



Consistent exploration improves convergence of reinforcement learning on POMDPs

Paul A. Crook, Gillian Hayes

`pcrook@inf.ed.ac.uk`, `gmh@inf.ed.ac.uk`

Institute of Perception Action and Behaviour (IPAB)

School of Informatics

University of Edinburgh



Background

Reinforcement Learning (RL) algorithms learn a policy π through the exploration of a task.

This policy sets out the action a that an agent should take for each “observation” of a task which the agent perceives.



Background

For a Markov decision process (MDP) the agent perceives the actual state s of the task.

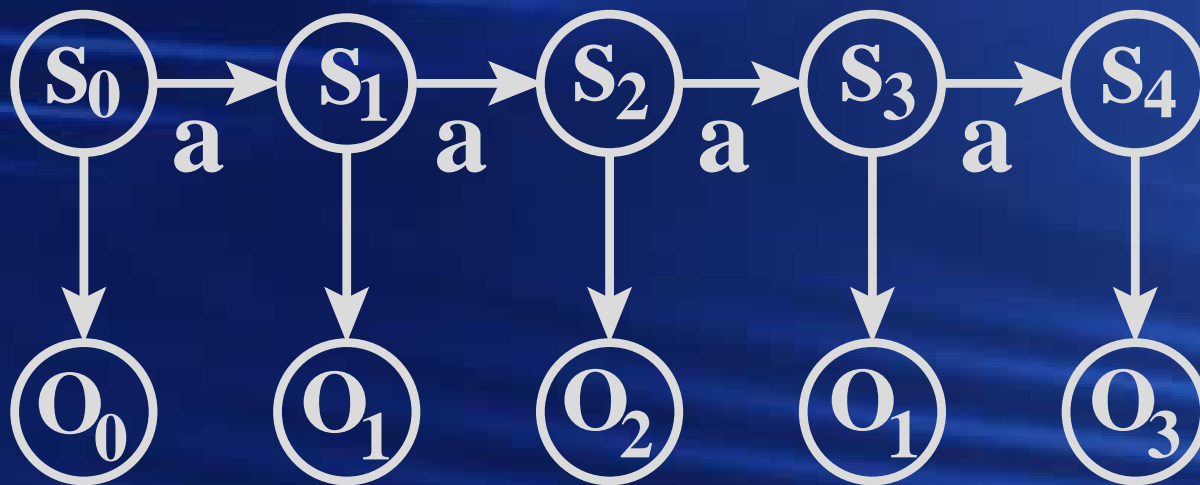


Thus the policy associates actions directly with the current state of the task; $\pi : S \rightarrow A$.



Background

For a partially observable Markov decision process (POMDP) the agent perceives observations o .



Thus the policy can only associate actions with the observations; $\pi : O \rightarrow A$.

Applying RL to POMDPs



Main issues:

- The policy convergence guarantees that exist for MDPs do not apply to POMDPs

Applying RL to POMDPs



Main issues:

- The policy convergence guarantees that exist for MDPs do not apply to POMDPs
- Performance does not degrade gracefully as the degree of non-Markovianness increases [Singh et al. ICML'94]

Applying RL to POMDPs

In response to these issues there are two schools of thought for on-line learning:

- State Estimation; map observations to estimated MDP states.



Applying RL to POMDPs



In response to these issues there are two schools of thought for on-line learning:

- State Estimation; map observations to estimated MDP states.
- Simply apply RL algorithms anyway, *i.e.* learn policies which map observations to actions and accept the reduction in value of the policies it is possible to learn.

State Estimation

- Attempts to estimate the MDP which underlies the POMDP and then applies RL using the states of the estimated MDP model.



State Estimation

- Attempts to estimate the MDP which underlies the POMDP and then applies RL using the states of the estimated MDP model.
- Advantage: RL is applied to an MDP model.



State Estimation



- Attempts to estimate the MDP which underlies the POMDP and then applies RL using the states of the estimated MDP model.
- Advantage: RL is applied to an MDP model.
- Disadvantage: no guarantees that the estimated MDP model will converge or be correct.

