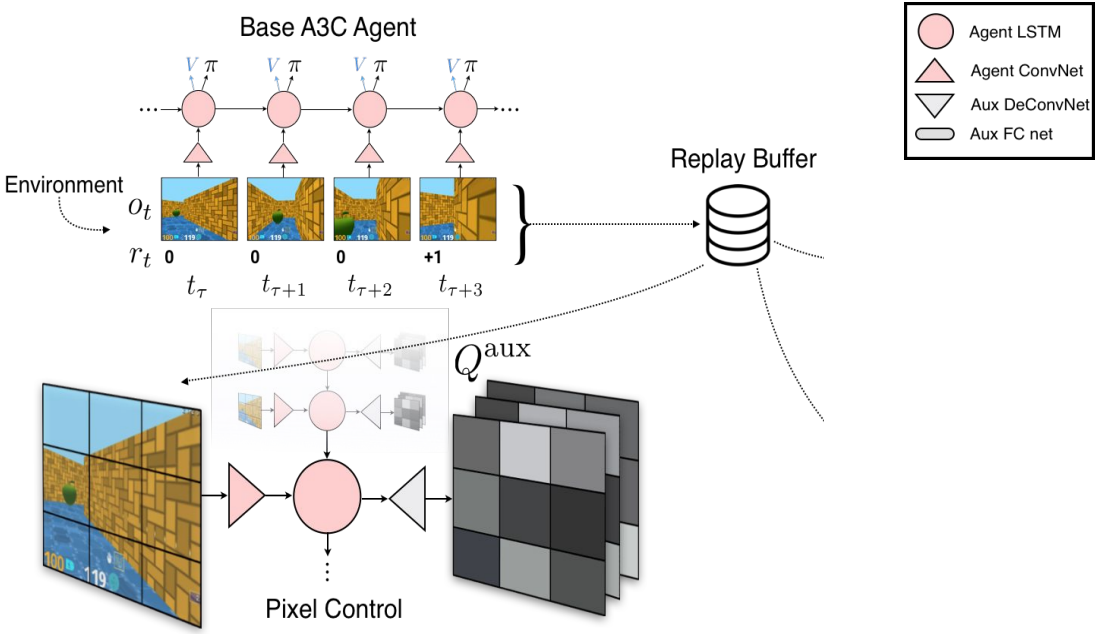
DeepMind

Auxiliary tasks

- Learn about many things, not just reward
- Even if the policy is still bad, there will at least be ground to learn it on
- Often has to do with learning values for other **cumulants** = reward functions:

$$c : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \to \mathbb{R}$$

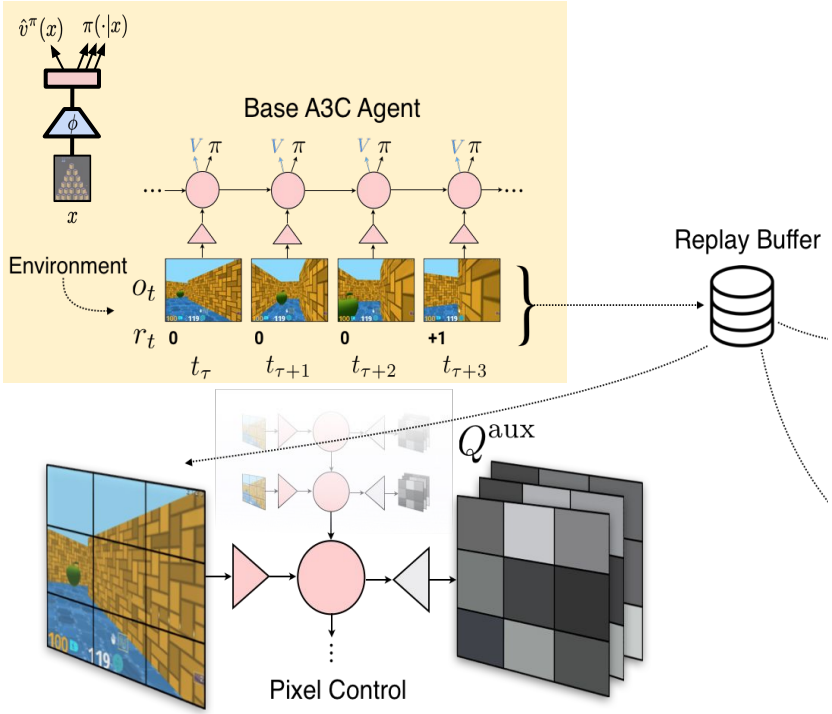
(a part of the) UNREAL architecture



Jaderberg, Max, et al. "Reinforcement learning with unsupervised auxiliary tasks." *ICLR*, 2017.

(a part of the) UNREAL architecture

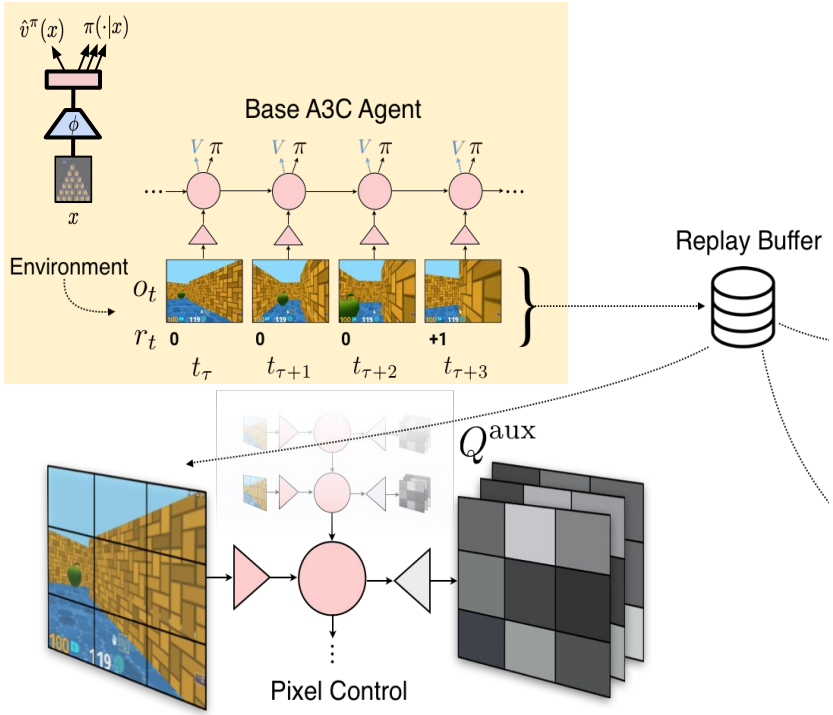
- A3C + a few types of auxiliary tasks



Jaderberg, Max, et al. "Reinforcement learning with unsupervised auxiliary tasks." *ICLR*, 2017.

(a part of the) UNREAL architecture

- A3C + a few types of auxiliary tasks
- Still **act** with π

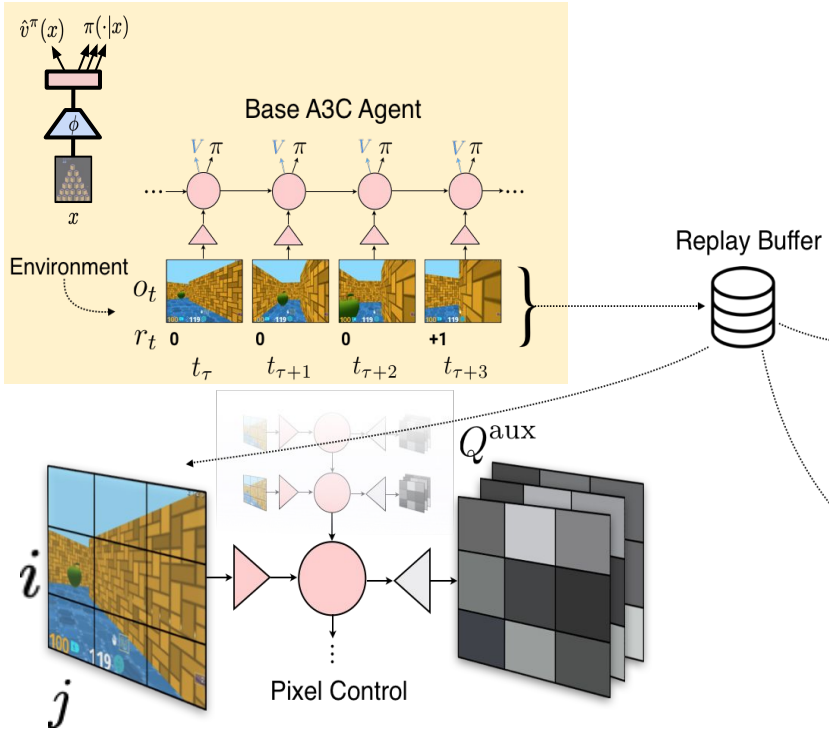


Jaderberg, Max, et al. "Reinforcement learning with unsupervised auxiliary tasks." *ICLR*, 2017.

(a part of the) UNREAL architecture

- A3C + a few types of auxiliary tasks
- Still **act** with π
- In parallel, **off-policy** learn to maximize change in pixel intensity

$$c(x, a, x') = x[i, j] - x'[i, j]$$



DeepMind

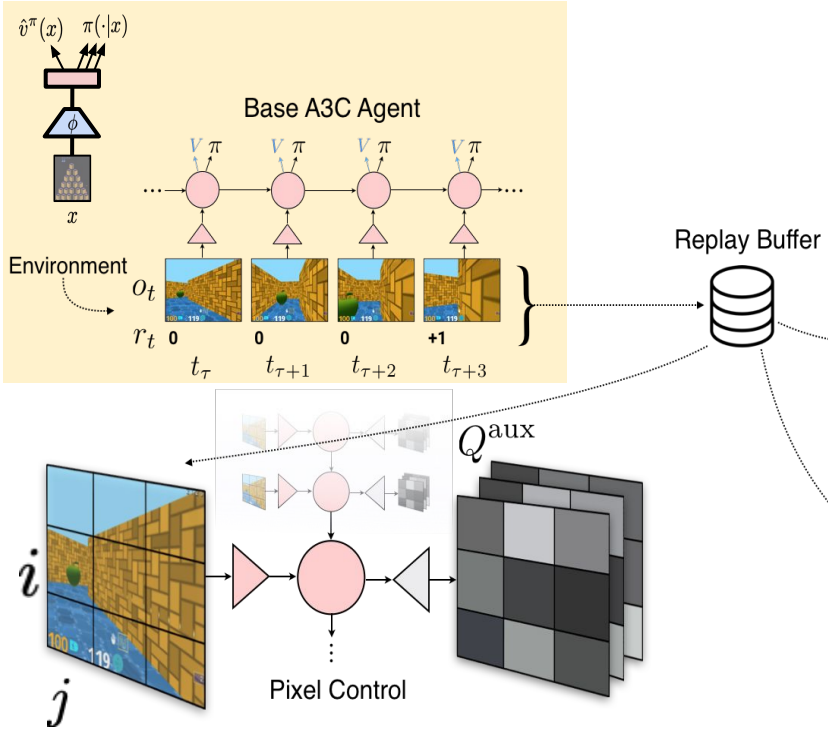
Jaderberg, Max, et al. "Reinforcement learning with unsupervised auxiliary tasks." *ICLR*, 2017.

(a part of the) UNREAL architecture

- A3C + a few types of auxiliary tasks
- Still **act** with π
- In parallel, **off-policy** learn to maximize change in pixel intensity

$$c(x, a, x') = x[i, j] - x'[i, j]$$

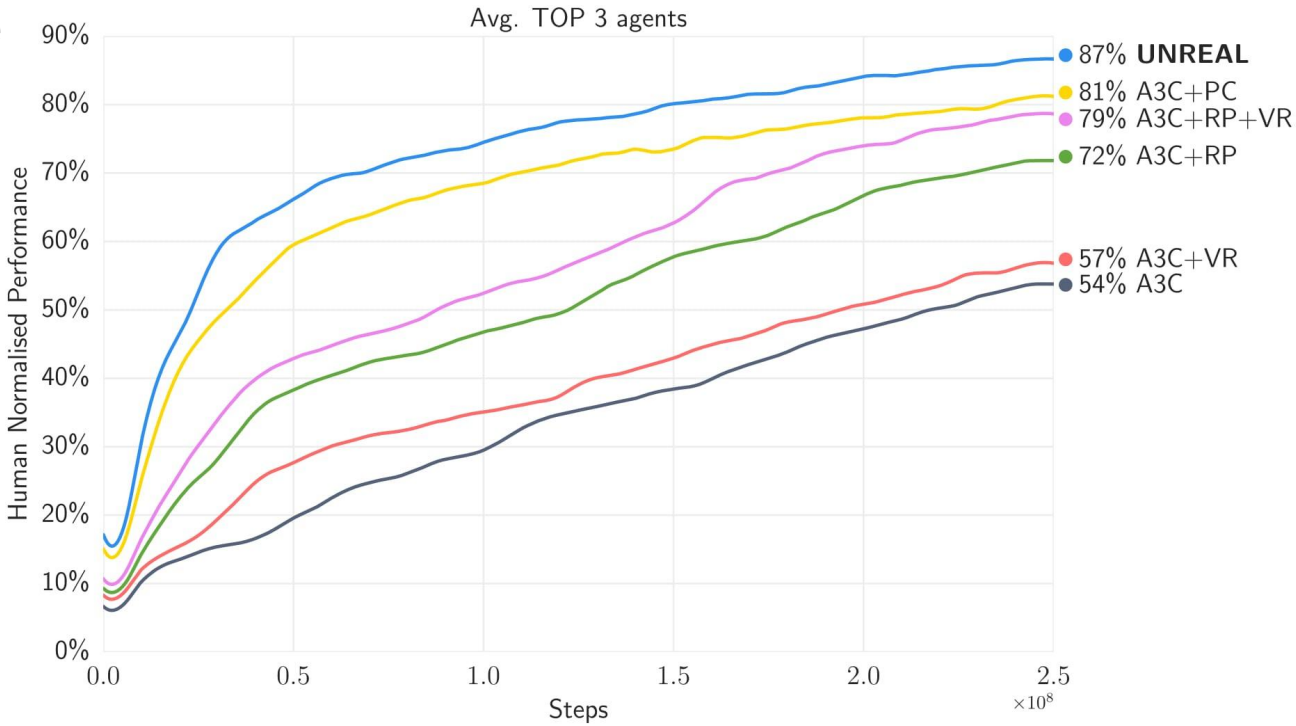
- **Unsupervised**, just observations
  - Follow bright lights, etc



Jaderberg, Max, et al. "Reinforcement learning with unsupervised auxiliary tasks." *ICLR*, 2017.

