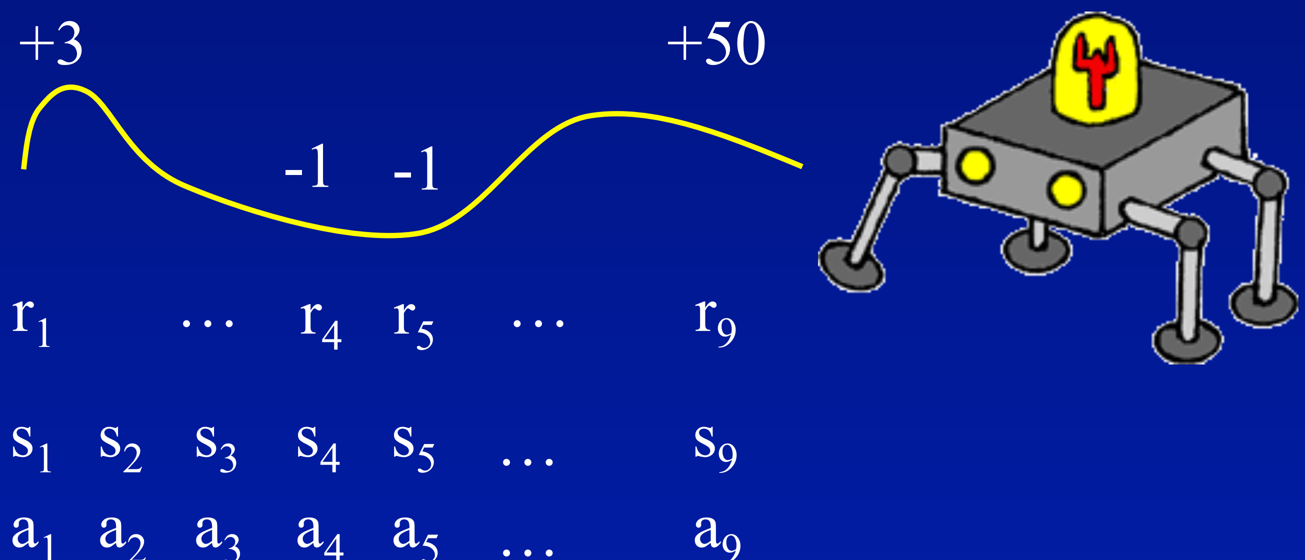
An Introduction to Reinforcement Learning

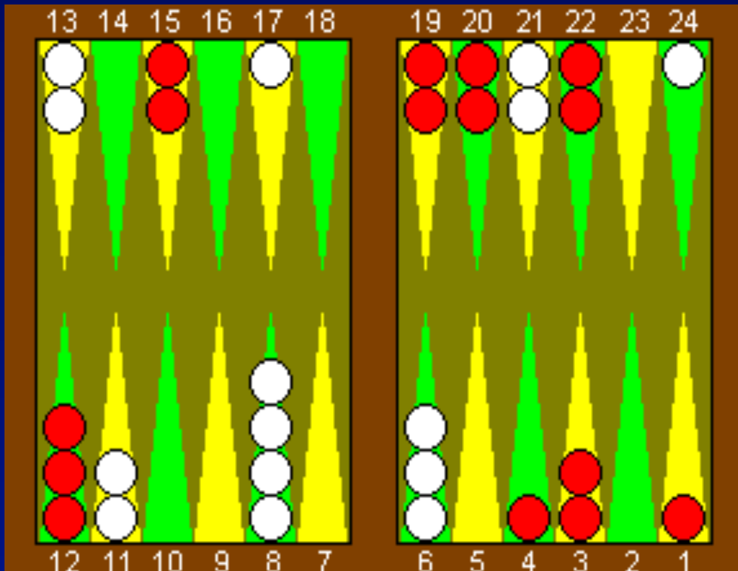
Jeremy Wyatt
Intelligent Robotics Lab
School of Computer Science
University of Birmingham
jlw@cs.bham.ac.uk
www.cs.bham.ac.uk/~jlw
www.cs.bham.ac.uk/research/robotics
www.cs.bham.ac.uk/~jlw/bcs_summer_school_handouts.ppt

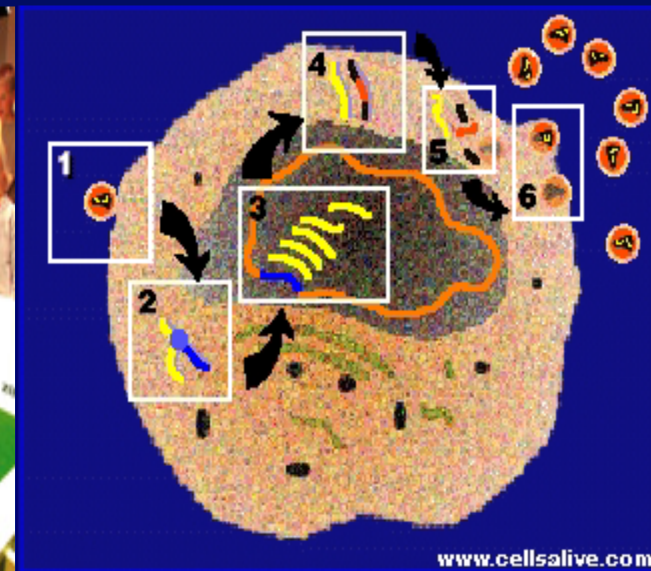
What is Reinforcement Learning (RL) ?

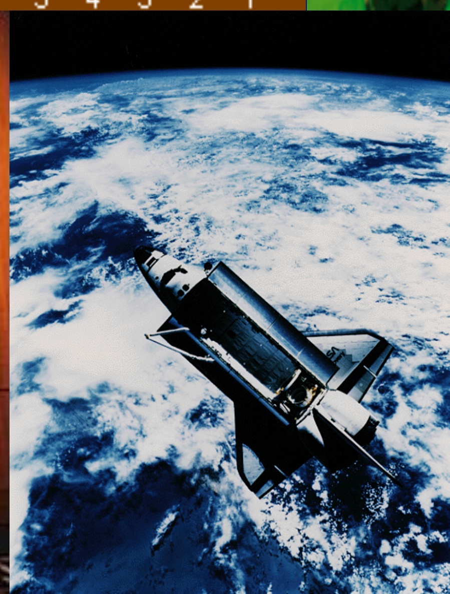
- Learning from punishments and rewards
- Agent moves through world, observing states and rewards
- Adapts its behaviour to maximise some function of reward

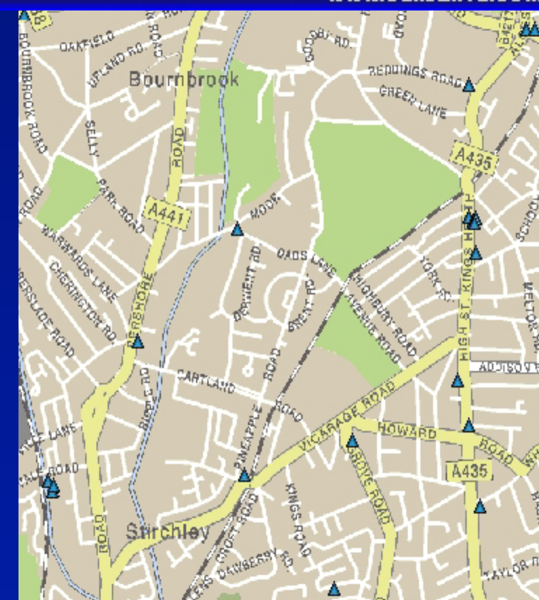


What is it good for?