State Estimation

- In practice simultaneous learning of state estimation mappings and RL policies based on those mappings has proved very brittle [Shatkay and Kaelbling IJCAI'97, Hartley 2004 PhD, Crook 2006 PhD],



Just simply apply RL to POMDPs

- Despite the lack of convergence guarantees and a reduction in the value of the policies it is possible to learn, this approach has proved surprisingly effective.



Just simply apply RL to POMDPs



- Despite the lack of convergence guarantees and a reduction in the value of the policies it is possible to learn, this approach has proved surprisingly effective.
- Especially eligibility trace algorithms, *i.e.* SARSA(λ) [Loch and Singh ICML'98].

Just simply apply RL to POMDPs



- Despite the lack of convergence guarantees and a reduction in the value of the policies it is possible to learn, this approach has proved surprisingly effective.
- Especially eligibility trace algorithms, *i.e.* SARSA(λ) [Loch and Singh ICML'98].
- My work attempts to understand and build on these results.

Just simply apply RL to POMDPs

Having accepted the reduction in value (i.e. optimality) of policies, my aim is to achieve *reliable* convergence to *satisficing* policies.



Just simply apply RL to POMDPs



Having accepted the reduction in value (*i.e.* optimality) of policies, my aim is to achieve *reliable* convergence to *satisficing* policies.

I define satisficing as reaching some minimum level of performance, *e.g.* reaching the goal from all starting states.

“Evidently, organisms adapt well enough to ‘satisfice’; they do not, in general, ‘optimize’.” [Herbert Simon, Psychological Review 1956]

Just simply apply RL to POMDPs



One problem is estimating the value of changing a policy action on POMDPs.

- Most RL algorithms take advantage of the fact that each “observation” uniquely represents the current state of a MDP.

Just simply apply RL to POMDPs



One problem is estimating the value of changing a policy action on POMDPs.

- Most RL algorithms take advantage of the fact that each “observation” uniquely represents the current state of a MDP.
- To evaluate a change to the policy these algorithms try an exploratory action and then follow the current policy – the value arrived at indicates if that change should be preferred.

Just simply apply RL to POMDPs



One problem is estimating the value of changing a policy action on POMDPs.

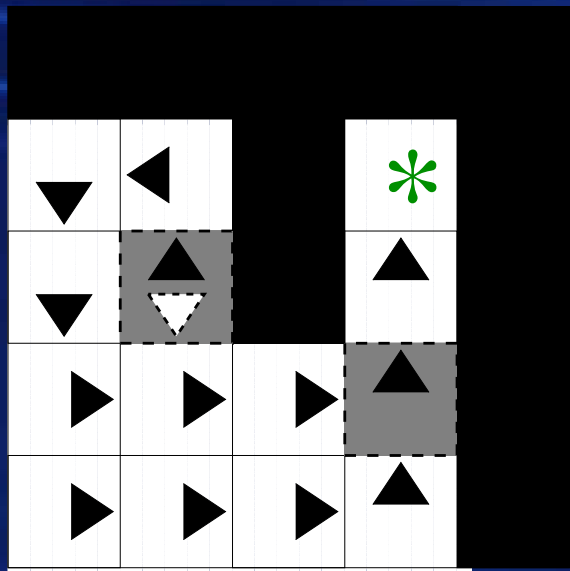
- Most RL algorithms take advantage of the fact that each “observation” uniquely represents the current state of a MDP.
- To evaluate a change to the policy these algorithms try an exploratory action and then follow the current policy – the value arrived at indicates if that change should be preferred.
- This doesn't work for POMDPs

