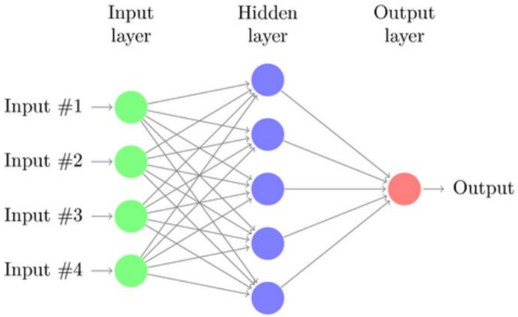
Model-Based RL

Roberto Capobianco

SAPIENZA
UNIVERSITÀ DI ROMA

Recap

Deep Neural Networks

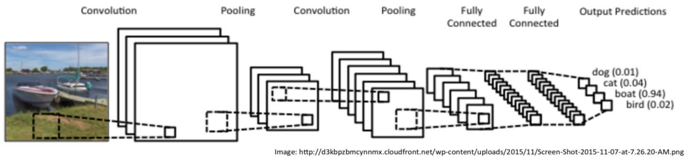


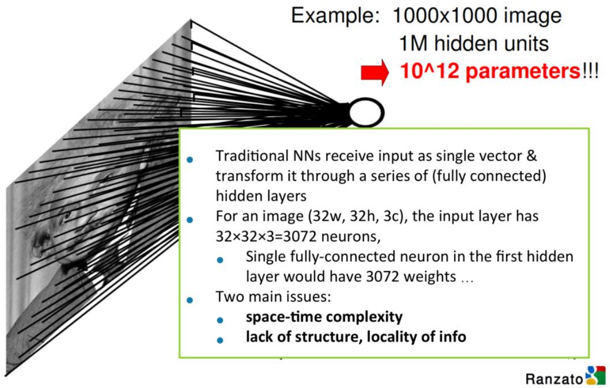
- Composition of multiple layers (functions)
- To fit the parameters, require a loss function (a measure of error)
- Use chain rule to propagate the error
- Combine linear and non-linear transformations

Why DNNs instead of feature engineering + other function approximators?

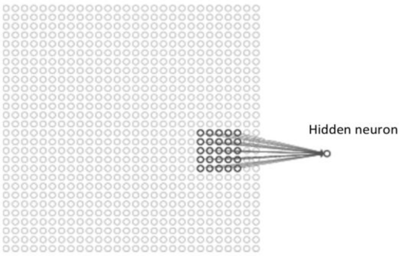
- Use distributed representations instead of local representations (like Kernels in SVMs)
- Universal function approximators
- Can potentially need exponentially less nodes/parameters (compared to a shallow net) to represent the same function
- Can learn the parameters using stochastic gradient descent

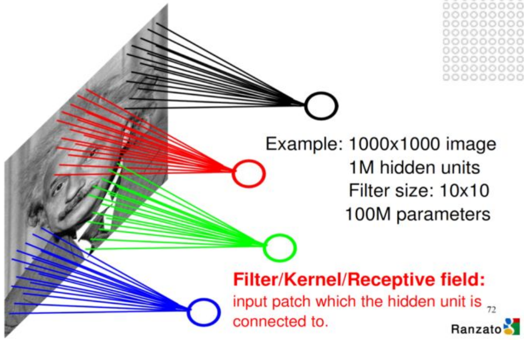
Convolutional Neural Networks





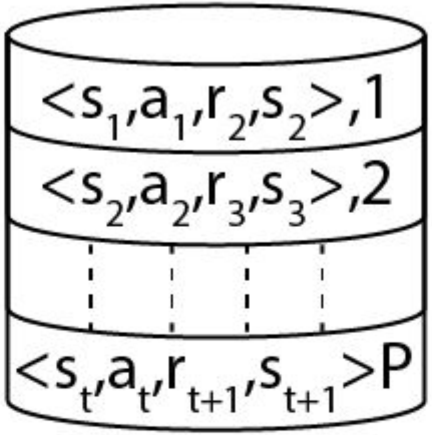
- Traditional NNs receive input as single vector & transform it through a series of (fully connected) hidden layers
- For an image (32w, 32h, 3c), the input layer has 32×32×3=3072 neurons,
  - Single fully-connected neuron in the first hidden layer would have 3072 weights …
- Two main issues:
  - **space-time complexity**
  - **lack of structure, locality of info**





Replay Buffer

- Used in most recent RL algorithms
- Stores previous transitions experienced by an agent
- Transitions are used later in time for training



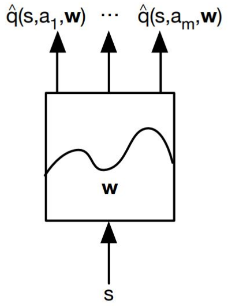
- **Capacity:** total number of transitions stored in the buffer
- **Age of transitions:** number of gradient steps taken by the learner since the transition was generated

- Efficient use of previous experience
  - Same data used multiple times
  - Collecting real data costs
- Better convergence with function approximators
  - Makes data more i.i.d.
  - More similar to a supervised learning task

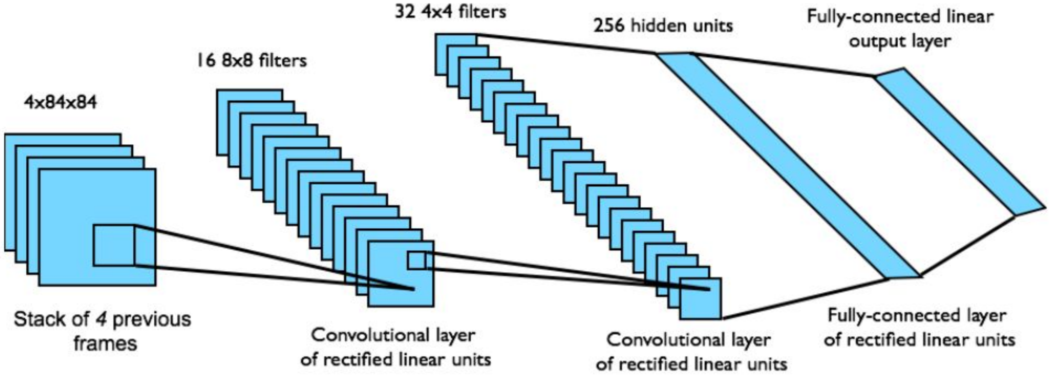
DQN

s_1, a_1, r_2, s_2
s_2, a_2, r_3, s_3
s_3, a_3, r_4, s_4
$\dots$
$s_t, a_t, r_{t+1}, s_{t+1}$

$\rightarrow$ s, a, r, s'



- End-to-end learning of values Q(s, a) from pixels s
- Input state s is stack of raw pixels from last 4 frames
- Output is Q(s, a) for 18 joystick/button positions
- Reward is change in score for that step



DQN Pseudo-code

Input $\boldsymbol{C}$, α, $D = \{\}$, Initialize $\boldsymbol{w}$, $\boldsymbol{w}^- = \boldsymbol{w}$, $t = 0$

Get initial state s_0

loop

  Sample action a_t given ϵ-greedy policy for current $\hat{Q}(s_t, a; \boldsymbol{w})$

  Observe reward r_t and next state s_{t+1}

  Store transition (s_t, a_t, r_t, s_{t+1}) in replay buffer D

  Sample random minibatch of tuples (s_i, a_i, r_i, s_{i+1}) from D

  for j in minibatch **do**

    if episode terminated at step $i + 1$ **then**

      $y_i = r_i$

    else

      $y_i = r_i + \gamma \max_{a'} \hat{Q}(s_{i+1}, a'; \boldsymbol{w}^-)$

    end if

    Do gradient descent step on $(y_i - \hat{Q}(s_i, a_i; \boldsymbol{w}))^2$ for parameters $\boldsymbol{w}$: $\Delta \boldsymbol{w} = \alpha(y_i - \hat{Q}(s_i, a_i; \boldsymbol{w}))\nabla_{\boldsymbol{w}} \hat{Q}(s_i, a_i; \boldsymbol{w})$

  end for

  $t = t + 1$

  if mod(t,C) == 0 **then**

    $\boldsymbol{w}^- \leftarrow \boldsymbol{w}$

  end if

end loop

Credits: Emma Brunskill

DQN Result Analysis

Game	Linear	Deep Network	DQN w/ fixed Q	DQN w/ replay	DQN w/replay and fixed Q
Breakout	3	3	10	241	317
Enduro	62	29	141	831	1006
River Raid	2345	1453	2868	4102	7447
Seaquest	656	275	1003	823	2894
Space Invaders	301	302	373	826	1089

Credits: Emma Brunskill

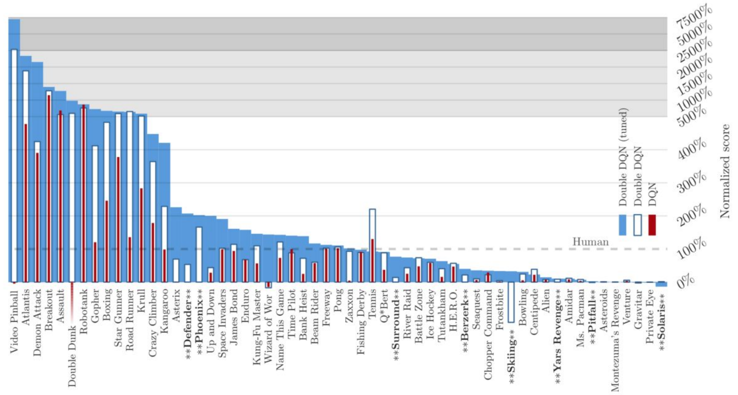
Maximization Bias

Greedy and epsilon greedy require a maximization step

This can lead to a positive bias:

- Consider s where many actions have Q(s,a) that are all zero
- Estimated values are uncertain, some above and some below zero
- Maximum of the true values is zero, but max over estimates is positive

DDQN



- Extend Double Q-Learning to DQN
- Use current Q-network for action selection
- Use older (the target) Q-network (w^-) for action evaluation

$$\Delta \boldsymbol{w} = \alpha\Big(r + \gamma \overbrace{\hat{Q}\big(\underbrace{\arg\max_{a'} \hat{Q}(s', a'; \boldsymbol{w})}_{\text{Action selection: } \boldsymbol{w}}; \boldsymbol{w}^-\big)}^{\text{Action evaluation: } \boldsymbol{w}^-} - \hat{Q}(s, a; \boldsymbol{w})\Big)$$

Prioritized Experience Replay (PER)

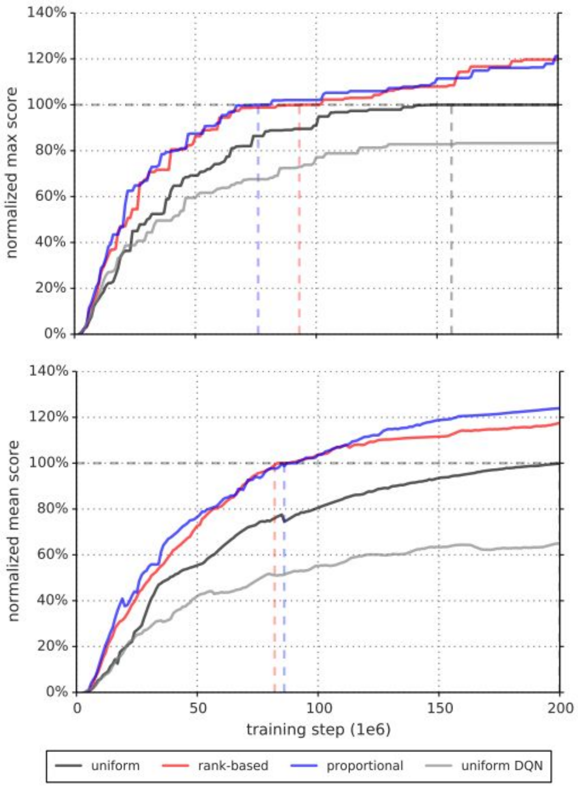
Schaul, Tom, et al. "Prioritized experience replay." *arXiv preprint arXiv:1511.05952* (2015). - Code: https://github.com/google/dopamine/tree/master/dopamine/replay_memory

