UCLA Samueli Computer Science

Uncertainty based Reinforcement Learning with Function Approximation

Weitong Zhang

Department of Computer Science, UCLA

Apr 18, 2022

Overview of my research

- Area: Reinforcement learning and online learning
- Reinforcement Learning with function approximation
  - Representation learning in reinforcement learning [**Z**HZZG21]
  - Exploration with function approximation [**Z**ZG21, NeurIPS] (this talk)
  - Convergence guarantee of popular RL algorithm [W**Z**XG20, NeurIPS]
- Deep learning based recommendation systems [**Z**ZLG20, ICLR], [J**Z**ZGW21, ICLR]
- Machine Learning in interdisciplinary fields
  - COVID-19 forecasting [CRLB21, PNAS]
  - Mechanistic deciphering for Chemistry reactions

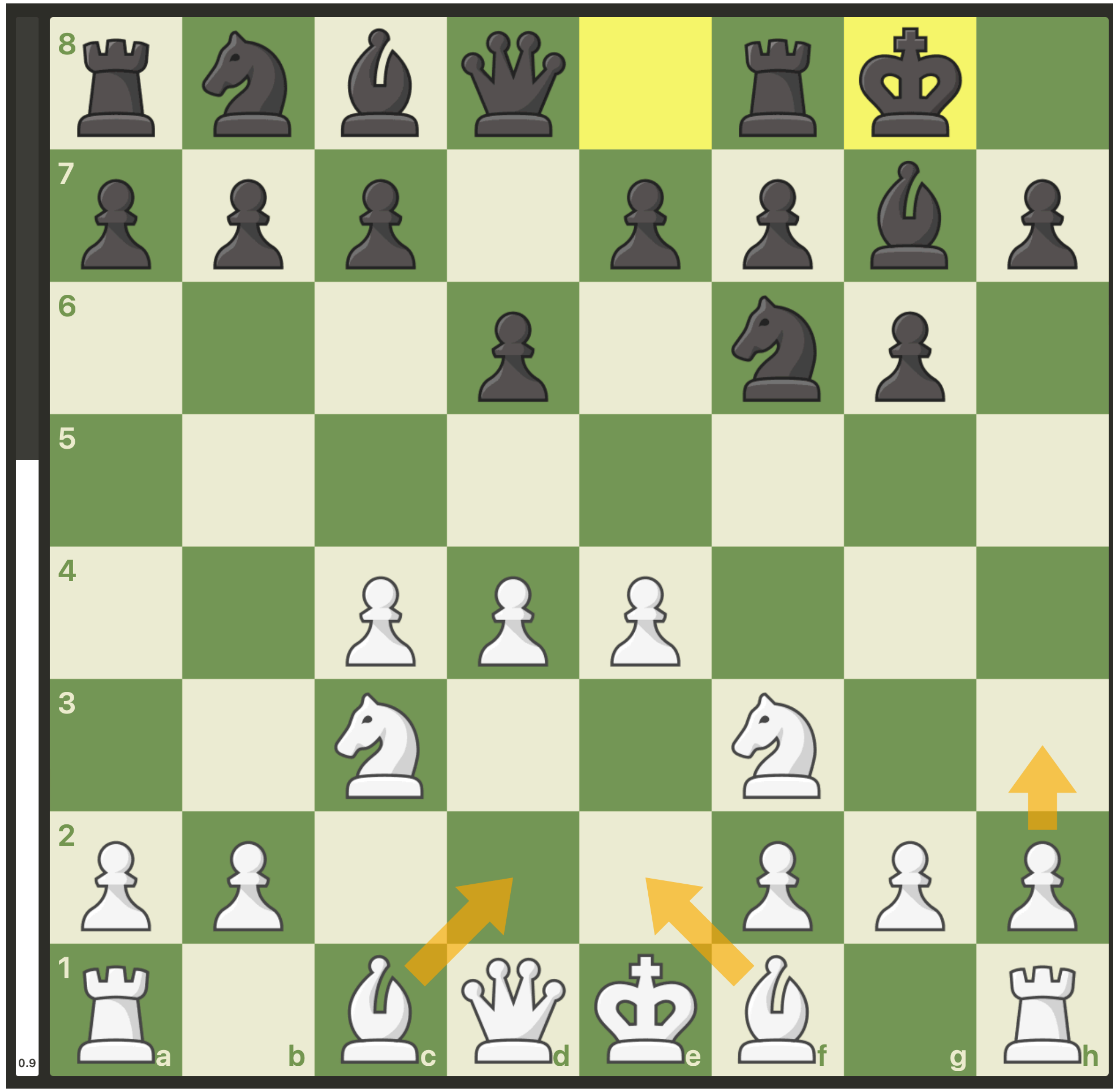
Reinforcement learning is useful

- Example: self-driving car
- Each time you will:
  - Get information from sensors, cameras, GPS…
  - Take actions: brake, accelerate or make turns
- Goal: reach the designation safely and efficiently

Image Credit: scharfsinn86 - stock.adobe.com

Reinforcement learning is interesting

- Example: chess game
- Each time you will:
  - Get information on board
  - Take actions:
    - Be2? Bd2? h3?
- Goal: Defend your king and checkmate your opponent



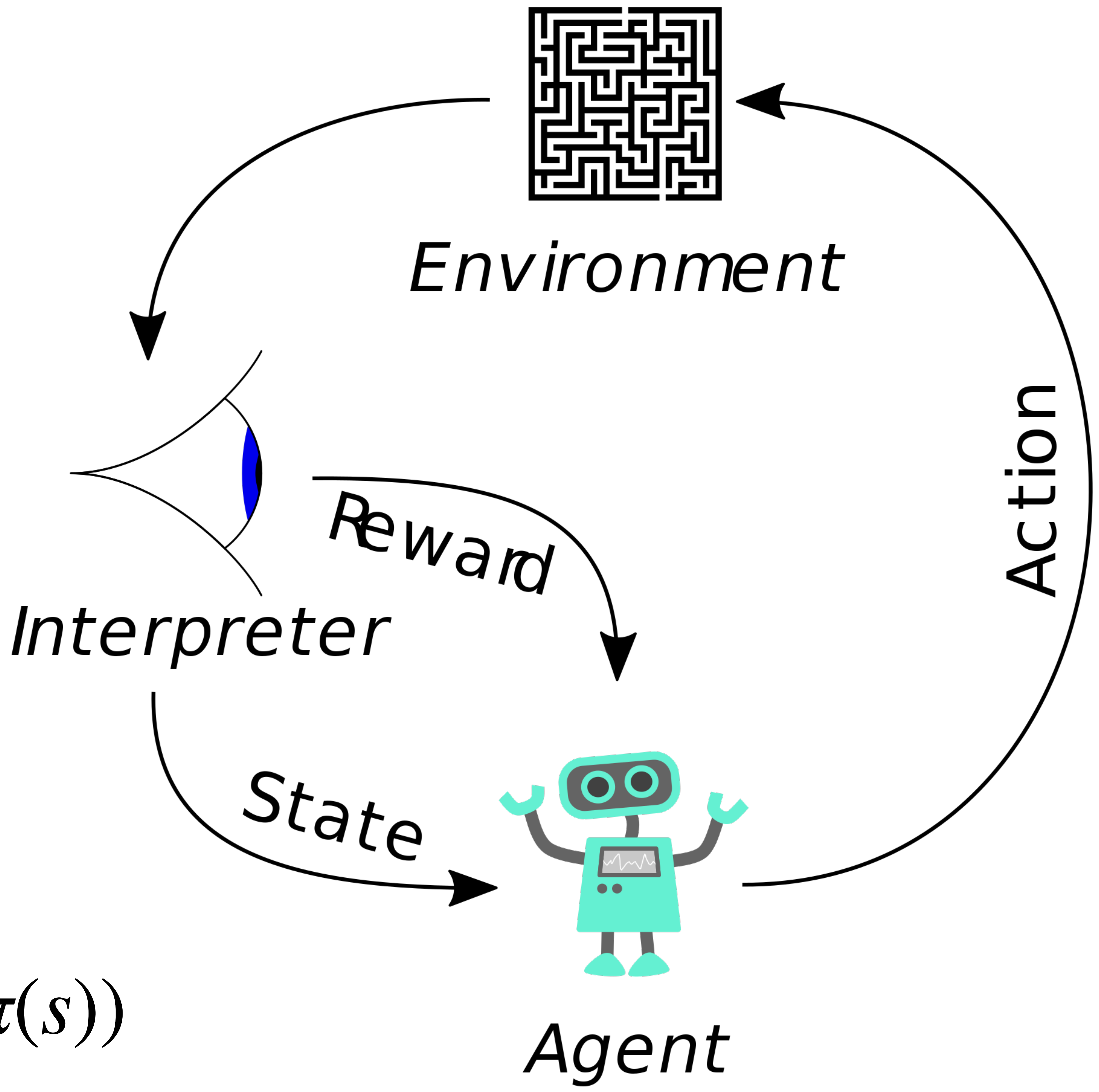
King's Indian Defense, known for Black's aggressive attack, was a big challenge for White until RL / search algorithms emerged

