Solving a problem via abstraction follows a generic formula. Transform the the problem into a new domain, solve the problem in this new domain, project the solution back into the original domain. In this section, we explore the way to design an abstraction so that it makes the optimisation problem "easier".

Firstly, what makes an optimisation problem easy? The search space is small, the search space has some structure or symmetry within it, allowing us to reduce the search space to something small(er), the search space gives us hints about how to search for what we are looking for (for example smooth and convex with respect to our loss function convex RL …), …

And within RL, properties that would make the optimisation problem easier; a sparse transition matrix, dense rewards, linearity, …

Note that it will not always be possible to find an efficient solution to an MDP. Are some MDPs just fundammentally harder to solve than others? Could mention no free lunch.

A quick review of MDPs

The bellman equation is a non-linear optimisation problem. Is there a way to turn this into a linear problem? What do we need to sacrifice to do so?

$$v(s) = \max_a \left[r(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim P(\cdot|s,a)} v(s') \right] \tag{1}$$

$$\tag{2}$$

^^^ What makes it non-linear?!?

Linear markov decision problems (LMDPs)

Define an infinite horizon LMDP to be $\{S, p, q, \gamma\}$. Where S is the state space, $p : S \to \Delta(S)$ is the unconditioned transition dynamics, $q : S \to \mathbb{R}$ is the state reward function an γ is the discount rate.

How can we exploit linearity to make reinforcement learning easier to understand, more efficient.

There are a few different ways we can introduce linearity to a MDP. Which one is best? We will see…

why linearity? - it has many mathematical tools for analysis. - we know linear systems can be solved efficiently. - ?

Linearity is a nice property that makes optimisation simpler and more efficient.

- Linear programming (see appendix: LP)

- Linear markov decision processes

Linear optimisation is … aka linear programming. Has a complexity of ???. Can Solving a system of linear relationships. Has a complexity of ???.

In fact. MDPs can actually be solved via LP. see [appendix].

Linear Markov decision process (Todorov 2009)

(Exponentiated and controlling state distributions)

How can we remove the sources of non-linearity from the bellman equation? The answer is a couple of "tricks";

- rather than optimising in the space of actions, optimise in the space of possible transition functions.
- set the policy to be
- ?

Let's unpack these tricks and see how they can allow us to convert an MDP into a linear problem. And what the conversion costs.

$$V(s) = \max_u \Big[q(s) - \mathsf{KL}(u(\cdot|s) \parallel p(\cdot|s)) + \gamma \mathop{\mathbb{E}}_{s' \sim u(\cdot|s)} V(s')\Big] \tag{3}$$

$$\tag{4}$$

Linear Markov decision process (Pires el at. 2016)

(Factored linear models)

$$\mathcal{R} : \mathcal{V} \to W \tag{5}$$

$$\mathcal{Q}_a : \mathcal{W} \to \mathcal{V}^A \tag{6}$$

$$P(s'|s,a) = \int_w \mathcal{Q}_a(s',w)\mathcal{R}(w,s) \tag{7}$$

$$\tag{8}$$

Great. But, how does this help?

$$
\begin{aligned}
T_{\mathcal{Q}}w &= r + \gamma \mathcal{Q}w & (9)\\
T_{\mathcal{R}^A\mathcal{Q}}w &= \mathcal{R}^A T_{\mathcal{Q}} & (10)\\
T_{\mathcal{Q}\mathcal{R}} &= T_{\mathcal{Q}}\mathcal{R} \quad (= T_P) & (11)
\end{aligned}
$$

It allows us to Bellman iterations in a lower dimensional space, $\mathcal{W}$, rather than the dimension of the transition function.

$$
\begin{aligned}
w^a &= T_{\mathcal{R}^A\mathcal{Q}}w & \text{(bellman evaluation operator)}\\
w &= M'w^a & \text{(greedy update)}\\
& & (12)
\end{aligned}
$$

When does this matter? Planning!! Simulating the transition function many times. Pick $\mathcal{W}\dots$

Linear Markov decision process (Jin el at. 2019)

()

$$
\begin{aligned}
P(\cdot|s,a) &= \langle \phi(s,a), \mu(\cdot) \rangle & (13)\\
r(s,a) &= \langle \phi(s,a), \theta \rangle & (14)
\end{aligned}
$$

Discussion

So which approach is best? What are the pros / cons of these linearisations?