- **Idea:** “*more frequently replay transitions with high expected learning progress, as measured by the magnitude of their temporal-difference (TD) error*”
- Prioritization can lead to:
  - Loss of diversity:
    - Use stochastic prioritization
  - Bias
    - Use importance sampling

DDQN with PER

Algorithm 1 Double DQN with proportional prioritization

Input: minibatch k, step-size η, replay period K and size N, exponents α and β, budget T.
Initialize replay memory $\mathcal{H} = \emptyset$, $\Delta = 0$, $p_1 = 1$
Observe S_0 and choose $A_0 \sim \pi_\theta(S_0)$
for $t = 1$ **to** T **do**
  Observe S_t, R_t, γ_t
  Store transition $(S_{t-1}, A_{t-1}, R_t, \gamma_t, S_t)$ in $\mathcal{H}$ with maximal priority $p_t = \max_{i<t} p_i$
  if $t \equiv 0 \mod K$ **then**
    for $j = 1$ **to** k **do**
      Sample transition $j \sim P(j) = p_j^\alpha / \sum_i p_i^\alpha$
      Compute importance-sampling weight $w_j = (N \cdot P(j))^{-\beta} / \max_i w_i$
      Compute TD-error $\delta_j = R_j + \gamma_j Q_{\text{target}}(S_j, \arg\max_a Q(S_j, a)) - Q(S_{j-1}, A_{j-1})$
      Update transition priority $p_j \leftarrow |\delta_j|$
      Accumulate weight-change $\Delta \leftarrow \Delta + w_j \cdot \delta_j \cdot \nabla_\theta Q(S_{j-1}, A_{j-1})$
    end for
    Update weights $\theta \leftarrow \theta + \eta \cdot \Delta$, reset $\Delta = 0$
    From time to time copy weights into target network $\theta_{\text{target}} \leftarrow \theta$
  end if
  Choose action $A_t \sim \pi_\theta(S_t)$
end for



SAPIENZA
UNIVERSITÀ DI ROMA

Importance Sampling for MC

The same idea is applied to RL for off-policy learning

Consider the MC setting: we want to use the returns from policy μ to evaluate π

Compute $G^{\pi/\mu}$ by multiplying **importance sampling corrections**

$$G_t^{\pi/\mu} = \frac{\pi(A_t|S_t)}{\mu(A_t|S_t)} \frac{\pi(A_{t+1}|S_{t+1})}{\mu(A_{t+1}|S_{t+1})} \cdots \frac{\pi(A_T|S_T)}{\mu(A_T|S_T)} G_t$$

if μ is zero and π non-zero this cannot be used

Importance Sampling for TD

The same idea is applied to RL for off-policy learning

Consider the TD setting: we want to weight the TD target

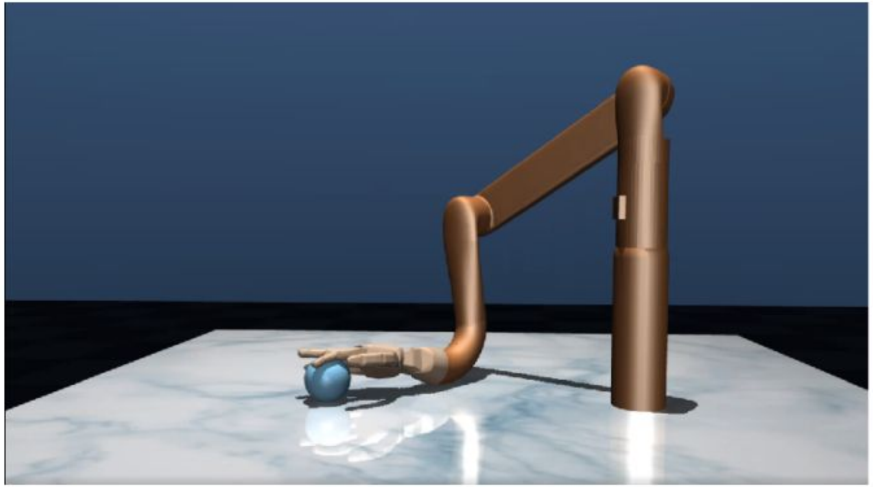
$$\frac{\pi(A_t|S_t)}{\mu(A_t|S_t)}(R_{t+1} + \gamma V(S_{t+1}))$$

Q-Learning does not need it, why?

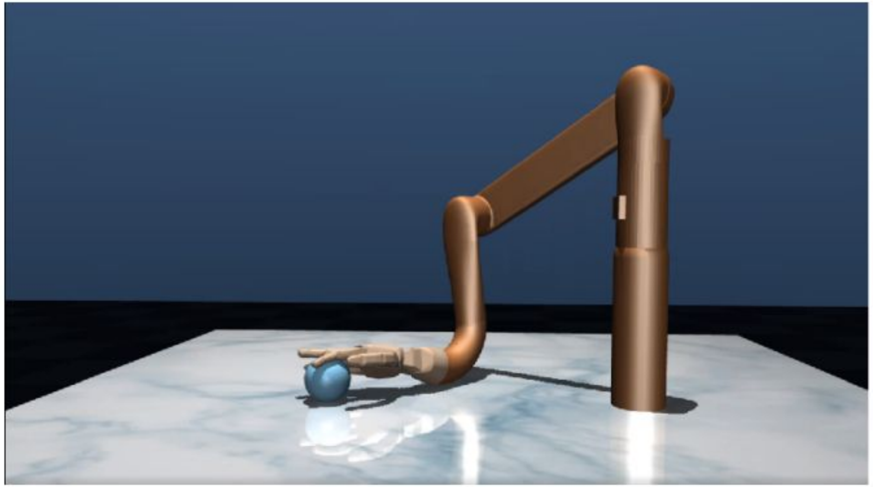
We directly use the action from the target policy

End Recap

Model-Based RL: Motivation



Model-Based RL: Motivation

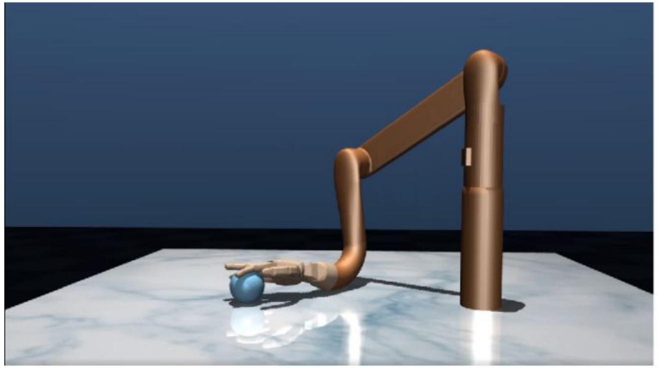


We cannot write out the exact analytical dynamics, but we can learn it from data {s,a,s'}

Model-Based RL: Motivation

We cannot write out the exact analytical dynamics, but we can learn it from data {s,a,s'}

And then find a policy by planning on such dynamic model

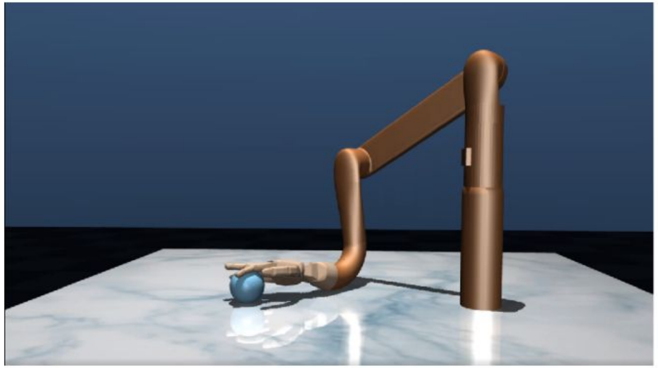


Model-Based RL

- **Model-free RL:** rely on learning alone
- **Model-based RL:** rely both on learning and planning on a *model*

Model: anything that an agent can use to predict how environment will respond to actions

- *Distribution models:* describe all possibilities and their probabilities
- *Sample models:* produce just one possibility sampled according to probabilities



Basic Algorithm

The simplest algorithm is the following:

1. Generate data

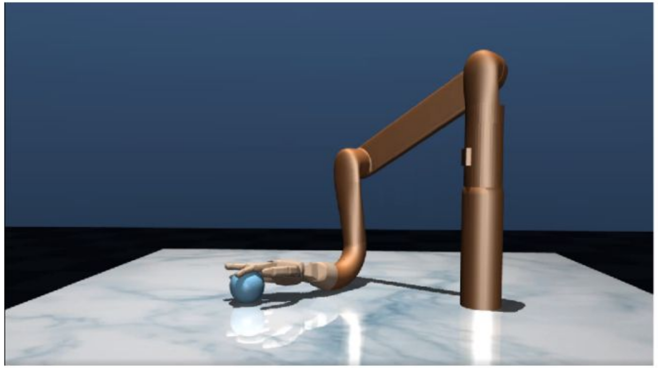
    (e.g., execute a starting policy)

2. Fit a model using data

    (e.g., using least-squares, or maximum likelihood)

3. Plan on the learned model

    (e.g., using VI, PI, or LQR)



Basic Algorithm

The simplest algorithm is the following:

1. Generate data

    (e.g., execute a starting policy)

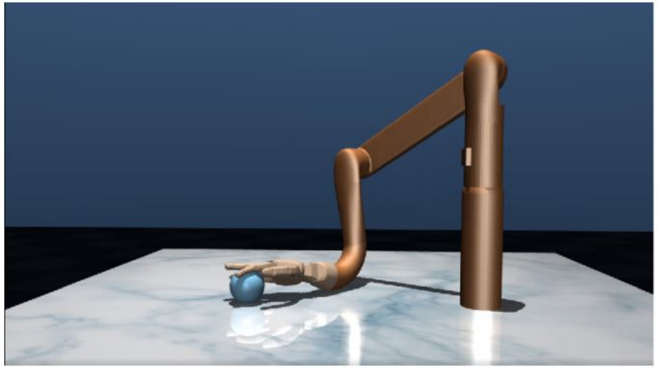
2. Fit a model using data

    (e.g., using least-squares, or maximum likelihood)

3. Plan on the learned model

    (e.g., using VI, PI, or LQR)

Often iterate this process several times



Simulation Lemma

Question: If I have an estimator of the true dynamics, what's the performance of a policy under this estimator, wrt the real world?

Simulation Lemma

---

approximation!

Question: If I have an $\boxed{\text{estimator}}$ of the true dynamics, what's the performance of a policy under this estimator, wrt the real world?

Simulation Lemma

---

a model or simulator!

Question: If I have an estimator of the true dynamics, what's the performance of a policy under this estimator, wrt the real world?

Simulation Lemma

Question: If I have an estimator of the true dynamics, what's the performance of a policy under this estimator, wrt the real world?

Example: if I have a driving policy and a self-driving car simulator: if I test such policy in the simulator and in the real world, what's the difference in terms of policy performance?