DeepMind Lab Results

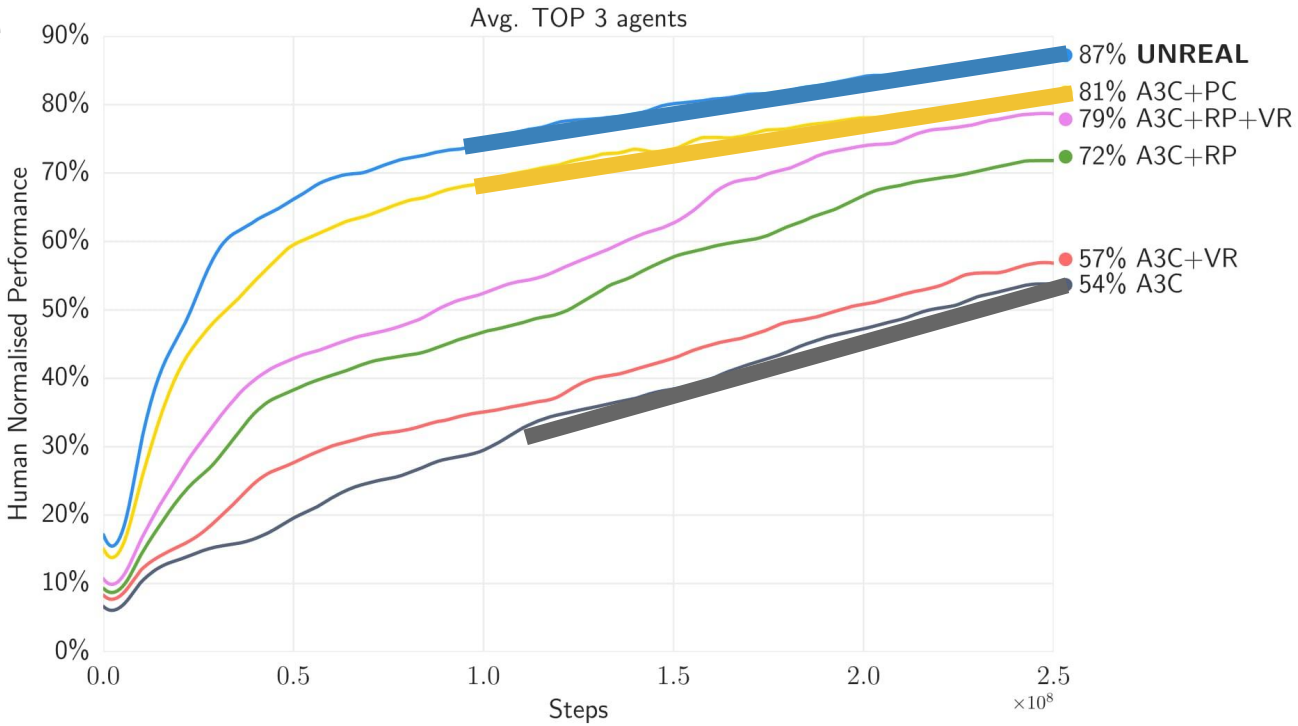
- Average human-normalized performance on 13 3D environments from DeepMind Lab.
- Tasks include random maze navigation and laser tag.
- Roughly a 10x improvement in data efficiency over A3C.
- 60% improvement in final performance.



DeepMind

DeepMind Lab Results

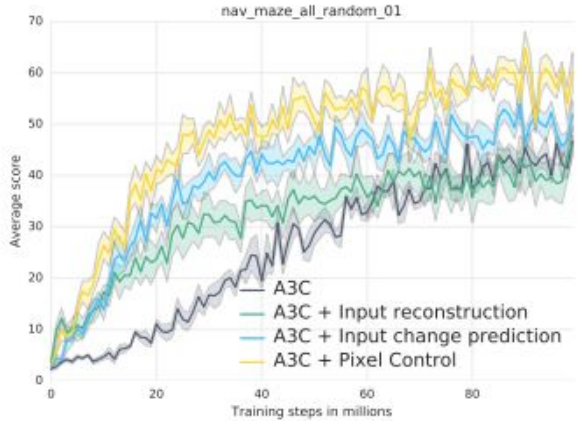
- Average human-normalized performance on 13 3D environments from DeepMind Lab.
- Tasks include random maze navigation and laser tag.
- Roughly a 10x improvement in data efficiency over A3C.
- 60% improvement in final performance.

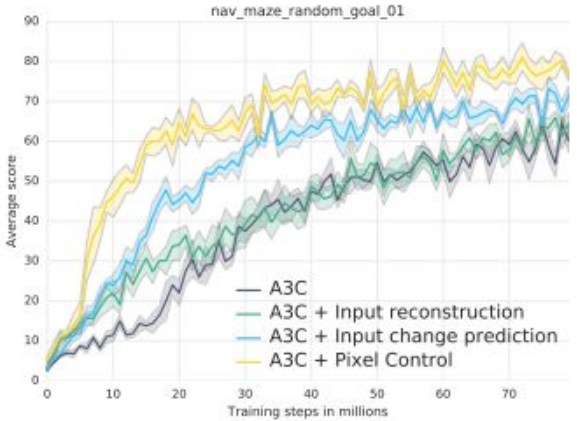


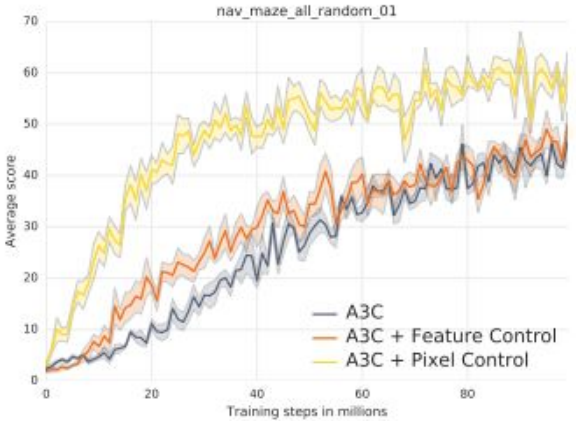
DeepMind

What types of tasks can be helpful?

- Random maze results
- Input change > input reconstruction
- Pixel control > prediction







DeepMind

Jaderberg, Max, et al. "Reinforcement learning with unsupervised auxiliary tasks." *ICLR*, 2017.

Auxiliary tasks

- Learn about many things, not just reward
  - Predict / control pixels, features, actions; past, present or future
- Even if the policy is still bad, there will at least be ground to learn it on
- Often has to do with learning values for other **cumulants** = reward functions:

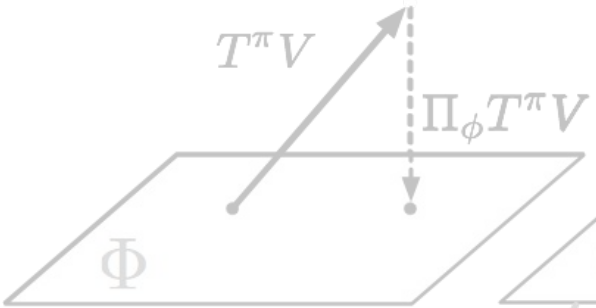
$$c : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \to \mathbb{R}$$

- Don't have to use the extra predictions in the RL algorithm itself
- Just shaping the representation can be very powerful
- Learning about lots of things was also the key idea behind **Horde**
  - Intuition was to build predictive knowledge
- Condensing that knowledge into the representation (or the agent state) is a way to think about it

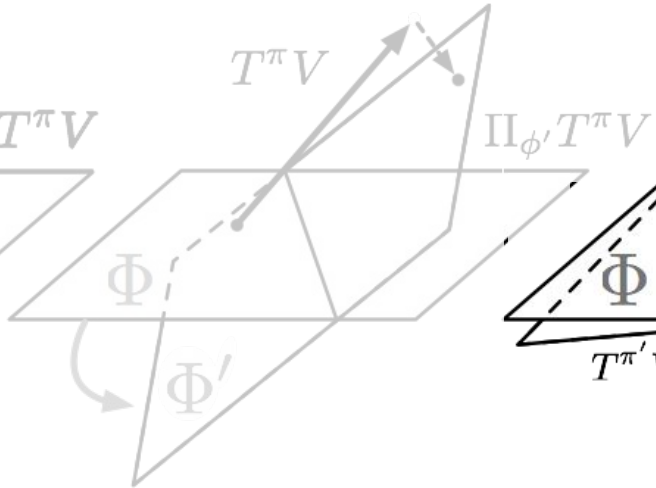
Sutton, Richard S., et al. "Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction." *AAMAS,* 2011.
Jaderberg, Max, et al. "Reinforcement learning with unsupervised auxiliary tasks." *ICLR*, 2017.

Regularizing the representation?

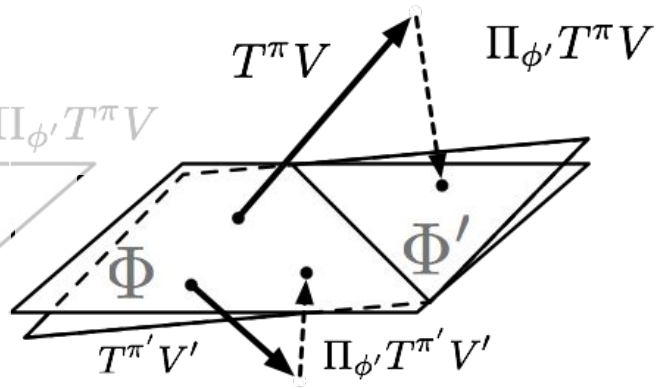
Linear RL



Deep RL



Deep RL
(with auxiliary)



DeepMind

Distributional RL

- Algorithms that learn statistics of the return beyond the mean:

$$Z^\pi(x,a) = \sum_{t=0}^{\infty} \gamma^t R_{t+1}$$

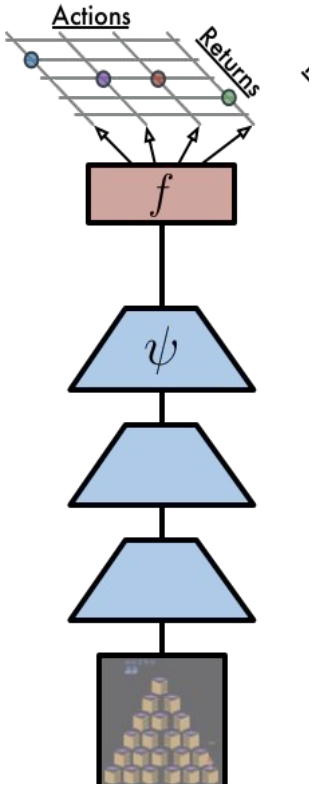
Distributional RL

- Algorithms that learn statistics of the return beyond the mean:

$$Z^\pi(x, a) = \sum_{t=0}^{\infty} \gamma^t R_{t+1}$$

- A particular family of auxiliary tasks

Q-Learning / DQN



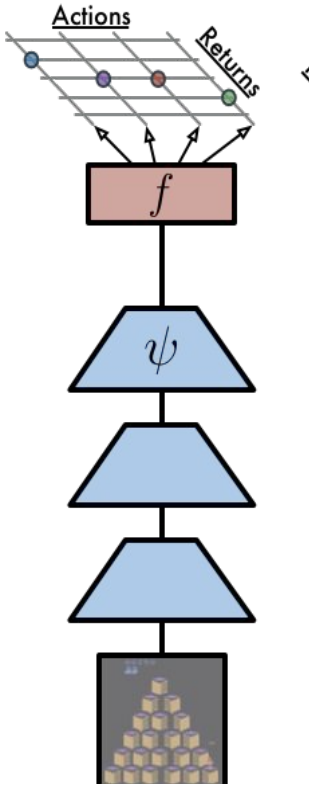
Targets

Loss

DeepMind

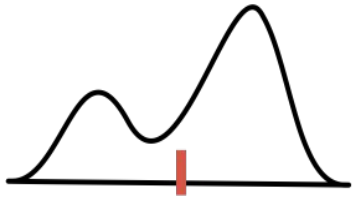
Watkins, Christopher JCH, and Peter Dayan. "Q-learning." *Machine learning* 8.3-4 (1992): 279-292.

Q-Learning / DQN



Targets

$$\mathbb{E}[Z^{\pi}(x, a)]$$



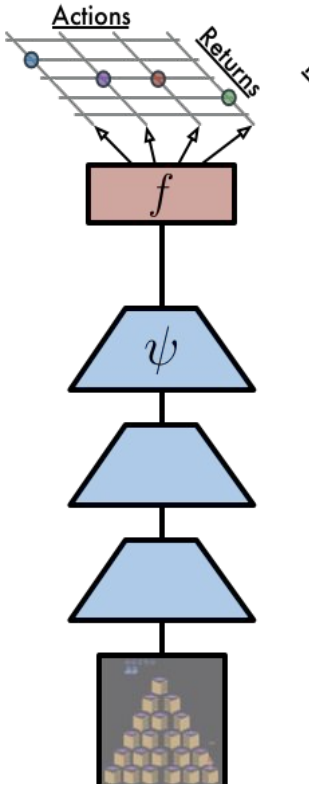
Estimate the expected value of the return distribution.

Loss

DeepMind

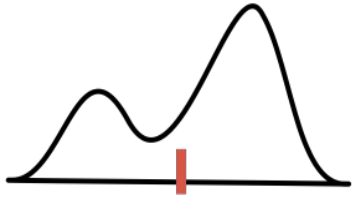
Watkins, Christopher JCH, and Peter Dayan. "Q-learning." *Machine learning* 8.3-4 (1992): 279-292.

Q-Learning / DQN



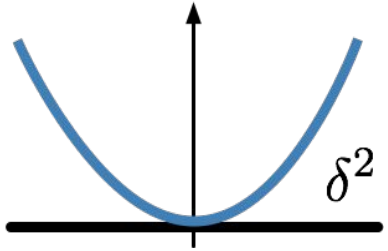
Targets

$$\mathbb{E}[Z^\pi(x,a)]$$



Loss

$$\mathcal{T}^\pi Q - Q$$

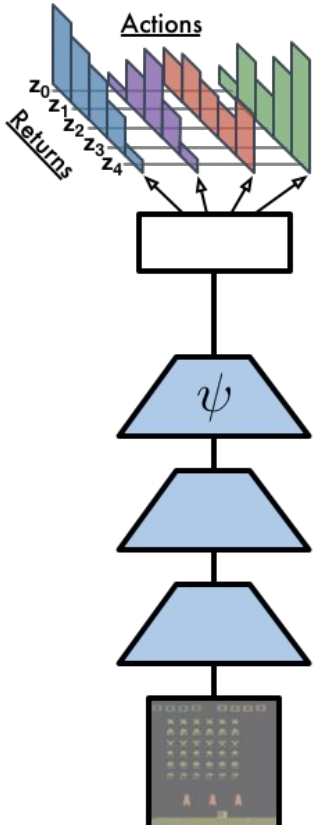


Estimate the expected value of the return distribution.

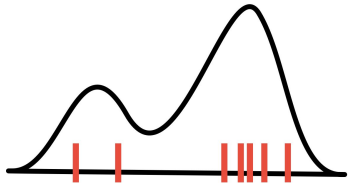
DeepMind

Watkins, Christopher JCH, and Peter Dayan. "Q-learning." *Machine learning* 8.3-4 (1992): 279-292.

Categorical Distributional RL



Targets

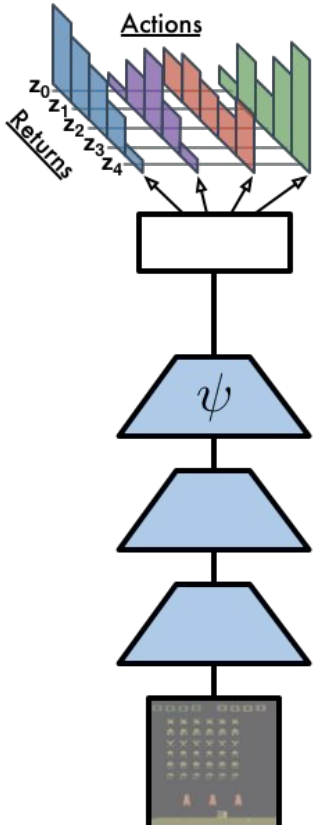


Loss

DeepMind

Bellemare, Marc G., Will Dabney, and Rémi Munos. "A distributional perspective on reinforcement learning." *ICML,* 2017.