Screen shot from chess.com

Reinforcement learning: formal definition

- Markov Decision Processes (MDPs)
- For each time step $h = 1,2,\cdots,H$, the agent will:
  - Observe state s_h
  - Take action $a_h = \pi_h(s_h)$ by policy π
  - Receive reward $r_h(s_h, a_h)$
  - Transit to next state $s_{h+1} \sim \mathbb{P}(\cdot \mid s_h, a_h)$
- Goal: maximize the cumulative rewards:

$$Q_h^\pi(s,a) = \mathbb{E}\left[\sum_{h'=h}^{H} r_h(s,a) \,\middle|\, s,a\right], V_h^\pi(s) = Q_h^\pi(s,\pi(s))$$



By Megajuice - Own work, CC0, https://commons.wikimedia.org/w/index.php?curid=57895741

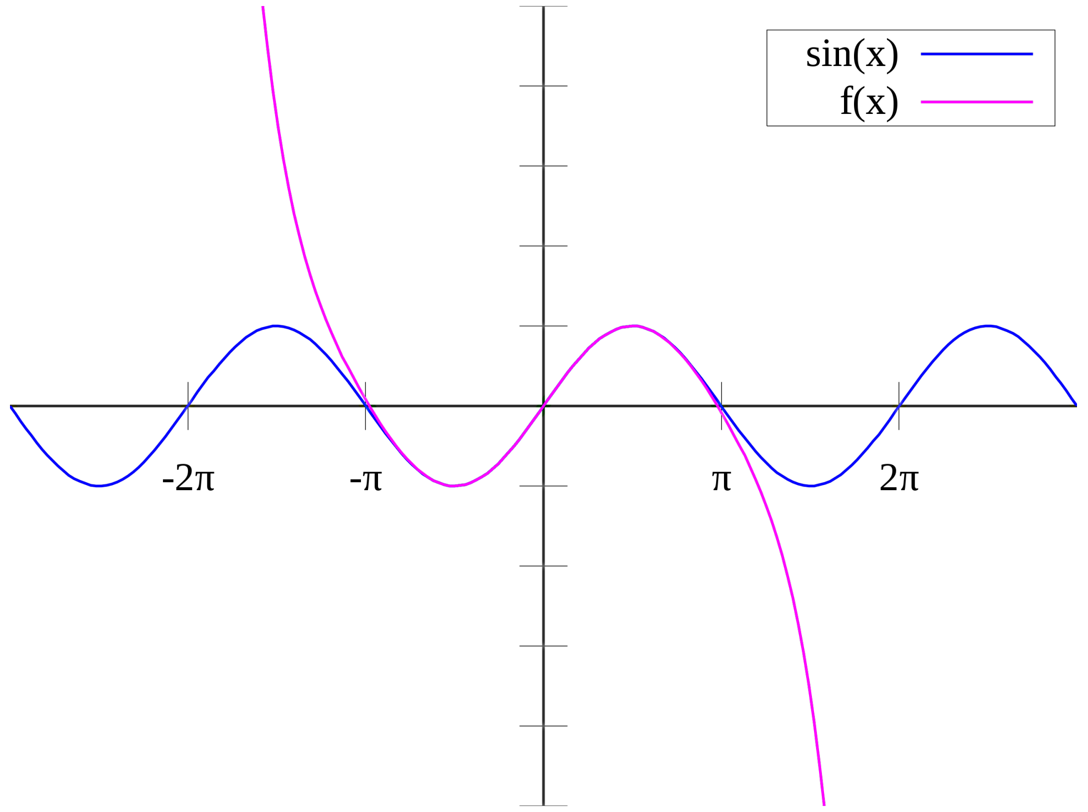
Towards large state space using function approximation

- Tabular methods are infeasible in practice
- Go game has approximately 10^{360} states
- Deep neural networks can perfectly extract features of the states
- RL with function approximation widely studied [JYWJ19, YW19, ZGS20]