Simulation Lemma

Question: If I have an estimator of the true dynamics, what's the performance of a policy under this estimator, wrt the real world?

$$\widehat{V}^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \,|\, \pi,\ \widehat{P}\right]$$

value of policy in the simulator

$$V^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \,|\, \pi, P\right]$$

value of policy in the true dynamics

Simulation Lemma

Question: If I have an estimator of the true dynamics, what's the performance of a policy under this estimator, wrt the real world?

$$\widehat{V}^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \mid \pi, \widehat{P}\right]$$

value of policy in the simulator

$$V^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \mid \pi, P\right]$$

value of policy in the true dynamics

$$\widehat{V}^{\pi}(s_0) - V^{\pi}(s_0) \quad ?$$

Simulation Lemma

Question: If I have an estimator of the true dynamics, what's the performance of a policy under this estimator, wrt the real world?

$$\widehat{V}^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \mid \pi, \widehat{P}\right]$$

value of policy in the simulator

$$V^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \mid \pi, P\right]$$

value of policy in the true dynamics

$$\widehat{V}^{\pi}(s_0) - V^{\pi}(s_0) = \frac{\gamma}{1-\gamma} \mathbb{E}_{s,a \sim d^{\pi}_{s_0}}\left[\mathbb{E}_{s' \sim \widehat{P}(s,a)} \widehat{V}^{\pi}(s') - \mathbb{E}_{s' \sim P(s,a)} \widehat{V}^{\pi}(s')\right]$$

Simulation Lemma

Question: If I have an estimator of the true dynamics, what's the performance of a policy under this estimator, wrt the real world?

$$\widehat{V}^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \mid \pi, \widehat{P}\right]$$

value of policy in the simulator

$$V^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \mid \pi, P\right]$$

value of policy in the true dynamics

$$\widehat{V}^{\pi}(s_0) - V^{\pi}(s_0) = \frac{\gamma}{1-\gamma}\mathbb{E}_{s,a\sim \boxed{d^{\pi}_{s_0}}}\left[\mathbb{E}_{s'\sim \widehat{P}(s,a)} \widehat{V}^{\pi}(s') - \mathbb{E}_{s'\sim P(s,a)} \widehat{V}^{\pi}(s')\right]$$

distribution of s,a under the policy and the true dynamics

Simulation Lemma

Question: If I have an estimator of the true dynamics, what's the performance of a policy under this estimator, wrt the real world?

$$\widehat{V}^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \mid \pi, \widehat{P}\right]$$

value of policy in the simulator

$$V^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \mid \pi, P\right]$$

value of policy in the true dynamics

$$\widehat{V}^{\pi}(s_0) - V^{\pi}(s_0) = \boxed{\frac{\gamma}{1-\gamma}} \mathbb{E}_{s,a \sim d^{\pi}_{s_0}}\left[\mathbb{E}_{s' \sim \widehat{P}(s,a)} \widehat{V}^{\pi}(s') - \mathbb{E}_{s' \sim P(s,a)} \widehat{V}^{\pi}(s')\right]$$

as usual, from the sum of the gammas

Simulation Lemma

Question: If I have an estimator of the true dynamics, what's the performance of a policy under this estimator, wrt the real world?

$$\widehat{V}^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \mid \pi, \widehat{P}\right]$$

value of policy in the simulator

$$V^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \mid \pi, P\right]$$

value of policy in the true dynamics

$$\widehat{V}^{\pi}(s_0) - V^{\pi}(s_0) = \frac{\gamma}{1-\gamma} \mathbb{E}_{s,a \sim d^{\pi}_{s_0}} \boxed{\left[\mathbb{E}_{s' \sim \widehat{P}(s,a)} \widehat{V}^{\pi}(s') - \mathbb{E}_{s' \sim P(s,a)} \widehat{V}^{\pi}(s')\right]}$$

difference between the two distribution (model and real dynamics)

Simulation Lemma

Question: If I have an estimator of the true dynamics, what's the performance of a policy under this estimator, wrt the real world?

$$\widehat{V}^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \mid \pi, \widehat{P}\right]$$

value of policy in the simulator

$$V^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \mid \pi, P\right]$$

value of policy in the true dynamics

$$\widehat{V}^{\pi}(s_0) - V^{\pi}(s_0) = \frac{\gamma}{1-\gamma}\mathbb{E}_{s,a\sim d^{\pi}_{s_0}} \boxed{\left[\mathbb{E}_{s'\sim \widehat{P}(s,a)} \widehat{V}^{\pi}(s') - \mathbb{E}_{s'\sim P(s,a)} \widehat{V}^{\pi}(s')\right]}$$

difference between two distributions P and Q can be computed as

$|E_{x\sim P}[f(x)] - E_{x\sim Q}[f(x)]|$

difference between the two distribution (model and real dynamics)

Simulation Lemma

$$\widehat{V}^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \,|\, \pi, \widehat{P}\right]$$

value of policy in the simulator

$$V^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \,|\, \pi, P\right]$$

value of policy in the true dynamics

Question: If I have an estimator of the true dynamics, what's the performance of a policy under this estimator, wrt the real world?

$$\widehat{V}^{\pi}(s_0) - V^{\pi}(s_0) = \frac{\gamma}{1-\gamma} \mathbb{E}_{s,a \sim d^{\pi}_{s_0}} \left[\mathbb{E}_{s' \sim \widehat{P}(s,a)} \widehat{V}^{\pi}(s') - \mathbb{E}_{s' \sim P(s,a)} \widehat{V}^{\pi}(s') \right]$$

Simulation Lemma

$$\widehat{V}^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \mid \pi, \widehat{P}\right]$$

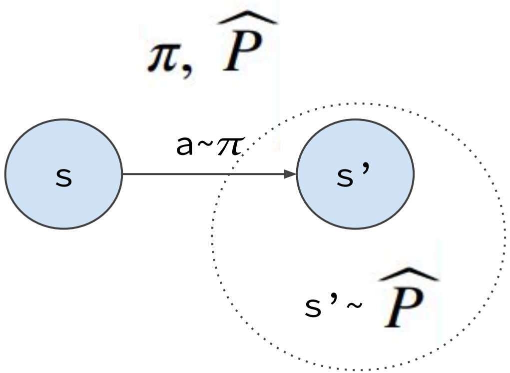
value of policy in the simulator

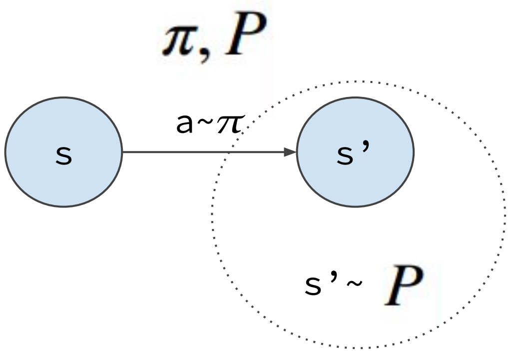
$$V^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \mid \pi, P\right]$$

value of policy in the true dynamics

Question: If I have an estimator of the true dynamics, what's the performance of a policy under this estimator, wrt the real world?

$$\widehat{V}^{\pi}(s_0) - V^{\pi}(s_0) = \frac{\gamma}{1-\gamma} \mathbb{E}_{s,a\sim d^{\pi}_{s_0}}\left[\mathbb{E}_{s'\sim \widehat{P}(s,a)} \widehat{V}^{\pi}(s') - \mathbb{E}_{s'\sim P(s,a)} \widehat{V}^{\pi}(s')\right]$$





Simulation Lemma

$$\widehat{V}^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \,|\, \pi, \widehat{P}\right]$$

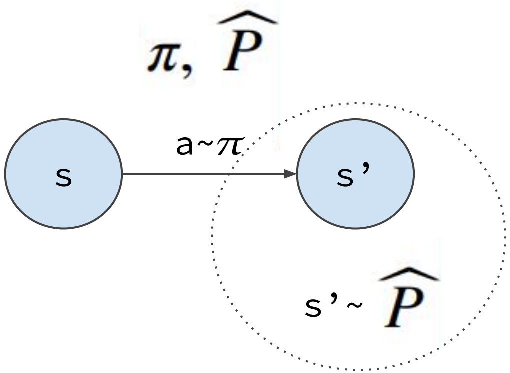
value of policy in the simulator

$$V^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \,|\, \pi, P\right]$$

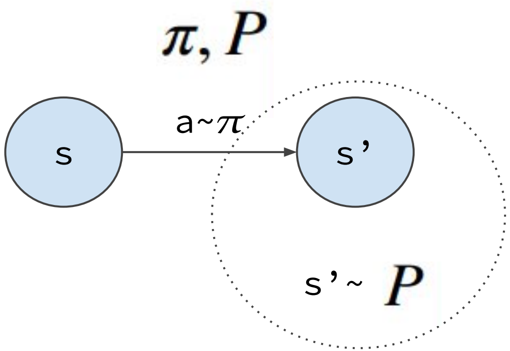
value of policy in the true dynamics

Question: If I have an estimator of the true dynamics, what's the performance of a policy under this estimator, wrt the real world?

$$\widehat{V}^{\pi}(s_0) - V^{\pi}(s_0) = \frac{\gamma}{1-\gamma}\mathbb{E}_{s,a\sim d^{\pi}_{s_0}}\left[\mathbb{E}_{s'\sim \widehat{P}(s,a)}\widehat{V}^{\pi}(s') - \mathbb{E}_{s'\sim P(s,a)}\widehat{V}^{\pi}(s')\right]$$



the action distribution is the same, the only difference, at one step, is in the next state



Simulation Lemma

$$\widehat{V}^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \mid \pi, \widehat{P}\right]$$

value of policy in the simulator

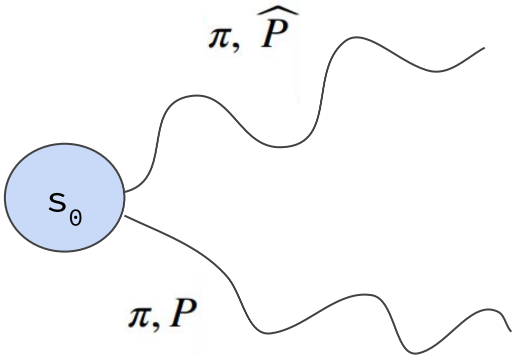
$$V^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \mid \pi, P\right]$$

value of policy in the true dynamics

Question: If I have an estimator of the true dynamics, what's the performance of a policy under this estimator, wrt the real world?

$$\widehat{V}^{\pi}(s_0) - V^{\pi}(s_0) = \frac{\gamma}{1-\gamma}\mathbb{E}_{s,a\sim d^{\pi}_{s_0}}\left[\mathbb{E}_{s'\sim \widehat{P}(s,a)}\widehat{V}^{\pi}(s') - \mathbb{E}_{s'\sim P(s,a)}\widehat{V}^{\pi}(s')\right]$$

Computing this is very difficult, but we can do it one step at a time



Simulation Lemma

$$\widehat{V}^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \mid \pi, \widehat{P}\right]$$

value of policy in the simulator

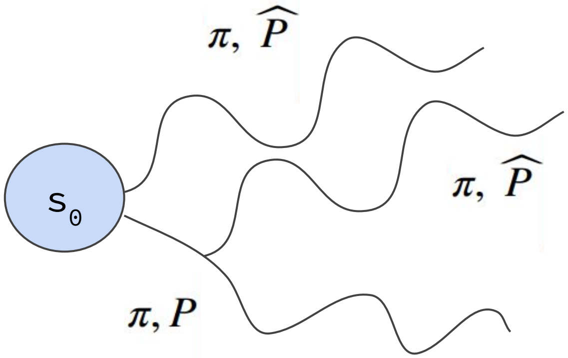
$$V^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \mid \pi, P\right]$$

value of policy in the true dynamics

Question: If I have an estimator of the true dynamics, what's the performance of a policy under this estimator, wrt the real world?

$$\widehat{V}^{\pi}(s_0) - V^{\pi}(s_0) = \frac{\gamma}{1-\gamma}\mathbb{E}_{s,a\sim d^{\pi}_{s_0}}\left[\mathbb{E}_{s'\sim \widehat{P}(s,a)} \widehat{V}^{\pi}(s') - \mathbb{E}_{s'\sim P(s,a)} \widehat{V}^{\pi}(s')\right]$$

