

# **Balancing Multiple Sources of Reward in Reinforcement Learning**

By Christian R. Shelton (cshelton@ai.mit.edu)

Presented by Eric Wiewiora

CSE 254 Spring 2002

Instructor: Charles Elkan

## **Traditional View of Reinforcement Learning**

One reward is associated with every state transition  $(x, a, x')$ .

The agent learns to maximize the sum of rewards received over time.

What if rewards are coming from many incompatible sources?

2

## **Example: Home Entertainment System**

A “smart” TV changes the channel to something the audience wants to see.

Only some members of the family are watching at any given time.

How can the system adapt to conflicting preferences of changing audiences?

3

## Example: Elevator

An elevator must schedule which floors to visit.

Different goal metrics are possible:

- Average response time
- Throughput

When are the right decisions important for optimizing each of these goals?

## The Paper's Solution

Independently learn the optimal strategy for each source of reward.

Let the sources vote to determine the combined policy of the agent.

Requirements:

- Balanced voting system
- Sources benefit by being honest about reward function

## Notation

The state observed at time step  $t$  is  $x(t)$ .

The action chosen after observing  $x(t)$  is  $a(t)$ .

A policy gives the probability,  $\pi(x, a) = p(a|x)$ , of taking action  $a$  after observing  $x$

The reward from source  $s$  after taking action  $a(t)$  is  $r_s(t)$

The expected Return from source  $s$  for following policy  $\pi$ :

$$R_s^\pi = \lim_{n \rightarrow \infty} \frac{1}{n} E\left(r_s(1) + r_s(2) + \cdots + r_s(n) | \pi\right)$$

## Votes

Each source wants to change the policy of the agent to best serve the source's interest.

Let the vote for source  $s$  for taking action  $a$  in state  $x$  be  $v_s(x, a)$ , where

$$\sum_a v_s(x, a) = 1 \quad \text{and} \quad v_s(x, a) \geq 0.$$

For each state  $x$ , think of  $v_s(x, a)$  as the desired probability of taking action  $a$ .

## Balancing Votes

We limit the total influence of a source's vote over all states to 1.

The amount of influence expended per state is  $0 \leq \alpha_s(x) \leq 1$ .

The final policy of the agent is

$$\pi(x, a) = \frac{\sum_s \alpha_s(x) v_s(x, a)}{\sum_s \alpha_s(x)}$$

Note: A source cannot directly learn its optimal vote!

## Example: Meal Time

Two people are voting on what to have for dinner

- *A* loves potatoes and hates meat.
- *B* likes a balanced meal of meat and potatoes.

*A* will always vote for a policy of

$$v_A(x, \textit{meat}) = 0; v_A(x, \textit{potatoes}) = 1$$

To offset *A*'s preference, *B* will vote

$$v_B(x, \textit{meat}) = 1; v_B(x, \textit{potatoes}) = 0$$

In the absence of *A*, *B* should vote for an even split.

## Returns as Preferences

Each source can measure its expected return of a policy.

Each source will vote in a way that maximizes its expected return from the consensus policy.

In game theory, this is known as a *general gum game*, with the sources as players.

10

## General Sum Games

A game is defined as:

- A set of players  $\{1, \dots, S\}$
- A set of strategies available to each player  $\Pi_s$
- A set of situations,  $\Pi$ , where each player chooses a strategy
- A reward for each player  $R_s$  defined over  $\Pi$ .

A game is *general sum* if the players are not competing for a fixed amount of total reward.

11

## Nash Equilibrium

If every player is playing such that no player can benefit by changing its strategy, this is known as a *Nash equilibrium*.

The algorithm in the paper presents a way to set the votes of sources to reach a Nash equilibrium.

Note: A Nash equilibrium does not maximize sum reward over all sources, nor the minimum reward received by any source.

## Top Level Description of the Algorithm

For multiple epochs do:

1. Calculate a Nash equilibrium policy based on current estimates of each source's optimal policy.
2. With probability  $\epsilon$  change to a random policy.
3. Execute policy and evaluate  $R_s^\pi$  for each source.
4. Update estimates of optimal policies.

## Estimating the Return of a Policy

A source  $s$  needs to have an estimate of the expected return of a policy before it can know how to change it.

We'll assume that the expected return of a policy for  $s$  is a linear function of the difference in the optimal policy for  $s$  and the current policy.

Use KL-divergence to measure the difference between policies.

14

## Estimating the Return of a Policy

The estimate expected return of any policy  $\pi$  for source  $s$  takes the form:

$$\hat{R}_s^\pi = b_s - a_s \sum_x \beta_s(x) \sum_a \pi_s(x, a) \log \frac{\pi_s(x, a)}{\pi(x, a)}$$

where

$b_s$  is maximum possible reward,

$a_s$  is the sensitivity of this source to policy differences,

$\beta_s(x)$  is the relative importance of state  $x$  to  $s$ ,

$\pi_s(x, a)$  is the preferred policy of  $s$ .