Estimating policy value

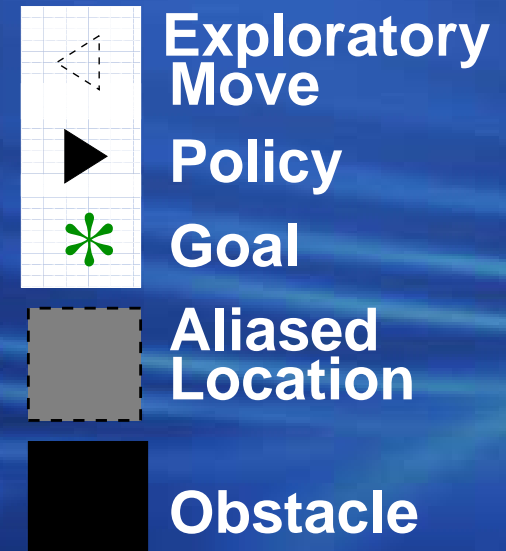
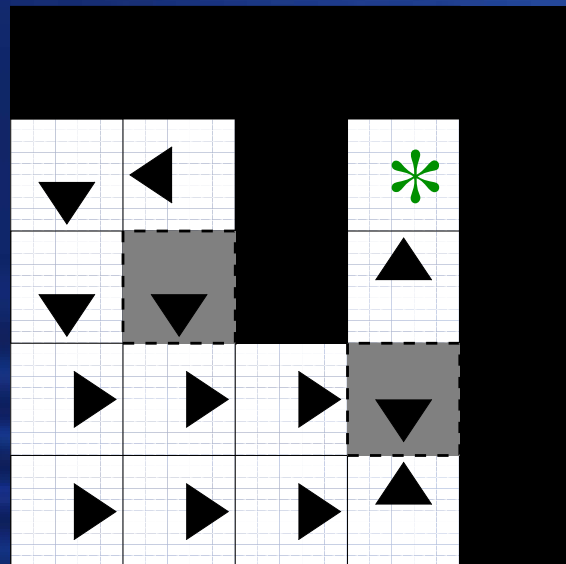


Illustration of the problem

Estimating Value of Explorative Move



Policy Updated



Consistent Exploration

- A solution is to evaluate complete policies rather than individual exploratory actions.



Consistent Exploration

- A solution is to evaluate complete policies rather than individual exploratory actions.
- That is, to *consistently* execute the same exploratory action for each occurrence of the observation against which it was first selected, until some evaluation of the modified policy has been arrived at.



Consistent Exploration



- A solution is to evaluate complete policies rather than individual exploratory actions.
- That is, to *consistently* execute the same exploratory action for each occurrence of the observation against which it was first selected, until some evaluation of the modified policy has been arrived at.
- This is what our algorithm CEQ(λ) and the algorithm MCESP [Perkins AAAI'02] attempt to do.

Why CEQ(λ)?

We introduce CEQ(λ) to address the following issues with MCESP.

- MCESP only varies the policy it explores and updates policy values *once per episode*, thus;



Why CEQ(λ)?

We introduce CEQ(λ) to address the following issues with MCESP.

- MCESP only varies the policy it explores and updates policy values *once per episode*, thus;
 - there is an issue of modified policies becoming stuck in infinite loops,



Why CEQ(λ)?

We introduce CEQ(λ) to address the following issues with MCESP.

- MCESP only varies the policy it explores and updates policy values *once per episode*, thus;
 - there is an issue of modified policies becoming stuck in infinite loops,
 - there is a question of applicability to non-episodic tasks,



Why CEQ(λ)?



We introduce CEQ(λ) to address the following issues with MCESP.

- MCESP only varies the policy it explores and updates policy values *once per episode*, thus;
 - there is an issue of modified policies becoming stuck in infinite loops,
 - there is a question of applicability to non-episodic tasks,
 - learning is potentially slower than temporal difference methods, such as SARSA(λ).

Why CEQ(λ)?

In contrast CEQ(λ);

- is a temporal difference style algorithm,



Why CEQ(λ)?

In contrast CEQ(λ);

- is a temporal difference style algorithm,
- generalises the period of consistent exploration from complete episodes to arbitrary length periods of time.



Consistent Exploration $Q(\lambda)$

- CEQ(λ) based on Watkins's $Q(\lambda)$



Consistent Exploration $Q(\lambda)$

- CEQ(λ) based on Watkins's $Q(\lambda)$
- when exploratory action a_1 is selected for observation o_1 follows modified policy

$$\pi \leftarrow (o_1, a_1)$$



Consistent Exploration $Q(\lambda)$

- CEQ(λ) based on Watkins's $Q(\lambda)$
- when exploratory action a_1 is selected for observation o_1 follows modified policy
 $\pi \leftarrow (o_1, a_1)$
- when new exploratory action a_2 chosen:



Consistent Exploration Q(λ)



- CEQ(λ) based on Watkins's Q(λ)
- when exploratory action a_1 is selected for observation o_1 follows modified policy
 $\pi \leftarrow (o_1, a_1)$
- when new exploratory action a_2 chosen:
 - clear eligibility trace