Let's step in the real dynamics for one step, and then go back to the simulator



Simulation Lemma

$$\widehat{V}^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \mid \pi, \widehat{P}\right]$$

value of policy in the simulator

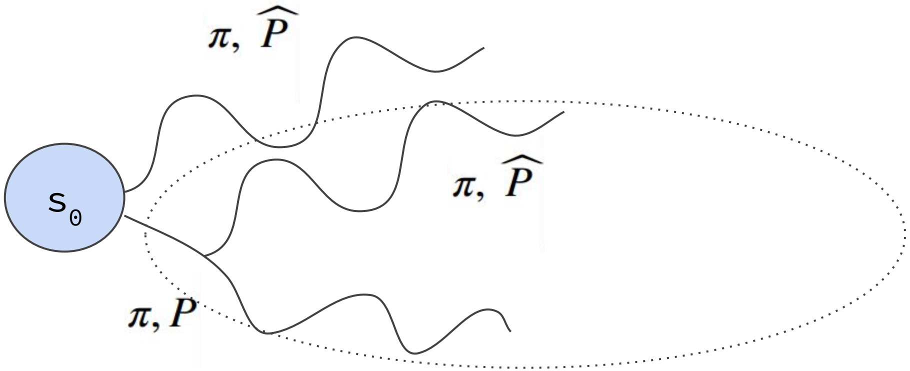
$$V^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \mid \pi, P\right]$$

value of policy in the true dynamics

Question: If I have an estimator of the true dynamics, what’s the performance of a policy under this estimator, wrt the real world?

$$\widehat{V}^{\pi}(s_0) - V^{\pi}(s_0) = \frac{\gamma}{1-\gamma}\mathbb{E}_{s,a\sim d^{\pi}_{s_0}}\left[\mathbb{E}_{s'\sim \widehat{P}(s,a)}\widehat{V}^{\pi}(s') - \mathbb{E}_{s'\sim P(s,a)}\widehat{V}^{\pi}(s')\right]$$

We can do recursion and follow the same reasoning again



Simulation Lemma

$$\widehat{V}^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \mid \pi, \widehat{P}\right]$$

value of policy in the simulator

$$V^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \mid \pi, P\right]$$

value of policy in the true dynamics

Question: If I have an estimator of the true dynamics, what's the performance of a policy under this estimator, wrt the real world?

$$\widehat{V}^{\pi}(s_0) - V^{\pi}(s_0) = \frac{\gamma}{1-\gamma}\mathbb{E}_{s,a\sim d^{\pi}_{s_0}}\left[\mathbb{E}_{s'\sim \widehat{P}(s,a)}\widehat{V}^{\pi}(s') - \mathbb{E}_{s'\sim P(s,a)}\widehat{V}^{\pi}(s')\right]$$

$$\widehat{V}^{\pi}(s_0) - V^{\pi}(s_0) = \gamma\mathbb{E}_{a_0\sim\pi(\cdot|s_0)}\left[\mathbb{E}_{s_1\sim \widehat{P}(s_0,a_0)}\widehat{V}^{\pi}(s_1) - \mathbb{E}_{s_1\sim P(s_0,a_0)}V^{\pi}(s_1)\right]$$

$$V^{\pi}(s_0) = r(s_0,\pi(s_0)) + \gamma E_{s1\sim P(s0,\pi(s0))}[V^{\pi}(s_1)]$$

and similar for the V hat, so $r(s_0,\pi(s_0))$ cancels out

Simulation Lemma

$$\widehat{V}^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \mid \pi, \widehat{P}\right]$$

value of policy in the simulator

$$V^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \mid \pi, P\right]$$

value of policy in the true dynamics

Question: If I have an estimator of the true dynamics, what's the performance of a policy under this estimator, wrt the real world?

$$\widehat{V}^{\pi}(s_0) - V^{\pi}(s_0) = \frac{\gamma}{1-\gamma}\mathbb{E}_{s,a\sim d^{\pi}_{s_0}}\left[\mathbb{E}_{s'\sim \widehat{P}(s,a)}\widehat{V}^{\pi}(s') - \mathbb{E}_{s'\sim P(s,a)}\widehat{V}^{\pi}(s')\right]$$

$$\widehat{V}^{\pi}(s_0) - V^{\pi}(s_0) = \gamma\mathbb{E}_{a_0\sim\pi(\cdot|s_0)}\left[\mathbb{E}_{s_1\sim \widehat{P}(s_0,a_0)}\widehat{V}^{\pi}(s_1) - \mathbb{E}_{s_1\sim P(s_0,a_0)}V^{\pi}(s_1)\right]$$

$$= \gamma\mathbb{E}_{a_0\sim\pi(\cdot|s_0)}\left[\mathbb{E}_{s_1\sim \widehat{P}(s_0,a_0)}\widehat{V}^{\pi}(s_1) \boxed{- \mathbb{E}_{s_1\sim P(s_0,a_0)}\widehat{V}^{\pi}(s_1) + \mathbb{E}_{s_1\sim P(s_0,a_0)}\widehat{V}^{\pi}(s_1)} - \mathbb{E}_{s_1\sim P(s_0,a_0)}V^{\pi}(s_1)\right]$$

Simulation Lemma

$$\widehat{V}^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty}\gamma^h r(s_h, a_h)\,|\,\pi, \widehat{P}\right]$$

value of policy in the simulator

$$V^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty}\gamma^h r(s_h, a_h)\,|\,\pi, P\right]$$

value of policy in the true dynamics

Question: If I have an estimator of the true dynamics, what’s the performance of a policy under this estimator, wrt the real world?

$$\widehat{V}^{\pi}(s_0) - V^{\pi}(s_0) = \frac{\gamma}{1-\gamma}\mathbb{E}_{s,a\sim d^{\pi}_{s_0}}\left[\mathbb{E}_{s'\sim \widehat{P}(s,a)}\widehat{V}^{\pi}(s') - \mathbb{E}_{s'\sim P(s,a)}\widehat{V}^{\pi}(s')\right]$$

$$\widehat{V}^{\pi}(s_0) - V^{\pi}(s_0) = \gamma\mathbb{E}_{a_0\sim\pi(\cdot|s_0)}\left[\mathbb{E}_{s_1\sim \widehat{P}(s_0,a_0)}\widehat{V}^{\pi}(s_1) - \mathbb{E}_{s_1\sim P(s_0,a_0)}V^{\pi}(s_1)\right]$$

$$= \gamma\mathbb{E}_{a_0\sim\pi(\cdot|s_0)}\left[\mathbb{E}_{s_1\sim \widehat{P}(s_0,a_0)}\widehat{V}^{\pi}(s_1) - \mathbb{E}_{s_1\sim P(s_0,a_0)}\widehat{V}^{\pi}(s_1) + \mathbb{E}_{s_1\sim P(s_0,a_0)}\widehat{V}^{\pi}(s_1) - \mathbb{E}_{s_1\sim P(s_0,a_0)}V^{\pi}(s_1)\right]$$

$$= \gamma\mathbb{E}_{a_0\sim\pi(\cdot|s_0)}\left[\mathbb{E}_{s_1\sim \widehat{P}(s_0,a_0)}\widehat{V}^{\pi}(s_1) - \mathbb{E}_{s_1\sim P(s_0,a_0)}\widehat{V}^{\pi}(s_1)\right] + \gamma\mathbb{E}_{a_0\sim\pi(\cdot|s_0),s_1\sim P(s_0,a_0)}\left[\widehat{V}^{\pi}(s_1) - V^{\pi}(s_1)\right]$$

Just play with the formula and re-order stuff

Simulation Lemma

$$\widehat{V}^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \mid \pi, \widehat{P}\right]$$

value of policy in the simulator

$$V^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \mid \pi, P\right]$$

value of policy in the true dynamics

Question: If I have an estimator of the true dynamics, what’s the performance of a policy under this estimator, wrt the real world?

$$\widehat{V}^{\pi}(s_0) - V^{\pi}(s_0) = \frac{\gamma}{1-\gamma}\mathbb{E}_{s,a\sim d^{\pi}_{s_0}}\left[\mathbb{E}_{s'\sim \widehat{P}(s,a)}\widehat{V}^{\pi}(s') - \mathbb{E}_{s'\sim P(s,a)}\widehat{V}^{\pi}(s')\right]$$

$$\boxed{\widehat{V}^{\pi}(s_0) - V^{\pi}(s_0)} = \gamma\mathbb{E}_{a_0\sim\pi(\cdot|s_0)}\left[\mathbb{E}_{s_1\sim \widehat{P}(s_0,a_0)}\widehat{V}^{\pi}(s_1) - \mathbb{E}_{s_1\sim P(s_0,a_0)}V^{\pi}(s_1)\right]$$

$$= \gamma\mathbb{E}_{a_0\sim\pi(\cdot|s_0)}\left[\mathbb{E}_{s_1\sim \widehat{P}(s_0,a_0)}\widehat{V}^{\pi}(s_1) - \mathbb{E}_{s_1\sim P(s_0,a_0)}\widehat{V}^{\pi}(s_1) + \mathbb{E}_{s_1\sim P(s_0,a_0)}\widehat{V}^{\pi}(s_1) - \mathbb{E}_{s_1\sim P(s_0,a_0)}V^{\pi}(s_1)\right]$$

$$= \gamma\mathbb{E}_{a_0\sim\pi(\cdot|s_0)}\left[\mathbb{E}_{s_1\sim \widehat{P}(s_0,a_0)}\widehat{V}^{\pi}(s_1) - \mathbb{E}_{s_1\sim P(s_0,a_0)}\widehat{V}^{\pi}(s_1)\right] + \gamma\mathbb{E}_{a_0\sim\pi(\cdot|s_0),s_1\sim P(s_0,a_0)}\boxed{\left[\widehat{V}^{\pi}(s_1) - V^{\pi}(s_1)\right]}$$

We can do recursion

Simulation Lemma

$$\widehat{V}^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \mid \pi, \widehat{P}\right]$$

value of policy in the simulator

$$V^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \mid \pi, P\right]$$

value of policy in the true dynamics

Question: If I have an estimator of the true dynamics, what's the performance of a policy under this estimator, wrt the real world?

$$\widehat{V}^{\pi}(s_0) - V^{\pi}(s_0) = \boxed{\frac{\gamma}{1-\gamma}\mathbb{E}_{s,a\sim d^{\pi}_{s_0}}\left[\mathbb{E}_{s'\sim \widehat{P}(s,a)} \widehat{V}^{\pi}(s') - \mathbb{E}_{s'\sim P(s,a)} \widehat{V}^{\pi}(s')\right]}$$

...and we get to this just by doing an exponential averaging over all the timesteps

Simulation Lemma

$$\widehat{V}^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \mid \pi, \widehat{P}\right]$$

value of policy in the simulator

$$V^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \mid \pi, P\right]$$

value of policy in the true dynamics

Question: If I have an estimator of the true dynamics, what's the performance of a policy under this estimator, wrt the real world?

$$\widehat{V}^{\pi}(s_0) - V^{\pi}(s_0) = \boxed{\frac{\gamma}{1-\gamma}\mathbb{E}_{s,a\sim d^{\pi}_{s_0}}\left[\mathbb{E}_{s'\sim \widehat{P}(s,a)} \widehat{V}^{\pi}(s') - \mathbb{E}_{s'\sim P(s,a)} \widehat{V}^{\pi}(s')\right]}$$

We also know that

$$|E_{x\sim P}[f(x)] - E_{x\sim Q}[f(x)]| \leq \sup_x f(x) \|P-Q\|_1$$

where we use the l1-norm and the *total variation distance* between the two distributions

Simulation Lemma

$$\widehat{V}^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \mid \pi, \widehat{P}\right]$$

value of policy in the simulator

$$V^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \mid \pi, P\right]$$

value of policy in the true dynamics

Question: If I have an estimator of the true dynamics, what's the performance of a policy under this estimator, wrt the real world?

$$\widehat{V}^{\pi}(s_0) - V^{\pi}(s_0) = \frac{\gamma}{1-\gamma}\mathbb{E}_{s,a\sim d^{\pi}_{s_0}}\left[\mathbb{E}_{s'\sim \widehat{P}(s,a)}\widehat{V}^{\pi}(s') - \mathbb{E}_{s'\sim P(s,a)}\widehat{V}^{\pi}(s')\right]$$

$$\leq \boxed{\frac{1}{(1-\gamma)^2}}\mathbb{E}_{s,a\sim d^{\pi}_{s_0}}\left\| \widehat{P}(\cdot \mid s,a) - P(\cdot \mid s,a) \right\|_1$$

