PAGE 0: RECONFIGURABLE REINFORCEMENT LEARNING NETWORKS

In humans, the structure and form of learning is not only driven by the environment and structure of the brain. The growing of the brain-structure itself defines what learning may take place, and thus the conditions and patterns which direct brain formation are primary and total for the success of learning. Thus, as artifical intelligence research continually generates and publishes on novel structures discovered by humans, this work is centered around how the discovery of this structure may occour automatically. This subject is frequently included in the subject of general intelligence, and is famous for both its philosopical and computational complexity, as well as its difficulty in finding funding. There have been previous works on this subject, such as [Consciousness as a State of Matter] [] [].

Specifically, This work presents a unification method for online learning (Reinforcement Learning) and offline learning (Backpropagation). In addition, this work demonstrates an apporach to the self-structuring of parametric models. First, it is demonstrated that Concurrent Markov Decision Processes (CMDPs) can discover parametric structure and optimal bhaviour with even when subject to large state spaces and generous state uncertainty. Second, it is shown that a variation of CMDPs called Reconfigurable Learning Networks (RLNs) can learn parametric decision networks. RLNs in structure and behaviour turn out to be equilivent to the structure and behaviour of feed-forward neural networks. Lastly, a few empirical examples are demonstrated, beginning with the MINST dataset. Two main contributions are made: First, RLNs can be trained offline and online, using Reinforcement Learning and then Backpropagation; online learning stimulates network growth and adaptation immediately, whereas backpropagation seems to be an ideal phase for network pruning. Second, an empty RLN can enjoy empirical success even when the reward function for the system is changed. Thus both a degree of empirical success and general learning have been achieved.

In order for a generally intelligent system to operate, solutions to several open problems need to be solved anaytically and/or heuristically. In this work, we present the related problem categories in the Introduction (Section 1), and include background on each area. Second, most of this work is focused around the reconfiguration of existing CMDP problems, so Section 2 includes work on transfer learning and analytical anaysis. Third, we express how convergence of behaviour policies can be preserved despite online RLN restructuring (Section 3). The tradeoff between network structure and computation time in learning is expressed analytically (Section 5). Lastly, it is shown that RLNs are actually just feed-forward Neural Networks, which adds the ability to use back propagation and other techniques on discovered models (Section 6).

In this work due to the diffiucty of the subject matter initally, models are assumed noiseless and stochastically stable. It is expected that later work will broaden this work by considering state uncertainty, and non-stationary problems.

NOTATION

In general, most online optimization problems can be expressed as fully observable Markov Decision Processes (MDPs) as $\langle S, A, T, R, \pi \rangle$ tuples:

- $S \subseteq R^n$: A discrete collection of states

- $A \subseteq R^n$: A discrete collection of actions
- $T(s|a, s') \in R$: A stably stochastic transition function, where $\sum_{s \in S} T(s|a, s') = 1$
- $R(s|a, s') \in R$: A stable stochastic reward fuction
- $\pi : S \times A \rightarrow R$: A non-negative behaviour policy with the general property, $\sum_{a \in A} \pi(s, a) = 1$

In general, we can express behaviour in this domain as a policy $\pi : S \rightarrow R$ **[? looked like this, but would $\pi : S \rightarrow A$ make more sense?]**. Particular attention is given to the optimal strategy.
In prior work the issue of tractability and subsequent decomposition have been articulated. In this work the subject of learning and generalizing this decomposition work into a General framework is discussed.

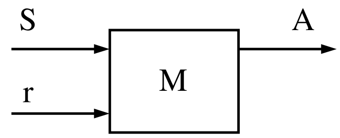
Ⓐ Theory
- Introduction (4): reconfigurable RL introduction & overview
- Mapping (11): a generalized set of mapping & deconstruction operations (parent, child)
- Convergence (16): parent, child, reward optimization, complexity
- Worst Case Performance (23): system behaviour with malformed problems

Ⓑ NN paper
- Neural Networks (24): RRLN are just feed forward Neural Networks
- $\hookrightarrow$ (N.)

Special topics:

Temporal Difference (A1): how to discover & change time basis/scale
Transitional Learning (A2): how to re-use and generalize transitional models
Financial Systems (A3): how to use with financial systems
Origins (E1-E4): original examples and sketches
Transitional Encoding (E5-E6): Continuous bnns **[?]** & applications

PAGE 4 – A RECONFIGURABLE REINFORCEMENT LEARNING SYSTEM



assume a Markov decision process M which can be completely represented as a tuple $M = \langle S, A, T, R, \pi, \tilde{T}, \tilde{R} \rangle$

S – a set of states $s \in S$ which may be experienced by M

A – a set of actions $a \in A$ that may be executed

T – a true transitional probability, $T(s'|a,s)$ expressing the probability of executing an action a in state s before ending up in later state s'.