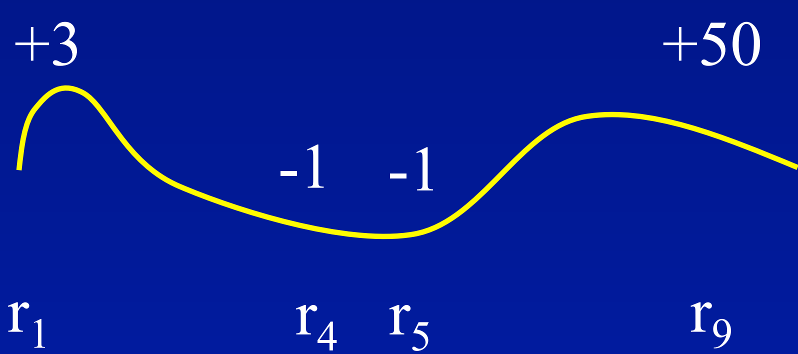


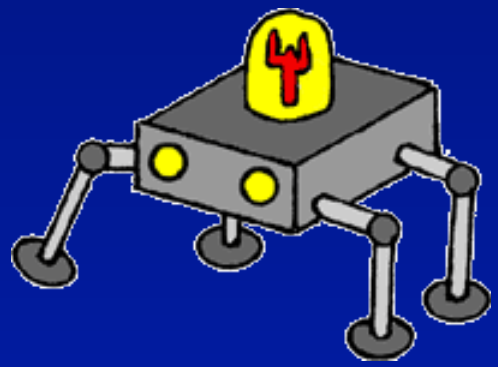
Return: A long term measure of performance

- Let’s assume our agent acts according to some rules, called a policy, p

- The return R_t is a measure of long term reward collected after time t

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \mathrm{K} = \sum_{k=0}^{\infty} \gamma^k r_{t+1+k} \qquad 0 \le \gamma \le 1$$





$$R_0 = 3 + \mathrm{K} - \gamma^3 1 - \gamma^4 1 + \mathrm{K} + \gamma^8 50 + \mathrm{K}$$

Value = Utility = Expected Return

- R_t is a random variable

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \mathrm{K} = \sum_{k=0}^{\infty} \gamma^k r_{t+1+k}$$

- So it has an expected value in a state under a given policy

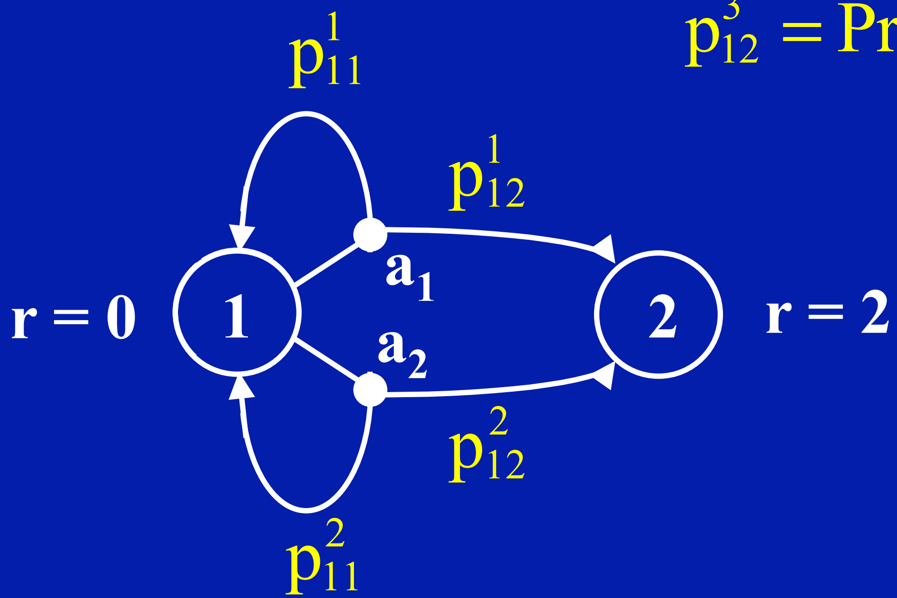
$$V^{\pi}(s_t) = E\{R_t \mid s_t, \pi\} = E\left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+1+k} \mid s_t, \pi \right\}$$

- RL problem is to find optimal policy p* that maximises the expected value in every state

$$\pi(s,a) = \Pr(A = a \mid S = s)$$

Markov Decision Processes (MDPs)

- The transitions between states are uncertain
- The probabilities depend only on the current state



$$p_{12}^{3} = \Pr(s_{t+1} = 2 \mid s_t = 1, a_t = 3)$$

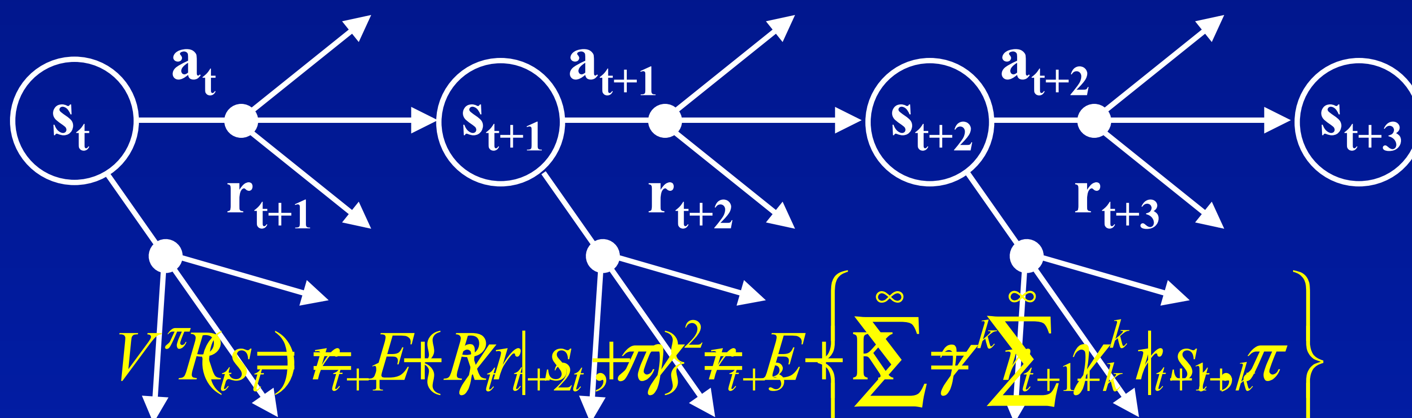
$$\mathbf{P} = \begin{bmatrix} p_{11}^{1} & p_{12}^{1} \\ p_{11}^{2} & p_{12}^{2} \end{bmatrix}$$

$$R = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$$

- Transition matrix **P**, and reward function $\mathcal{R}$

Summary of the story so far…

- Some key elements of RL problems:
  - A class of sequential decision making problems
  - We want p*
  - Performance metric: Short term ⟶ Long term
- Some common elements of RL solutions
  - Exploit conditional independence
  - Randomised interaction