Since we assume reward is in [0, 1], the max value of V is 1/(1-γ)

Simulation Lemma

$$\widehat{V}^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \,|\, \pi, \widehat{P}\right]$$

value of policy in the simulator

$$V^{\pi}(s_0) = \mathbb{E}\left[\sum_{h=0}^{\infty} \gamma^h r(s_h, a_h) \,|\, \pi, P\right]$$

value of policy in the true dynamics

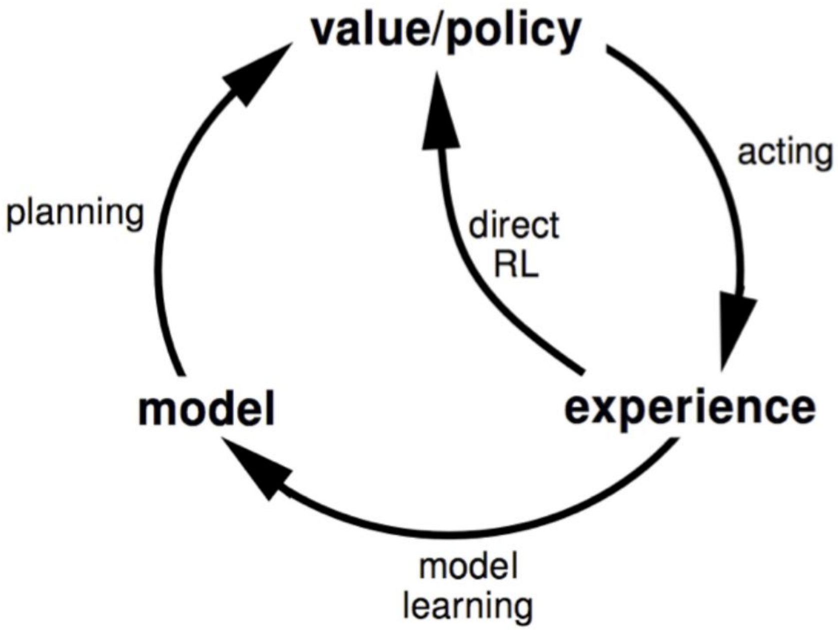
Question: If I have an estimator of the true dynamics, what’s the performance of a policy under this estimator, wrt the real world?

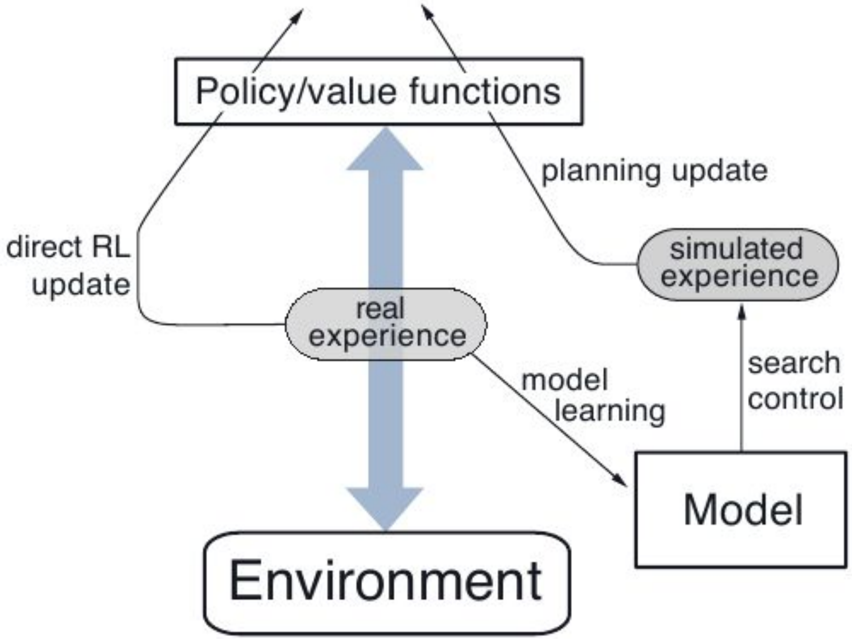
$$\widehat{V}^{\pi}(s_0) - V^{\pi}(s_0) = \frac{\gamma}{1-\gamma}\mathbb{E}_{s,a\sim d^{\pi}_{s_0}}\left[\mathbb{E}_{s'\sim \widehat{P}(s,a)}\widehat{V}^{\pi}(s') - \mathbb{E}_{s'\sim P(s,a)}\widehat{V}^{\pi}(s')\right]$$

$$\le \frac{1}{(1-\gamma)^2}\mathbb{E}_{s,a\sim d^{\pi}_{s_0}}\left\| \widehat{P}(\cdot\,|\,s,a) - P(\cdot\,|\,s,a)\right\|_1$$

We can bound the policy performance difference by the total model disagreement measured on the real trajectory

Full Model-Based RL Loop





Model Fitting

How can we fit a model?

For example, very simply, collect N data-points and estimate it as follows (note that we're using the indicator function **1**)

$$\widehat{P}(s'|s,a) = \frac{\sum_{i=1}^{N} \mathbf{1}\{s_i' = s'\}}{N}$$

Model Fitting

How can we fit a model?

For example, very simply, collect N data-points and estimate it as follows (note that we're using the indicator function **1**)

$$\widehat{P}(s' | s, a) = \frac{\sum_{i=1}^{N} \mathbf{1}\{s'_i = s'\}}{N}$$

At infinity this should converge to the true P

Planning

How can we plan using a model?

Use value iteration, policy iteration, LQR if we’re in continuous space, or other solutions like:

- Q-planning
- Monte-Carlo Tree Search

Dyna-Q

Initialize $Q(s,a)$ and $Model(s,a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$

Loop forever:

(a) $S \leftarrow$ current (nonterminal) state

(b) $A \leftarrow \varepsilon$-greedy(S, Q)

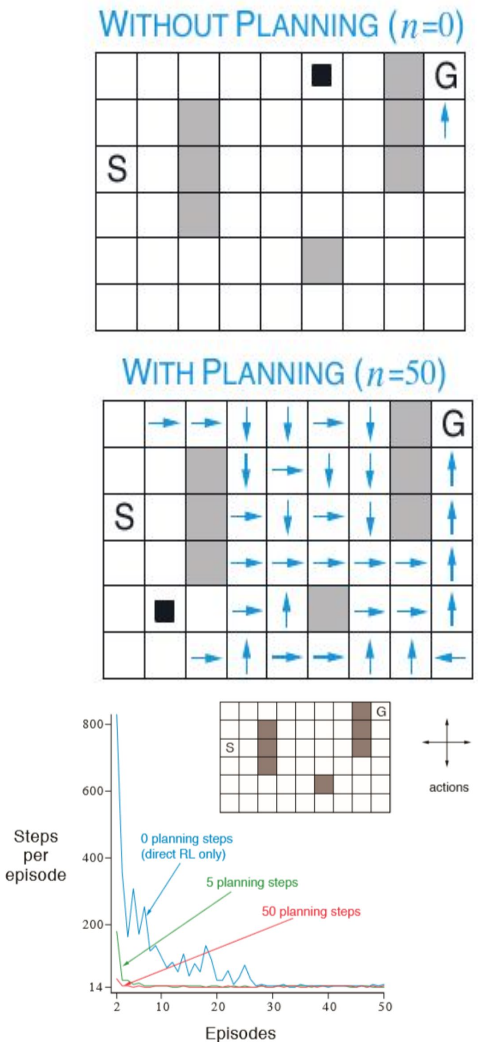
(c) Take action A; observe resultant reward, R, and state, S'

(d) $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$

(e) $Model(S,A) \leftarrow R, S'$ (assuming deterministic environment)

(f) Loop repeat n times:

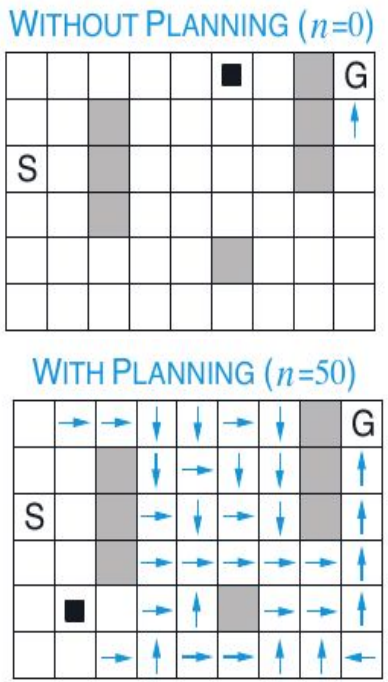
- $S \leftarrow$ random previously observed state
- $A \leftarrow$ random action previously taken in S
- $R, S' \leftarrow Model(S,A)$
- $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$

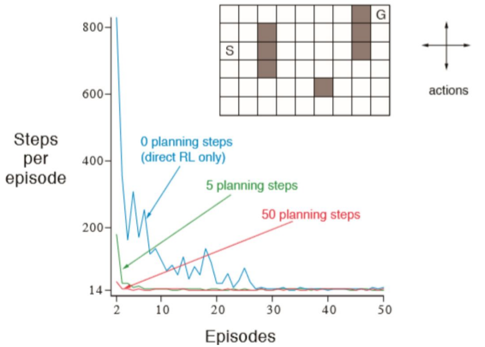


Dyna-Q

Initialize $Q(s,a)$ and $Model(s,a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$
Loop forever:
  (a) $S \leftarrow$ current (nonterminal) state
  (b) $A \leftarrow \varepsilon$-greedy(S, Q)
  (c) Take action A; observe resultant reward, R, and state, S'
  (d) $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$
  (e) $\boxed{Model(S,A) \leftarrow R, S' \text{ (assuming deterministic environment)}}$
  (f) Loop repeat n times:
    $S \leftarrow$ random previously observed state
    $A \leftarrow$ random action previously taken in S
    $R, S' \leftarrow Model(S,A)$
    $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$

Model Fitting





Dyna-Q

Initialize $Q(s,a)$ and $Model(s,a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$

Loop forever:

(a) $S \leftarrow$ current (nonterminal) state

(b) $A \leftarrow \varepsilon$-greedy(S, Q)

(c) Take action A; observe resultant reward, R, and state, S'

(d) $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$

(e) $Model(S,A) \leftarrow R, S'$ (assuming deterministic environment)

(f) Loop repeat n times:

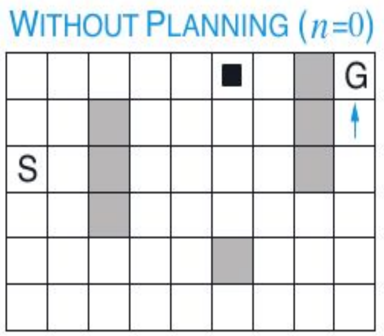
$S \leftarrow$ random previously observed state

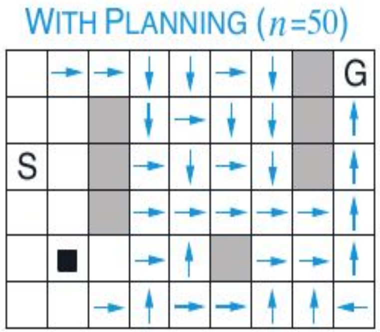
$A \leftarrow$ random action previously taken in S

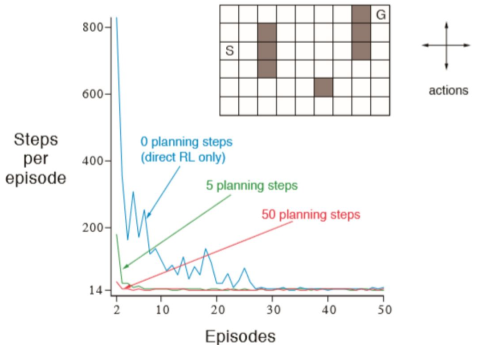
$R, S' \leftarrow Model(S,A)$

$Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$

Q-planning







More on Planning

Background planning (e.g., Dyna)