R – is a reward function which quantifies how desirable a transition $R(s'|a,s)$ is. $R : S \times S \to \mathbb{R}_{\geq 0}$

[I changed $\mathbb{R}^+$ to $\mathbb{R}_{\geq 0}$ because the former is ambiguous with respect to whether or not 0 is included (online research suggests there is no accepted convention) while the latter is unambiguous.]

π – is an action selection policy, ideally chosen to maximize expected reward, an optimal policy is denoted π^*. Typically

$$\pi^*(s) = \arg\max_a \sum_{s'} \underbrace{R(s'|a,s)T(s'|a,s) + \gamma V(s')}_{\text{expected reward}}$$

ENCODING

To encode the expected reward over all states, typically Q-values are kept: $Q(s,a) \sim \sum R(s'|a,s)T(s'|a,s) + \gamma V(s')$ and $Q_{t+1}(s,a) \leftarrow Q_t(s,a) + \alpha \left(R(s'|a,s) - Q_t(s,a) + \gamma \arg\max)_{a'} Q(s',a')\right)$.

In this paper we rely on a method of extracting dynamic Q-values from an encoded transition and reward function $(\tilde{T}, \tilde{R})$. The motivation for this encoding is that it allows mapping the transition function into multiple spaces, and allows the reward function to be altered. The significance of this finding is covered in **???** Price wash **???**.

PAGE 5

1 Reconfiguration

Reconfiguring **???** Process M allows some intractable MDPs to be rendered tractable. As an example, a three dimensional foraging experiment with three thousand positions on the x, y, and z axes respectively will consume over three billion memory locations and may be impossible to explore. If this system is broken into three sub problems, each targets a special axis, the only nine thousand memory locations need be consumed. This decreases memory requirements by an exponential factor.

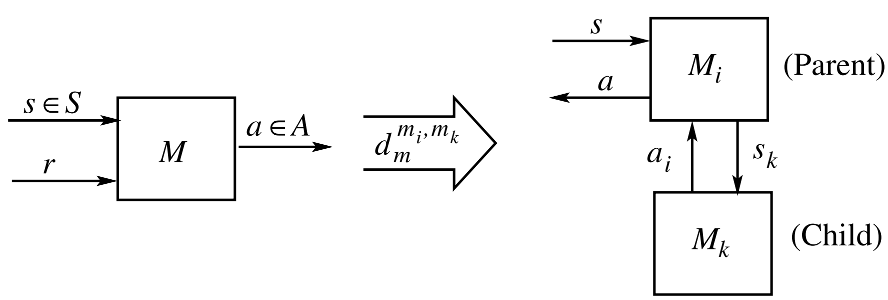
This paper presents a method of decomposition that, when followed, introduces no degeneration of the found policy $\pi^*(s,a)$. The summary of these conditions is presented.

Summary of Requirements

Introduction to approach

$$d_M^{M_i,M_k} = M \longrightarrow \left\{ M_i, M_k \,\middle|\, \begin{array}{l} S_i \times (S_k/s_i) = S, S_k \times (S_i/s_k) = S \\ A = A_i \cup A_k \\ \tilde{T} \sim d^{-1}(d(\tilde{T})), d(\tilde{T}) = \tilde{T}_i, \tilde{T}_k \\ \tilde{R} \sim d^{-1}(d(\tilde{R})), d(\tilde{R}) = \tilde{R}_i, \tilde{R}_k \end{array} \right\}$$

where d, d represent belief mapping functions that decompose and recompose mapping functions. This allows **???** to be mapped as new spaces and observes are encountered. The decomposition process breaks one MDP into a parent and child:



PAGE 7

The system can be broken into the following MDP definitions

$\underline{M_i \text{ – Parent}}$

S_i – a collection of states, $s_i \in S_i$

A_i – a collection of actions, $a_i \in A_i$

$$\left.\begin{matrix} \tilde{T} \\ \tilde{R} \end{matrix}\right\} \text{Covered Pages on BII p12-14}$$

$P(s_i'|s_i, a_i)$ is observed directly

$$R_t\left(\begin{matrix} s_i' \\ a_k' \end{matrix}\middle|\begin{matrix} \\ a_i \end{matrix}, \begin{matrix} s_i \\ a_k \end{matrix}\right) = R_t\left(\begin{matrix} s_i' \\ s_k' \end{matrix}, \begin{matrix} a_i \\ a_k \end{matrix}, \begin{matrix} s_i \\ s_k \end{matrix}\right)$$