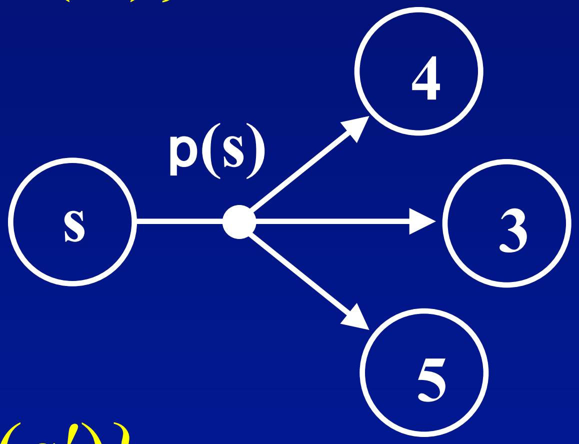
$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

$$V^\pi(s_t) = E\{R_t \mid s_t, \pi\} = E\left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+1+k} \mid s_t, \pi \right\}$$

Bellman equations

- Conditional independence allows us to define expected return in terms of a recurrence relation:

$$V^{\pi}(s) = \sum_{s' \in S} p_{\mathrm{ss'}}^{\pi(\mathrm{s})} \{ \mathrm{R}_{\mathrm{ss'}}^{\pi(\mathrm{s})} + \gamma V^{\pi}(s') \}$$



where $p_{ss'}^{\pi(s)} = \Pr(s' \mid s, \pi(s))$

and $\mathrm{R}_{ss'}^{\pi(s)} = \mathrm{E}\{ r_{t+1} \mid s_{t+1} = s', s_t = s, \pi \}$

$$Q^{*}(s,a) = \sum_{s' \in S} p_{\mathrm{ss'}}^{a} \{ \mathrm{R}_{\mathrm{ss'}}^{a} + \gamma V^{*}(s') \}$$

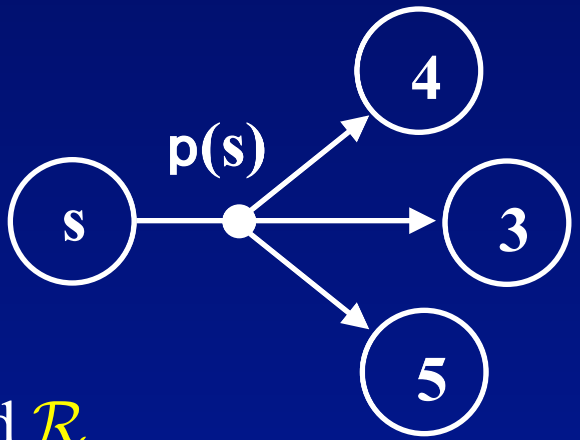
where

$$V^{*}(s) = \max_{a \in A} \left[Q^{*}(s,a) \right]$$

Two types of bootstrapping

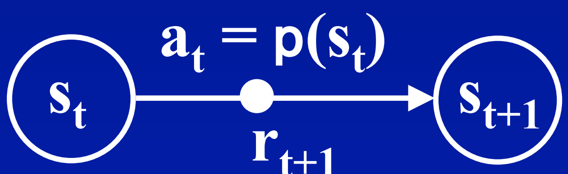
- We can bootstrap using explicit knowledge of **P** and $\mathcal{R}$ (Dynamic Programming)

$$\hat{V}_n^{\pi}(s) = \sum_{s' \in S} p_{ss'}^{\pi(s)} \{ R_{ss'}^{\pi(s)} + \gamma \hat{V}_{n-1}^{\pi}(s') \}$$



- Or we can bootstrap using samples from **P** and $\mathcal{R}$ (Temporal Difference learning)

$$\hat{V}_{t+1}^{\pi}(s_t) = \hat{V}_t^{\pi}(s_t) + \alpha(r_{t+1} + \gamma \hat{V}_t^{\pi}(s_{t+1}) - \hat{V}_t^{\pi}(s_t))$$



TD(0) learning

t:=0

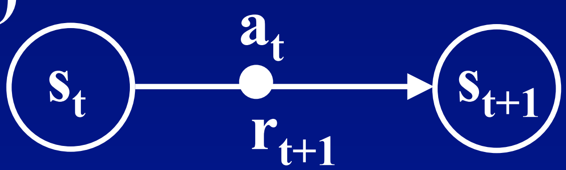
p is the policy to be evaluated

Initialise $\hat{V}_t^{\pi}(s)$ arbitrarily for all $s \in S$

Repeat

select an action a_t from $p(s_t)$

observe the transition



update $\hat{V}^{\pi}(s_t)$ according to

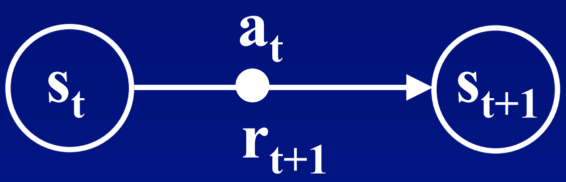
$$\hat{V}_{t+1}^{\pi}(s_t) = \hat{V}_t^{\pi}(s_t) + \alpha(r_{t+1} + \gamma \hat{V}_t^{\pi}(s_{t+1}) - \hat{V}_t^{\pi}(s_t))$$

t:=t+1

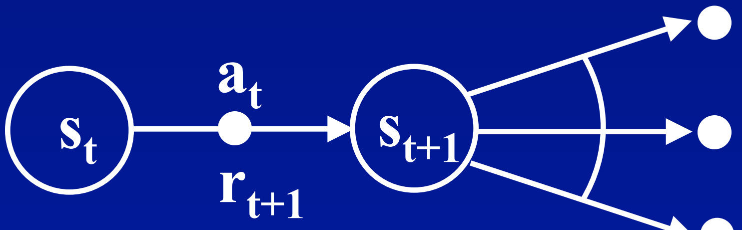
On and Off policy learning

- On policy: evaluate the policy you are following, e.g. TD learning

$$\hat{V}^{\pi}_{t+1}(s_t) = \hat{V}^{\pi}_{t}(s_t) + \alpha(r_{t+1} + \gamma\hat{V}^{\pi}_{t}(s_{t+1}) - \hat{V}^{\pi}_{t}(s_t))$$