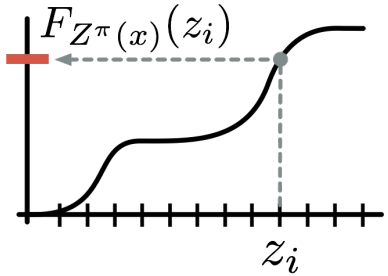
Categorical Distributional RL



Targets

Loss

$$\mathbb{P}\{Z^\pi(x,a) \leq z_k\}$$

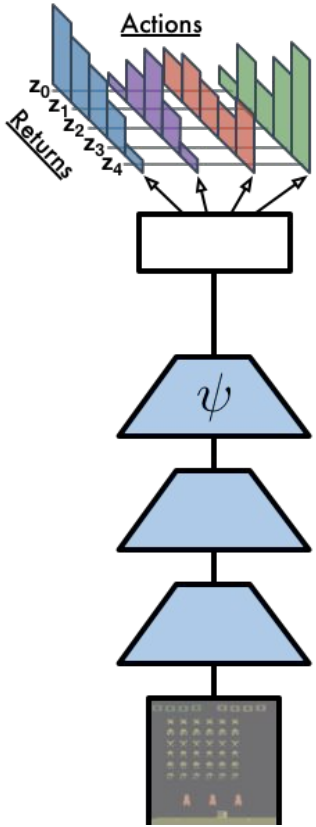


Estimate k probabilities of getting a return of a certain value

DeepMind

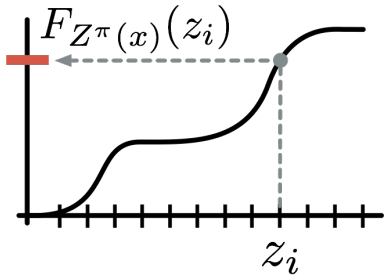
Bellemare, Marc G., Will Dabney, and Rémi Munos. "A distributional perspective on reinforcement learning." *ICML,* 2017.

Categorical Distributional RL



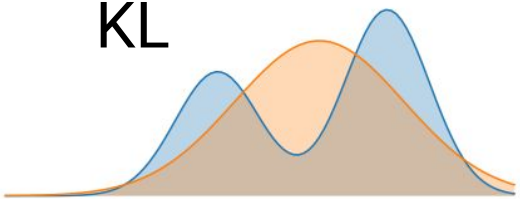
Targets

$$\mathbb{P}\{Z^\pi(x,a) \leq z_k\}$$



Estimate k probabilities of getting a return of a certain value

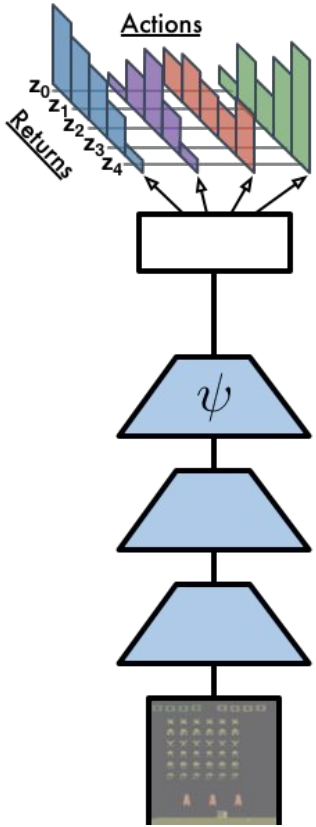
Loss



DeepMind

Bellemare, Marc G., Will Dabney, and Rémi Munos. "A distributional perspective on reinforcement learning." *ICML,* 2017.

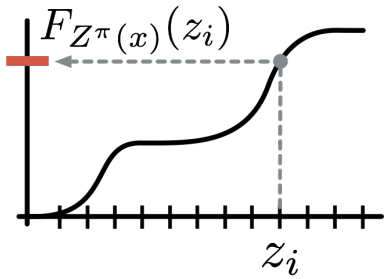
Categorical Distributional RL



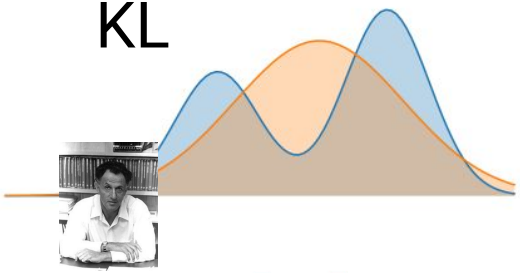
Targets

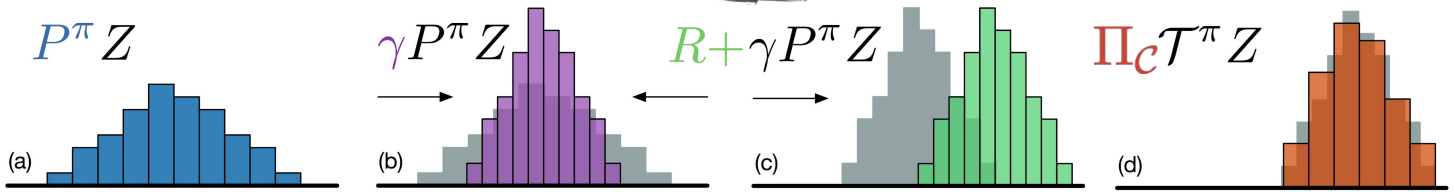
Loss

$$\mathbb{P}\{Z^\pi(x,a) \le z_k\}$$



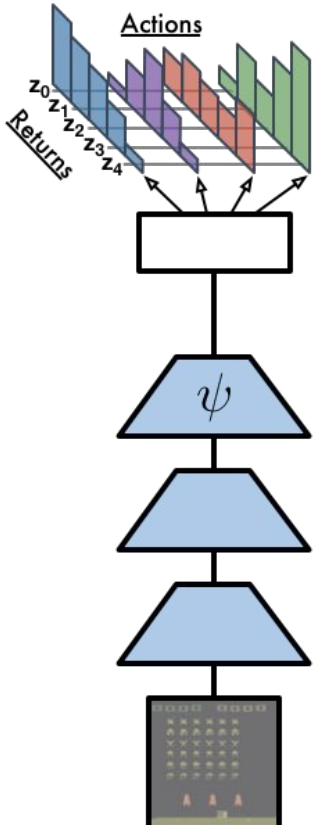
KL





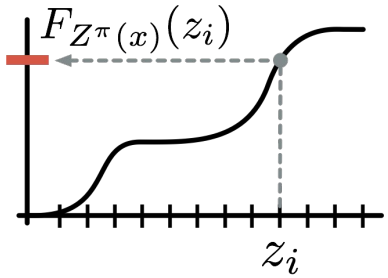
DeepMind

Categorical Distributional RL



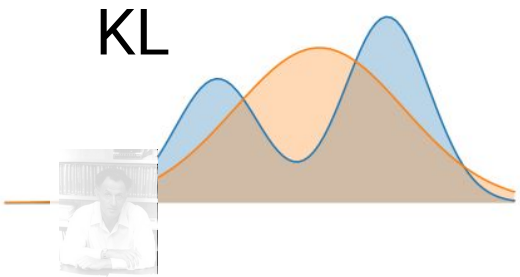
Targets

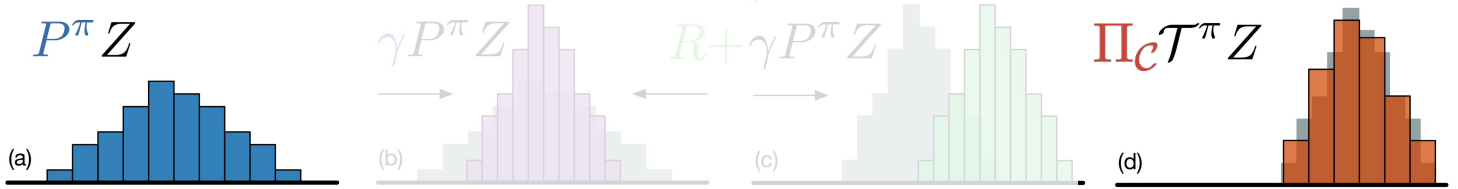
$$\mathbb{P}\{Z^\pi(x,a) \leq z_k\}$$



Loss

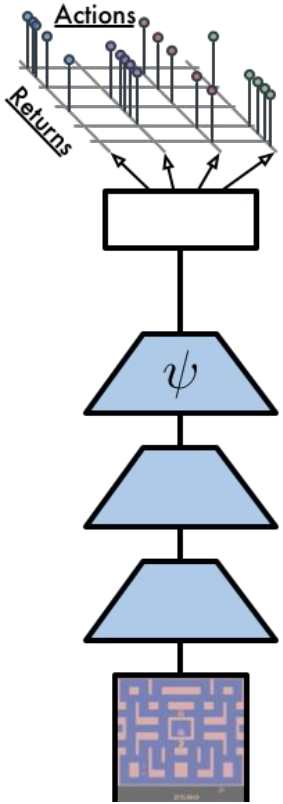
$$\Pi_c \mathcal{T}^\pi Z \| Z$$





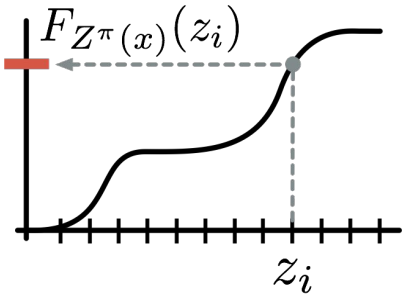
DeepMind

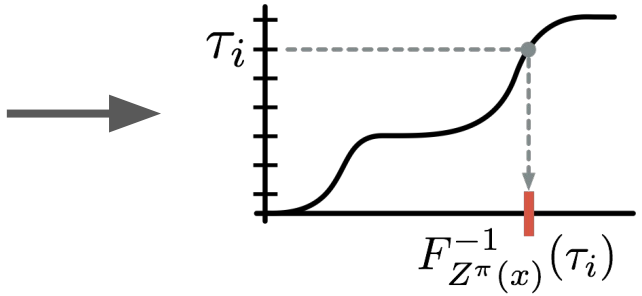
Quantile Distributional RL



Targets

Loss



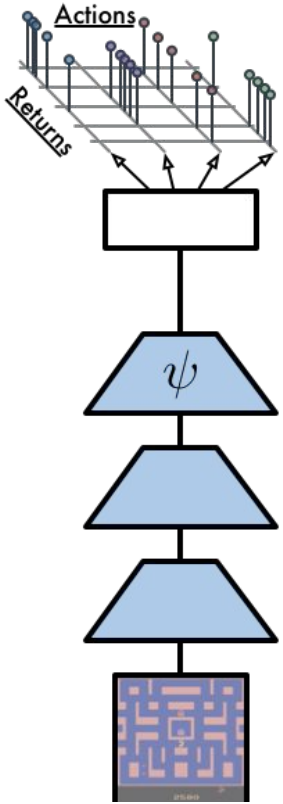


Can also do the inverse: Estimate the return values for given probability thresholds = distribution **quantiles**

Dabney, Will, et al. "Distributional reinforcement learning with quantile regression." *AAAI*, 2018.

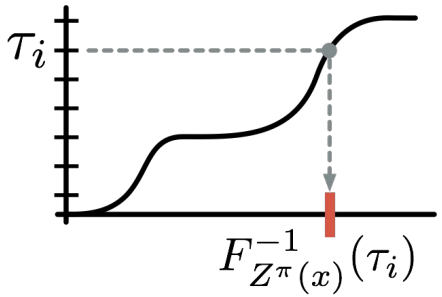
DeepMind

Quantile Distributional RL



Targets

$$F^{-1}_{Z^\pi(x)}(\tau_k)$$



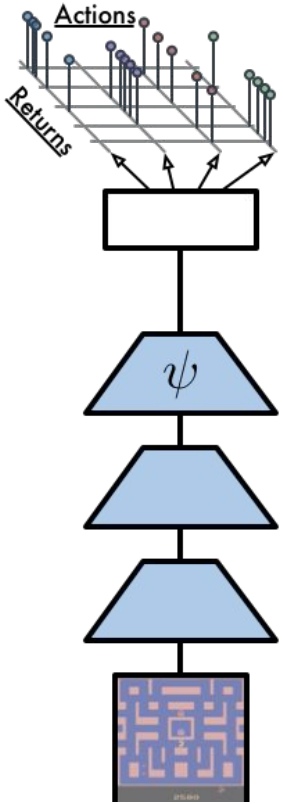
Can also do the inverse: Estimate the return values for given probability thresholds = distribution **quantiles**

Loss

Dabney, Will, et al. "Distributional reinforcement learning with quantile regression." *AAAI*, 2018.

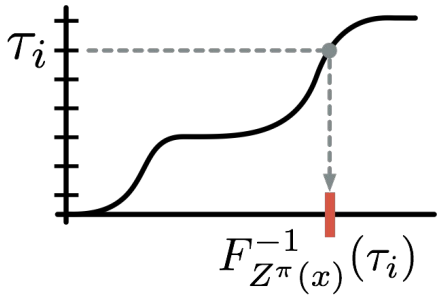
DeepMind

Quantile Distributional RL



Targets

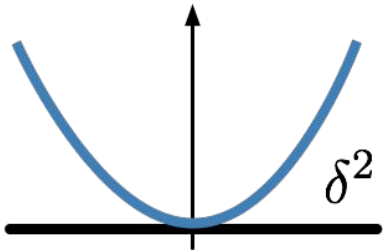
$$F^{-1}_{Z^\pi(x)}(\tau_k)$$



Can also do the inverse: Estimate the return values for given probability thresholds = distribution **quantiles**

Loss

$$\mathcal{T}^\pi Z - Z$$



Dabney, Will, et al. "Distributional reinforcement learning with quantile regression." *AAAI,* 2018.

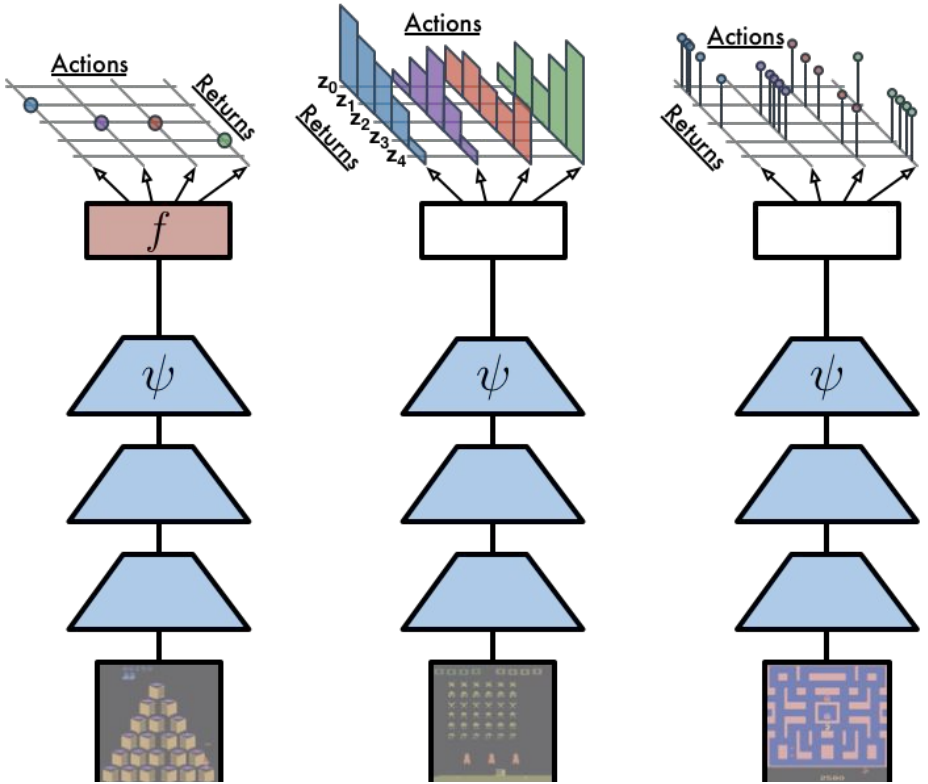
DeepMind

Distributional RL

- Algorithms that learn statistics of the return beyond the mean:

$$Z^\pi(x,a) = \sum_{t=0}^{\infty} \gamma^t R_{t+1}$$

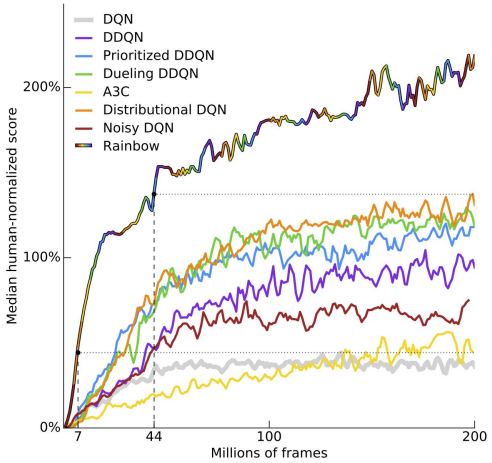
- A particular family of auxiliary tasks