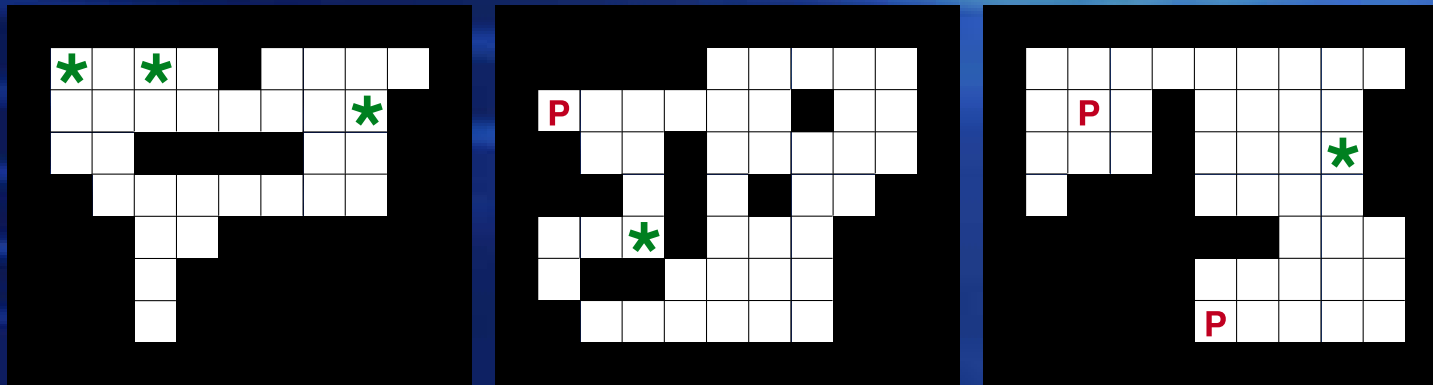
Consistent Exploration Q(λ)



- CEQ(λ) based on Watkins's Q(λ)
- when exploratory action a_1 is selected for observation o_1 follows modified policy $\pi \leftarrow (o_1, a_1)$
- when new exploratory action a_2 chosen:
 - clear eligibility trace
 - change policy followed to $\pi \leftarrow (o_2, a_2)$

Comparison with SARSA(λ)