- Off-policy: evaluate one policy while following another policy



- E.g. One step Q-learning

$$\hat{Q}_{t+1}(s_t, a_t) = \hat{Q}_t(s_t, a_t) + \alpha\left(r_{t+1} + \gamma \max_{b\in A}\left\{\hat{Q}_t(s_{t+1}, b)\right\} - \hat{Q}_t(s_t, a_t)\right)$$

Off policy learning of control

- Q-learning is powerful because
  - it allows us to evaluate p^*
  - while taking non-greedy actions (explore)

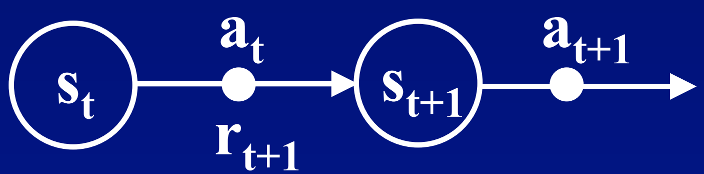
- h-greedy is a simple and popular exploration rule:
  - take a greedy action with probability h
  - Take a random action with probability 1-h

- Q-learning is guaranteed to converge for MDPs
  (with the right exploration policy)

- Is there a way of finding p^* with an on-policy learner?

On policy learning of control: Sarsa

- Discard the max operator
- Learn about the policy you are following



$$\hat{Q}^{\pi}_{t+1}(s_t, a_t) = \hat{Q}^{\pi}_t(s_t, a_t) + \alpha\left(r_{t+1} + \gamma \hat{Q}^{\pi}_t(s_{t+1}, a_{t+1}) - \hat{Q}^{\pi}_t(s_t, a_t)\right)$$

- Change the policy gradually
- Guaranteed to converge for Greedy in the Limit Infinite Exploration Policies

On policy learning of control: Sarsa

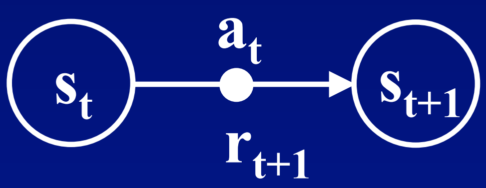
```
t:=0
```

Initialise $\hat{Q}_t^{\pi}(s,a)$ **arbitrarily for all** $s \in S, a \in A$

select an action a_t **from explore(** $\hat{Q}_t^{\pi}(s_t,a)$ **)**

Repeat

observe the transition



select an action a_{t+1} **from explore(** $\hat{Q}_t^{\pi}(s_{t+1},a)$ **)**

update $\hat{Q}_t^{\pi}(s_t,a_t)$ **according to**

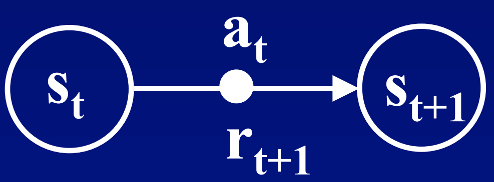
$$\hat{Q}_{t+1}^{\pi}(s_t,a_t) = \hat{Q}_t^{\pi}(s_t,a_t) + \alpha\left(r_{t+1} + \gamma \hat{Q}_t^{\pi}(s_{t+1},a_{t+1}) - \hat{Q}_t^{\pi}(s_t,a_t)\right)$$

```
t:=t+1
```

Summary: TD, Q-learning, Sarsa

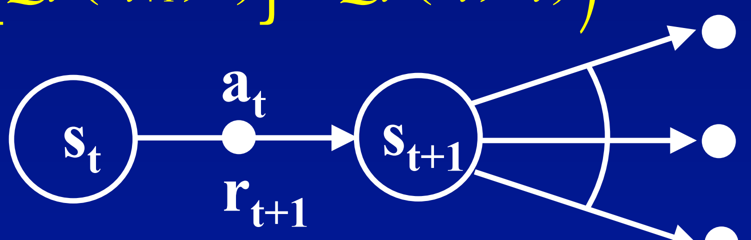
- TD learning

$$\hat{V}^{\pi}_{t+1}(s_t) = \hat{V}^{\pi}_{t}(s_t) + \alpha(r_{t+1} + \gamma \hat{V}^{\pi}_{t}(s_{t+1}) - \hat{V}^{\pi}_{t}(s_t))$$



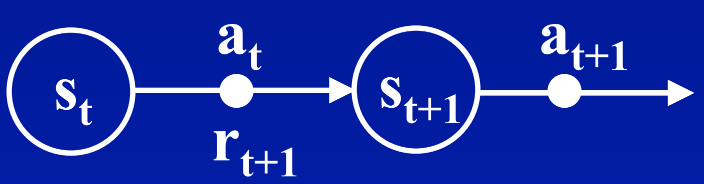
- One step Q-learning

$$\hat{Q}_{t+1}(s_t, a_t) = \hat{Q}_{t}(s_t, a_t) + \alpha\left(r_{t+1} + \gamma \max_{b \in A}\left\{\hat{Q}_{t}(s_{t+1}, b)\right\} - \hat{Q}_{t}(s_t, a_t)\right)$$



- Sarsa learning

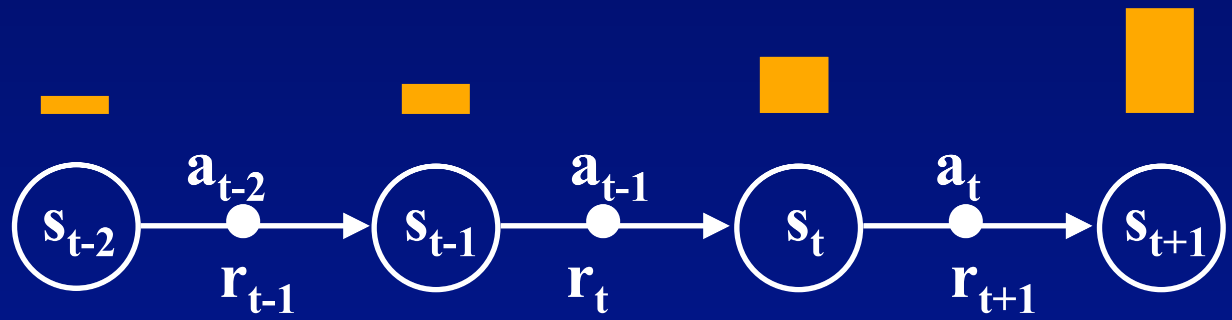
$$\hat{Q}^{\pi}_{t+1}(s_t, a_t) = \hat{Q}^{\pi}_{t}(s_t, a_t) + \alpha\left(r_{t+1} + \gamma \hat{Q}^{\pi}_{t}(s_{t+1}, a_{t+1}) - \hat{Q}^{\pi}_{t}(s_t, a_t)\right)$$



Speeding up learning: Eligibility traces, TD(l)

- TD learning only passes the TD error to one state

$$\hat{V}^{\pi}_{t+1}(s_t) = \hat{V}^{\pi}_{t}(s_t) + \alpha(r_{t+1} + \hat{V}^{\pi}_{t}(s_{t+1}) - \hat{V}^{\pi}_{t}(s_t))$$