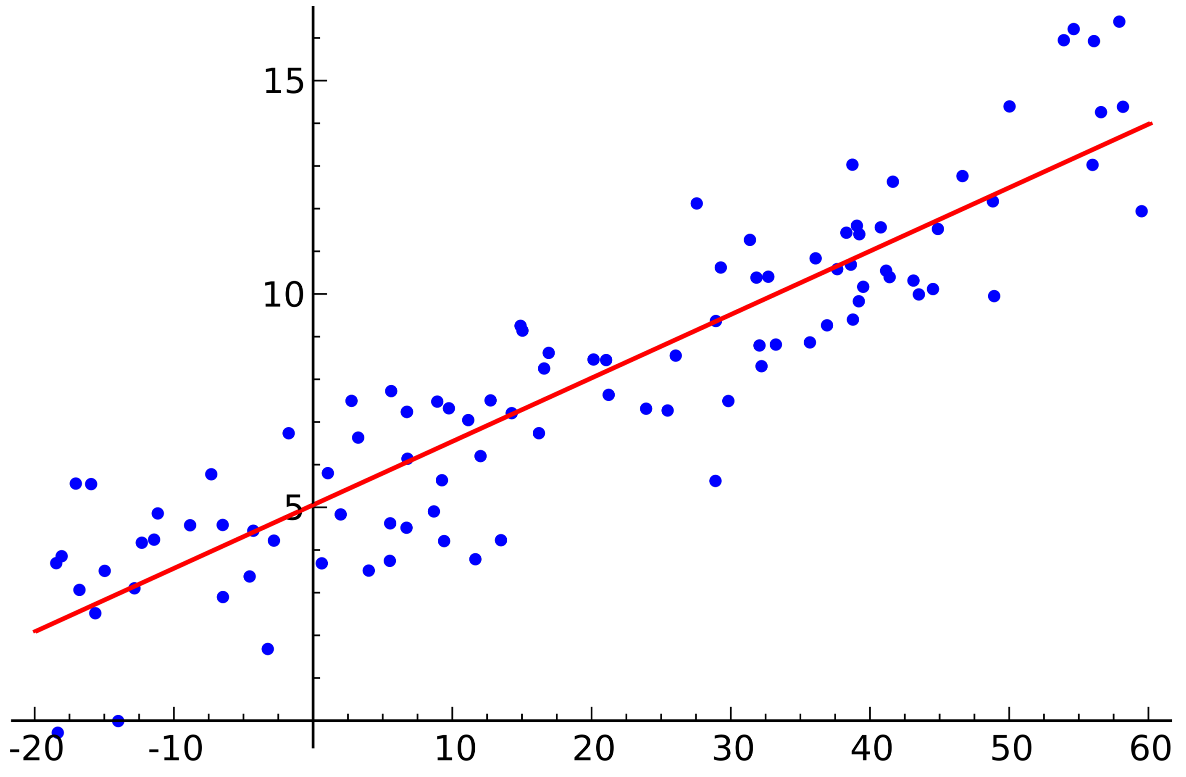
By Goban1 - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=15223468

Uncertainty in RL with function approximation

- Uncertainty is omnipresent!
  - Errors in the model…
  - Noise in the data…
  - Missing information…
- Uncertainty is important!
  - Performance issue
  - Efficiency issue
  - Fairness issue…



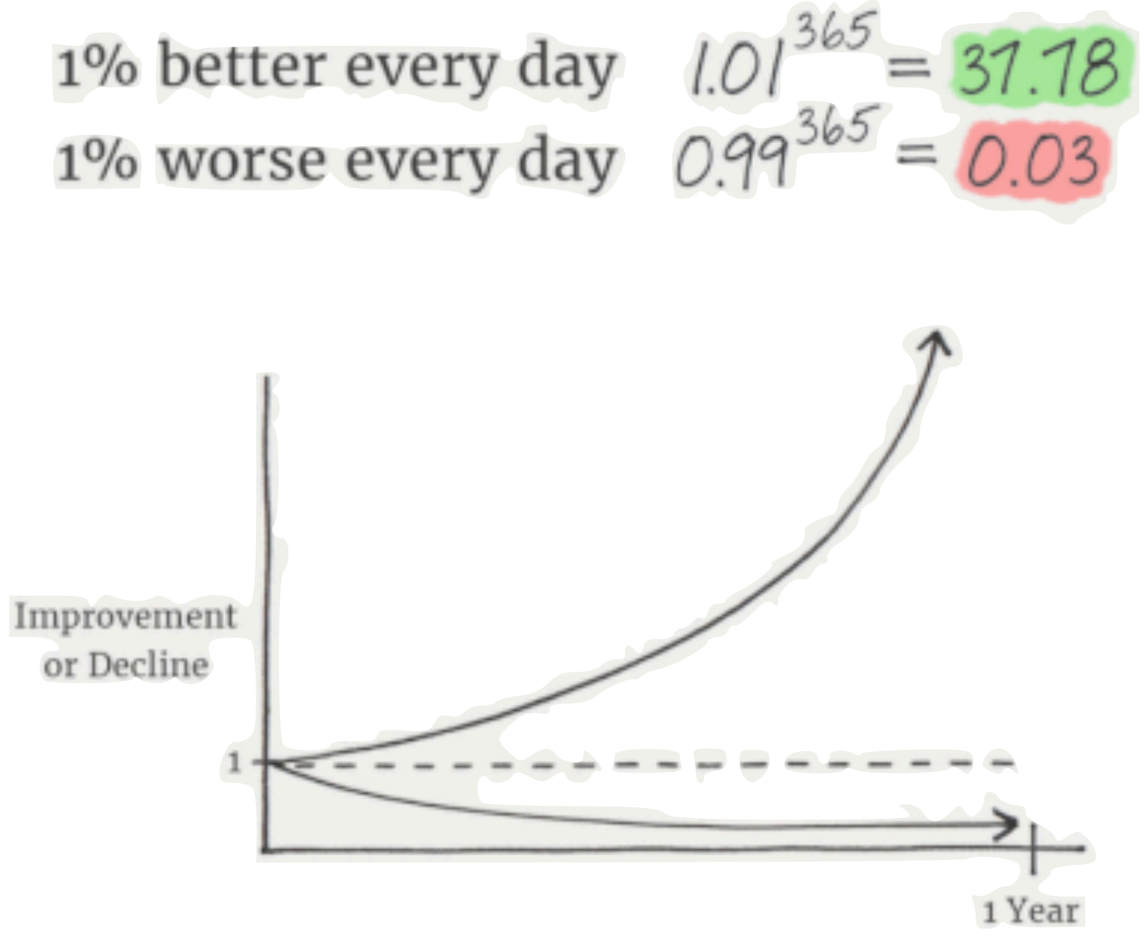
Uncertainty from an inaccurate model (Using Taylor approx.)



Uncertainty from the noise in the model

ID	Color	Weight	Broken	Class
1	Black	80	Yes	1
2	Yellow	100	No	2
3	Yellow	120	Yes	2
4	Blue	90	No	2
5	Blue	85	No	2
6	?	60	No	1
7	Yellow	100	?	2
8	?	40	?	1