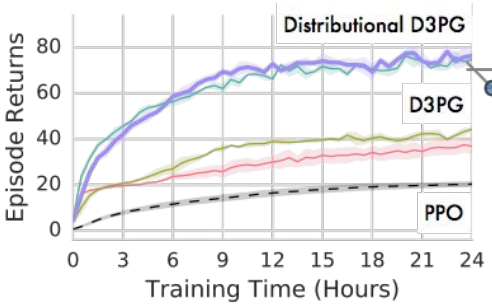


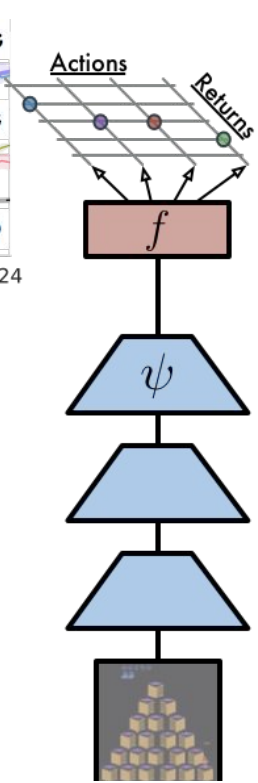
DeepMind

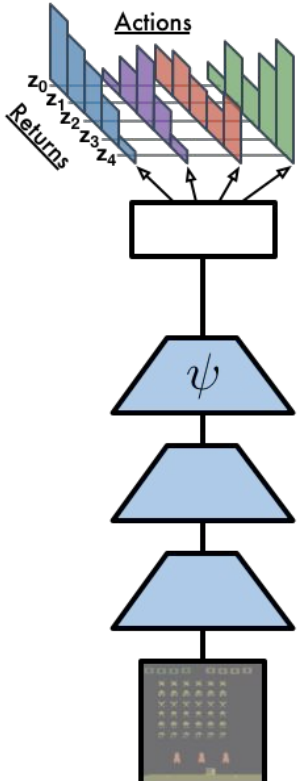
Bellemare, Marc G., Will Dabney, and Rémi Munos. "A distributional perspective on reinforcement learning." *ICML,* 2017.
Dabney, Will, et al. "Distributional reinforcement learning with quantile regression." *AAAI,* 2018.

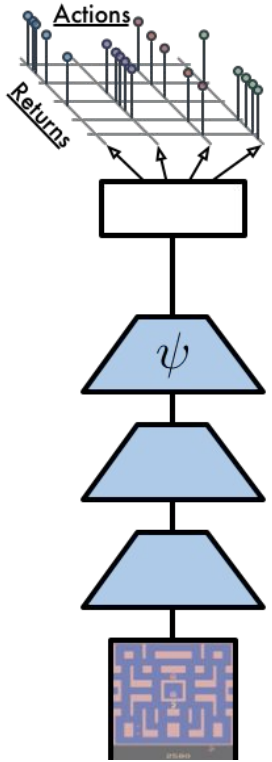
Distributional RL





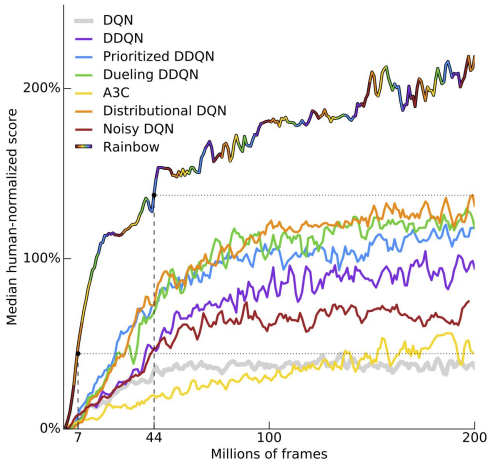


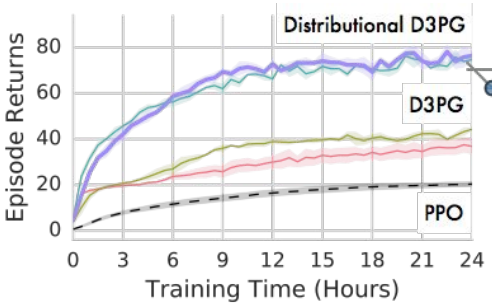


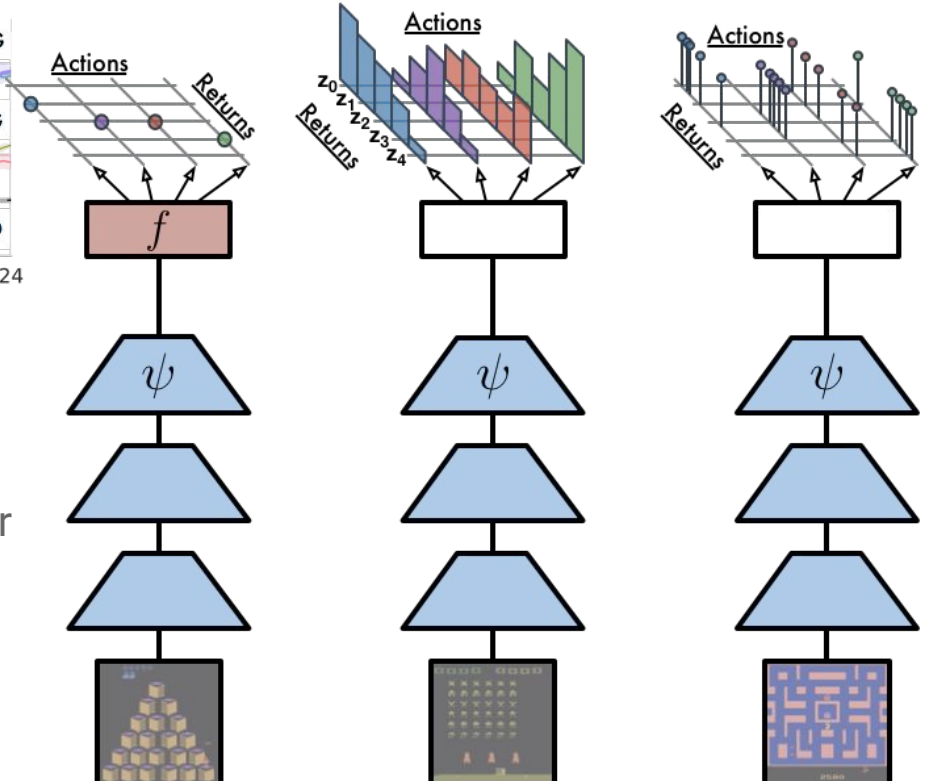


DeepMind

Distributional RL







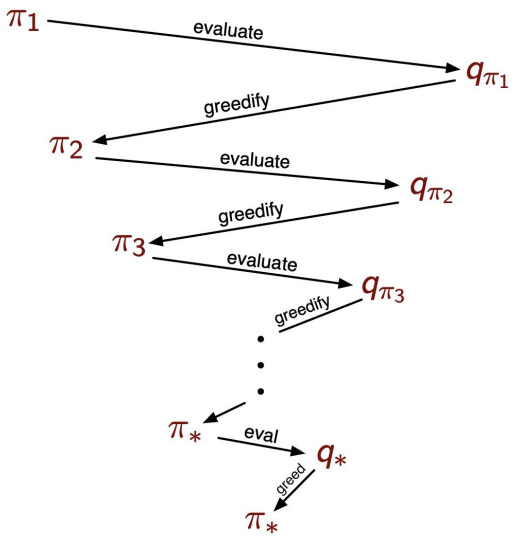
- **Why** does it help?
  - We still act w.r.t. the mean
  - Doesn't affect anything in the tabular case
- **Hypothesis:** return statistics are particularly good auxiliary tasks

DeepMind

Lyle, Clare, Marc G. Bellemare, and Pablo Samuel Castro. "A comparative analysis of expected and distributional reinforcement learning." *AAAI,* 2019.

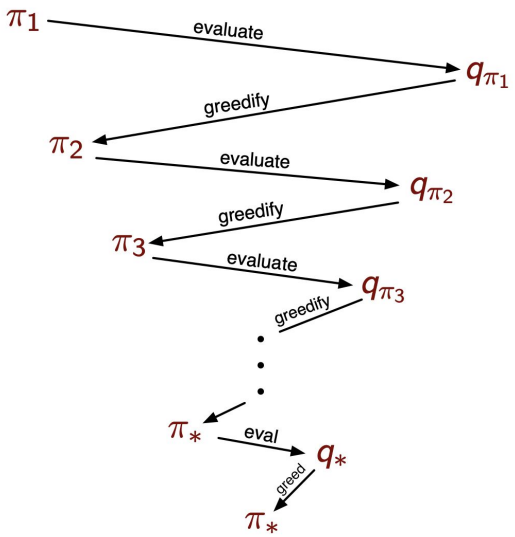
Value functions are temporarily useful

Value functions are temporarily useful



“The dance of policy and value”, cf. Rich

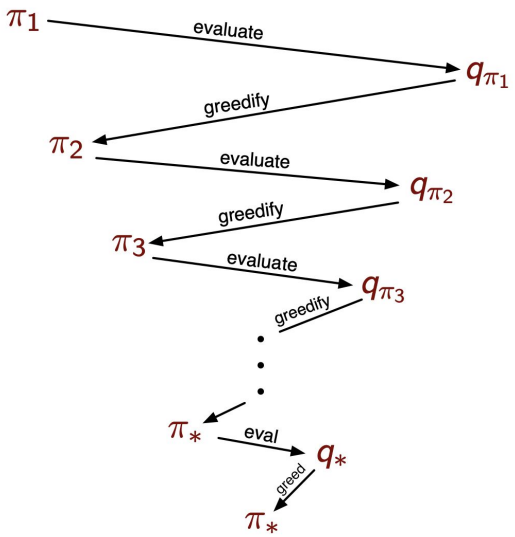
Value functions are temporarily useful



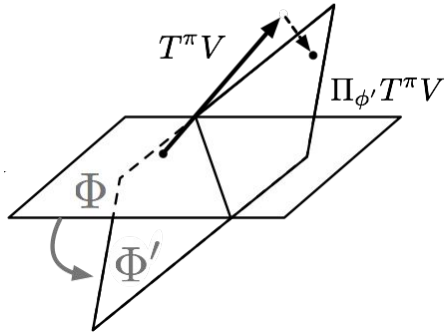
- The policy iteration dance is particularly treacherous in deep RL

“The dance of policy and value”, cf. Rich

Value functions are temporarily useful

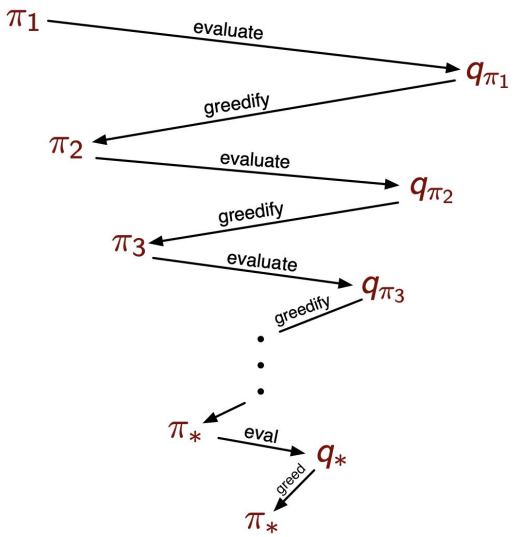


- The policy iteration dance is particularly treacherous in deep RL
- Every new policy/value wants its own set of features!



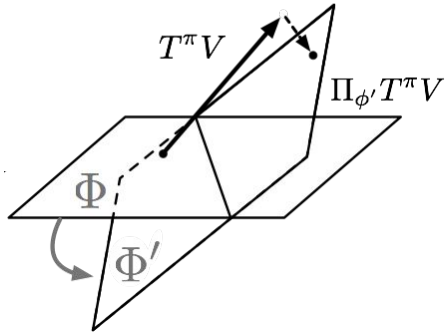
“The dance of policy and value”, cf. Rich

Value functions are temporarily useful



"The dance of policy and value", cf. Rich

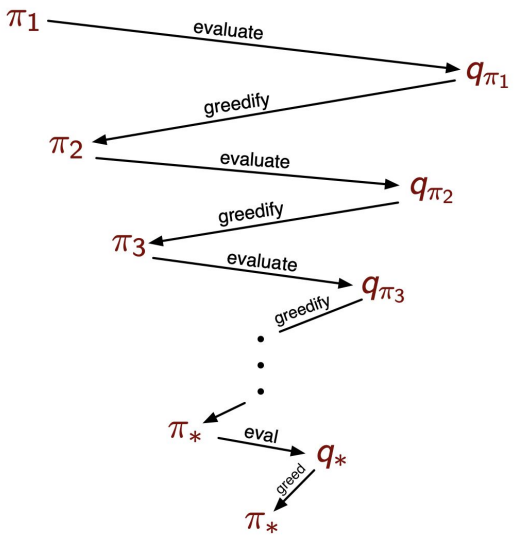
- The policy iteration dance is particularly treacherous in deep RL
- Every new policy/value wants its own set of features!



- There is a risk that the features overfit and are **unable** to learn the future optimal policy

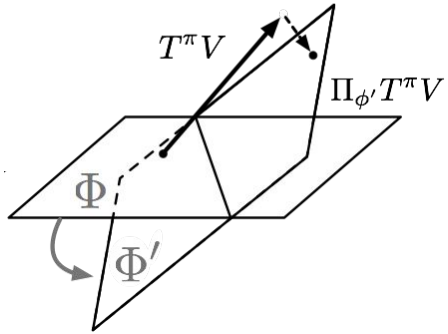
DeepMind

Value functions are temporarily useful



"The dance of policy and value", cf. Rich

- The policy iteration dance is particularly treacherous in deep RL
- Every new policy/value wants its own set of features!

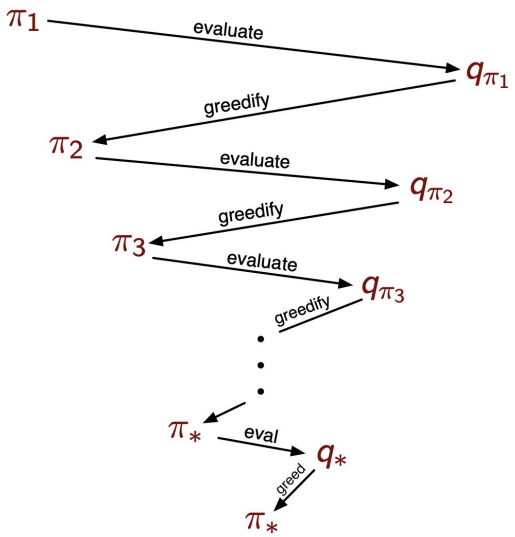


- There is a risk that the features overfit and are **unable** to learn the future optimal policy

Need to generalize across **time** as well

DeepMind

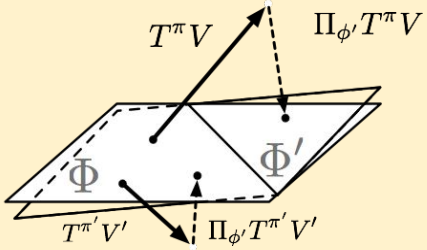
Value functions are temporarily useful



"The dance of policy and value", cf. Rich

- The policy iteration dance is particularly treacherous in deep RL
- Every new policy/value wants its own set of features!