S, T_i, S_k, S_k' are not directly observable

ii) $a_k = \pi_k(s_k)$

iii) $a_k' = \pi_k(s_k)$

iiii) (s_k, s_k') chosen indirectly by $\pi_k(\cdot)$ in a manner that

$$\boxed{A^*} \quad \longrightarrow \quad E[R_{t+1}(\cdot)] \geq E[R_t(\cdot)]$$

$\underline{M_k \text{ – child}}$

S_i' – all child states, $s_k \in S_k$

$a_k \in A_k$

$P(s_k'|s_k, a_k)$

$R_t(s_k', a_k, s_k) = R_t\left(\begin{smallmatrix} s_i \\ s_k' \end{smallmatrix}, \begin{smallmatrix} a_i \\ a_k \end{smallmatrix}, \begin{smallmatrix} s_i \\ s_k \end{smallmatrix}\right)$ s.t. s_i, s_i' are chosen by another process, and

$$\boxed{A^*} \quad \longrightarrow \quad E[R_{t+1}(\cdot)] \geq E[R_t(\cdot)]$$

PAGE 8

Definitions

$\underline{M}$ $S = (S_i/S_k) \times (S_k/S_i)$

$A = A_i \cup A_k$

$T = P(S \times A \times S)$

$R =$ real, positive, convergent stochastic as $t \to \infty$

$$R(s', a, s) = R\begin{pmatrix} s'_i & & a_i & & s_i \\ & , & & , & \\ s'_k & & a_k & & s_k \end{pmatrix}$$

Parent MDP

$\underline{M_i}$ S_i, $s_i \in S_i$

$a_i \in A_i$

$$P\left(\begin{array}{c|ccc} s'_i & a_i & & s_i \\ & & , & \\ a'_k & a_k & & s_k \end{array}\right)$$

$$R_t\begin{pmatrix} s'_i & & & & s_i \\ & , & a_i & , & \\ a'_k & & & & a_k \end{pmatrix} = R_t\begin{pmatrix} s'_i & & a_i & & s_i \\ & , & & , & \\ s'_k & & a_k & & s_k \end{pmatrix} \quad \text{s.t. } s_k, s'_k \text{ are not directly observable}$$

$a_k = \pi_k(s_k)$

$a'_k = \pi_k(s'_k)$

$*$ $\boxed{\text{assume } s_k,\ s'_k \text{ chosen s.t. as } t \to \infty \quad E[R_{t+1}()] \geq E[R_t(\cdot)]}$

Child MDP

$\underline{M_k}$ $S_k, s_k \in S_k$

$a_k \in A_k$

$P(s'_k|s_k, a_k)$

$$R_t(s'_k, a_k, s_k) \;=\; R_t\begin{pmatrix} s'_i & & a_i & & s_i \\ & , & & , & \\ s'_k & & a_k & & s_k \end{pmatrix} \quad \text{s.t. } s_i, s'_i \text{ are chosen by another process}$$

$a_i = \pi_i(s_i)$

$*$ time monotonicity assumed.

PAGE 9

<u>Basic mapping requirements</u>

S_i, S_j, S_k: $\quad S_j \times (S_k/S_j) \supseteq S_i, \quad S_k \times (S_j/S_k) \supseteq S_i$

A_i, A_j, A_k: $\quad A_i \subseteq A_j \cup A_k$

T_i, $R_i \sim$ unknown/unknowable, stable decomposition

more$\longrightarrow$ $\boxed{\text{assumed} \quad * \text{ important to select so that } \tilde{T}_i \text{ \& } \tilde{T}_k \text{ seem independent}}$

$\exists f_1 : \tilde{T}_i \to \tilde{T}_j, \tilde{T}_k$, invertible; $\tilde{T}_i = f_1\left(f^{-1}\left(\tilde{T}_i\right)\right)$
$\exists f_2 : \tilde{R}_i \to \tilde{R}_j, \tilde{R}_k$, invertible; $\tilde{R}_i = f_2\left(f_2^{-1}\left(\tilde{R}_i\right)\right)$

(Network approach)

<u>Parent/child augmentation</u>

j – parent $\quad k$ – child
$\tilde{R}_j \leftarrow E[\tilde{R}_k]$
$s_j \in S_j \leftarrow \{S_j, a_k = \pi_k(\cdot)\}$

$\boxed{\text{Continuous (Now)}}$

old approach

Brech
Reword
(BI, p.72)

PAGE 11

Total Mapping

$$* A_R = \left\{ \begin{array}{l} \text{State 1, State 2, Action 1, Action 2} \\ \text{merge, time up, time down} \end{array} \right\}$$

Given $S \in \mathbb{R}^n$, define dimensions $\{i_s\}_{i_s=1}^n$

$A \in \mathbb{N}^m$, define dimensions $\{i_r\}_{i_r=1}^m$

Then, with an initial MDP $M = \langle S, A, T, R, \pi, M_R \rangle$, all possible "sub mdps" M_1, M_2, M_3, … represent the family of MDPs which can be created from M, $\mathcal{P}(M) = \{M_x | S_x \subseteq S, A_x \subseteq A, R, \pi$ from MDPs **???**$\}$ and each member M_x is characterized by a language $J_{sx} \subseteq \{i_s\}_{i_s=1}^n$ or $J_{sy} \subseteq \{i_r\}_{i_r=1}^m$ where $J_{sx} \times J_{sy}$ defines a space S_R, for the reconfiguration MDP to explore, with actions from A_R.