All of them are trying to insert some kind of linearity into the transition function.

A closer look at LMDPs

(the Todorov ones…)

A few things I want to explore; - the composability of policies / tasks - the embedding of actions - LMDPs as an abstraction

Insert section on theory of LMDPs. Convergence, approx error, …__

What are their properties?

- Efficently solvable
- Allows the composition of optimal controls.
- ???

And what are they lacking?

- Assumes we are working with a tabular representation

-

So now that we have explored LMDPs, how can we extract their nice properties into an architecture that might scale to more complex problems: larger state spaces and action spaces, sparse rewards, …?

The key steps that were taken;

- Exponentiated values
- learn a policy that chooses state distributions, rather than actions.

LMDPs; more formally

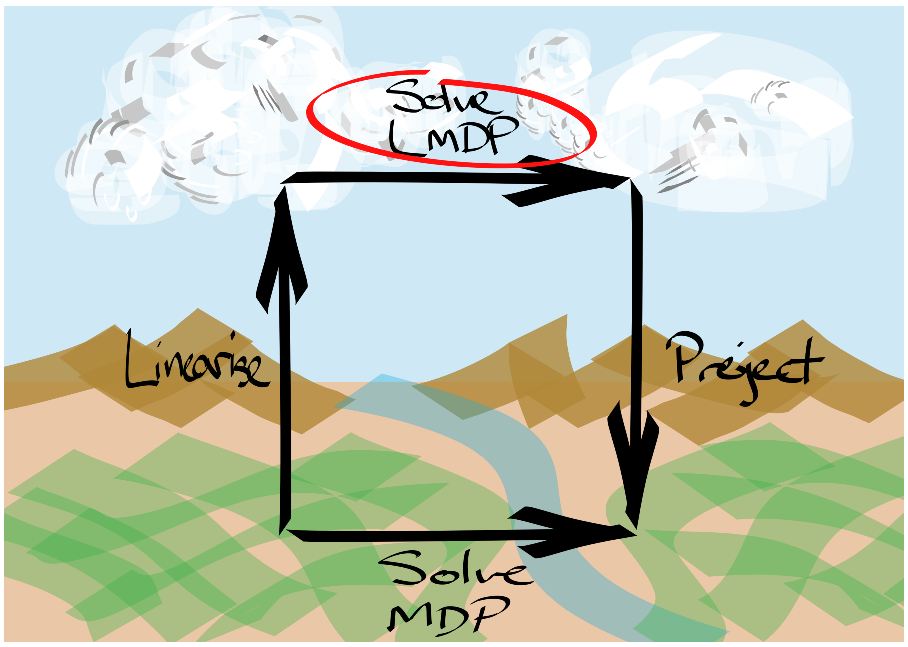

Figure 1: ”

Pick $a \in A$, versus, pick $\Delta(S)$. $f : S \to A$ vs $f : S \to \Delta(S)$.

In the original Todorov paper, they derive the LMDP equations for minimising a cost function. This maximisation derivation just changes a few negative signs around. Although there is also a change in the interpretation of what the unconstrained dynamics are doing. …?

$$V(s) = \max_u q(s) - \mathsf{KL}(u(\cdot|s) \parallel p(\cdot|s)) + \gamma \mathop{\mathbb{E}}_{s' \sim u(\cdot|s)} V(s') \tag{1}$$

$$\tag{15}$$

$$u^*(\cdot|s) = \frac{p(\cdot|s) \cdot z(\cdot)^\gamma}{\sum_{s'} p(s'|s) z(s')^\gamma} \tag{8}$$

$$z_{u^*} = e^{q(s)} \cdot P z_{u^*}^\gamma \tag{11}$$

$$\tag{16}$$

By definition, an LMDP is the optimisation problem in (1). (3) Define a new variable, $z(s) = e^{v(s)}$. (5) Define a new variable that will be used to normalise $p(s'|s)z(s')^\gamma$. (8) Set the optimal policy to minimise the KL distance term. (9) Since we picked the optimal control to be the form in (8), the KL divergence term is zero. (11) Rewrite the equations for the tabular setting, giving a z vector, uncontrolled dynamics matrix.