- We add an eligibility for each state: $\bar{e}_{t+1}(s) = \begin{cases} 1 & \text{if } s_t = s \\ \gamma\lambda\bar{e}_t(s) & \text{otherwise} \end{cases}$
  where $0 \le \lambda \le 1$

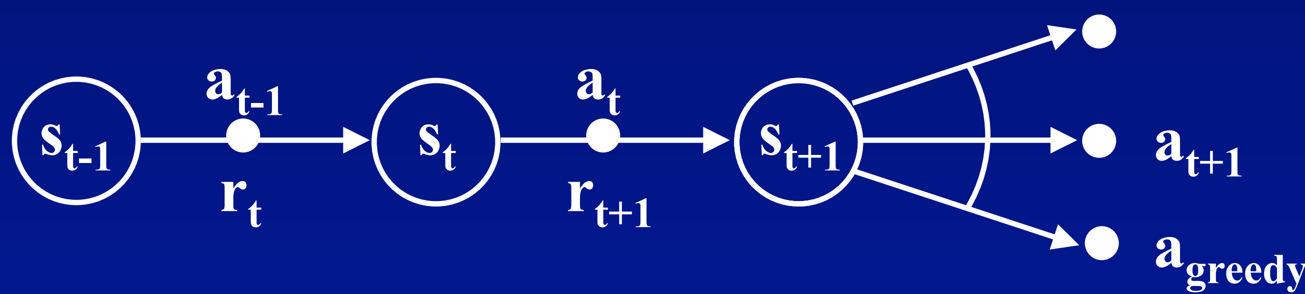
- Update $\hat{V}^{\pi}_{t}(s)$ in **every** state $s \in S$ proportional to the eligibility

$$\hat{V}^{\pi}_{t+1}(s) = \hat{V}^{\pi}_{t}(s) + \alpha(r_{t+1} + \hat{V}^{\pi}_{t}(s_{t+1}) - \hat{V}^{\pi}_{t}(s_t))\bar{e}_t(s)$$

Eligibility traces for learning control: Q(l)

- There are various eligibility trace methods for Q-learning
- Update for every s,a pair

$$\hat{Q}_{t+1}(s,a)=\hat{Q}_t(s,a)+\alpha\left(r_{t+1}+\gamma\max_{b\in A}\left\{\hat{Q}_t(s_{t+1},b)\right\}-\hat{Q}_t(s_t,a_t)\right)\bar{e}_t(s,a)$$

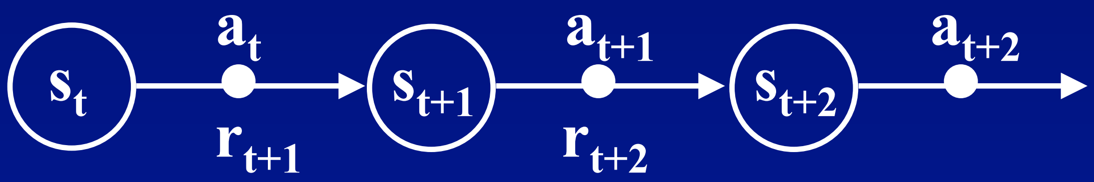


- Pass information backwards through a non-greedy action
- Lose convergence guarantee of one step Q-learning
- Watkin's Solution: zero all eligibilities after a non-greedy action
- Problem: you lose most of the benefit of eligibility traces

Eligibility traces for learning control: Sarsa(l)

- Solution: use Sarsa since it’s on policy
- Update for every s,a pair

$$\hat{Q}^{\pi}_{t+1}(s,a)=\hat{Q}^{\pi}_{t}(s,a)+\alpha\Big(r_{t+1}+\gamma\hat{Q}^{\pi}_{t}(s_{t+1},a_{t+1})-\hat{Q}^{\pi}_{t}(s_{t},a_{t})\Big)\bar{e}_{t}(s,a)$$



- Keeps convergence guarantees

Approximate Reinforcement Learning

- Why?
  - To learn in reasonable time and space
    (avoid Bellman's curse of dimensionality)
  - To generalise to new situations

- Solutions
  - Approximate the value function
  - Search in the policy space
  - Approximate a model (and plan)

Linear Value Function Approximation

- Simplest useful class of problems
- Some convergence results
- We' ll focus on linear TD(l)

Weight vector at time *t* $\vec{\theta}_t = \left(\theta_t(1), \theta_t(2) \ldots \theta_t(n)\right)'$

Feature vector for state *s* $\vec{\phi}_s = \left(\phi_s(1), \phi_s(2) \ldots \phi_s(n)\right)$

Our value estimate $\hat{V}_t^{\pi}(s) = \vec{\theta}_t' \vec{\phi}_s$

Our objective is to minimise $\mathrm{MSE}\left(\vec{\theta}_t\right) = \sum_{s \in S} P(s)\left[V^{\pi}(s) - \hat{V}_t^{\pi}(s)\right]^2$

Value Function Approximation: features

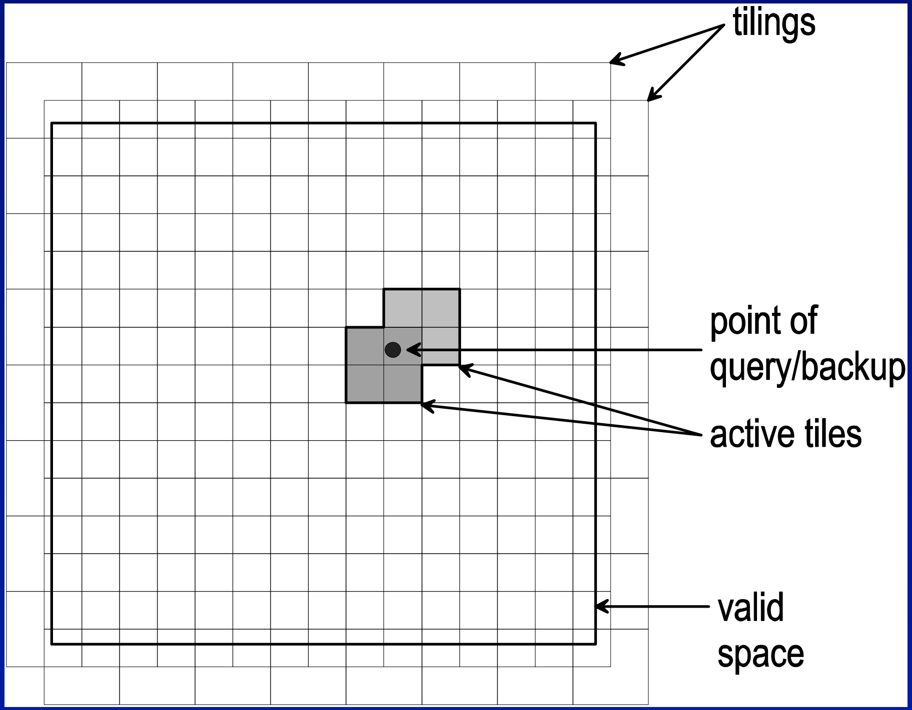
- There are numerous schemes, CMACs and RBFs are popular