J | Reward is defined as average expected reward over an epoch e.
in terms of transition

$$* S_R = \mathcal{P}\left(\{i_s\}_{i_s=1}^n\right) \times \mathcal{P}\left(\{i_r\}_{i_r=1}^m\right) \qquad \longleftarrow \text{ exponentional increase in space (stupid!)}$$

Problems 1) exponential space consumption

2) how to handle chaining/nesting

3) how to structure action choice policy

PAGE 12

Mapping function rewards

Given $(S_{\text{map}}, A_{\text{map}}, T_{\text{map}}, R^i_{\text{map}}, \pi_{\text{map}})$, applied to $M = \langle S_x, A_x, \tilde{T}_x, R_x, \tilde{\pi}_x \rangle$, we may trivially define $M_y = \langle S_y, A_y, \tilde{T}_y, R_y, \pi_y \rangle$ in a method consistent with Bush, p. 74, with M_x being the parent process and M_y being the child.

a) for R_{map^i}, there are five versions $i \in \{1, \ldots, 5\}$
b) Given M_x, M_y, a merge is also possible, so recovering M
c) we can perform temporal Sink actions on an MDP (Book I, p. 88)
  ↪ reduce resolution
  ↪ re-increase resolution

Actions

∴ Seven "actions" can be performed on an MDP: (M_R)

???

$$\left\{R^i_{\text{map}}\right\}_{i=0}^{5} \cup \{\text{merge}\} \times \left\{\text{scale up}^i\right\}_{i=0}^{5} \cup \{\text{normal}\} \cup \{\varnothing\}$$

Reward

$$R(s', a, s) = \sum_{l \in e} R(l) \qquad \text{reward during a trajectory}$$

$e =$ epoch

Transition

-easy to explain in MS Word

$$T = \begin{cases} 1 \text{ – allow } \textbf{???} \\ 0 \text{ – otherwise} \end{cases}$$

PAGE 13

Reward function mapping (5 way)

knowing $R\left(\{s_x, s_y\}|a, \{s_x', s_y'\}\right) = R(s|a, s)$

1) Average method:

$$R\left(s_x'|a, s_x\right) = \underbrace{\sum_{s_y}\sum_{s_y'} R\left(\left\{s_x', s_y'\right\}|a, \{s_x, s_y\}\right)}_{|S_y|^2}$$

$m \in \{\max, \min\}$

$m' \in \{\max, \min\}$

$\max, \max$	2)	max method!
$\max, \min$	3)	$\min$
	4)	$R\left(s_x'|a, s_x\right) = m \quad m' \quad s_x \in S_x \quad s_y' \in S_y \quad R\left(\left\{s_x', s_y'\right\}|a, \{s_x, s_y\}\right)$
$\min, \max$	5)	
$\min, \min$		
	6)	

PAGE 14

Mapping Policy (1 way)

finding: $f\tilde{\pi}(s) \to f\tilde{\pi}(s_y)$

$$\tilde{\pi}(a_y|s_y) \leftarrow \sum_{s_x} \sum_{a_x} \tilde{\pi}\left(\{a_x, a_y\}|\{s_x, s_y\}\right) P(s_x)$$

PAGE 15

Action mapping (1 way)

Next, we can consider an action mapping where actions from A can be randomly assigned to A_x, A_y: $A_x \leftarrow \{a \in A' | A' \subseteq A\}$, $A_x, A_y \subseteq A$, $A_x \cup A_y = A$, $A_x \neq \{\}$, $A_y \neq \{\}$.

General approach: High reward for **???**ve states

Given $\tau \in \mathbb{R}$, $\pi(a|s)$, $\tilde{Q}(a,s)$ then

$$A_x \leftarrow A_x \cup \left\{ a \middle| \underbrace{\pi(a|s)\tilde{Q}(a,s) > \tau}_{\text{condition}} \right\}$$

or, more usefully/generally

$$A_x \leftarrow A_x \cup \left\{ a \middle| \underbrace{\left(\pi\left(a \middle| S_s^{R'}\right) \tilde{Q}(a, S^{*\prime}) > \tau\right.}_{\text{condition}} \right\}$$

where

$$S_s^{*\prime} = \{S' | S/s_s^* \neq S\} \qquad \text{(see p. 12)}$$

Condition options:

$$*\text{reformulation over set } S \text{ vs. } s \in S \qquad \begin{cases} \text{a) } \pi(a|s)\tilde{Q}(a,s) > \tau & \cdots \text{ High reward} \\ \text{b) } \pi(a|s)\tilde{Q}(a,s) > \tau, \quad \pi(a|s) > 0 & \cdots \text{ small reward} \end{cases}$$