(see appendix [] for a full derivation)

Main transformations of the LMDP, everything else follows.

1. Allow the direct optimisation of transitions, $u(s'|s)$, rather than policies.
2. $r(s,a) = q(s) + KL(P(\cdot|s,a) \parallel p(s'|s)), \forall s, a$
3. Set the optimal policy to be …

Another way to frame. **Q:** If we want to optimise the space of transitions, what augmentations of the MDP are necessary to ensure solutions in the LMDP are optimal in the MDP?

- Prove that 2. is necessary and sufficient for optimality. (probs not possible?!)

A linearisation of an MDP

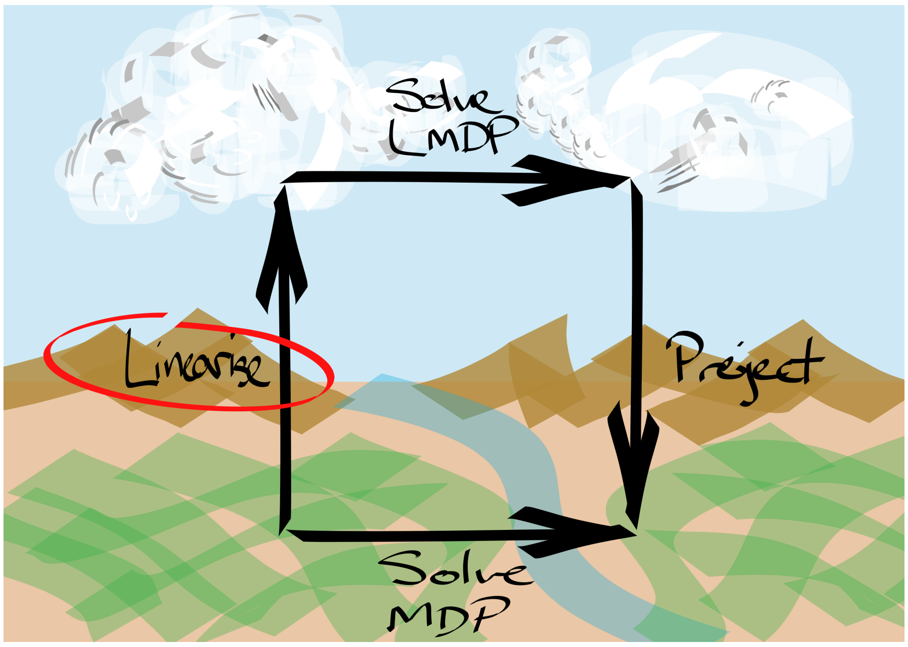

Figure 2: ”

> Ok great, we can solve LMDPs. But how does being able to solve an LMDP help us solve MDPs?

We want a way to transform a MDP into a LMDP, while preserving the “structure” of the MDP. But what do we mean by a MDP’s structure?

The LMDP, $\{S, p, q, \gamma\}$ should;

- be able to represent the same transition dynamics as the original MDP,
- give the the same rewards was the original MDP,
- have the same optima.

(It turns out that (1) and (2) imply (3) given some assumptions. See Optimality)

So, given a reward function, r, and a transition function, P, from the MDP, we must translate them into a p and a q. Thus we have built a LMDP with the same “structure”.

$$\forall s, s' \in S, \forall a \in A, \exists u_a \text{ such that;} \tag{17}$$

$$P(s'|s,a) = u_a(s'|s)p(s'|s) \tag{1}$$

$$r(s,a) = q(s) - \text{KL}(P(\cdot|s,a) \parallel u_a(\cdot|s)) \tag{2}$$

$$\tag{18}$$

Which leads to $|A|$ linear equations to solve, for each state in the MDP.