Auxiliary tasks?



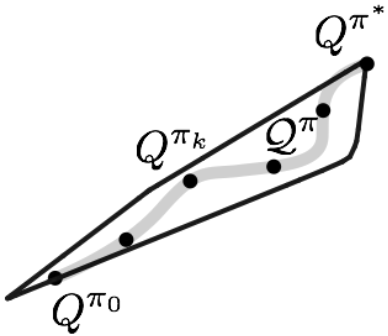
- There is a risk that the features overfit and are **unable** to learn the future optimal policy

Need to generalize across **time** as well

DeepMind

Value improvement path

Value-Improvement Path

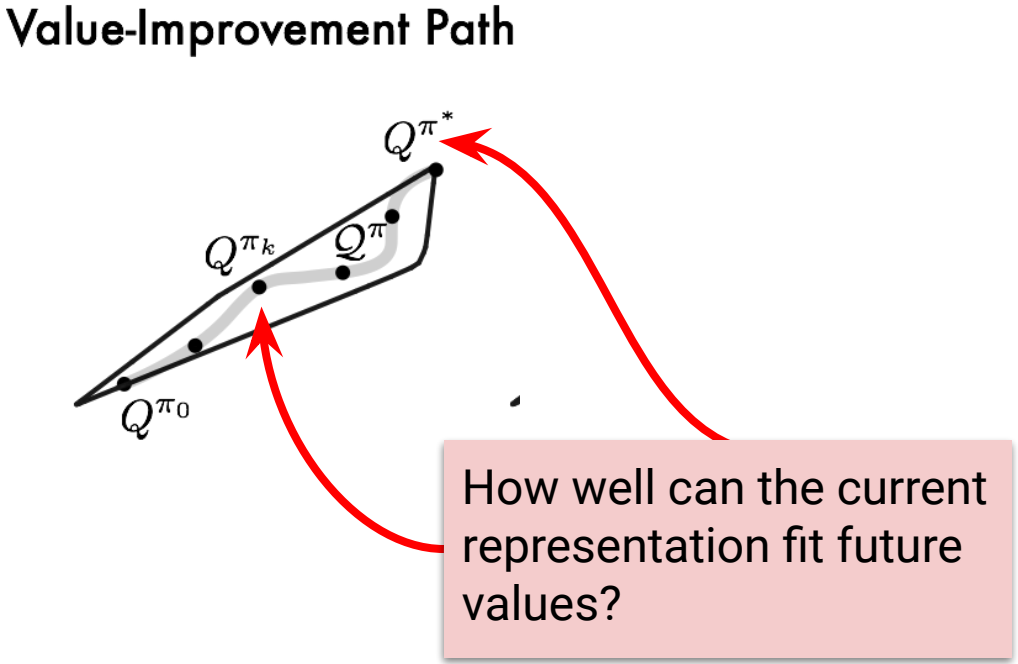


Value function polytope = space of possible value functions w.r.t. the reward

DeepMind

Dabney, Will, et al.. "The value improvement path: understanding representation for reinforcement learning." In submission.

Value improvement path



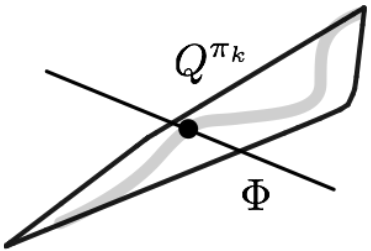
DeepMind

Dabney, Will, et al.. "The value improvement path: understanding representation for reinforcement learning." In submission.

Value improvement path

Value-Improvement Path

Value-Only



DeepMind

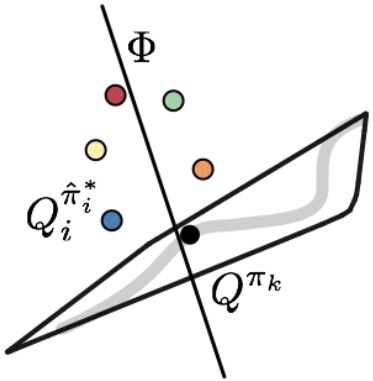
Dabney, Will, et al.. "The value improvement path: understanding representation for reinforcement learning." In submission.

Value improvement path

Value-Improvement Path

Value-Only

Cumulant Value



Auxiliary tasks / cumulants can help, but it's arbitrary if they're aligned correctly

DeepMind

Dabney, Will, et al.. "The value improvement path: understanding representation for reinforcement learning." In submission.

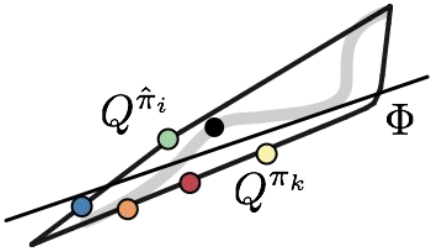
Value improvement path

Value-Improvement Path

Value-Only

Cumulant Value

Cumulant Policy



When evaluated w.r.t. the same reward, they do lie on the polytope, so it's more likely

DeepMind

Dabney, Will, et al.. "The value improvement path: understanding representation for reinforcement learning." In submission.

Value improvement path

Path | Value-Only | Cumulant Value | Cumulant Policy | **Past Policies**

Can explicitly ask to continue accommodating values of past policies

DeepMind

Dabney, Will, et al.. "The value improvement path: understanding representation for reinforcement learning." In submission.

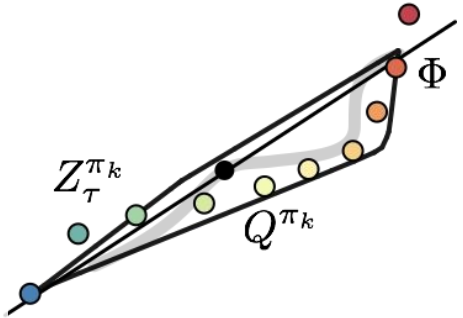
Value improvement path

Cumulant Value

Cumulant Policy

Past Policies

Distributional RL

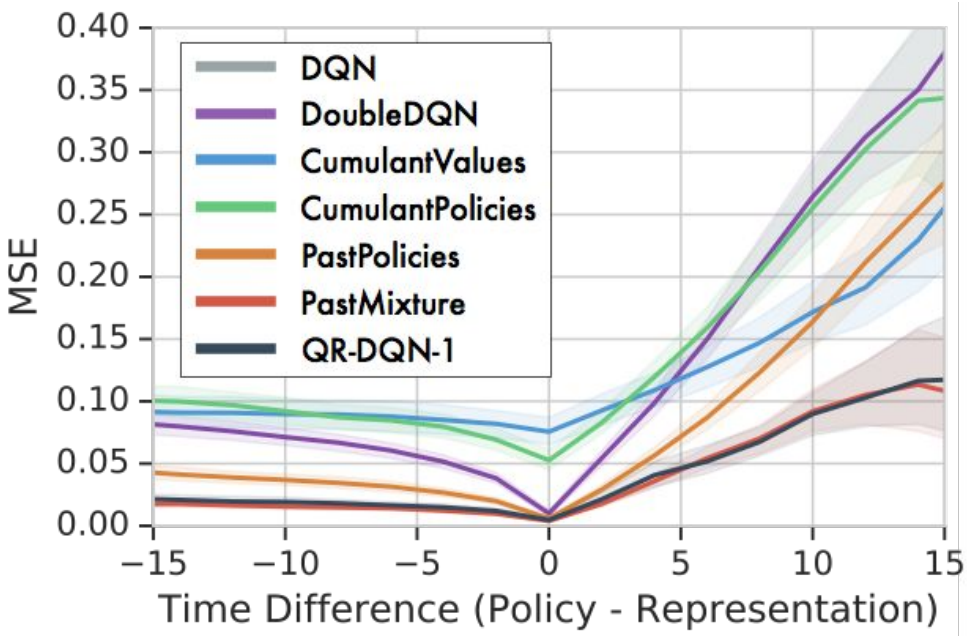


Distributional RL might be estimating statistics that are well aligned with value improvement

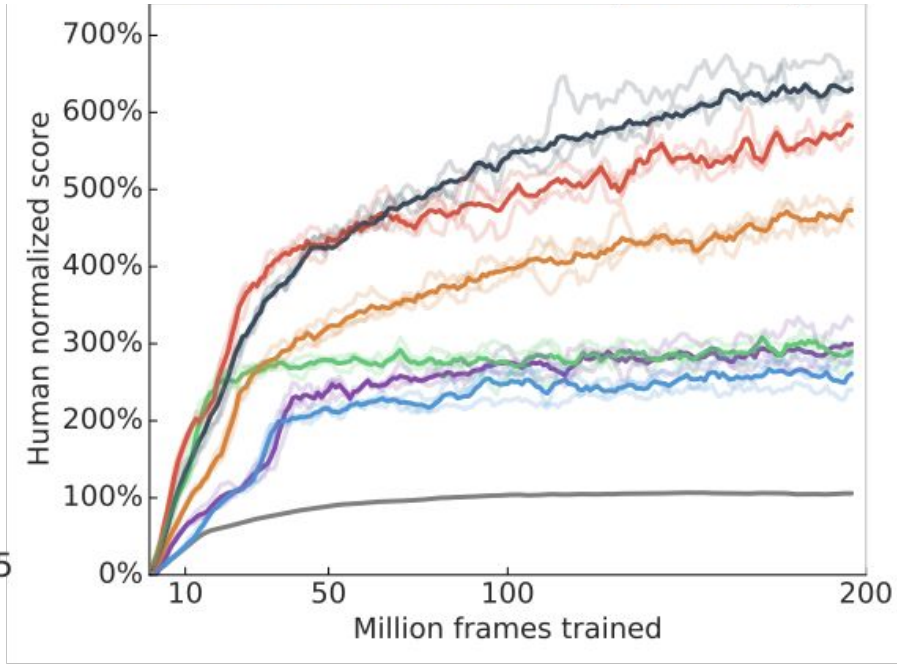
DeepMind

Dabney, Will, et al.. "The value improvement path: understanding representation for reinforcement learning." In submission.

Generalization error



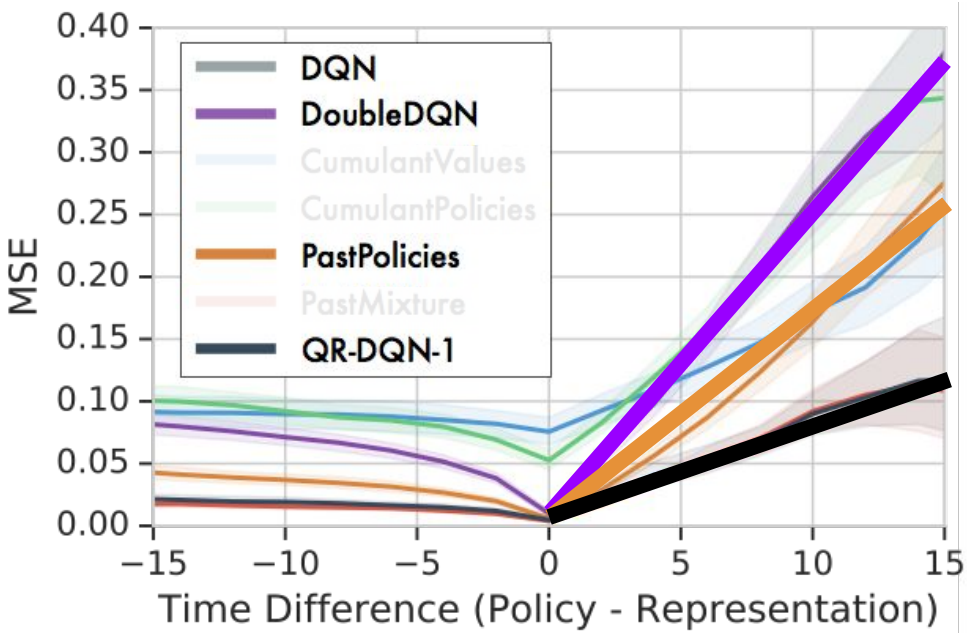
Performance (mean)



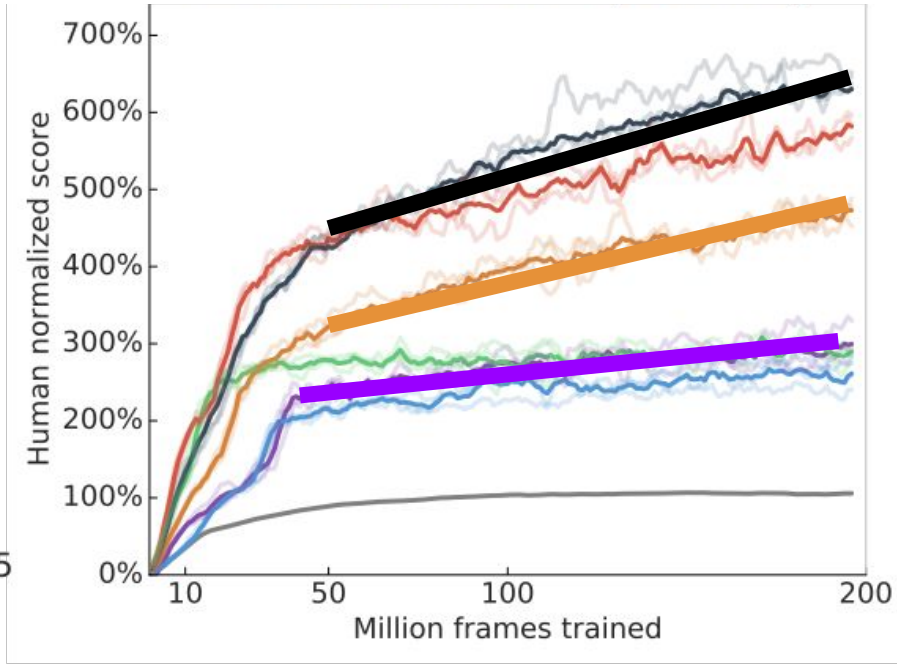
DeepMind

Dabney, Will, et al.. "The value improvement path: understanding representation for reinforcement learning." In submission.