Transition function mapping: knowing $s_x \in S_x$, $s_y \in S_y$ (1 way)

Goal $\exists f : P(S_x|A, S_x) \leftarrow P(S_x|A, S)$
knowing $P(S_x \times S_y | A_x \cup A_y, S_x \times S_y) = P(S|A,S)$
Clearly:

$$P(s'_x|s_x, a_x) = \sum_{s'_y} \sum_{a_y} \sum_{s_y} P\left(s'_x, s'_y | s_x, s_y, a_x, a_y\right) P\left(a_y|S_y\right) P\left(s_y\right)$$

$$\therefore \quad \tilde{T}\left(s'_x|s_x, a_x\right) = \sum_{s'_y} \sum_{a_y} \sum_{s_y} \tilde{T}\left(s'|a,s\right) \quad \underbrace{\tilde{\pi}\left(a_y|s_y\right)}_{\text{require policy mapping}} \quad P(s_y)$$

PAGE 15.1

MDP Policy Decomposition

$$\pi^*(s) = \arg\max_a \sum_{s'} R(s,a,s')P(s'|s,a) + \gamma V(s')$$

Given $\pi_i^*(S_i, A_k)$, $\pi_k^*(S_k)$

$$1. \quad \pi^*(s_i,s_k) = \arg\max_{a_i,a_k} \sum_{s_i'} \sum_{s_k'} R\left((s_i,s_k),(a_i,a_k),(s_i',s_k')\right) P\left((s_i',s_k')|(s_i,s_k),(a_i,a_k)\right)$$

$*$ Lemma 1

–augmentation with a_k where $a_k' = \pi^*(s_k')$

$$2. \quad \pi^*(s_i,s_k) = \arg\max_{a_i,a_k} \sum_{s_i'} \sum_{s_k'} R\left((s_i,s_k,a_k),(a_i,a_k),(s_i',s_k',a_k')\right) P\left((s_i',s_k',a_k')|(s_i,s_k),(a_i,a_k)\right)$$

$*$ Lemma 2 – Simplification

$$\overbrace{* \quad \text{assume} \quad \arg\max_{a_i,a_k} \equiv \arg\max_{a_i} \arg\max_{a_k}}^{\text{separability}}$$

$$3. \quad \pi^*(s_i,s_k) = \arg\max_{a_i,a_k} \sum_{s_i'} R\left((s_i,a_k),(a_i,a_k),(s_i',a_k')\right) P\left(s_i',a_k'|(a_i,a_k),(s_i,s_k)\right)$$

Lemma 3

$*$ separation of a_k, and $a_k \leftarrow \pi_k^*(s_k)$

$$4. \quad \pi^*(s_i,s_k) = \left(\arg\max_{a_i} \sum_{s_i'} R\left((s_i,a_k),a_i,(s_i',a_k')\right) P\left(s_i',a_k'|a_i,(s_i,s_k)\right)\right)$$

$*$ Lemma 4

$$a_i = \pi_i^*(s_i) \longrightarrow^{\cup} \left(\arg\max_{a_k} \sum_{s_k'} R\left(s_k,a_k,s_k',a_k'\right) P\left(s_k'|a_k,s_k\right)\right)$$

$$\boxed{5. \quad \pi^*(s_i,s_k) = \pi_i^*(s_i,a_k) \cup P(s_k)}$$

PAGE 15.2

MDP Policy Decomposition

$$\pi^*(s) = \arg\max_{a} \sum_{s'} R(s,a,s')P(s'|s,a) + \gamma V(s)$$

Given $\pi_i^*(S_i, A_k)$, $\pi^*(S_k)$

1. $$\pi^*(s_i,s_k) = \arg\max_{a_i,a_k} \sum_{s'_i}\sum_{s'_k} R\left((s_i,s_k),(a_i,a_k),(s'_i,s'_k)\right) P\left((s'_i,s'_k)|(s_i,s_k),(a_i,a_k)\right)$$

Note: $R\left((s_i,s_k,a_k),(a_i,a_k),(s'_i,s'_k)\right) \leftarrow R\left((s_i,s_k),(a_i,a_k),(s'_i,s'_k)\right)$

$R\left((s_i,s_k,a_k),(a_i,a_k),(s'_i,s'_k,a'_k)\right) \leftarrow R\left((s_i,s_k),(a_i,a_k),(s'_i,s'_k)\right)$