Uncertainty from missing data



Minor uncertainty on beginning can lead to significant performance issue

Tasks in RL with Function Approximation

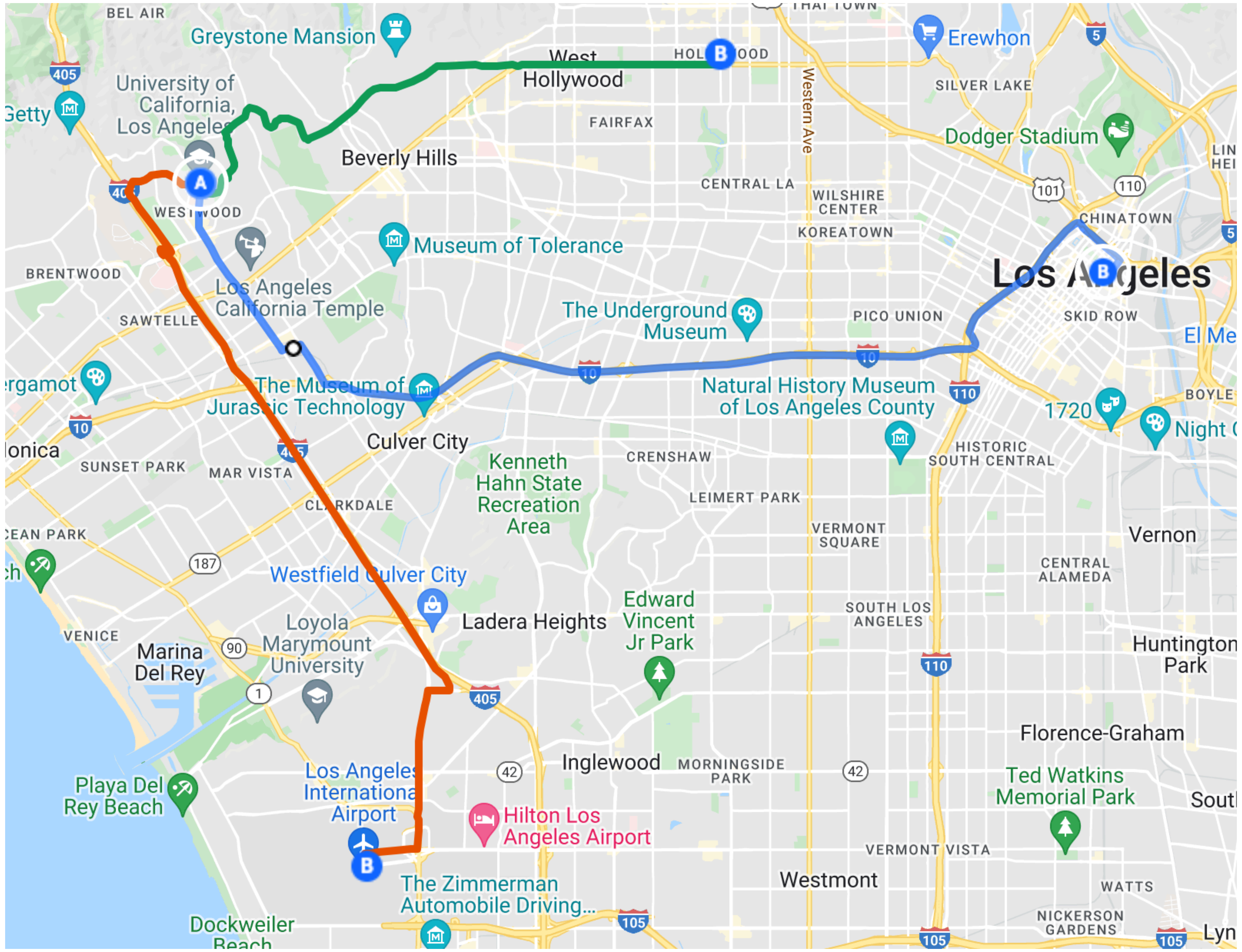
- [x] Exploration: Pure exploration without reward signals (this talk)
- [x] Model Misspecification Issue: Control the approximation error in the model
- [x] Representation Learning: Select good representation to improve performance
- [ ] Partially observed RL and non-Markovian RL: Missing information in current observation
- [ ] Fairness in RL: Make fair decision when uncertainty exists
- [ ] Deep RL: quantify the uncertainty in neural networks used in RL

Key technical issue: How to precisely quantify and utilize the uncertainty in RL with function approximation?

Using uncertainty to guide exploration in RL

Efficiency Challenge in Reinforcement Learning

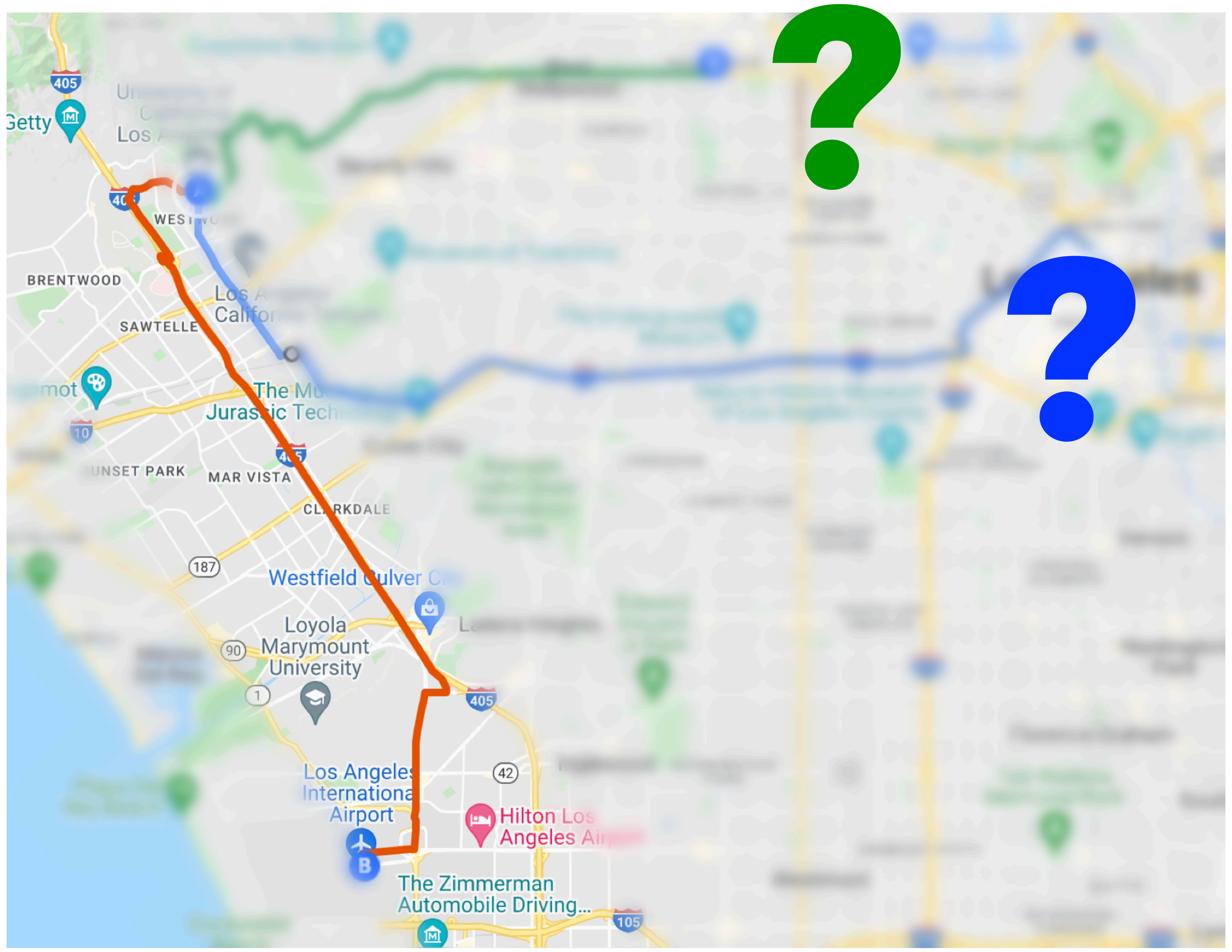
- Learning the environment (model) can be inefficient!
- Challenge: Can we reuse the model on different goals?
- E.g. a map (same environment) can be used to navigate to different destination (different targets)



We can use the same map of LA to get to DTLA, LAX or Hollywood from UCLA!

Why conventional reward-driven RL fails?