15

## Estimating a Weighted $\pi(x, a)$

Let  $\pi'_s(x, a) = a_s \beta(x) \pi_s(x, a)$ .

Given experiences of different policies and their empirical returns, we can estimate  $\pi'_s(x, a)$  using linear least-squared error.

The training examples for linear least-squares will be defined as  $\langle \pi(x, a), R_s^\pi \rangle$ .

$\pi'_s(x, a)$  will be the maximum value the learned function takes over the appropriate range.

Note that the  $\pi'_s(x, a)$  do not sum to one over actions like  $\pi(x, a)$  must.

16

## Deriving Our Parameters from $\pi'_s(x, a)$

From  $\pi'_s(x, a) = a_s \beta(x) \pi_s(x, a)$  we can calculate

$$\begin{aligned} a_s &= \sum_{x,a} \pi'_s(x, a) \\ \beta(x) &= \frac{1}{a_s} \sum_a \pi'_s(x, a) \\ \pi_s(x, a) &= \frac{\pi'_s(x, a)}{\sum_{a'} \pi'_s(x, a')} \end{aligned}$$

$b_s$  need not be calculated. We are only interested in the relative return of policies.

This completes the method for evaluating policies.

17

## Algorithm for Finding Equilibrium

1. For all  $s, x, a$ , initialize  $\alpha_s(x) = \beta_s(x)$ ,  $v_s(x, a) = \pi_s(x, a)$ .
2. Repeat until convergence:
  - (a) **For each  $s$** , set all  $v_s(x, a)$  based on current  $\pi$ .
  - (b) **For each  $s$** , adjust all  $\alpha_s(x)$  based on the new  $v_s(x, a)$  values.
  - (c) If any  $\alpha_s(x)v_s(x, a)$  changed by more this round than the last, set the change of **all the  $\alpha_s(x)$  and  $v_s(x, a)$  for state  $x$**  to the average of the new and old.
  - (d) Calculate the new  $\pi$  based on the new  $v_s$  and  $\alpha_s$ .

This should converge in a few tens of iterations.

18

## Best Response Algorithm

We must set  $\alpha_s(x) > 0$  and  $v_s(x, a) > 0$  to satisfy

$$\sum_x \alpha_s(x) = 1, \quad \sum_a v_s(x, a) = 1$$

while maximizing expected return for each source.

This is the same as minimizing the KL-divergence of  $\pi$  from  $\pi_s$ .

$$-\sum_x \beta_s(x) \sum_a \pi_s(x, a) \log \frac{\sum_{s'} \alpha_{s'} v_{s'}(x, a)}{\sum_{s'} \alpha_{s'}}$$

We'll do this by adjusting  $v_s$  and  $\alpha_s$  separately.

19

## Setting $v_s$

We wish to set each source's vote to push the sum of votes closer to the source's optimal policy.

The optimal vote is

$$v_s(x, a) = \frac{\sum_{s' \neq s} \alpha_{s'}(x) (\pi_s(x, a) - v_{s'}(x, a))}{\alpha_s(x)}$$

Any negative  $v_s$  are set to zero and the votes in state  $x$  are renormalized to sum to one.

20

## Adjusting $\alpha$

We wish to place more of a source's influence on states that

- the source finds important.
- the current vote is not in the source's favor.

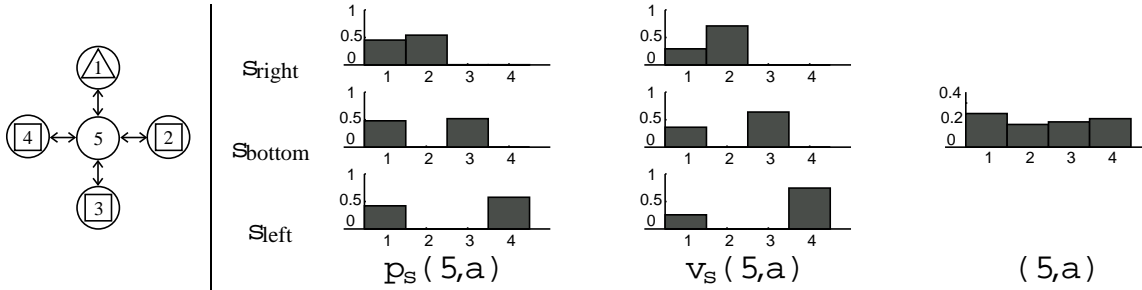
Use gradient descent on  $\alpha$ , yielding

$$\Delta \alpha_s(x) \propto \frac{\beta_s(x)}{\sum_{s'} \alpha_{s'}(x)} \sum_a \pi_s(x, a) \frac{\sum_{s' \neq s} \alpha_{s'}(x) (\pi_s(x, a) - v_{s'}(x, a))}{\sum_{s'} \alpha_{s'}(x) v_{s'}(x, a)}$$

The updated  $\alpha_s$  are renormalized to sum to one.