$*$assume separability $\longrightarrow$

2. $$\pi^*(s_i,s_k) = \arg\max_{a_i}\arg\max_{a_k} \sum_{s'_i}\sum_{s'_k} R\left((s_i,s_k,a_k),(a_i,a_k),(s'_i,s'_k,a'_k)\right) P\left((s'_i,s'_k,a'_k)|\cdot\right)$$

$$= \arg\max_{a_i} \sum_{s'_i}\sum_{s'_k} R\left((s_i,s_k,a_k),\ \begin{array}{c}a_i\\a_k\end{array} \middle| \begin{array}{c}s'_i\\s'_k\\a'_k\end{array}\right) P\left(\begin{array}{c}s'_i\\s'_k\\a'_k\end{array} \middle| \begin{array}{c}a_i\\a_k\end{array},\ \begin{array}{c}s_i\\s_k\\a_k\end{array}\right)$$

$$\cup \arg\max_{a_k} \sum_{s'_k} R\left(\begin{array}{c}s_i\\s_k\\a_k\end{array},\ \begin{array}{c}a_i\\a_k\end{array} \middle| \begin{array}{c}s'_i\\s'_k\\a'_k\end{array}\right) P\left(\begin{array}{c}s'_i\\s'_k\\a'_k\end{array} \middle| \begin{array}{c}a_i\\a_k\end{array},\ \begin{array}{c}s_i\\s_k\\a_k\end{array}\right)$$

$*$ let $a_k^* = \arg\max_{a_k} \sum_{s'_k} R\left(\begin{array}{c}s_i\\s_k\\a_k\end{array},\ \begin{array}{c}a_i\\a_k\end{array} \middle| \begin{array}{c}s'_i\\s'_k\\a'_k\end{array}\right) P\left(\begin{array}{c}s'_i\\s_k\\a_k\end{array} \middle| \begin{array}{c}a_i\\a_k\end{array},\ \begin{array}{c}s_i\\s_k\\a_k\end{array}\right)$

s. t. $s_i, s'_i \leftarrow \pi_i^*($

$a_i^* = \pi_i^*(s)$

3. $$= \arg\max_{a_i} \sum_{s'_i} R\left(\begin{array}{c}s_i\\a_k\end{array},\ \begin{array}{c}a_k\\a_k^*\end{array},\ \begin{array}{c}s'_i\\a'_k\end{array}\right) P\left(\begin{array}{c}s'_i\\a'_k\end{array} \middle| \begin{array}{c}a_i\\a_k^*\end{array},\ \begin{array}{c}s_i\\a_k\end{array}\right) \cup \pi_k^*(s_k) = a_k$$

$$= \pi_i^*\left(s_i|\pi_k^*\right) \cup \pi_k^*(s_k)$$

PAGE 16

Parent Policy Convergence

For the Parent MDP M_k

–from definition $E\left[R_t\left(s'_k|a_k, s_k\right)\right] \geq E\left[R_{t+1}\left(s'_k|a_k, s_k\right)\right]$

$$R_t\left(s'_i|a'_i, s'_i\right) = R\left(\begin{array}{c|cc} s'_i & a_i & s_i \\ s'_k & a_k \, , & s_k \end{array}\right)$$

i. $a_k = \pi_k(s_k)$

(s'_k, s_k) result from a_i s.t.

ii. $s'_k \sim T(S_k|\pi_i(s_k), s_k)$

iii. $s_k \sim T(S_k|\pi_i(s^*_k), s^*_k)$

A)

$\boxed{*}$ – $\pi_k(\cdot)$ is effective: $E\left[R\left(\begin{array}{c|cc} s'_i & a_i & s_i \\ s'_k & a^*_k \, , & s_k \end{array}\right)\right] \geq E\left[R\left(\begin{array}{c|cc} s'_i & A_i & s_i \\ s'_k & A_k \, , & s_k \end{array}\right)\right]$

$\exists a_i, a^*_k \leftarrow \pi_k(s_k)$

B)

$\boxed{*}$ – $\pi_k(\cdot)$ is convergent:

$$E\left[R_{t+1}\left(\cdot \middle| \pi_{\substack{k \\ t+1}}, \cdot\right)\right] \geq E\left[R_{t+1}\left(\cdot \middle| \pi_{\substack{k \\ t+1}}, \cdot\right)\right]$$

assume some policy $\pi_k(\cdot)$ is both effective and convergent, then:

A + B $\rightarrow$ C

C)

$$\boxed{*} \qquad \lim_{t \to \infty} R_t\left(s'_i|a_i, s_i\right) \sim R_{t+1}\left(s'_i|a_i, s_i\right)$$