- Not focused on current state
- Gradually improve policy on the basis of simulated experience from model
- Planning plays a part well before an action is selected

Decision-time planning

- Begin planning after encountering each new state
- Evaluates action choices leading to different predicted states
- Use simulated experience to select an action for the current state
- Values and policy are updated specifically for current state

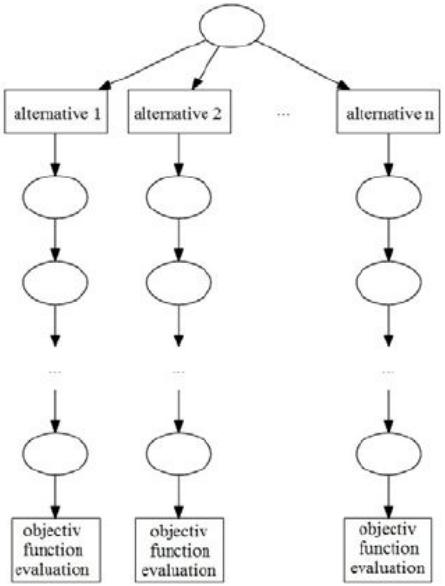
Can be blended together

Rollout Planning Algorithm

Decision-time planning

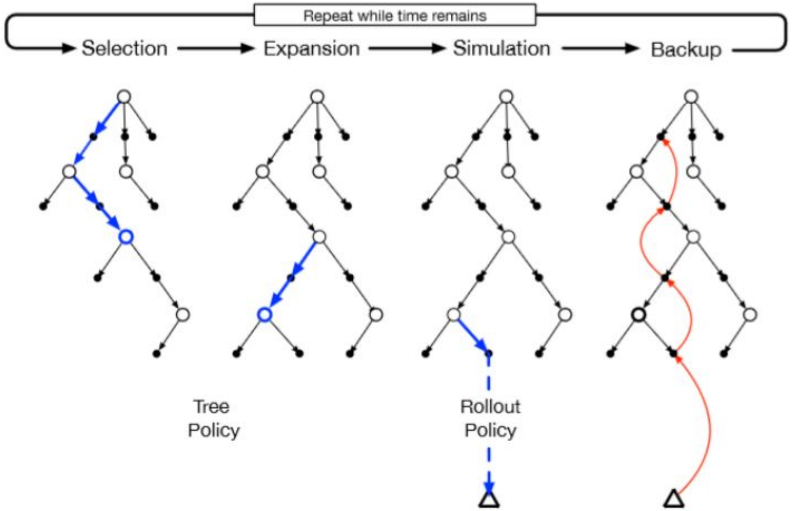
- Uses MC control applied to simulated trajectories starting at current state
- Estimate action values by averaging returns of many simulated trajectories: try each possible action for one step and then follow rollout policy
- When estimate accurate, highest value action is executed

Does not estimate (unlike MC) full value-function, but only value of actions for current state and given policy



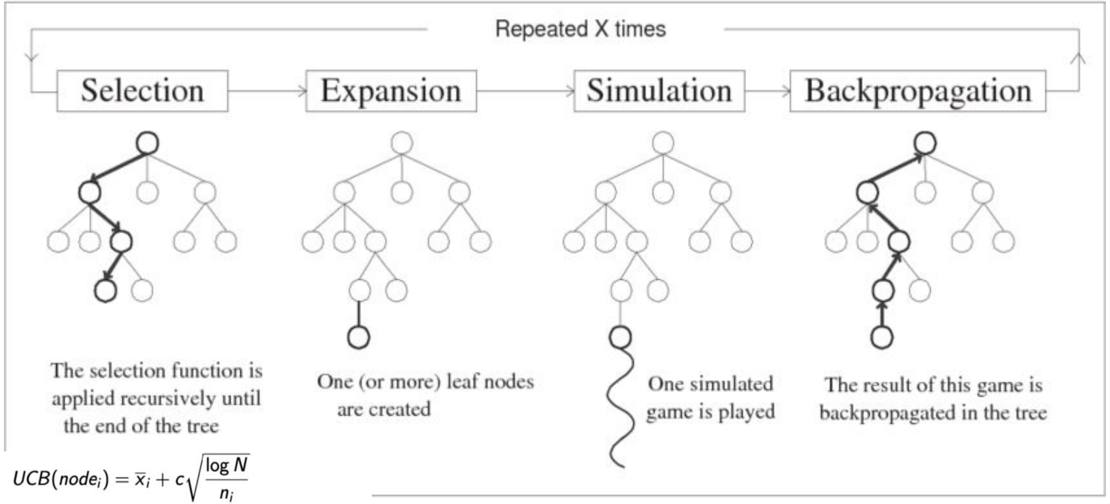
Monte-Carlo Tree Search

Decision-time planning, like a rollout algorithm BUT accumulating value estimates



Monte-Carlo Tree Search

Decision-time planning, like a rollout algorithm BUT accumulating value estimates



$$UCB(node_i) = \overline{x}_i + c\sqrt{\frac{\log N}{n_i}}$$

$\overline{x}_i$: mean node value; n_i: #visits of node i; N #visits parent;

AlphaGo

Silver, David, et al. "Mastering the game of Go with deep neural networks and tree search." *nature* 529.7587 (2016): 484-489.

AlphaGo - Networks

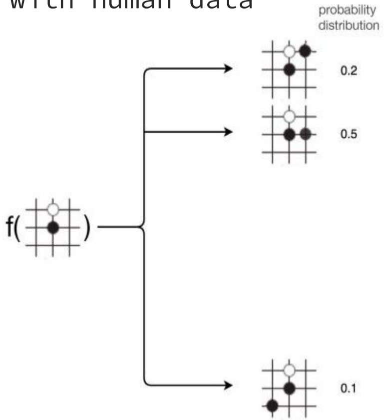
- Faster than SL Policy
- Less accurate than SL Policy
- Used during simulation

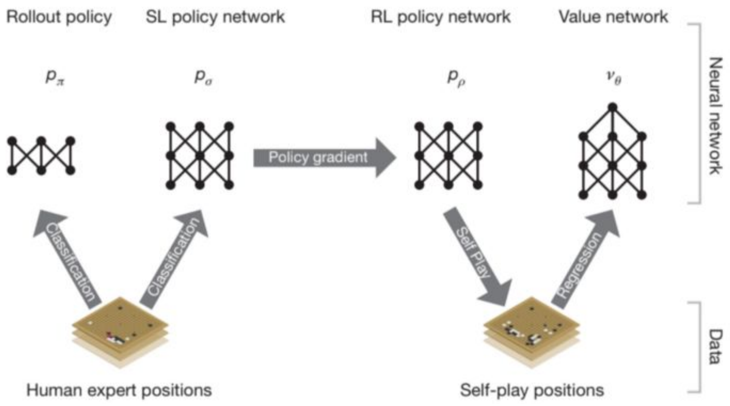
- RL policy initialized at SL policy
- RL training occurs using outcomes z (+1 win, -1 lose)

- 19×19×48 input feature to represent the board
- Trained on millions of board positions with human data

Extended Data Table 2 | **Input features for neural networks**

Feature	# of planes	Description
Stone colour	3	Player stone / opponent stone / empty
Ones	1	A constant plane filled with 1
Turns since	8	How many turns since a move was played
Liberties	8	Number of liberties (empty adjacent points)
Capture size	8	How many opponent stones would be captured
Self-atari size	8	How many of own stones would be captured
Liberties after move	8	Number of liberties after this move is played
Ladder capture	1	Whether a move at this point is a successful ladder capture
Ladder escape	1	Whether a move at this point is a successful ladder escape
Sensibleness	1	Whether a move is legal and does not fill its own eyes
Zeros	1	A constant plane filled with 0
Player color	1	Whether current player is black





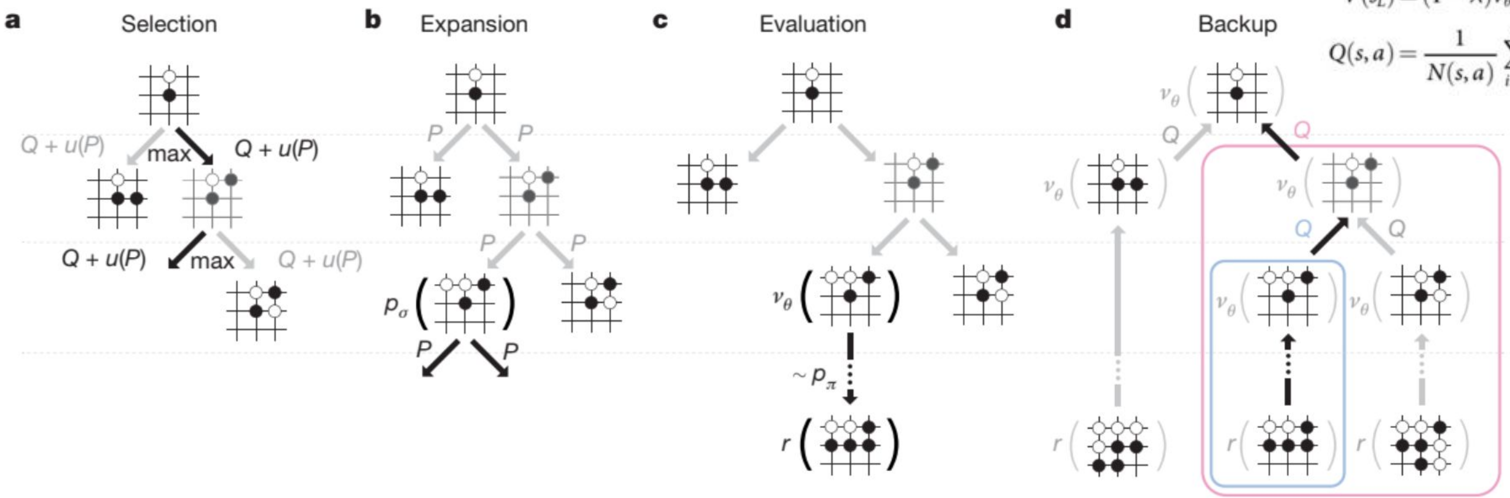
SAPIENZA
UNIVERSITÀ DI ROMA

AlphaGo - MCTS

$$N(s,a) = \sum_{i=1}^{n} 1(s,a,i)$$

$$V(s_L) = (1-\lambda)v_\theta(s_L) + \lambda z_L$$

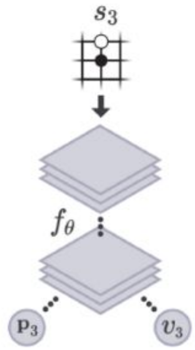
$$Q(s,a) = \frac{1}{N(s,a)} \sum_{i=1}^{n} 1(s,a,i) V(s_L^i)$$

