PAGE 0: RECONFIGURABLE REINFORCEMENT LEARNING NETWORKS

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

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
  - ↪ (N.)

Special topics:

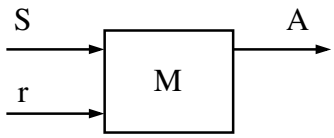
Temporal Difference (A1): how to discover & change time basis/scale

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



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

M_i – Parent

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

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

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

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

M_k – child

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

$a_k \in A_k$

$P(s_k'|s_k, a_k)$

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