E) Show other typical convergence **???**,

Done

For the child

$\boxed{*}$ trivial

PAGE 16.1

3 Tractability: it is not possible to map infinite state spaces in practice, so it is advantageous to set up f_Q on a subspace

$$\begin{aligned} &\mathfrak{z} = S \times A \times S \\ \text{s.t.:} \quad &f_Q : \tilde{T}_t(\mathfrak{z}) \times \tilde{R}_t(\mathfrak{z}) \to Q_t(\mathfrak{z}) \qquad\qquad \leftarrow \text{(sloppy)} \end{aligned}$$

which more or less can be directly incorporated into Q_t:

$$f_m : Q_{t+1}(S \times A) \times Q_t(\mathfrak{z}) \to Q_t(S \times A)$$

We also need to keep $\tilde{T}$ and $\tilde{R}_t$ updated and can employ **???** regression instances to do this

$$\begin{aligned} f_T :&\tilde{T}_{t-1}(\mathfrak{z}) \times \{s, a, s'\} \to \tilde{T}_t(\mathfrak{z}) \\ f_R :&\tilde{R}_{t-1}(\mathfrak{z}) \times \{R(s'|a, s)\} \to \tilde{R}_t(\mathfrak{z}) \end{aligned}$$

---

4 Implementation: I use instances of stochastic gradient descent to regress $\boxed{f_T}$ and $\boxed{f_R}$

$$\tilde{T}_t(s'|a, s) \leftarrow f_T(s, a, s', T_{t-1}) = \tilde{T}_{t-1}(s'|a, s) + \alpha_T \left(\frac{f_r(s, a, s')}{f_r(s, a)} - T_{t-1}(s'|a, s) \right)$$

$$\tilde{R}_t \leftarrow \text{ user defined (in this case), and is readily ``pulled''}$$

where $f_r(s, a, s')$ and $f_r(s, a)$ reflect visitation frequencies.

f_Q is more difficult, and can be broken into exact solutions and approximate solutions

$$Q_t(s, a) \leftarrow f_Q^{\text{exact}} \left(\mathfrak{z}, \tilde{T}_t, \tilde{R}_t \right) = \sum_{s' \in \mathfrak{z}} \tilde{T}(s'|s, a) \tilde{R}(s'|s, a) + \gamma V(s')$$

where $V(s) \approx \arg\max_a Q_{t-1}(s, a)$, where $\mathfrak{z} = S \times A$ yields the more accurate and intractable model, it may be desired to focus on estimation, f_a^{est}.

PAGE 17

On The Generalization and reuse of transitional knowledge #2

---

① Setting the state, taking a general FOMDP given usual expections (stably stochastic etc.) $m = \langle S, A, T, R \rangle$ want to find $\pi^* : \{S \times A\} \cup Q(S, A) \rightarrow A$ s. t. for some value function $V(s)$, $\pi^*(s) = \arg\max_a \sum_{s'} T\left(s'|a, s\right) R\left(s'|a, s\right) + \gamma V(s')$

Traditionally, convergence can be found directly, using stochastic gradient descent

$$Q_{t+1}(s, a) = Q_t(s, a) + \alpha \left(R(s'|s, a) - Q_t(s, a) + \gamma \arg\max_a \left(Q_t(s', a^*)\right) \right)$$

which is limited because as $Q(S, A)$ converges, it becomes difficult to adjust to changes in $R(S, A, S)$.

---

② Optimization objectives change, meaning the basis of $Q(S, A)$ is typically malleable in real-life scenarios. In this paper we present a method for separating transitional models and reward models.
We hold reward and transitional functioning separate as $\tilde{T}$ and $\tilde{R}$; and attempt to regress to true values s. t. $\tilde{T} \approx T$ and $\tilde{R} \approx R$. We then develop a Q_{map} function f_Q to **???** $Q(S, A)$ space as needed:

$$f_Q : \tilde{T}\left(S \times A \times S\right) \times \tilde{R}\left(S \times A \times S\right) \rightarrow Q_t\left(S, A\right)$$

PAGE 18

MDP: Linearization of Reward/Optimal Policy

P_a – $P_a(i,j)$ represents $T(s_i, a, s_j)$ S– all states

γ – decay factor $(0,1)$

π – policy

$V^\pi(s)$ – typical value function

$\mathbf{V}^\pi$ – vector of all values $\{V^\pi(s_1), \ldots, V^\pi(s_n)\}$

$\prec$ and $\preceq$ denote strict and non-strict vectoral inequality.