See appendix [] for more details.

Alternative views of linearisation.

- A relaxation of the MDP
- Linelihood interpretation

Unconstrained dynamics and state rewards

> Let's try and understand this thing we have contructed.

The state rewards are not capable of giving rewards for actions taken. Rather, the differences in reward, by taking another action, is captured by the KL divergence between the control and the unconstrained dynamics.

- What is their function?
- What do they look like?

Does it make sense to treat the q(s) like rewards?! They reward for bing in state s. But cant capture action specific rewards!?

Decoding

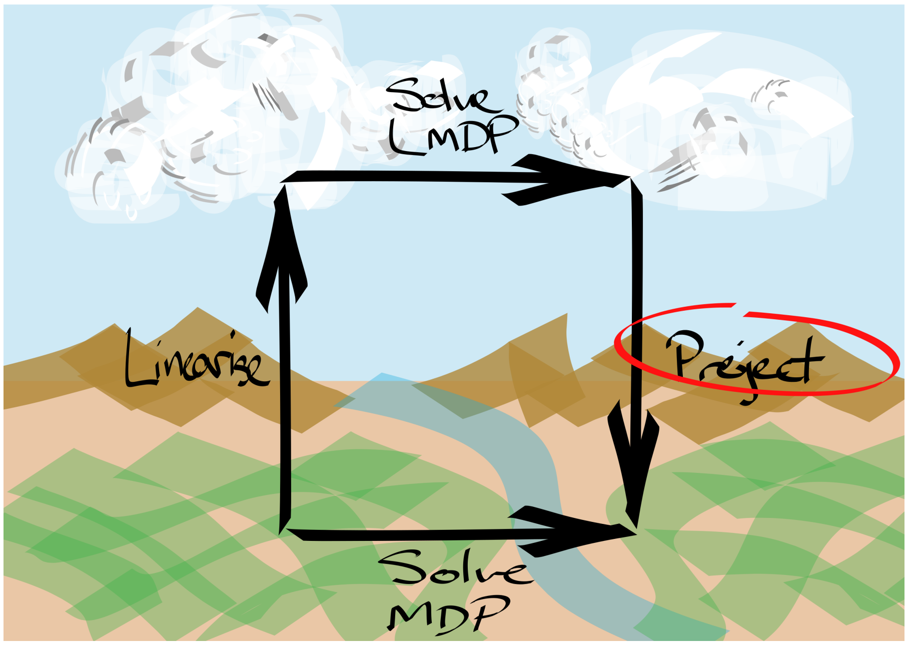

Figure 3: ”

Ok, so now we get a glimpse at why LMDPs are an interesting abstraction. THe LMDP has disentangled the search for the behaviour (go to this or that state) and the search for optimal controls (how to actually achieve that behaviour). This can be seen in the decoding step. As we know which states we want to be in, via the optimal control from solving the LMDP, u^*, but, we do not know how to implement those controls using the actions we have available.

$$P_\pi(\cdot|s) = \sum_a P_k(\cdot|s,a)\pi(a|s) \tag{19}$$

$$\pi = \underset{\pi}{\text{argmin}}\ \text{KL}\Big(u(\cdot|s)) \parallel P_\pi(\cdot|s)\Big) \tag{20}$$

Maybe this isnt enough? Do we need to add a reward sensitive part as well?!? (but what if the actual path we take to get there has a neg rewards?!?)

Optimality of solutions via LMDPs

> Do these two paths lead to the same place?