- CMAC: n tiles in the space (aggregate over all tilings)

$$\vec{\phi}_s = \left(\phi_s(1), \phi_s(2) \ldots \phi_s(n)\right)$$

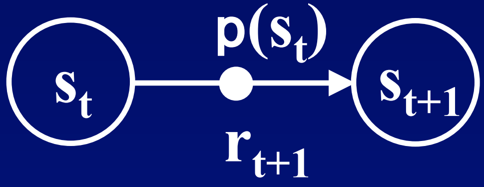
- Features $\phi_s(i) = 1$ or 0

- Properties
  - Coarse coding
  - Regular tiling ⇨ efficient access
  - Use random hashing to reduce memory



Linear Value Function Approximation

- We perform gradient descent using $\vec{\theta}_t$



- The update equation for TD(l) becomes

$$\vec{\theta}_{t+1} = \vec{\theta}_t + \alpha\left(r_t + \gamma \hat{V}_t^{\pi}\left(s_{t+1}\right) - \hat{V}_t^{\pi}\left(s_t\right)\right)\vec{e}_t$$

- Where the eligibility trace is an n-dim vector updated using

$$\vec{e}_t = \gamma\lambda\vec{e}_{t-1} + \vec{\varphi}_t$$

- If the states are presented with the frequency they would be seen under the policy p you are evaluating TD(l) converges close to $\vec{\theta}^*$

Value Function Approximation (VFA) Convergence results

- Linear TD(lambda) converges if we visit states using the on-policy distribution
- Off policy Linear T(lambda) and linear Q learning are known to diverge in some cases
- Q-learning, and value iteration used with some averagers (including k-Nearest Neighbour and decision trees) has almost sure convergence if particular exploration policies are used
- A special case of policy iteration with Sarsa style updates and linear function approximation converges

- Residual algorithms are guaranteed to converge but only very slowly

Value Function Approximation (VFA)
TD-gammon

- TD(l) learning and a Backprop net with one hidden layer
- 1,500,000 training games (self play)
- Equivalent in skill to the top dozen human players
- Backgammon has ~10^{20} states, so can't be solved using DP

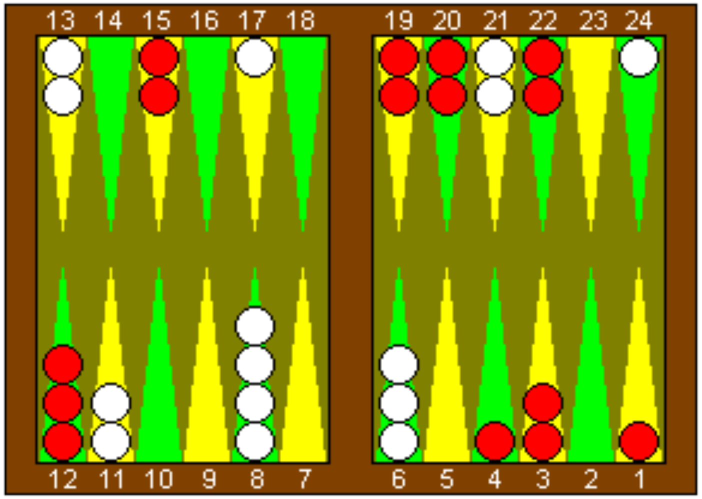
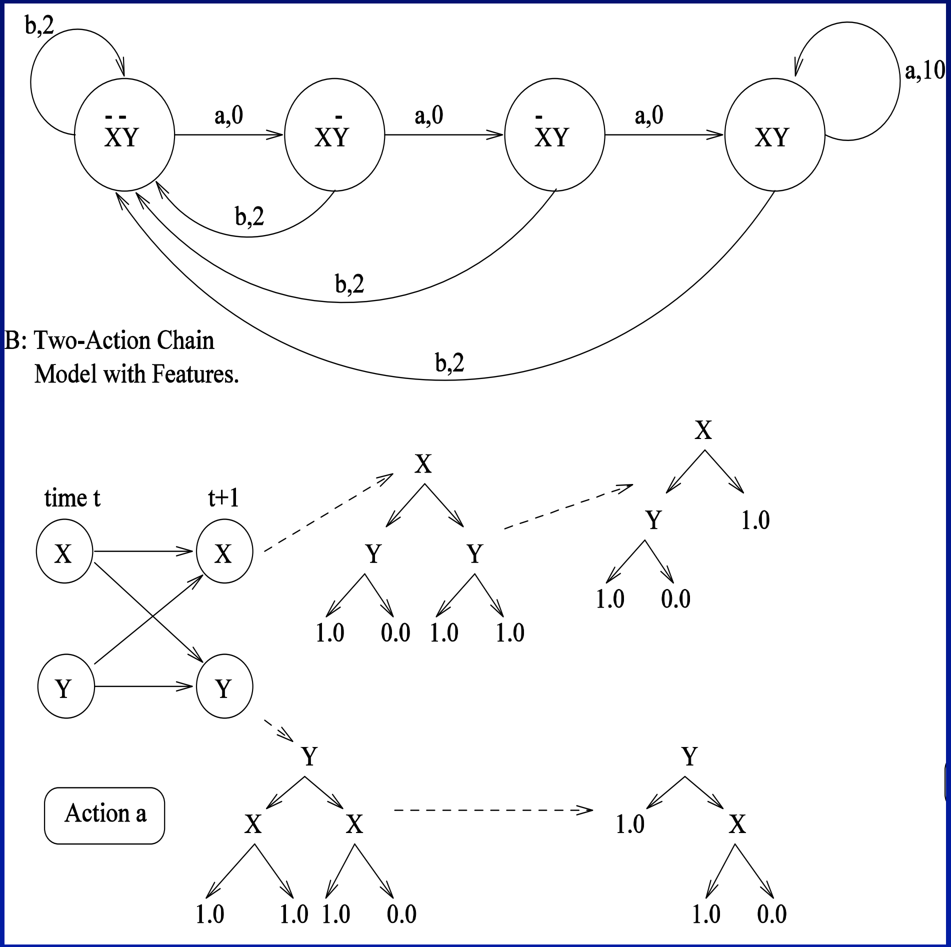


Figure 3. A complex situation where TD-Gammon's positional judgment is apparently superior to traditional expert thinking. White is to play 4-4. The obvious human play is 8-4*, 8-4, 11-7, 11-7. (The asterisk denotes that an opponent checker has been hit.) However, TD-Gammon's choice is the surprising 8-4*, 8-4, 21-17, 21-17! TD-Gammon's analysis of the two plays is given in Table 3.