21

## Example 1: Load-unload



One state loads cargo on agent.

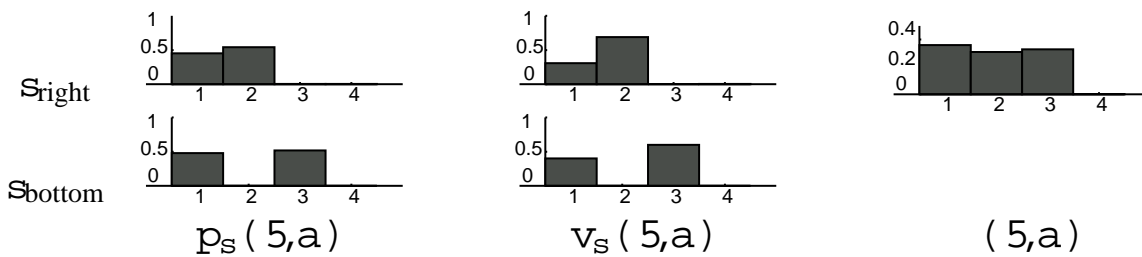
One cargo destination per source, where if the agent arrives with cargo, the source gives reward.

Note that the agent does not know if it is carrying cargo.

At state 5, the learned policy have almost uniform action probabilities.

22

## Example 1: Adaption to Source Removal



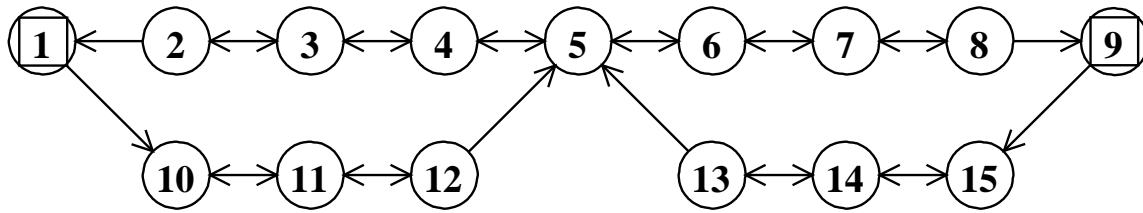
When one source is removed, the agent's policy adapts once a Nash equilibrium is recalculated.

The new policy doesn't visit the removed source's destination state.

The remaining states are visited with almost uniform probability.

23

## Example 2: One-Way Doors



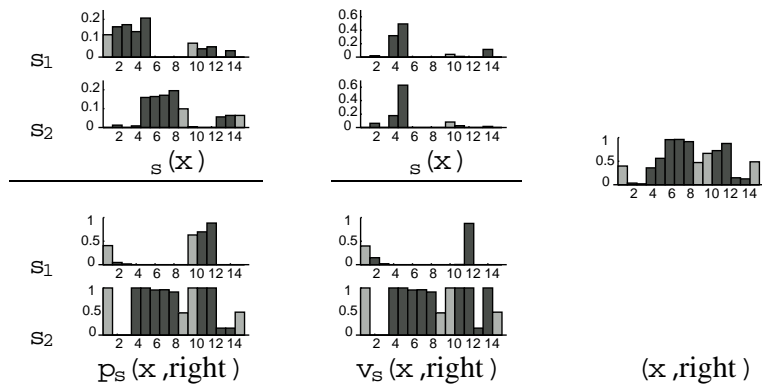
Two sources of reward:

- Source 1 gives reward when the agent enters the leftmost state.
- Source 2 gives reward when the agent enters the rightmost state.

After one of the reward-giving states is reached, the one-way doors force the agent to pass back through the middle state.

24

## Example 2: Learned Policy



Both sources place most of their emphasis on State 5.

This corresponds well with the best overall policy:

- Compete over the action in state 5, but cooperate afterwards to get back to state 5 ASAP.

If the sources battled over all states, fewer rewards will be given by both sources.

25

## Insights

The paper approaches balancing reward sources from a game theoretic perspective.

The vote setting algorithm acts as a mediator between the sources, which leaves open the possibility of optimizing in less greedy ways.

There are other, more philanthropic game theory metrics.

One example is a Pareto optimal condition. This is defined as a set of strategies where there is no other situation more preferable to all players.

## Conclusion

The paper presents a framework for balancing multiple sources of reward.

By taking a game theoretic approach, the algorithm:

- balances the influence of the sources,
- adapts to changes in the sources,
- performs a best effort assignment of the sources' votes.