One of the main questions we have not addressed yet is; if we solve the MDP directly, or linearise, solve and project, do we end up in the same place? This is a question about the completeness of our abstraction. Can our abstraction represent (and find) the same solutions that the original can?

$$\begin{aligned}
\parallel V_{\pi^*} - V_{\pi_{u^*}} \parallel_\infty &= \epsilon \qquad (1)\\
&= \parallel (I-\gamma P_{\pi^*})^{-1} r_{\pi^*} - (I-\gamma P_{\pi_{u^*}})^{-1} r_{\pi_{u^*}} \parallel_\infty \qquad (2)\\
&\le \parallel (I-\gamma P_{\pi^*})^{-1} r - (I-\gamma P_{\pi_{u^*}})^{-1} r \parallel_\infty \qquad (3)\\
&= \parallel \Big((I-\gamma P_{\pi^*})^{-1} - (I-\gamma P_{\pi_{u^*}})^{-1} \Big) r \parallel_\infty \qquad (4)\\
&\le r_{\text{max}} \parallel (I-\gamma P_{\pi^*})^{-1} - (I-\gamma P_{\pi_{u^*}})^{-1} \parallel_\infty \qquad (5)\\
&= r_{\text{max}} \parallel \sum_{t=0}^{\infty} \gamma^t P_{\pi^*} - \sum_{t=0}^{\infty} \gamma^t P_{\pi_{u^*}} \parallel_\infty \qquad (6)\\
&= r_{\text{max}} \parallel \sum_{t=0}^{\infty} \gamma^t (P_{\pi^*} - P_{\pi_{u^*}}) \parallel_\infty \qquad (7)\\
&= \frac{r_{\text{max}}}{1-\gamma} \parallel P_{\pi^*} - P_{\pi_{u^*}} \parallel_\infty \qquad (7)
\end{aligned} \tag{21}$$

(1) We want to compare the optimal policies value and the value achieved by the optimal LDMP

solution.

(2) Assume that there exists a policy that can generate the optimal control dynamics (as given by the LMDP). In that case we can set $P_{\pi_{u^*}} = U^*$.
(3) r_{u^*} doesnt really make sense as the reward is action dependent. We could calculate it as $r_{\pi_{u^*}}$, but we dont explicity know π_{u^*}. $(I - \gamma P_{\pi^*})^{-1} r$ represents the action-values, or Q values. By doing this exhange, we might over estimate the diffference under the infinity norm as two non-optimal actions may have larger difference. Also, use the element wise infinity norm.

Notes

- why are we using the infinity norm?!!
- ?

What does $\delta \geq$ KL$(P_{\pi^*} \parallel U^*)$ imply about $\parallel P_{\pi^*} - P_{\pi_{u^*}} \parallel_\infty$?

$$Q = r + \gamma P \cdot_{(s')} V \tag{22}$$
$$Q = (I - \gamma P_\pi)^{-1} r??? \tag{23}$$
$$\tag{24}$$

Ok, great. Insights from optimality bounds.

Need to be able to approximate the optimal controls. When is it hard to approximate the optimal controls? When our basis set of distributions oer future states (aka our actions) have little weight…?

Potential solution? Use options.

Option decoding

What about using options to help solve the optimal control decoding? Does this actually help?!

$$P_\pi(\cdot|s) = \sum_\omega P_k(\cdot|s, \omega)\pi(\omega|s) \tag{25}$$
$$\pi = \underset{\pi}{\text{argmin}}\, \text{KL}\Big(u(\cdot|s)) \parallel P_\pi(\cdot|s)\Big) \tag{26}$$

Options would allow greater flexibility in the $P_\pi(\cdot|s)$ distribution, making is possible to match $u(s'|s)$ with greater accuracy (and possibly cost).

- First need to demonstrate that action decoding is lossy.
- Then show that using options is less lossy.

The complexity of solutions via LMDPs

> Is my path actually shorter?

The whole point of this abstraction was to make the problem easier to solve. So hasit actually made it any easier?

The complexity of solving our abstraction can be broken down into the three steps;

- linearisation: $|S| \times \text{min}(|S|, |A|)^{2.3}$
- solve the LMDP: $\text{min}(|S|, |A|)^{2.3}$
- project back: $???$

Giving a total complexity of $\dots$

Contrasted with the complexity of solving an MDP.