- Reward-guided exploration cannot well explore environment!
- Reward signal might not be given during the exploration.
  - *You will never know where you will go when constructing the map!*



Reward-guided exploration with target LAX cannot well explore the whole map

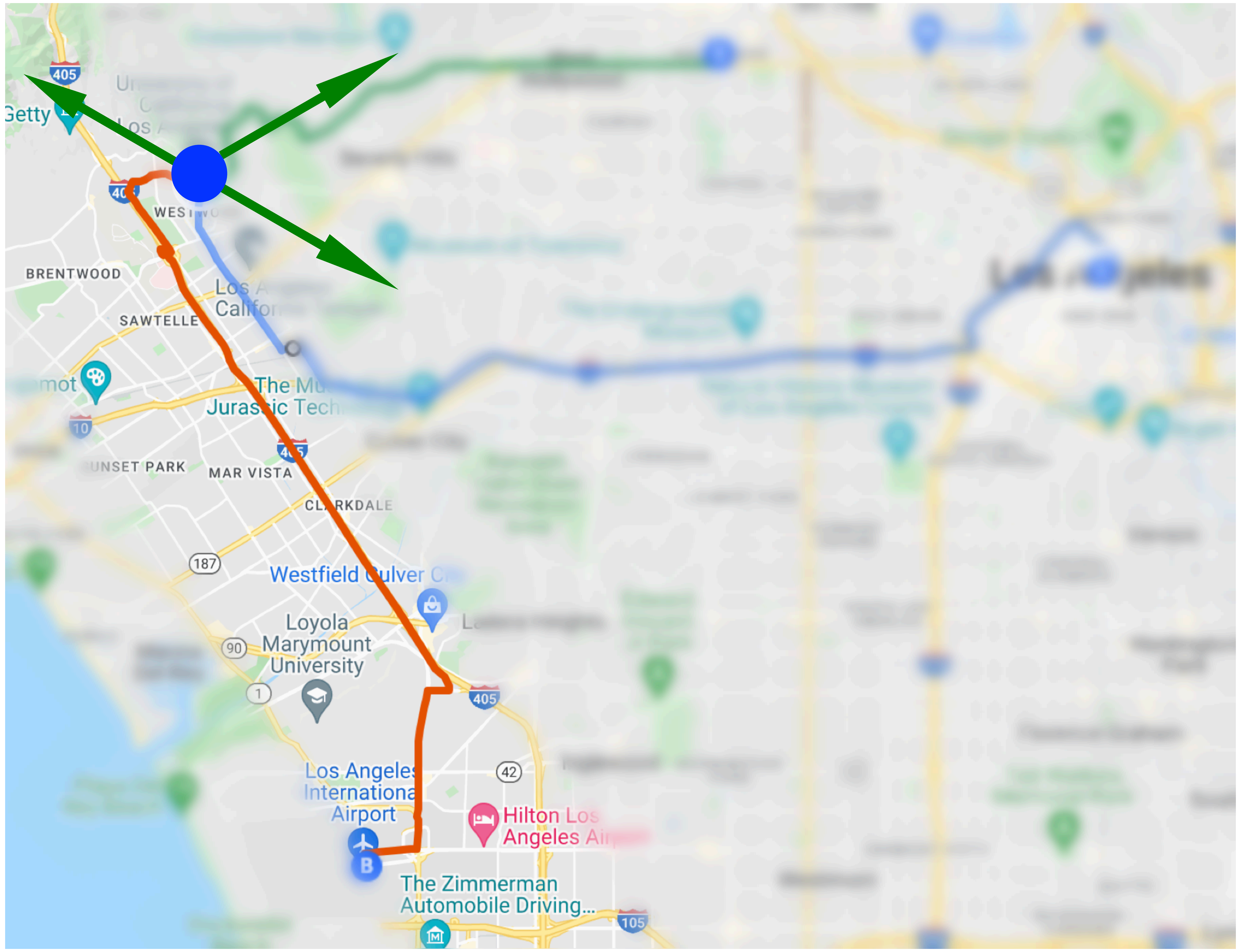
*How can we efficiently explore an environment **without** using any reward function?*

Reward Free Exploration Paradigm

- Reward-free exploration [JKSY20], pure exploration without reward signals
- Two phases algorithm:
  - Exploration Phase: Explore the environment for K episodes
    - No reward signals provided
    - e.g. starting from campus, drive until running out fuel for K times
    - Build estimation of the environment (e.g. road connection in the map)
  - Planning Phase: Given reward signals, plan a near optimal policy
    - No more exploration, only based on estimated environment
    - Can be done multiple times for different reward input
    - e.g. find route to different destinations (UCLA -> LAX, UCLA -> Hollywood, UCLA-> DTLA)

Reward-free exploration: Intuition

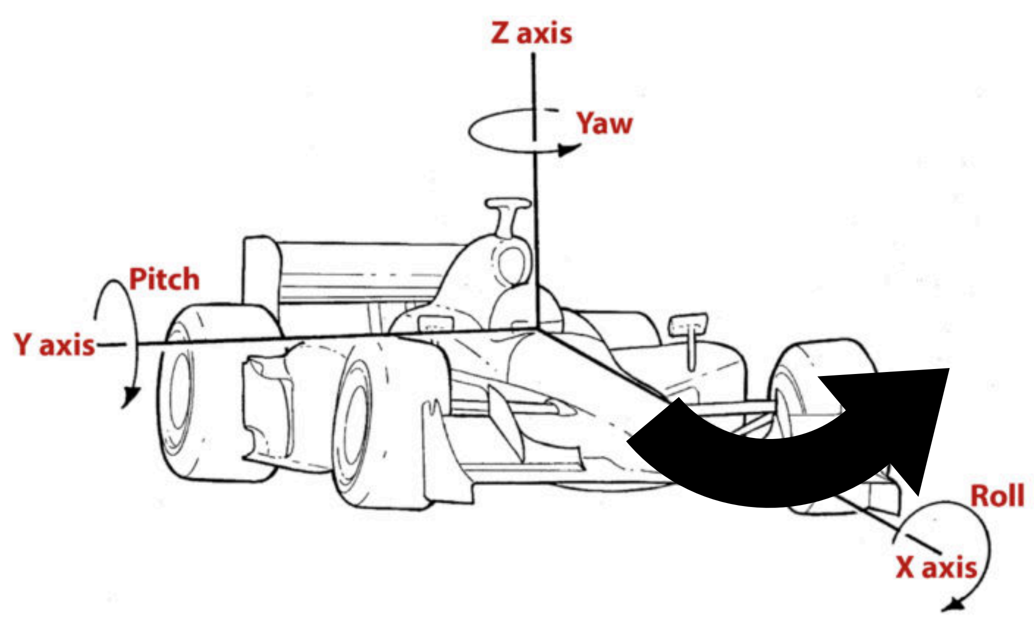
- Favoring uncertainty:
  - Explore the environment with more uncertainty!
  - How to measure the uncertainty?
- In tabular setting (i.e. $\mathbb{P}(s' \mid s, a)$ is explicitly represented by tables)
  - Visiting all states with reasonable probability [JKSY20]
  - Other following up work in tabular setting [MDJKLV20, ZDJ20, KMDJLV21]



We need to explore the areas where we are not sure (by green arrows) starting the initial state (red point)