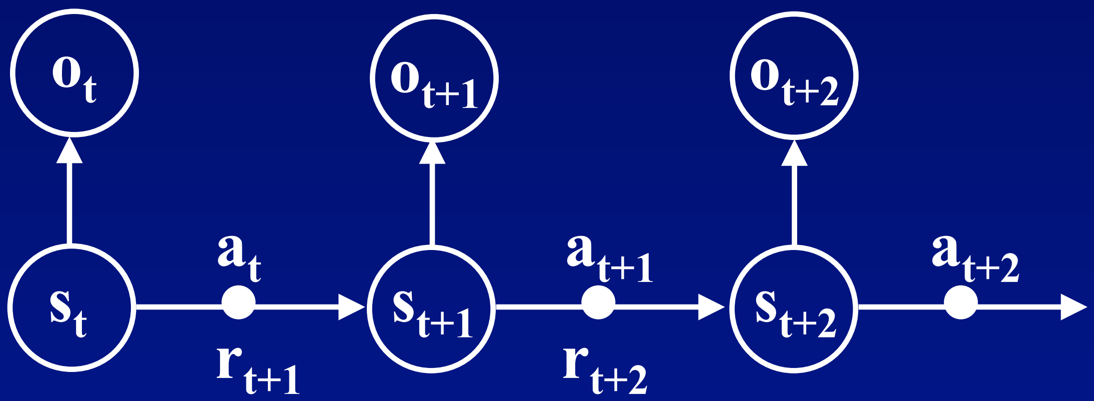
Model-based RL: structured models

- Transition model P is represented compactly using a Dynamic Bayes Net
  (or factored MDP)
- V is represented as a tree
- Backups look like goal regression operators
- Converging with the AI planning community



Reinforcement Learning with Hidden State

- Learning in a POMDP, or k-Markov environment



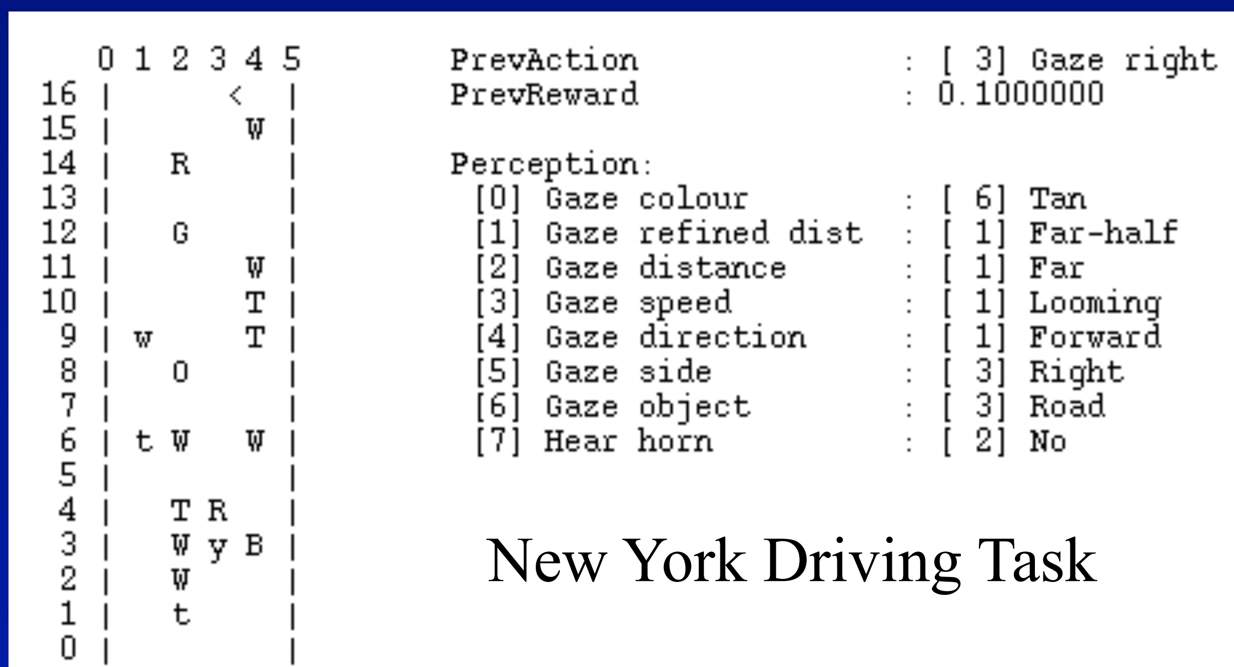
- Planning in POMDPs is intractable

- Factored POMDPs are a hot topic

- Policy search can work well

Policy Search

- Why not search directly for a policy?
- Policy gradient methods
- Evolutionary methods
- Particularly good for problems with hidden state



New York Driving Task

Other RL applications

- Elevator Control (Barto & Crites)
- Space shuttle job scheduling (Zhang & Dietterich)
- Dynamic channel allocation in cellphone networks (Singh & Bertsekas)

Hot Topics in Reinforcement Learning

- Efficient Exploration and Optimal learning
- Learning with structured models (eg. Bayes Nets)
- Learning with relational models
- Learning in continuous state and action spaces
- Hierarchical reinforcement learning
- Learning in processes with hidden state (eg. POMDPs)
- Policy search methods

Reinforcement Learning: key papers

Overviews

R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.

J. Wyatt, Reinforcement Learning: A Brief Overview. *Perspectives on Adaptivity and Learning*. Springer Verlag, 2003.

L.Kaelbling, M.Littman and A.Moore, Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 4:237-285, 1996.

Value Function Approximation

D. Bersekas and J.Tsitsiklis. *Neurodynamic Programming*. Athena Scientific, 1998.

Eligibility Traces

S.Singh and R. Sutton. Reinforcement learning with replacing eligibility traces. *Machine Learning*, 22:123-158, 1996.

Reinforcement Learning: key papers

Structured Models and Planning

C. Boutillier, T. Dean and S. Hanks. Decision Theoretic Planning: Structural Assumptions and Computational Leverage. *Journal of Artificial Intelligence Research*, 11:1-94, 1999.

R. Dearden, C. Boutillier and M.Goldsmidt. Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 121(1-2):49-107, 2000.

B. Sallans. *Reinforcement Learning for Factored Markov Decision Processes* Ph.D. Thesis, Dept. of Computer Science, University of Toronto, 2001.

K. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. Ph.D. Thesis, University of California, Berkeley, 2002.

Reinforcement Learning: key papers

Policy Search

R. Williams. Simple statistical gradient algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229-256.

R. Sutton, D. McAllester, S. Singh, Y. Mansour. Policy Gradient Methods for Reinforcement Learning with Function Approximation. *NIPS 12*, 2000.

Hierarchical Reinforcement Learning

R. Sutton, D. Precup and S. Singh. Between MDPs and Semi-MDPs: a framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112:181-211.

R. Parr. *Hierarchical Control and Learning for Markov Decision Processes*. PhD Thesis, University of California, Berkeley, 1998.

A. Barto and S. Mahadevan. Recent Advances in Hierarchical Reinforcement Learning. *Discrete Event Systems Journal* 13: 41-77, 2003.