Scaling to more complex problems

Now that we have some evidence that this LMDP solution strategy makes sense, it efficiently (see complexity) yields high value (see optimality) policies. We want to test it out on some real world problems. But the real world isn't as nice as the setting we have been working in. There are a few added complexities;

- sample based / incremental
- large / cts state spaces
- sparse rewards

Incremental implementation

Generalise to a more complex problem. We are only given samples. A first step to tackling more complex problems.

Model based

Learn p, q based on samples.

$$\mathcal{L}(\theta,\phi) = \mathop{\mathbb{E}}_{s,a,} \Big[r(s,a) - q_\theta(s) + \mathsf{KL}(p_\phi(\cdot|s) \parallel P(\cdot|s,a)) \Big] \tag{27}$$

$$\mathcal{L}(\theta,\phi) = \mathop{\mathbb{E}}_{s,r,s'} \Big[r - q_\theta(s) - p_\phi(s'|s) \log \frac{1}{p_\phi(s'|s)} \Big] \tag{28}$$

$$\tag{29}$$

Model free

z-iterations. But we need to find a way to project $(s_t, a_t, r_t, s_{t+1}) \to (s_t, u_t, q_t, s_{t+1})$.

$$z_{t+1}(s_i) = z_{t+1}(s_i) - \eta \Big(e^{q(s_i)} z({s'}_i)^\gamma - z_t(s_i) \Big) \tag{30}$$

$$\tag{31}$$

- Is there a way to learn p, q incrementally!?!?
- What is the essence of what is being done here?

$$r(s,a) = q(s) - \sum_{s'} P(s'|s,a) \log(p(s'|s)) \tag{32}$$

$$u^*(s'|s) = \frac{p(s'|s) \cdot z(s')^\gamma}{\sum_{s'} p(s'|s) z(s')^\gamma} \tag{33}$$

$$\tag{34}$$

\begin{align} q_{t+1}(s) = (1-⊠);q_{t+1}(s) + ⊠;r(s_{t-1}, a_{t-1}) \ p_{t+1}(s'|s) = (1-⊠);q_{t+1}(s) + ⊠;r(s_{t-1}, a_{t-1}) \ u_t = \ \begin{end}

So we need counts?!?! The $p(s'|s)$? Hmm. That will be expensive. Or inaccurate. Maybe both.

Ok. Lets take a different approach. **Q:** Why is it a bad idea to try to do incremental RL with this linearisation trick? Not sure.

Alternative perspective. The high value trajectories are the most likely ones.

Distributions over states

What if we wanted to approximate these distributions? Generalise subgoal methods to work with distributions? The distribution could be constructed via; parzen window / GMM, neural flow, ?!.

Connections to distributional RL?

Questions

- What is p(s'|s)!?!?
- Want some examples of MDPs they cannot solve.
- What is the relationship to other action embedding strategies?
- How does p(s'|s) bias the controls found??? I can imagine the unconstrained dynamics acting as a prior and prefering some controls over others.
- If we have m states and n actions. Where m >> n. Then $u(s'|s)$ is much larger than $\pi(a|s)$. Also, $u(s'|s)$ should be low rank?! $u_{s's} = \sum_a u_a \alpha_a u_a^T$

Other properties

LMDPs have the property that if we have already solved two LMDPs, with the same state space, action space, unconditioned transition dynamics, but different state rewards, q_1, q_2. Then we can solve a new LMDP, again with the same, ..., and state rewards in the span of q_1, q_2, $z_3 = w_1 z_1 + w_2 z_2, \dots$

Problem. What does it even mean for two LMDPs to have the same unconditioned dynamics but different state rewards? The MDPs must have been the same up to some additive constant (constant in the actions), $r(s,a) = r(s,a) + c(s)$. Does this really capture what we mean by different tasks?!?

AND HRL!?!?

Refs

- Efficient computation of optimal actions
- Linearly-solvable Markov decision problems
- Moving Least-squares Approximations for Linearly-solvable MDP
- Aggregation Methods for Lineary-solvable Markov Decision Process
- A Unifying Framework for Linearly Solvable Control
- A Stability Result for Linear Markov Decision Processes