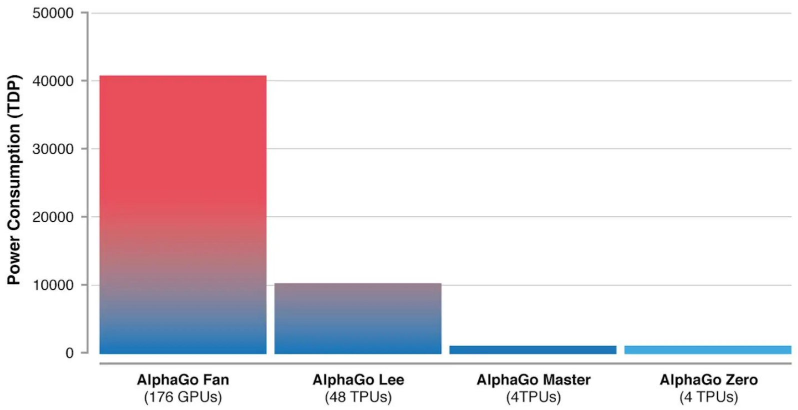


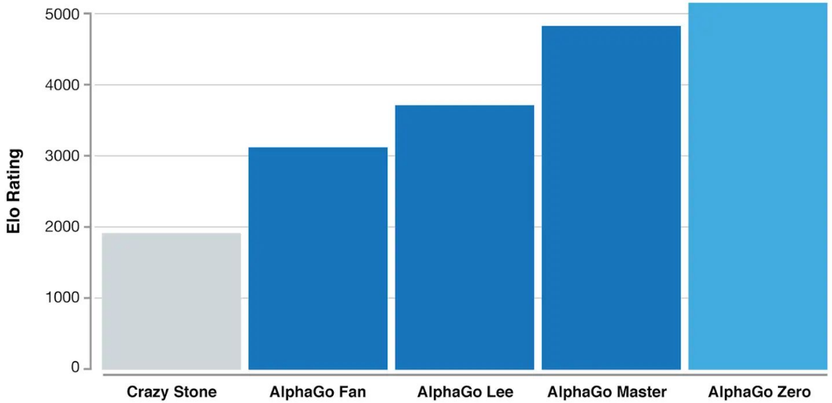
- Selection makes use of UCT
- Rollouts completed using rollout policy
- Backup done by mixing value network prediction and outcome of a rollout
- AlphaGo plays against previous versions of itself (self-play)

AlphaGo Zero VS AlphaGo

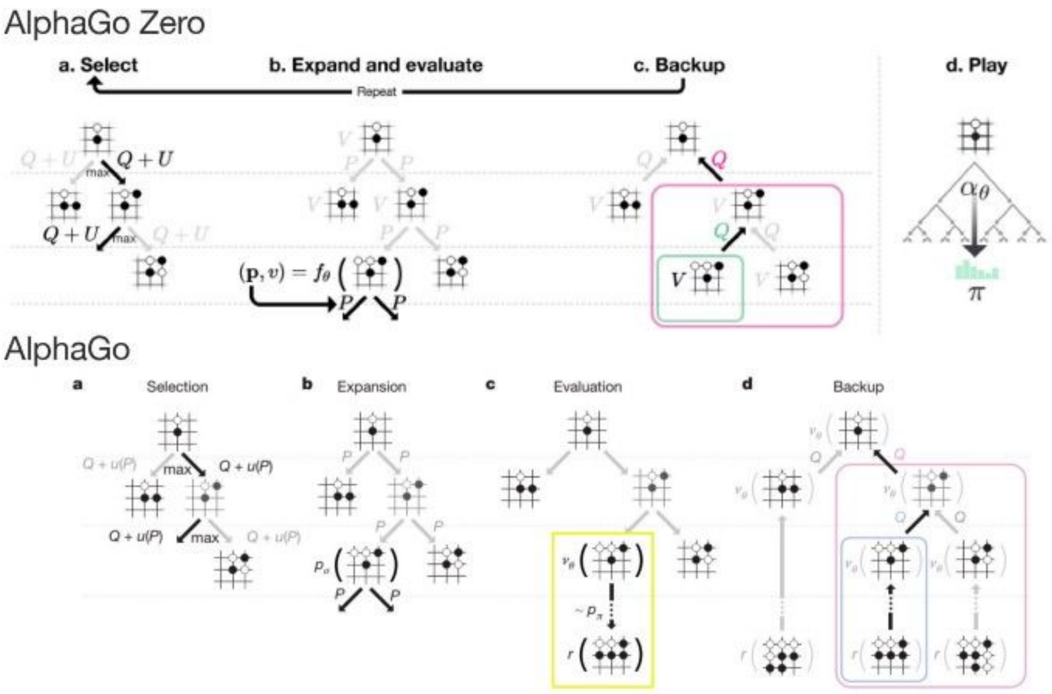
- Does not use human knowledge (SL policy network)
- Trained using only black and white stones from the Go board as input
- Uses one neural network rather than two
- Does not use rollouts

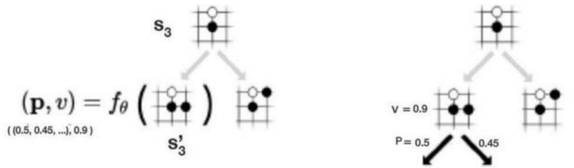




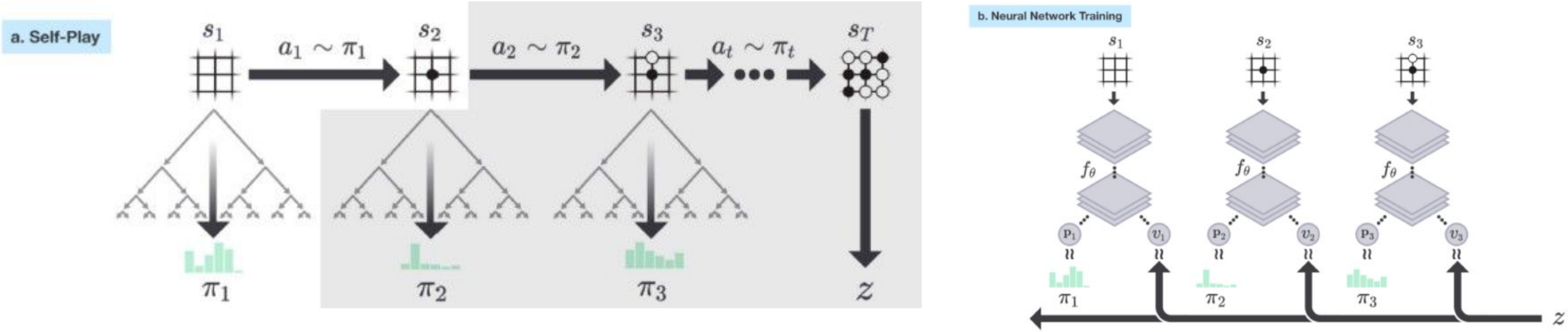


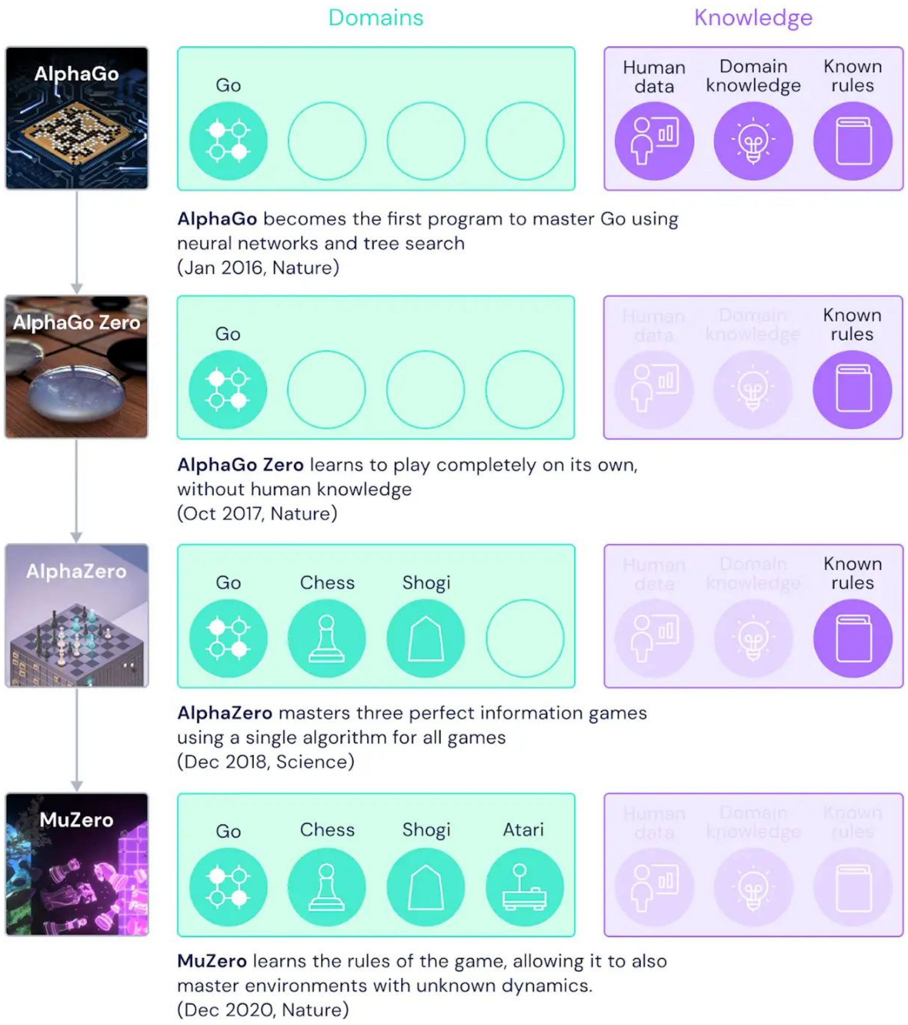
AlphaGo Zero VS AlphaGo - MCTS





AlphaGo Zero VS AlphaGo - Training





Domains
Knowledge
AlphaGo
Go
Human data
Domain knowledge
Known rules
AlphaGo becomes the first program to master Go using neural networks and tree search
(Jan 2016, Nature)
AlphaGo Zero
Go
Human data
Domain knowledge
Known rules
AlphaGo Zero learns to play completely on its own, without human knowledge
(Oct 2017, Nature)
AlphaZero
Go
Chess
Shogi
Human data
Domain knowledge
Known rules
AlphaZero masters three perfect information games using a single algorithm for all games
(Dec 2018, Science)
MuZero
Go
Chess
Shogi
Atari
Human data
Domain knowledge
Known rules
MuZero learns the rules of the game, allowing it to also master environments with unknown dynamics.
(Dec 2020, Nature)

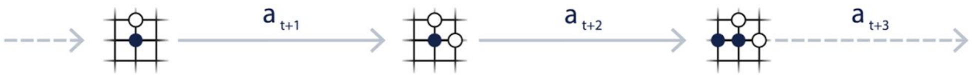
MuZero - MCTS

Models:

- policy
- value function
- reward
- new hidden state

MuZero - Training

Learned model unrolled with the collected experience

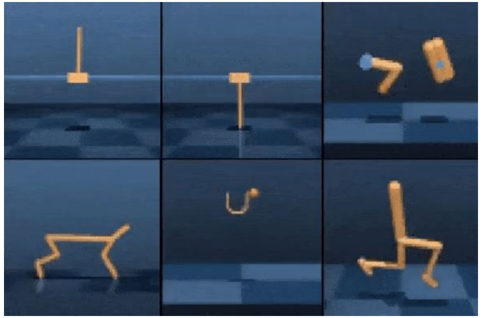


Learning Dynamics from Pixels

Hafner, Danijar, et al. "Learning latent dynamics for planning from pixels." *International Conference on Machine Learning*. PMLR, 2019.