Generalization error



Performance (mean)



DeepMind

Dabney, Will, et al.. "The value improvement path: understanding representation for reinforcement learning." *In submission.*

Recap

- Generalizing to new observations
  - **Architecture** + **loss** determine what features are learned
- Auxiliary tasks
  - **Issue 1:** The RL loss based on reward might be uninformative,
  - So let's learn other things -- pixels, features, etc
- Distributional RL
  - One type of auxiliary tasks is to learn various statistics of return
  - It helps, but we don't quite know why
- Value improvement path
  - **Issue 2:** Policies change! Past values should fit future policies
  - Perhaps distributional RL / aux tasks help this kind of generalization across time

2) Generalizing across tasks

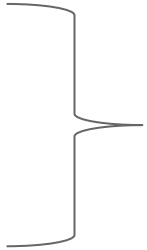
Contents

- Successor representation
- Successor features
- Universal Value Function Approximators (UVFAs)
- Generalized Policy Iteration (GPI)
- Universal Successor Features (USFAs)

Contents

- Successor representation

1) Successor features
2) Universal Value Function Approximators (UVFAs)
3) Generalized Policy Iteration (GPI)



USFAs

Successor representation

DeepMind

Dayan, Peter. "Improving generalization for temporal difference learning: The successor representation." *Neural Computation* (1993)

Successor representation

Value function

$$q^{\pi}(x,a) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} | X_0 = x, A_0 = a\right]$$

DeepMind

Dayan, Peter. "Improving generalization for temporal difference learning: The successor representation." *Neural Computation* (1993)

Successor representation

Value function

$$q^{\pi}(x,a) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} | X_0 = x, A_0 = a\right]$$

Successor representation

$$s^{\pi}(x,a,x') = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t \mathbb{I}_{X_t = x'} | X_0 = x, A_0 = a\right]$$

DeepMind

Dayan, Peter. "Improving generalization for temporal difference learning: The successor representation." *Neural Computation* (1993)

Successor representation

Value function

$$q^{\pi}(x, a) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} | X_0 = x, A_0 = a\right]$$

Successor representation

$$s^{\pi}(x, a, x') = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t \mathbb{I}_{X_t = x'} | X_0 = x, A_0 = a\right]$$

DeepMind

Dayan, Peter. "Improving generalization for temporal difference learning: The successor representation." *Neural Computation* (1993)

Successor representation

Value function

$$q^{\pi}(x,a) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} | X_0 = x, A_0 = a\right]$$

Successor representation

$$s^{\pi}(x,a,x') = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t \mathbb{I}_{X_t = x'} | X_0 = x, A_0 = a\right]$$

= how much time are we expected to spend in x' from x,a under π

DeepMind

Dayan, Peter. "Improving generalization for temporal difference learning: The successor representation." *Neural Computation* (1993)

Successor representation

Value function

$$q^{\pi}(x,a) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} | X_0 = x, A_0 = a\right]$$

$$= \sum_{x' \in \mathcal{X}} s^{\pi}(x,a,x') r^{\pi}(x')$$

Successor representation

$$s^{\pi}(x,a,x') = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t \mathbb{I}_{X_t = x'} | X_0 = x, A_0 = a\right]$$

= how much time are we expected to spend in x' from x,a under π

DeepMind

Dayan, Peter. "Improving generalization for temporal difference learning: The successor representation." *Neural Computation* (1993)

Successor representation

Value function

$$q^{\pi}(x,a) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} | X_0 = x, A_0 = a\right]$$

$$= \sum_{x' \in \mathcal{X}} s^{\pi}(x,a,x') r^{\pi}(x')$$

How much time in x'?

How much reward in x'?

Successor representation

$$s^{\pi}(x,a,x') = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t \mathbb{I}_{X_t = x'} | X_0 = x, A_0 = a\right]$$

= how much time are we expected to spend in x' from x,a under π

DeepMind

Dayan, Peter. "Improving generalization for temporal difference learning: The successor representation." *Neural Computation* (1993)

Successor representation

Value function

$$q^{\pi}(x,a) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}\gamma^{t}R_{t+1}|X_0 = x, A_0 = a\right]$$

$$= \sum_{x' \in \mathcal{X}} s^{\pi}(x,a,x')r^{\pi}(x')$$

Reward independent!

DeepMind

Dayan, Peter. "Improving generalization for temporal difference learning: The successor representation." *Neural Computation* (1993)

Successor representation

Value function

$$q^{\pi}(x,a) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}\gamma^{t}R_{t+1}|X_0 = x, A_0 = a\right]$$

$$= \sum_{x' \in \mathcal{X}} s^{\pi}(x,a,x') \begin{matrix} r_1^{\pi}(x') \\ \vdots \\ r_k^{\pi}(x') \end{matrix}$$

Reward independent!

If we have s, we can instantly evaluate any cumulant!

DeepMind

Dayan, Peter. "Improving generalization for temporal difference learning: The successor representation." *Neural Computation* (1993)

Successor representation

Value function

$$q^{\pi}(x,a) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} | X_0 = x, A_0 = a\right]$$

$$= \sum_{x' \in \mathcal{X}} s^{\pi}(x,a,x') \begin{bmatrix} r_1^{\pi}(x') \\ \vdots \\ r_k^{\pi}(x') \end{bmatrix}$$

Problem:
Quickly evaluate **new** reward functions w.r.t. the **same** policy

Reward independent!

If we have s, we can instantly evaluate any cumulant!

DeepMind

Dayan, Peter. "Improving generalization for temporal difference learning: The successor representation." *Neural Computation* (1993)

How to use with function approximation?

1) Successor features

Assume the reward is linear in some features:

$$r(x, a) = \phi(x, a)^T \mathbf{w} \qquad R_{t+1} = \phi_t^T \mathbf{w}$$

DeepMind

Barreto, André, et al. "Successor features for transfer in reinforcement learning." *NeurIPS,* 2017.

1) Successor features

Assume the reward is linear in some features:

$$r(x, a) = \phi(x, a)^T \mathbf{w} \qquad R_{t+1} = \phi_t^T \mathbf{w}$$

$$q^\pi(x, a) = \mathbb{E}_\pi \Big[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \Big]$$

DeepMind

Barreto, André, et al. "Successor features for transfer in reinforcement learning." *NeurIPS,* 2017.

1) Successor features

Assume the reward is linear in some features:

$$r(x, a) = \phi(x, a)^T \mathbf{w} \qquad R_{t+1} = \phi_t^T \mathbf{w}$$

$$q^\pi(x, a) = \mathbb{E}_\pi \Big[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \Big] = \mathbb{E}_\pi \Big[\sum_{t=0}^{\infty} \gamma^t \phi_t^T \mathbf{w} \Big]$$

DeepMind

Barreto, André, et al. "Successor features for transfer in reinforcement learning." *NeurIPS,* 2017.

1) Successor features

Assume the reward is linear in some features:

$$r(x,a) = \phi(x,a)^T \mathbf{w} \qquad R_{t+1} = \phi_t^T \mathbf{w}$$

Linearity

$$q^\pi(x,a) = \mathbb{E}_\pi\Big[\sum_{t=0}^{\infty} \gamma^t R_{t+1}\Big] = \mathbb{E}_\pi\Big[\sum_{t=0}^{\infty} \gamma^t \phi_t^T \mathbf{w}\Big] = \mathbb{E}_\pi\Big[\sum_{t=0}^{\infty} \gamma^t \phi_t\Big]^T \mathbf{w}$$

DeepMind

Barreto, André, et al. "Successor features for transfer in reinforcement learning." *NeurIPS,* 2017.

1) Successor features

Assume the reward is linear in some features:

$$r(x,a) = \phi(x,a)^T \mathbf{w} \qquad R_{t+1} = \phi_t^T \mathbf{w}$$

Linearity

$$q^\pi(x,a) = \mathbb{E}_\pi\Big[\sum_{t=0}^{\infty} \gamma^t R_{t+1}\Big] = \mathbb{E}_\pi\Big[\sum_{t=0}^{\infty} \gamma^t \phi_t^T \mathbf{w}\Big] = \mathbb{E}_\pi\Big[\sum_{t=0}^{\infty} \gamma^t \phi_t\Big]^T \mathbf{w}$$

= expected cumulative reward features from x,a

DeepMind

Barreto, André, et al. "Successor features for transfer in reinforcement learning." *NeurIPS,* 2017.

1) Successor features

Assume the reward is linear in some features:

$$r(x,a) = \phi(x,a)^T \mathbf{w} \qquad R_{t+1} = \phi_t^T \mathbf{w}$$

Linearity

$$q^\pi(x,a) = \mathbb{E}_\pi\Big[\sum_{t=0}^{\infty} \gamma^t R_{t+1}\Big] = \mathbb{E}_\pi\Big[\sum_{t=0}^{\infty} \gamma^t \phi_t^T \mathbf{w}\Big] = \mathbb{E}_\pi\Big[\sum_{t=0}^{\infty} \gamma^t \phi_t\Big]^T \mathbf{w}$$

$$= \psi^\pi(x,a)^T \mathbf{w}$$

= expected cumulative reward features from x,a

Successor features!

DeepMind

Barreto, André, et al. "Successor features for transfer in reinforcement learning." *NeurIPS,* 2017.

1) Successor features

Assume the reward is linear in some features:

$$r(x,a) = \phi(x,a)^T \mathbf{w} \qquad R_{t+1} = \phi_t^T \mathbf{w}$$

Linearity

$$q^\pi(x,a) = \mathbb{E}_\pi\Big[\sum_{t=0}^{\infty} \gamma^t R_{t+1}\Big] = \mathbb{E}_\pi\Big[\sum_{t=0}^{\infty} \gamma^t \phi_t^T \mathbf{w}\Big] = \mathbb{E}_\pi\Big[\sum_{t=0}^{\infty} \gamma^t \phi_t\Big]^T \mathbf{w}$$

$$= \psi^\pi(x,a)^T \begin{matrix} \mathbf{w}_1 \\ \bullet \\ \bullet \\ \bullet \\ \mathbf{w}_k \end{matrix}$$

= expected cumulative reward features from x,a

Successor features!

If we have ψ, we can instantly evaluate any reward parameters **w**!

DeepMind

Barreto, André, et al. "Successor features for transfer in reinforcement learning." *NeurIPS,* 2017.

1) Successor features

Problem:
Quickly evaluate **new** reward *parameters* w.r.t. the **same** policy

$$r(x,a) = \phi(x,a)^T \mathbf{w} \qquad R_{t+1} = \phi_t^T \mathbf{w}$$

Linearity

$$q^\pi(x,a) = \mathbb{E}_\pi\Big[\sum_{t=0}^{\infty} \gamma^t R_{t+1}\Big] = \mathbb{E}_\pi\Big[\sum_{t=0}^{\infty} \gamma^t \phi_t^T \mathbf{w}\Big] = \mathbb{E}_\pi\Big[\sum_{t=0}^{\infty} \gamma^t \phi_t\Big]^T \mathbf{w}$$

= expected cumulative reward features from x,a

$$= \psi^\pi(x,a)^T \begin{matrix} \mathbf{w}_1 \\ \bullet \\ \bullet \\ \bullet \\ \mathbf{w}_k \end{matrix}$$

Successor features!

If we have ψ, we can instantly evaluate any reward parameters **w**!

DeepMind

Barreto, André, et al. "Successor features for transfer in reinforcement learning." *NeurIPS,* 2017.

Learning SFs

$$\hat{q}^{\pi}(x, a) = \hat{\psi}^{\pi}(x, a)^{T} \hat{\mathbf{w}}$$

DeepMind

Barreto, André, et al. "Successor features for transfer in reinforcement learning." *NeurIPS,* 2017.

Learning SFs

$$\hat{q}^{\pi}(x, a) = \hat{\psi}^{\pi}(x, a)^{T} \hat{\mathbf{w}}$$

DeepMind

Barreto, André, et al. "Successor features for transfer in reinforcement learning." *NeurIPS,* 2017.

Learning SFs

$$\hat{q}^{\pi}(x,a) = \hat{\psi}^{\pi}(x,a)^{T}\hat{\mathbf{w}}$$

Regression

$$r(x,a) - \phi(x,a)^{T}\hat{\mathbf{w}}$$

DeepMind

Barreto, André, et al. "Successor features for transfer in reinforcement learning." *NeurIPS,* 2017.

Learning SFs

$$\hat{q}^{\pi}(x,a) = \hat{\psi}^{\pi}(x,a)^{T}\hat{\mathbf{w}}$$

RL

$$\phi + \gamma\hat{\psi}' - \hat{\psi}$$

Regression

$$r(x,a) - \phi(x,a)^{T}\hat{\mathbf{w}}$$

DeepMind

Barreto, André, et al. "Successor features for transfer in reinforcement learning." *NeurIPS,* 2017.

Learning SFs

$$\hat{q}^{\pi}(x,a) = \hat{\psi}^{\pi}(x,a)^{T}\hat{\mathbf{w}}$$

RL

$$\phi + \gamma\hat{\psi}' - \hat{\psi}$$

Regression

$$r(x,a) - \phi(x,a)^{T}\hat{\mathbf{w}}$$

Features φ also can be learned
(but are usually given)

DeepMind

Barreto, André, et al. "Successor features for transfer in reinforcement learning." *NeurIPS,* 2017.

Learning SFs

$$\hat{q}^{\pi}(x,a) = \hat{\psi}^{\pi}(x,a)^{T}\hat{\mathbf{w}}$$

RL

$$\phi + \gamma\hat{\psi}' - \hat{\psi}$$

Regression

$$r(x,a) - \phi(x,a)^{T}\hat{\mathbf{w}}$$

- Zero-shot (and good quality!) evaluation

- Policy dependent
- Reward linearity assumption

DeepMind

Barreto, André, et al. "Successor features for transfer in reinforcement learning." *NeurIPS,* 2017.

2) Universal Value Function Approximators

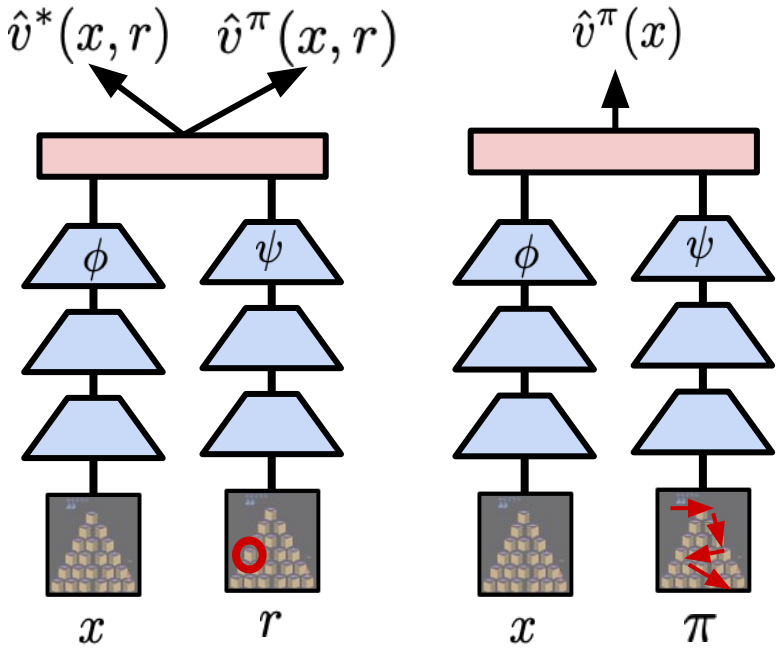
Problem:
Generalize in the space of value functions to **new** tasks, policies, rewards, etc

Problem:
Quickly evaluate **new** reward *parameters* w.r.t. the **same** policy

2) Universal Value Function Approximators

Problem:
Generalize in the space of value functions to **new** tasks, policies, rewards, etc

- A **UVFA** takes the extra conditioning as an input and learns a joint representation



Tasks / Rewards Policies

DeepMind

Schaul, Tom, et al. "Universal value function approximators." *ICML*. 2015.

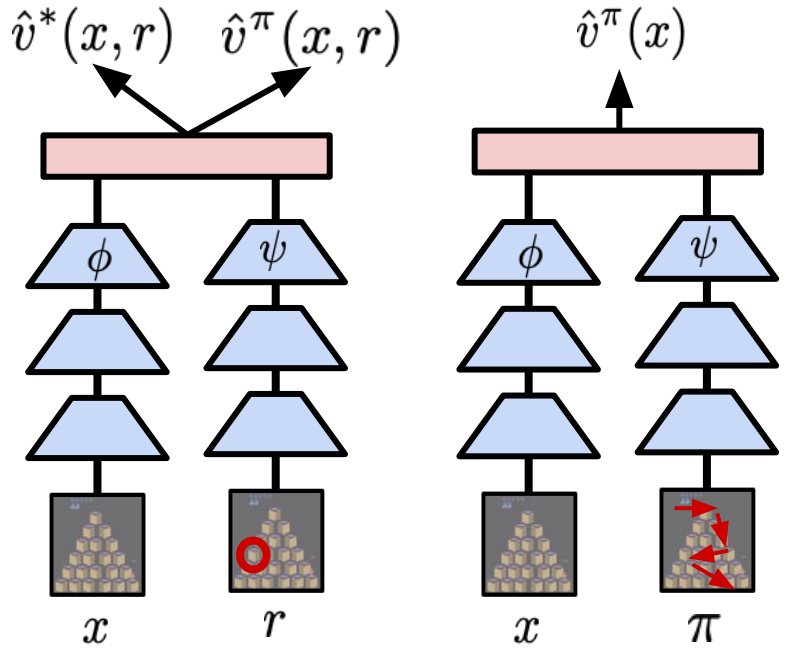
2) Universal Value Function Approximators

Problem:
Generalize in the space of value functions to **new** tasks, policies, rewards, etc

- A **UVFA** takes the extra conditioning as an input and learns a joint representation

- Zero-shot, cheap generalization

- Hard work for the network



Tasks / Rewards Policies

DeepMind

Schaul, Tom, et al. "Universal value function approximators." *ICML*. 2015.