Example randomly generated grid worlds



parameter values used: learning rate (α) 0.01, 0.02, 0.05, 0.1 and 0.3

discount rate (γ) 0.8, 0.9, 0.99, and 1.0

eligibility trace (λ) 0.8, 0.9, 0.99, and 1.0

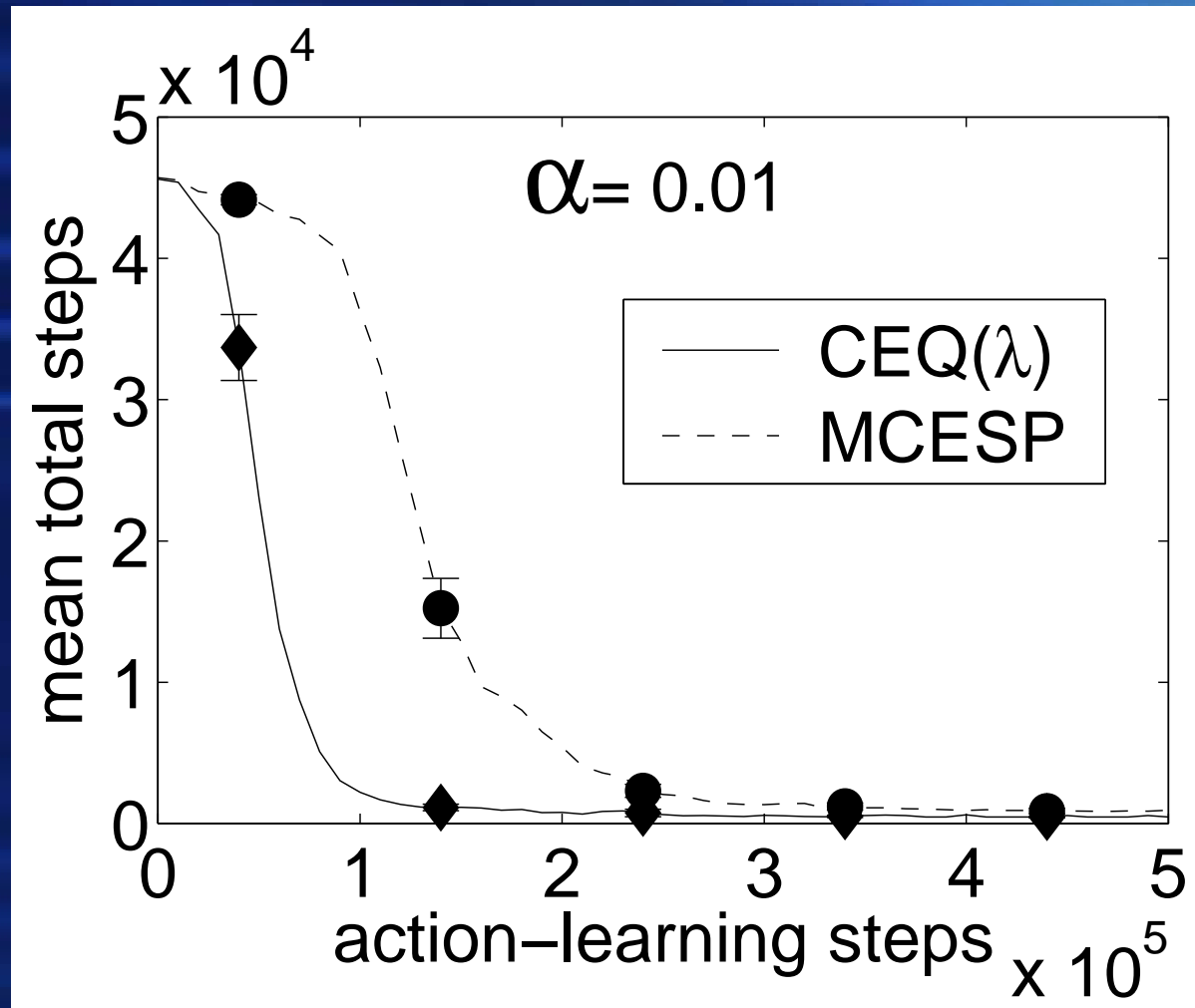
3 agent types, 20 trials of each combination.

Comparison with SARSA(λ)

Mean number of satisficing policies for each agent-algorithm combination as learnt on 10 random grid worlds

Agent \ Algorithm	CEQ(λ)	SARSA(λ)
Basic	90% \pm 28%	80% \pm 37%
Active Perception	99% \pm 6%	88% \pm 29%
1-bit Memory	99% \pm 2%	90% \pm 27%
Overall	96% \pm 17%	86% \pm 31%

Comparison with MCESP



Conclusions

- Successfully demonstrate the concept of consistent exploration can result in algorithms which learn deterministic reactive policies more reliably in partially observable environments.



Conclusions

- Successfully demonstrate the concept of consistent exploration can result in algorithms which learn deterministic reactive policies more reliably in partially observable environments.
- Shown that the algorithm CEQ(λ) outperforms SARSA(λ) on a range of POMDP problems and can learn quicker than MCESP.



Future Directions

- Development of a theory of learning deterministic policies in partially observable domains.



Future Directions

- Development of a theory of learning deterministic policies in partially observable domains.
 - Is it possible to identify tasks where a deterministic policy exists?



Future Directions



- Development of a theory of learning deterministic policies in partially observable domains.
 - Is it possible to identify tasks where a deterministic policy exists?
- Comparison of approach with PSRs, recurrent-networks (LSTM), etc.

Future Directions



- Development of a theory of learning deterministic policies in partially observable domains.
 - Is it possible to identify tasks where a deterministic policy exists?
- Comparison of approach with PSRs, recurrent-networks (LSTM), etc.
- Look at a larger range of tasks, *i.e.* non-episodic tasks.

Future Directions



- Development of a theory of learning deterministic policies in partially observable domains.
 - Is it possible to identify tasks where a deterministic policy exists?
- Comparison of approach with PSRs, recurrent-networks (LSTM), etc.
- Look at a larger range of tasks, *i.e.* non-episodic tasks.

Thank You