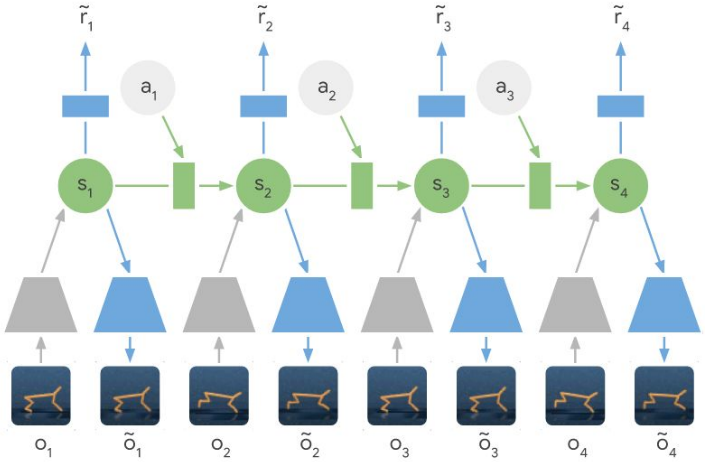
- Not always the state is available (POMDP): learn a compact representation
  - A recurrent model is needed
- Plan in the learned (latent!) dynamics space

$$
\begin{aligned}
&\text{Transition function:} && s_t \sim \mathrm{p}(s_t \mid s_{t-1}, a_{t-1}) \\
&\text{Observation function:} && o_t \sim \mathrm{p}(o_t \mid s_t) \\
&\text{Reward function:} && r_t \sim \mathrm{p}(r_t \mid s_t) \\
&\text{Policy:} && a_t \sim \mathrm{p}(a_t \mid o_{\leqslant t}, a_{<t}),
\end{aligned}
\tag{1}
$$



PlaNet

- Learns a
  - transition model
  - observation model
  - reward model
- Policy obtained using planning in MPC fashion



SAPIENZA UNIVERSITÀ DI ROMA

PlaNet - Data Collection & Training

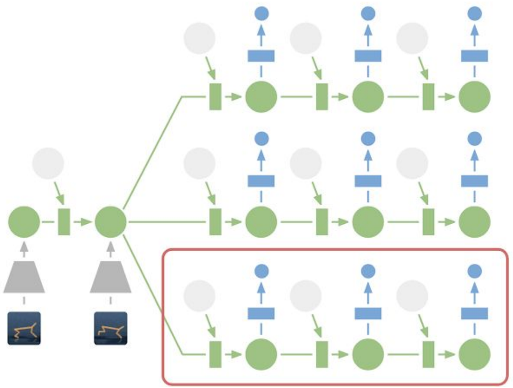
Algorithm 1: Deep Planning Network (PlaNet)

Input :

R Action repeat		$p(s_t \mid s_{t-1}, a_{t-1})$	Transition model
S Seed episodes		$p(o_t \mid s_t)$	Observation model
C Collect interval		$p(r_t \mid s_t)$	Reward model
B Batch size		$q(s_t \mid o_{\leq t}, a_{<t})$	Encoder
L Chunk length		$p(\epsilon)$	Exploration noise
α Learning rate			

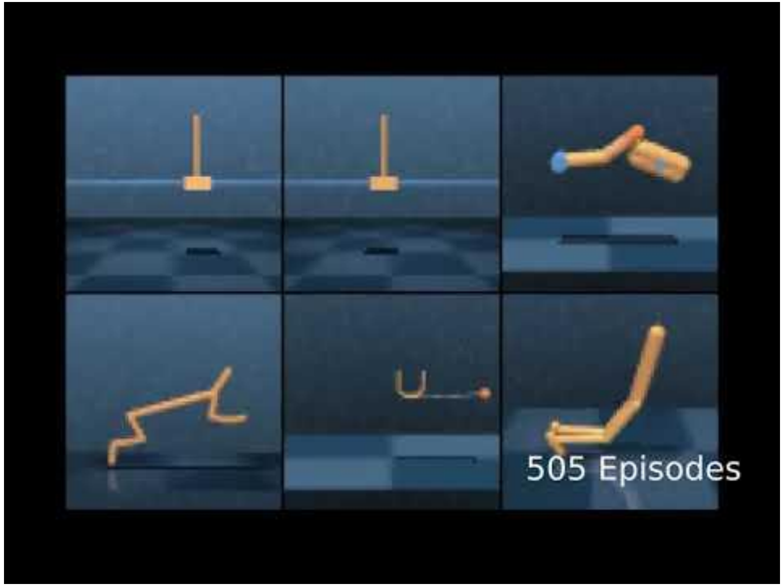
Initialize dataset $\mathcal{D}$ with S random seed episodes.
Initialize model parameters θ randomly.
while *not converged* **do**
  `// Model fitting`
  for *update step* $s = 1..C$ **do**
    Draw sequence chunks $\{(o_t, a_t, r_t)_{t=k}^{L+k}\}_{i=1}^{B} \sim \mathcal{D}$ uniformly at random from the dataset.
    Compute loss $\mathcal{L}(\theta)$ from Equation 8.
    Update model parameters $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta)$.
  `// Data collection`
  $o_1 \leftarrow$ `env.reset()`
  for *time step* $t = 1..\lceil \frac{T}{R} \rceil$ **do**
    Infer belief over current state $q(s_t \mid o_{\leq t}, a_{<t})$ from the history.
    $a_t \leftarrow$ `planner`$(q(s_t \mid o_{\leq t}, a_{<t}), p)$, see Algorithm 2 in the appendix for details.
    Add exploration noise $\epsilon \sim p(\epsilon)$ to the action.
    for *action repeat* $k = 1..R$ **do**
      $r_t^k, o_{t+1}^k \leftarrow$ `env.step`(a_t)
    $r_t, o_{t+1} \leftarrow \sum_{k=1}^{R} r_t^k, o_{t+1}^R$
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(o_t, a_t, r_t)_{t=1}^{T}\}$

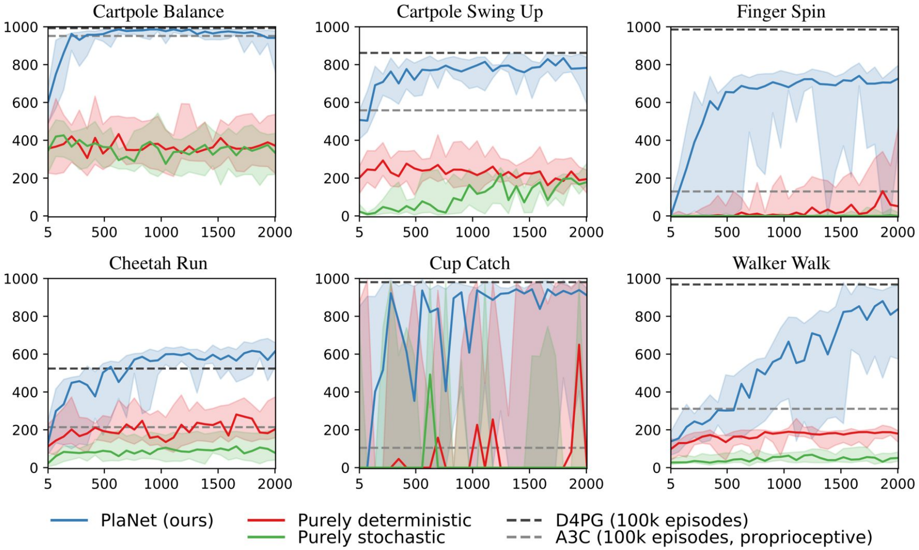
- Planning using Cross-Entropy Method
  - Population-based optimization algorithm that infers a distribution over action sequences that maximize the objective
- Encode past images
- Execute only first planned action



SAPIENZA
UNIVERSITÀ DI ROMA

PlaNet Results

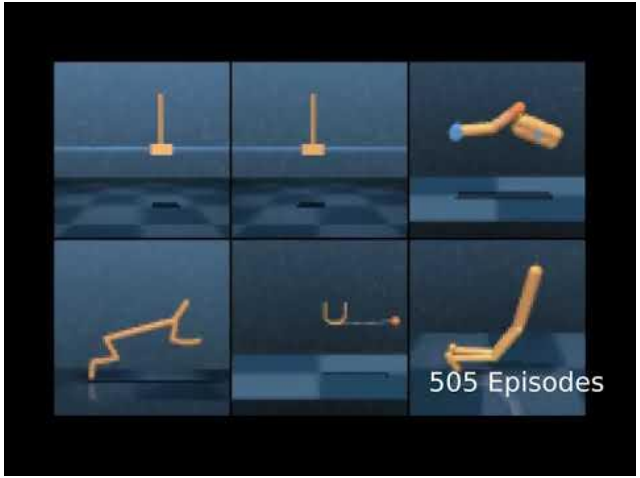




SAPIENZA
UNIVERSITÀ DI ROMA

Sim-to-Real

Tan, Jie, et al. "Sim-to-real: Learning agile locomotion for quadruped robots." *arXiv preprint arXiv:1804.10332* (2018).



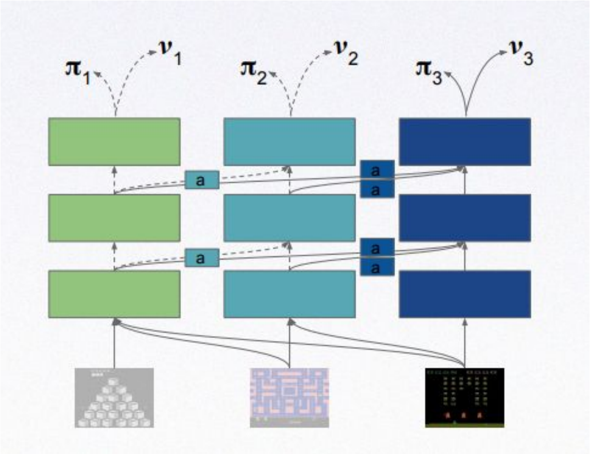
Narrowing the Reality Gap

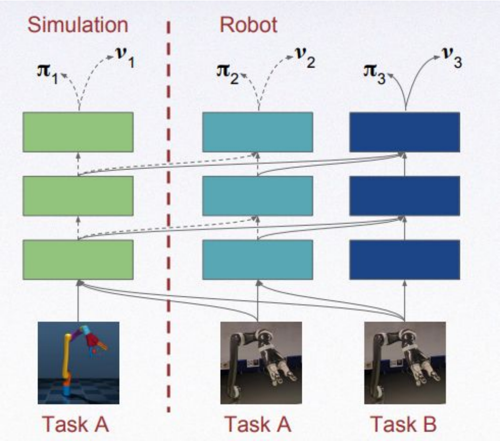
- Fine-tuning
- Progressive nets
- Improved modeling
  - System identification, better models, etc.
- Randomization
  - Random perturbations, Randomization in dynamics parameters

Progressive Networks

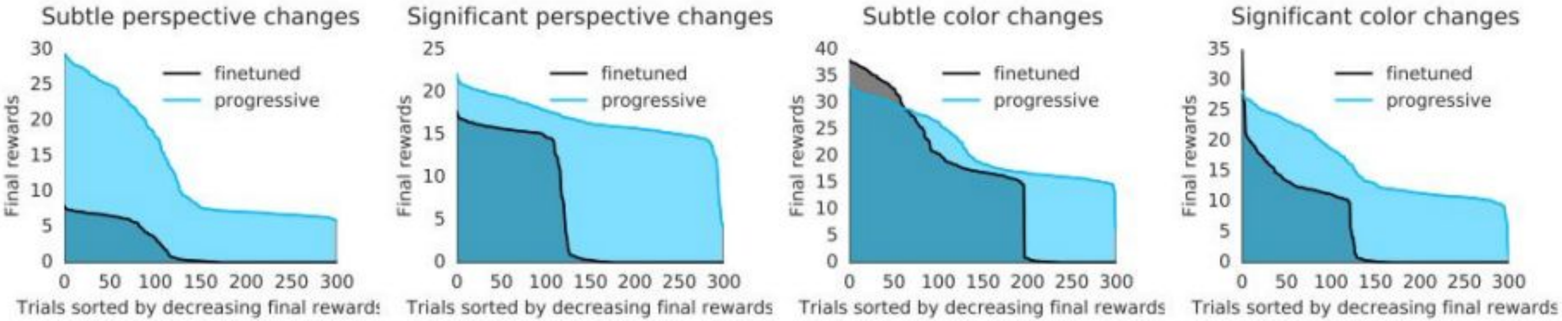
Rusu, Andrei A., et al. "Sim-to-real robot learning from pixels with progressive nets." arXiv preprint arXiv:1610.04286 (2016).

- Avoid catastrophic forgetting
- Allow transfer and multi-task
- Can be used to transfer from simulation to real robot





Progressive Nets VS Fine-Tuning



Dynamics Randomization