3) Generalized Policy Improvement (GPI)

Problem:
Given a set of existing policies, solve an arbitrary new task

Problem:
Generalize in the space of value functions to **new** tasks, policies, rewards, etc

DeepMind

Barreto, André, et al. "Successor features for transfer in reinforcement learning." *NeurIPS,* 2017.

3) Generalized Policy Improvement (GPI)

Problem:
Given a set of existing policies, solve an arbitrary new task

Problem:
Generalize in the space of value functions to **new** tasks, policies, rewards, etc

- **Option 1:** Feed the new task to a UVFA, get an estimate of the value of the new policy interpolated from the values of existing policies

DeepMind

Barreto, André, et al. "Successor features for transfer in reinforcement learning." *NeurIPS,* 2017.

3) Generalized Policy Improvement (GPI)

Problem:
Given a set of existing policies, solve an arbitrary new task

Problem:
Generalize in the space of value functions to **new** tasks, policies, rewards, etc

- **Option 1:** Feed the new task to a UVFA, get an estimate of the value of the new policy interpolated from the values of existing policies
- **Option 2:** Use the semantics of our problem (RL!):

$$\pi(\cdot|x) = \arg\max_a \max_i q^{\pi_i}(x, a)$$

DeepMind

Barreto, André, et al. "Successor features for transfer in reinforcement learning." *NeurIPS,* 2017.

3) Generalized Policy Improvement (GPI)

Problem:
Given a set of existing policies, solve an arbitrary new task

Problem:
Generalize in the space of value functions to **new** tasks, policies, rewards, etc

- **Option 1:** Feed the new task to a UVFA, get an estimate of the value of the new policy interpolated from the values of existing policies
- **Option 2:** Use the semantics of our problem (RL!):

$$\pi(\cdot|x) = \arg\max_a \max_i q^{\pi_i}(x, a)$$

- Theoretical guarantees -- better than any of the initial policies!

- Double-max might be expensive / unstable

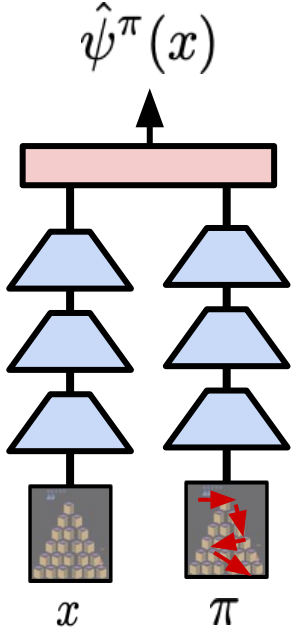
DeepMind

Barreto, André, et al. "Successor features for transfer in reinforcement learning." *NeurIPS,* 2017.

UVFAs vs. Successor Features

- Under the linear assumption, SFs give free accurate generalization to new **w**-s *(evaluation only)*
- How's this different from the way a UVFA generalizes?
  - UVFAs don't assume anything, but rely on the function approximator to figure it out
  - More general, but also a harder problem
- But the successor representation is **policy-dependent**
  - Have to learn a new one for every policy

- **Idea:** why not both! Use a UVFA to learn universal SFs!



Policies

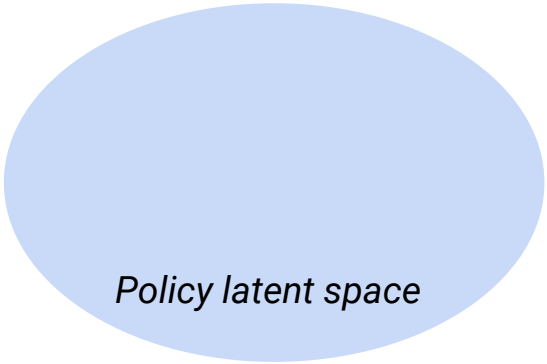
USFAs: How to Solve Anything

DeepMind

Borsa, Diana, et al. "Universal successor features approximators." *ICLR*, 2019.

USFAs: How to Solve Anything

1) **UVFA:** Get the successor features for many policies

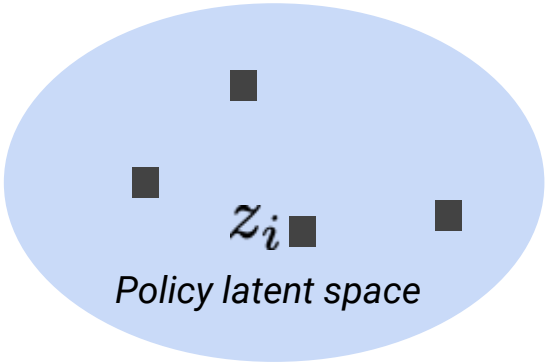


DeepMind

Borsa, Diana, et al. "Universal successor features approximators." *ICLR*, 2019.

USFAs: How to Solve Anything

1) **UVFA:** Get the successor features for many policies

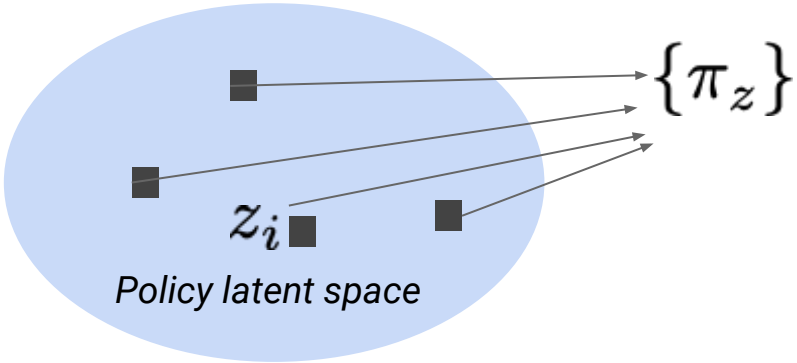


DeepMind

Borsa, Diana, et al. "Universal successor features approximators." *ICLR*, 2019.

USFAs: How to Solve Anything

1) **UVFA:** Get the successor features for many policies

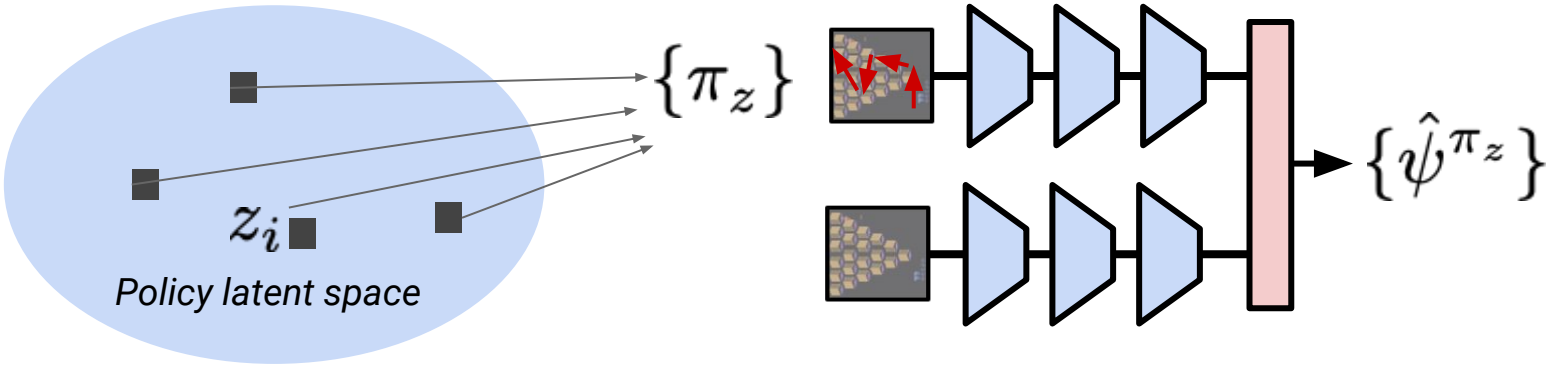


DeepMind

Borsa, Diana, et al. "Universal successor features approximators." *ICLR*, 2019.

USFAs: How to Solve Anything

1) **UVFA:** Get the successor features for many policies

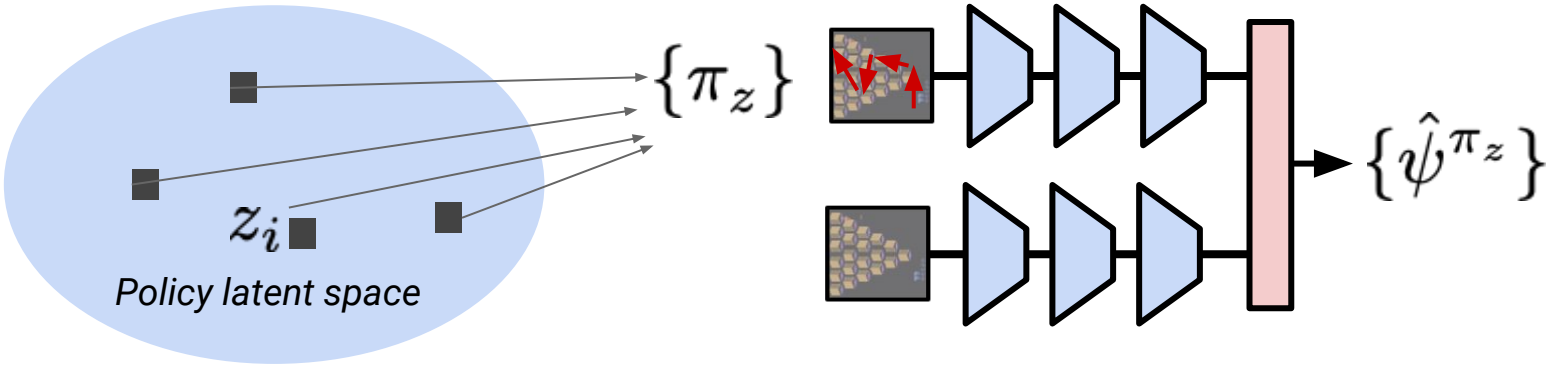


DeepMind

Borsa, Diana, et al. "Universal successor features approximators." *ICLR*, 2019.

USFAs: How to Solve Anything

1) **UVFA:** Get the successor features for many policies



2) **SF:** Evaluate the policies w.r.t. the reward of interest **w**, get Q-functions:

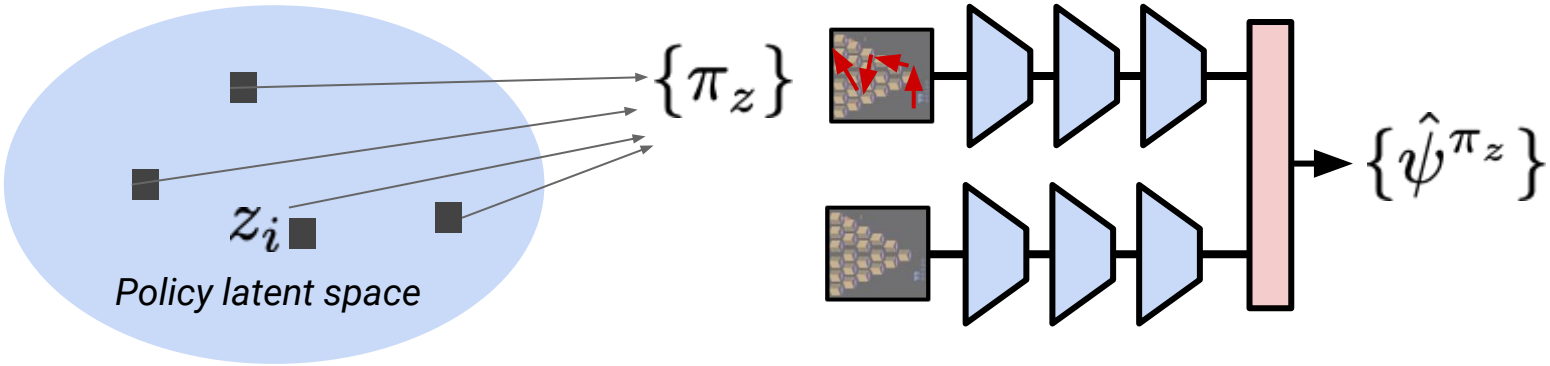
$$\{\hat{q}^{\pi_z}_{\mathbf{w}}(x,a) = \hat{\psi}^{\pi_z}(x,a)\mathbf{w}\}$$

DeepMind

Borsa, Diana, et al. "Universal successor features approximators." *ICLR*, 2019.

USFAs: How to Solve Anything

1) **UVFA:** Get the successor features for many policies



2) **SF:** Evaluate the policies w.r.t. the reward of interest **w**, get Q-functions:

$$\{\hat{q}^{\pi_z}_{\mathbf{w}}(x,a) = \hat{\psi}^{\pi_z}(x,a)\mathbf{w}\}$$

3) **GPI:** Recombine policies to do as well as possible:

$$\pi(\cdot|x) = \arg\max_a \max_z \hat{q}^{\pi_z}_{\mathbf{w}}(x,a)$$

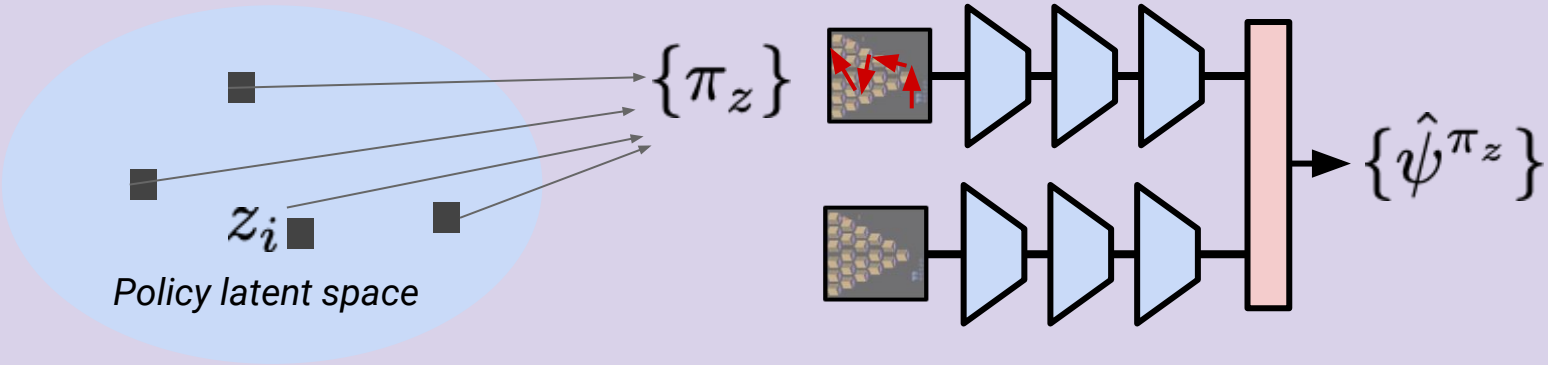
DeepMind

Borsa, Diana, et al. "Universal successor f
approximators." *ICLR*, 2019.