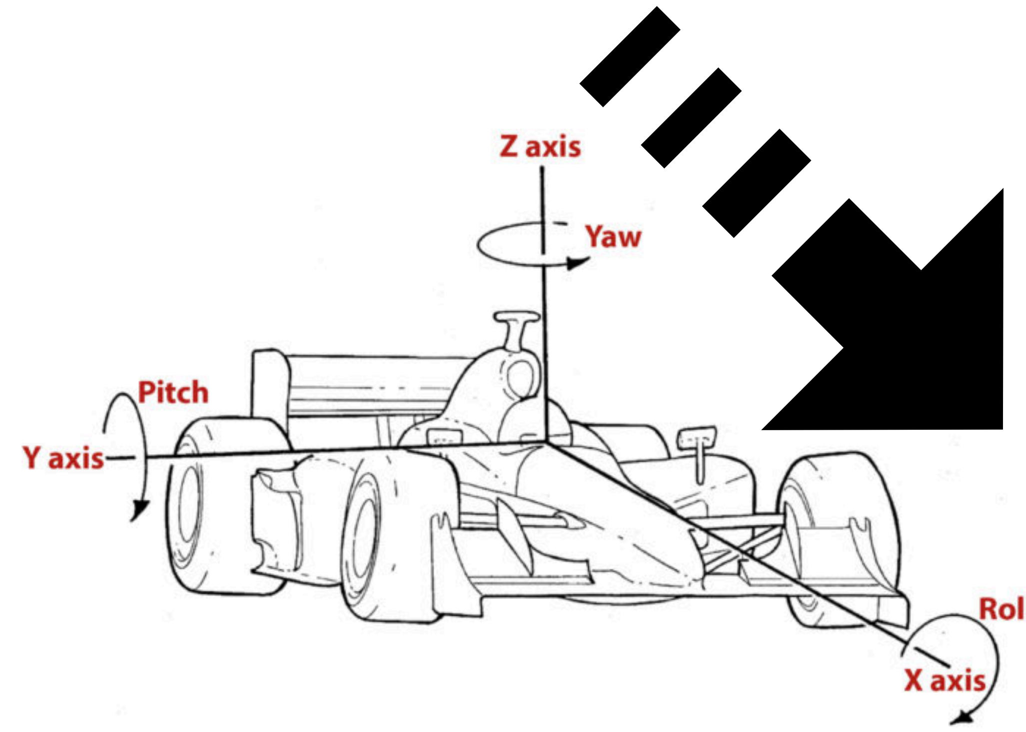
Function approximation: Mixture of distributions

- Linear Mixture MDPs [JYSW19, AJSWY20, ZHG20]
- Environment $\mathbb{P}(s'|s,a)$ is linear combination of several distributions
- Combining several components of the environment
- $$\mathbb{P}(s'|s,a) = \sum_{i=1}^{d} \theta_i \mathbb{P}_i(s'|s,a)$$



Simulated car dynamic when turning, with no acceleration

$\mathbb{P}_1(s'|s,a)$ (known)



Simulated car dynamic when accelerating

$\mathbb{P}_2(s'|s,a)$ (known)

θ_1 θ_2

A real car with both turning and accelerating

How to estimate the real car's dynamic?

$$\mathbb{P}(s'|s,a) = \theta_1 \mathbb{P}_1(s'|s,a) + \theta_2 \mathbb{P}_2(s'|s,a)$$

Linear Mixture MDPs: estimating total rewards

- Results from [JYSW19, AJSWY20, ZHG20]:

$$Q_h(s,a) = \underbrace{r_h(s,a)}_{\text{current reward}} + \underbrace{\mathbb{E}[V_{h+1} \mid s,a]}_{\text{future rewards}}$$

$$\mathbb{E}[V_{h+1} \mid s,a] = \sum_{i=1}^{d} \theta_i \mathbb{E}[V_{h+1}(s') \mid s,a,\mathbb{P}_i] = \langle \boldsymbol{\theta}, \boldsymbol{\psi}_{V_{h+1}}(s,a) \rangle$$

- Plan by Dynamic Programming! $Q_H \to V_H \to Q_{H-1} \to \cdots \to Q_1 \to V_1$

Uncertainty quantification for Linear Mixture MDPs

- Assume reward functions are known, no uncertainty
- Future uncertainty (Upper Confidence Bound):

$$\left| \hat{\mathbb{E}}[V_{h+1} \,|\, s, a] - \mathbb{E}[V_{h+1} \,|\, s, a] \right| \leq \beta \sqrt{\boldsymbol{\psi}_{V_{h+1}}^{\top}(s, a) \boldsymbol{\Sigma}^{-1} \boldsymbol{\psi}_{V_{h+1}}(s, a)}$$