$\mathbf{R}$ – vector of reward (like $\mathbf{V}^\pi(s)$)

for optimal reward:

$$(P_{a_i} - P_a)\,(I - \gamma P_{a_i})^{-1}\mathbf{R} \succeq 0 \qquad \Leftrightarrow$$

Proof (cool as fuck):

$$a_1 \equiv \pi(s) \in \arg\max_{a \in A} \sum_{s'} P_{s_a}(s')V^\pi(s') \quad \forall s \in S$$

$$\sum_{s'} P_{s_{a_1}} \geq \sum_{s'} P_{s_a}(s')V^\pi(s') \quad \forall s \in S, a \in A$$

$$\vdots \qquad a_1 \text{ is Pareto efficient (!)}$$

$$P_{a_1}\mathbf{V}^\pi \succeq P_a V^\pi \quad \forall a \in A \backslash a_1 \qquad \text{(non-strict improvement)}$$

$$\vdots$$

$$P_{a_1}(I - \gamma P_{a_1})^{-1}\mathbf{R} \succeq P_a(I - \gamma P_{a_1})^{-1}\mathbf{R} \quad \forall a \in A \backslash a_1$$

The hard part to verify: $\mathbf{V}^\pi = (I - \gamma P_{a_1})\mathbf{R}$

PAGE 19

Transitional Learning Continued

- $\{s, s', w\}$ can be controlled to both represent the state space and accurately represent $P+(s'|s', a)$
- $A+h$, $B+h$ and γ can be controlled to speed the algorithm R_γ
- Q-learning can still be used, if calculation of $\bar{R}_\gamma$ is too "slow".
- the reward function $R(s, a, s')$ can be redefined at an instant to allow immediate re-calculation of a policy $\bar{R}_\gamma(s, a)$.

Possible experiments:

- show speed of convergence is greater, due to the "storing" of the transitional model across all actions
- show that the generalized learning allows for redefinition of the reward function.

PAGE 20

Policy

Convergence of "Bad MDP"

Question, given π_i^*, is it possible to find

$$f : \pi_j^*, \pi_k^* \rightarrow \pi_i^*$$

$$f(\pi_j(s))$$

a) suppose $f\left(\pi_j\left(s, \pi_k(s)\right)\right) = \pi_j\left(s, \pi_k(s)\right) \cup \pi_k(s)$

b) S_k,S_j – assume a subspace that is independent of effect by A, $S_j \in S_i$
   A_k, A_j – assume a subset of A_j
   T_k, T_j – assume $T(s_i, a, s_j') = 0 \quad \forall (s_i, a_j, s_j) \in S_j \times A_j \times S_j$
   R_k, R_j – assume $R(s_i, a, s_j)$
   In this case, A_j must effect $T(s_k, a_k, s_k')$ and A_k must effect $T(s_j, a_j, s_j')$
   effect how:

$$\exists a_j, a_k \sum_{s_j' \neq s_j} T\left(s_j' \,\middle|\, \begin{matrix} a_j \\ a_k \end{matrix}, s_j\right) > 0 \qquad \boxed{*\text{(Derive this conclusion from } T(s_i, a_i, s_i))*}$$

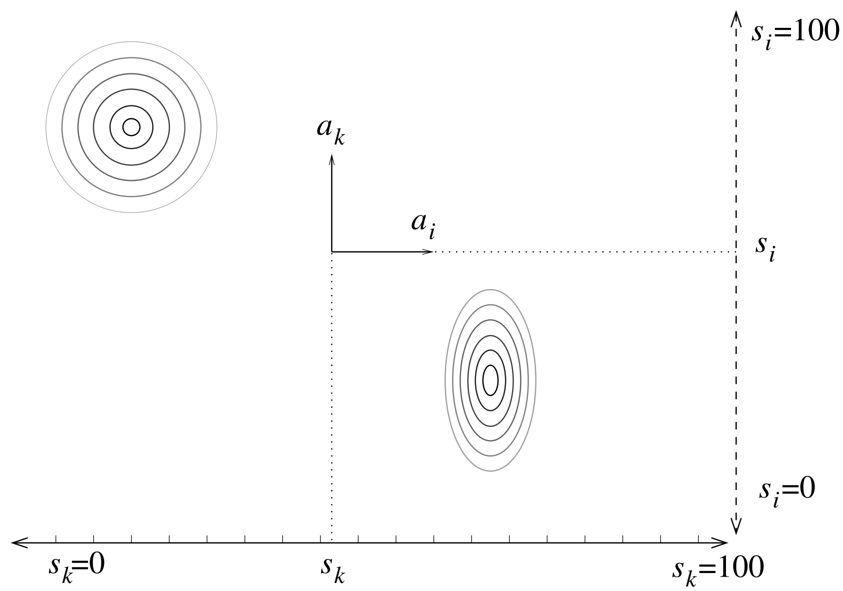
c) M_j, M_k execute concurrently, meaning at each time t $\exists (a_j, a_k) \in A_i$, chosen by $a_j \sim \pi_j(s_j, a_k) \qquad a_k \sim \pi_k(s_k)$

PAGE 21

child convergence assume during concurrent learning

- R_t must be time-monotonic and convergent stochastic.

- Evaluate selection of s'_i and s_i, assuming worst case: a_k has no impact on s'_k, s_k and only an impact on s_i and s'_i



consider policy learning: execution of $\pi_k(s_k)$ yielding s'_k