Two types of generalization

Parameteric

1) **UVFA:** Get the successor features for many policies



2) **SF:** Evaluate the policies w.r.t. the reward of interest **w**, get Q-functions:

$$\{\hat{q}^{\pi_z}_{\mathbf{w}}(x,a) = \hat{\psi}^{\pi_z}(x,a)\mathbf{w}\}$$

3) **GPI:** Recombine policies to do as well as possible:

$$\pi(\cdot|x) = \arg\max_a \max_z \hat{q}^{\pi_z}_{\mathbf{w}}(x,a)$$

Recomposition

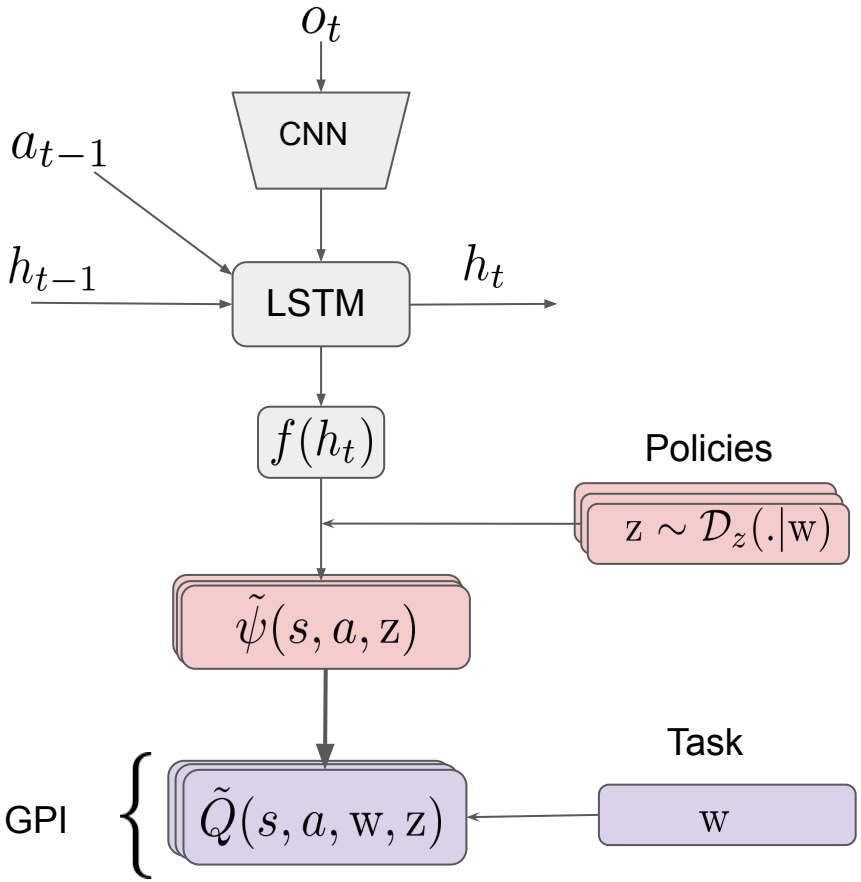
USFAs - Acting

- Sample in policy space:

$$\mathrm{z} \sim \mathcal{D}_z(.|\mathrm{w})$$

- Compute GPI policy:

$$\pi(s) = \arg\max_a \max_z \tilde{Q}(s, a, \mathrm{w}, \mathrm{z})$$



USFAs - Training

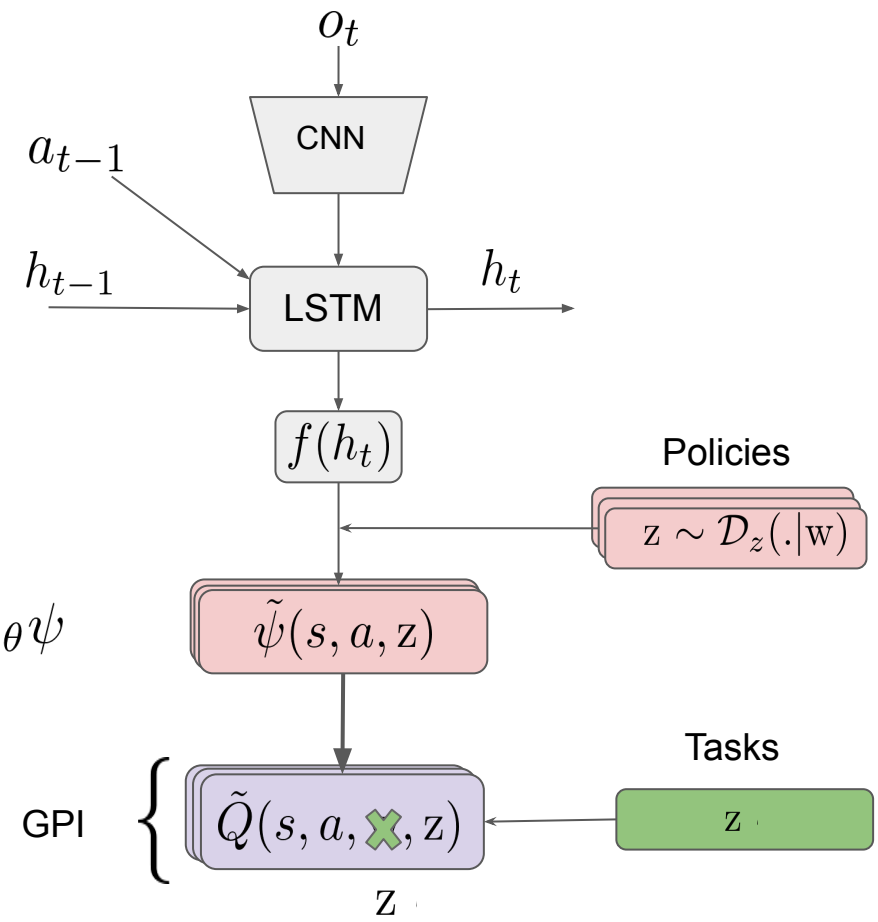
- Sample in policy space:

$$\mathrm{z} \sim \mathcal{D}_z(.|\mathrm{w})$$

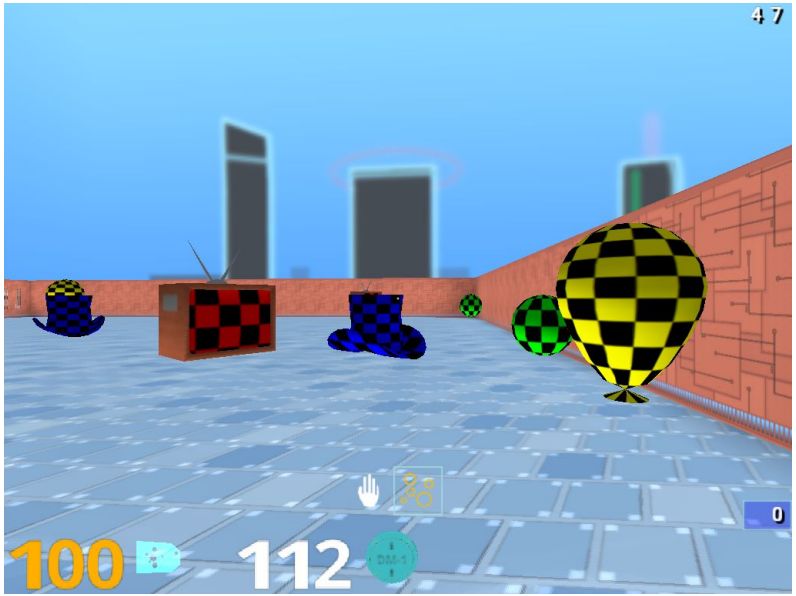
- **Pretend you're in task** z:

$$\theta \xleftarrow{\alpha} \left[\phi + \gamma\psi(s', \pi_{\mathbf{z}}, \mathbf{z}) - \psi(s, a, \mathbf{z})\right] \nabla_\theta \psi$$

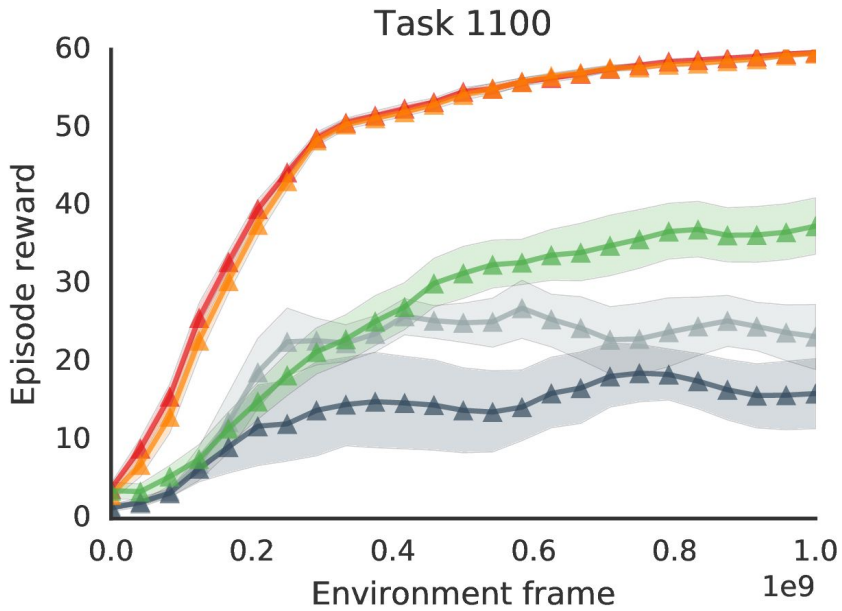
where: $\pi_{\mathbf{z}} = \arg\max_a Q_{\mathbf{z}}^{\pi_{\mathbf{z}}}$



Experiments: DMLab Treasure Island



Observation



UVFA
UVFA (offpolicy)
USFA (GPI over C = M + {w'})
USFA (GPI over C = M)
USFA (GPI over C = {w'})

Recap

Recap

- Successor representation
  - Future expected state visitations

Recap

- Successor representation
  - Future expected state visitations

Problem:
Quickly evaluate **new** reward functions w.r.t. the **same** policy

Recap

- Successor representation
  - Future expected state visitations
- Successor features
  - SR under the assumption of a linear reward decomposition

Problem:
Quickly evaluate **new** reward functions w.r.t. the **same** policy

Problem:
Quickly evaluate **new** reward *parameters* w.r.t. the **same** policy

Recap

- Successor representation
  - Future expected state visitations
- Successor features
  - SR under the assumption of a linear reward decomposition
- Universal Value Function Approximators (UVFAs)
  - Learn a joint value representation for an extra input

Problem:
Quickly evaluate **new** reward functions w.r.t. the **same** policy

Problem:
Quickly evaluate **new** reward *parameters* w.r.t. the **same** policy

Problem:
Generalize in the space of value functions to **new** policies, rewards, etc

Recap

- Successor representation
  - Future expected state visitations
- Successor features
  - SR under the assumption of a linear reward decomposition
- Universal Value Function Approximators (UVFAs)
  - Learn a joint value representation for an extra input
- Generalized Policy Iteration (GPI)
  - Recombine existing policies to a new task using RL principles (max-max)

Problem:
Quickly evaluate **new** reward functions w.r.t. the **same** policy

Problem:
Quickly evaluate **new** reward *parameters* w.r.t. the **same** policy

Problem:
Generalize in the space of value functions to **new** policies, rewards, etc

Problem:
Given a set of existing policies, solve an arbitrary new task

Recap

- Successor representation
  - Future expected state visitations
- Successor features
  - SR under the assumption of a linear reward decomposition
- Universal Value Function Approximators (UVFAs)
  - Learn a joint value representation for an extra input
- Generalized Policy Iteration (GPI)
  - Recombine existing policies to a new task using RL principles (max-max)
- Universal Successor Features (USFAs)
  - Elegantly combine previous components for knowledge acquisition and transfer

Problem:
Quickly evaluate **new** reward functions w.r.t. the **same** policy

Problem:
Quickly evaluate **new** reward *parameters* w.r.t. the **same** policy

Problem:
Generalize in the space of value functions to **new** policies, rewards, etc

Problem:
Given a set of existing policies, solve an arbitrary new task

Recap

- Successor representation
  - Future expected state visitations
- Successor features
  - SR under the assumption of a linear reward decomposition
- Universal Value Function Approximators (UVFAs)
  - Learn a joint value representation for an extra input
- Generalized Policy Iteration (GPI)
  - Recombine existing policies to a new task using RL principles (max-max)
- Universal Successor Features (USFAs)
  - Elegantly combine previous components for knowledge acquisition and transfer
- Different problems, different generalizations!
  - Parametric with UVFAs and SFs, RL with GPI

Problem:
Quickly evaluate **new** reward functions w.r.t. the **same** policy

Problem:
Quickly evaluate **new** reward *parameters* w.r.t. the **same** policy

Problem:
Generalize in the space of value functions to **new** policies, rewards, etc

Problem:
Given a set of existing policies, solve an arbitrary new task

What is an unseen example in RL?

Time

$$\mathcal{M} = (\mathcal{X}, \mathcal{A}, p, r, \gamma)$$

$$v^{\pi} | \pi, \mathcal{M}$$

$$\pi^{*} | \mathcal{M}$$

Observations
- Auxiliary tasks
- Distributional RL

Rewards
- Successor features

Anything
- Meta-RL
- Hierarchies

Dynamics
- Max entropy RL
- Empowerment

Policies
- UVFAs
- USFAs

DeepMind

Takeaways

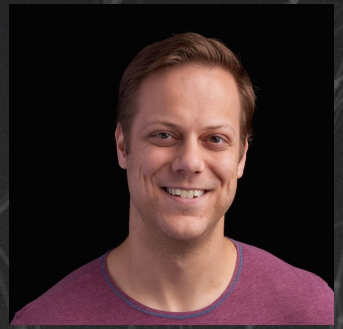
- Lots of **things** to generalize over in RL
  - Observations, rewards, policies, transitions, time, ...
- Lots of different **ways** to generalize with in deep RL
  - Pure parametric (like most ML), structured parametric, online recomposition using the structure of the RL problem, meta learning
- Always a tradeoff between **expressivity** and **sample complexity**
  - Related to the bitter lesson
  - Remain as general as possible, while keeping the structure of problem in mind
- The solutions might change, the problems are less likely to

Takeaways

- Lots of **things** to generalize over in RL
  - Observations, rewards, policies, transitions, time, ...
- Lots of different **ways** to generalize with in deep RL
  - Pure parametric (like most ML), structured parametric, online recomposition using the structure of the RL problem, meta learning
- Always a tradeoff between **expressivity** and **sample complexity**
  - Related to the bitter lesson
  - Remain as general as possible, while keeping the structure of problem in mind
- The solutions might change, the problems are less likely to

"Everything's exciting, nothing is solved!"

DeepMind





Thank you for your attention!

Questions?

DeepMind

References

Jaderberg, Max, et al. "Reinforcement learning with unsupervised auxiliary tasks." *ICLR*, 2017.

Sutton, Richard S., et al. "Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction." *AAMAS,* 2011.

Bellemare, Marc G., Will Dabney, and Rémi Munos. "A distributional perspective on reinforcement learning." *ICML,* 2017.

Dabney, Will, et al. "Distributional reinforcement learning with quantile regression." *AAAI,* 2018.

Dabney, Will, et al.. "The value improvement path: understanding representation for reinforcement learning." In submission.

Dayan, Peter. "Improving generalization for temporal difference learning: The successor representation." *Neural Computation* (1993)

Barreto, André, et al. "Successor features for transfer in reinforcement learning." *NeurIPS,* 2017.

Schaul, Tom, et al. "Universal value function approximators." *ICML*. 2015.

Borsa, Diana, et al. "Universal successor features approximators." *ICLR*, 2019.