Estimated Expectation

True Expectation

Confidence radius

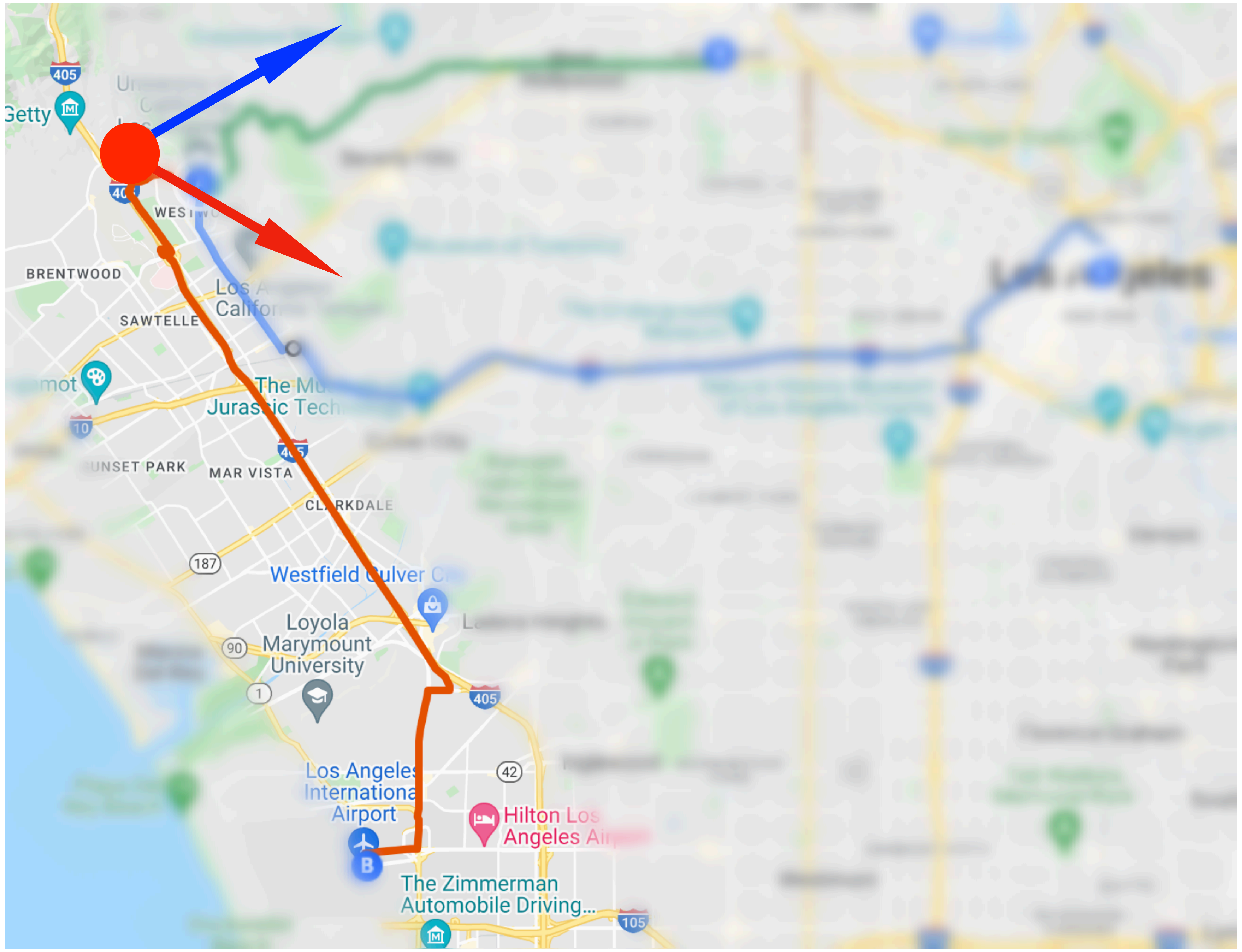
Covariance matrix

Reward free exploration: our approach (I)

- Encouraging exploration on unknown components
- Exploration driven reward function
  - Intuition: favoring actions lead to more uncertainty
  - $r(s,a) \propto \sqrt{\boldsymbol{\psi}_{V_{h+1}}^{\top}(s,a)\boldsymbol{\Sigma}^{-1}\boldsymbol{\psi}_{V_{h+1}}(s,a)}$
  - Issue: no reward, no value function V_{h+1}!
  - Solution: $r(s,a) \propto \sqrt{\max_{f:\mathcal{S}\mapsto\mathbb{R}} \boldsymbol{\psi}_{f}^{\top}(s,a)\boldsymbol{\Sigma}^{-1}\boldsymbol{\psi}_{f}(s,a)}$

Exploration Driven reward function: Explained

- $r(s,a) \propto \sqrt{\boldsymbol{\psi}_{V_{h+1}}^{\top}(s,a)\boldsymbol{\Sigma}^{-1}\boldsymbol{\psi}_{V_{h+1}}(s,a)}$
  - Encouraging exploration with more uncertainty towards a fixed next-step V_{h+1} (e.g. DTLA)
- $r(s,a) \propto \sqrt{\max_{f:\mathcal{S}\mapsto\mathbb{R}} \boldsymbol{\psi}_{f}^{\top}(s,a)\boldsymbol{\Sigma}^{-1}\boldsymbol{\psi}_{f}(s,a)}$
  - Encouraging exploration with more uncertainty globally



Red arrow: exploration towards a fixed goal (DTLA)
Blue arrow: exploration for a global uncertainty

Reward free exploration: our approach (II)

- Using more informative data for regression
- Previous ridge regression in UCRL-VTR [JYSW19, AJSWY20, ZHG20]

$$\mathbb{E}[V(s') \mid s, a] = \langle \boldsymbol{\theta}^*, \boldsymbol{\psi}_V(s, a) \rangle$$

$$\boldsymbol{\theta} = \arg\min_{\boldsymbol{\theta}} \sum_{(s,a,s')} \left(V(s') - \langle \boldsymbol{\theta}, \boldsymbol{\psi}_V(s, a) \rangle \right)^2 + \lambda \|\boldsymbol{\theta}\|_2^2$$

Value function from last step

- Why use V? Any function can be used to regression!
- Our approach: Uncertainty based regression targets.

Uncertainty based regression targets

- Uncertainty of any function f on fresh sampled data s, a, s':
  - $\left| \hat{\mathbb{E}}[f(s') \mid s, a)] - \mathbb{E}[f(s') \mid s, a)] \right| \leq \beta \sqrt{\boldsymbol{\psi}_f^\top(s, a)\boldsymbol{\Sigma}^{-1}\boldsymbol{\psi}_f(s, a)}$

Confidence radius

Covariance Matrix

- Conjecture: larger uncertainty ⇔ learn more with that target

$$\boldsymbol{\theta} = \arg\min_{\boldsymbol{\theta}} \sum_{(s,a,s')} \left(f(s') - \langle \boldsymbol{\theta}, \boldsymbol{\psi}_f(s, a) \rangle \right)^2 + \lambda \|\boldsymbol{\theta}\|_2^2$$

- $u = \arg\max_{f: \mathcal{S} \mapsto \mathbb{R}} \left(\boldsymbol{\psi}_f^\top(s, a)\boldsymbol{\Sigma}^{-1}\boldsymbol{\psi}_f(s, a) \right)$ as the target!

Algorithm pipeline: uncertainty based exploration and regression

- Initialize parameter $\boldsymbol{\theta}$, covariance matrix $\boldsymbol{\Sigma}$
- Exploration driven reward function in exploration
    - $r(s,a) \propto \sqrt{\max_{f:\mathcal{S}\mapsto\mathbb{R}} \boldsymbol{\psi}_f^\top(s,a)\boldsymbol{\Sigma}^{-1}\boldsymbol{\psi}_f(s,a)}$
    - Sample new data (s,a,s') by maximizing the cumulative reward r (by UCRL-VTR [JYSW19, AJSWY20, ZHG20])
- Learning the model efficiently using uncertainty

  $u = \arg\max_{f:\mathcal{S}\mapsto\mathbb{R}} \left(\boldsymbol{\psi}_f^\top(s,a)\boldsymbol{\Sigma}^{-1}\boldsymbol{\psi}_f(s,a)\right)$, add $u(s')$ and $\boldsymbol{\psi}_u(s,a)$ into regression
- Update $\boldsymbol{\theta}, \boldsymbol{\Sigma}$ by ridge regression $\boldsymbol{\theta} = \arg\min_{\boldsymbol{\theta}} \sum_{(s,a,s')} \left(u(s') - \langle \boldsymbol{\theta}, \boldsymbol{\psi}_u(s,a)\rangle\right)^2 + \lambda\|\boldsymbol{\theta}\|_2^2$

Repeat K times

Theoretical Results

Theorem (Sample Complexity)

Let the parameters be properly set, for any $0 < \epsilon < 1$, if $K = \tilde{\mathcal{O}}\left(H^5 d^2 \epsilon^{-2}\right)$, with probability at least $1 - \delta$, for any reward function r, the algorithm can provide a policy such that $V_1^*(s; r) - V_1^\pi(s; r) \leq \epsilon$

Take aways

- Collecting $K = \tilde{\mathcal{O}}\left(H^5 d^2 \epsilon^{-2}\right)$ is enough to estimate a good enough $\boldsymbol{\theta}$ (close to $\boldsymbol{\theta}^*$)
- A well estimated $\boldsymbol{\theta}$ is enough to provide a (near) optimal policy for any reward function
- Learning a longer MDP (larger H) is more difficult (why)
  - Longer MDP $\Leftrightarrow$ Larger value function
  - Longer MDP $\Leftrightarrow$ Larger noise in sampling
- Dependency on d and ϵ is standard in linear regression tasks

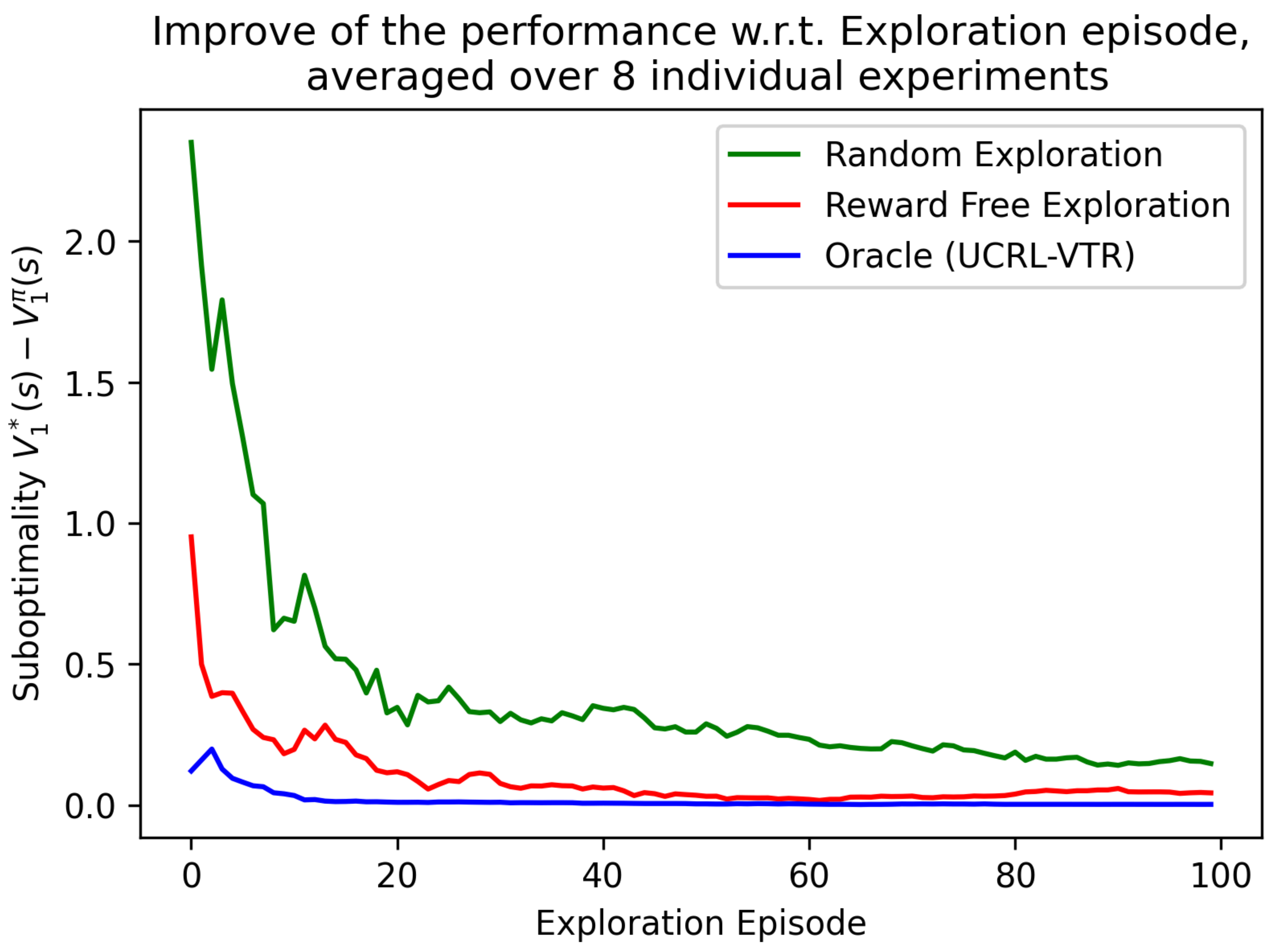
Comparison to Related Works

Algorithm	Setting	Model Based / Model Free	Sample Complexity
[JKSY20]	Tabular	Model Based	$\tilde{\mathcal{O}}(H^5S^2A\epsilon^{-2})$
[KMDJLV21]	Tabular	Model Based	$\tilde{\mathcal{O}}(H^4S^2A\epsilon^{-2})$
[MDJKLV20]	Tabular	Model Based	$\tilde{\mathcal{O}}(H^3S^2A\epsilon^{-2})$
[ZDJ20]	Tabular	Model Based	$\tilde{\mathcal{O}}(H^2S^2A\epsilon^{-2})$
Lower bound [JKSY20]	Tabular	Model Based	$\Omega(H^2S^2A\epsilon^{-2})$
[WDYS20]	Linear MDP	Model Free	$\tilde{\mathcal{O}}(H^5d^3\epsilon^{-2})$
[ZLKB20]	Linear MDP	Model Free	$\tilde{\mathcal{O}}(H^5d^3\epsilon^{-2})$
Lower bound [WDYS20]	Linear MDP	Model Free	Not Applicable
UCRL-RFE (ours)	Linear Mixture MDP	Model Based	$\tilde{\mathcal{O}}(H^5d^2\epsilon^{-2})$
UCRL-RFE+ (ours, improved)	Linear Mixture MDP	Model Based	$\tilde{\mathcal{O}}(H^4d(H+d)\epsilon^{-2})$
Lower bound (ours)	Linear Mixture MDP	Model Based	$\Omega(H^2d\epsilon^{-2})$
Lower bound (Improved) [CHYW21]	Linear Mixture MDP	Model Based	$\Omega(H^2d^2\epsilon^{-2})$

Sample complexity of the reward free exploration algorithms, the time-inhomogeneous results are translated to time-homogeneous results by removing an H factor

Experiment Results

- $d = 3,\ \mathcal{S} = 10,\ \mathcal{A} = 5,\ H = 10,\ K = 100$
- Random Exploration:
  - Random exploration policy
  - Random regression target
- Oracle (UCRL-VTR) [JYSW19, AJSWY20, ZHG20]:
  - Know the reward function during exploration
- Reward Free Exploration (Ours):
- Reward free exploration is significantly better than random exploration!



Further extensions

- Explore to non-linear function approximation
  - Once we know the uncertainty, we can use that to guide exploration
- Improve the theoretical bounds (Current upper bound: $\mathcal{O}(H^5 d^2 \epsilon^{-2})$)
  - Lower bound is improved to $\Omega(H^2 d^2 \epsilon^{-2})$ from $\Omega(H^2 d \epsilon^{-2})$ [CHYW21]
  - The dependency on H still have a large gap
- Empirical results can be applied to modern model-based algorithms.

Summary

Quantify uncertainty of RL with function approximation can ….

- ☑ Guide the exploration in RL
- ☑ Improve the efficiency of learning the parameters

The uncertainty of function approximated RL can be quantified by …

- ☑ Precisely controlled with linear function approximation
- ☑ Easy to immigrate to other models (e.g. neural networks [ZZLQ20])

My future plan

- [x] ~~Exploration: Pure exploration without reward signals~~
- [x] ~~Model Misspecification Issue: Control the approximation error in the model~~
- [x] ~~Representation Learning: Select good representation to improve performance~~

- [ ] Partially observed RL and non-Markovian RL: Missing information in current observation
- [ ] Fairness in RL: Make fair decision when uncertainty exists
- [ ] Deep RL: quantify the uncertainty in neural networks used in RL
- ○ Practical algorithms using modern neural networks, on modern tasks..

Thank you!