Reinforcement Learning: key papers

Exploration

N. Meuleau and P.Bourgnine. Exploration of multi-state environments: Local Measures and back-propagation of uncertainty. *Machine Learning*, 35:117-154, 1999.

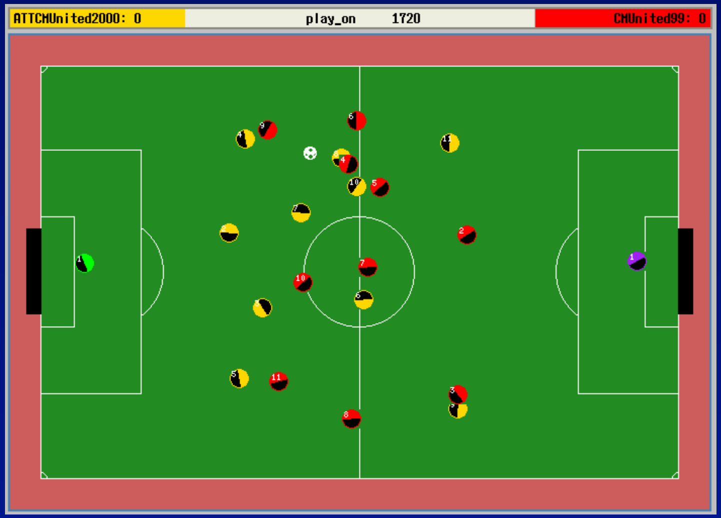
J. Wyatt. Exploration control in reinforcement learning using optimistic model selection. In *Proceedings of 18th International Conference on Machine Learning*, 2001.

POMDPs

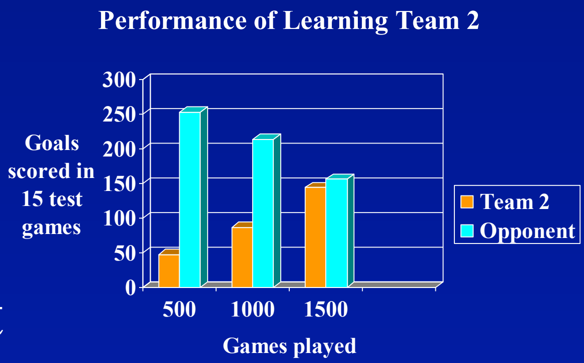
L. Kaelbling, M. Littman, A. Cassandra. Planning and Acting in Partially Observable Stochastic Domains. *Artificial Intelligence*, 101:99-134, 1998.

Extras: 1 Applications

Multi-agent RL: Learning to play football



- Learning to play in a team
- Too time consuming to do on real robots
- There is a well established simulator league
- We can learn effectively from reinforcement



Extras: 2 Exploration

The Exploration problem: intuition

- We are learning to maximise performance
- But how should we act while learning?
- Trade-off: exploit what we know or explore to gain new information?
- Optimal learning: maximise performance while learning given your imperfect knowledge

The optimal learning problem

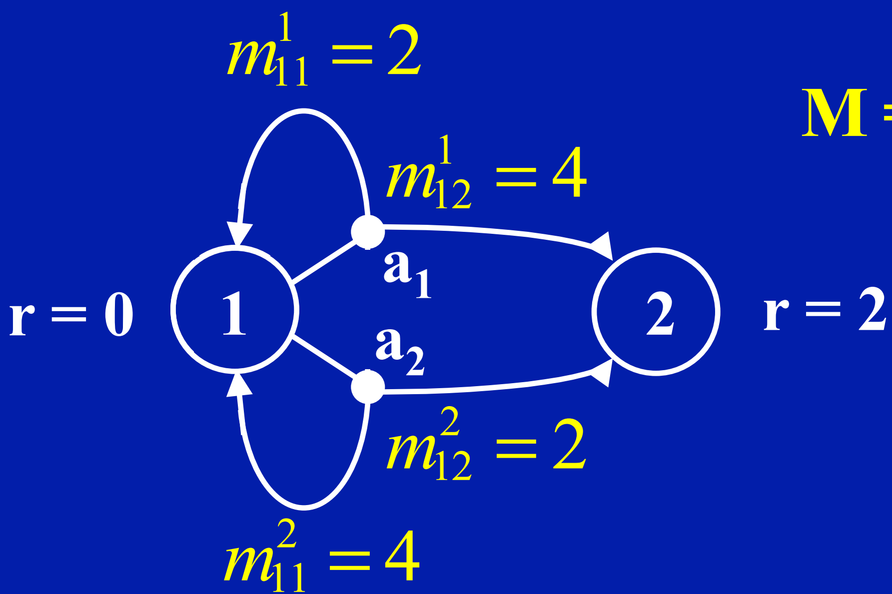
- If we knew **P** it would be easy

$$Q^*(i,a) = \sum_{j \in S} p_{\mathrm{ij}}^{a} \{\mathrm{R}_j + \gamma \max_{b \in \mathrm{A}} \{Q^*(j,b)\}\}$$

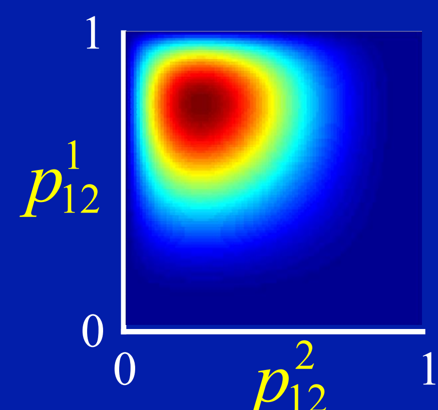
- However …
  - We estimate **P** from observations
  - **P** is a random variable
  - There is a density *f*(**P**) over the space of possible MDPs
  - What is it? How does it help us solve our problem?

A density over MDPs

- Suppose we've wandered around for a while
- We have a matrix **M** containing the transition counts



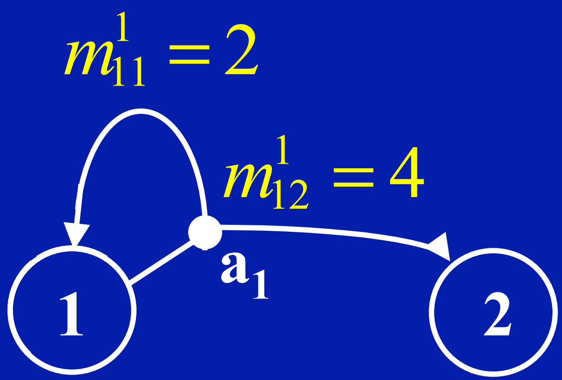
$$\mathbf{M} = \begin{bmatrix} m_{11}^{1} & m_{12}^{1} \\ m_{11}^{2} & m_{12}^{2} \end{bmatrix} \quad \mathbf{P} = \begin{bmatrix} \mathrm{p}_{11}^{1} & \mathrm{p}_{12}^{1} \\ \mathrm{p}_{11}^{2} & \mathrm{p}_{12}^{2} \end{bmatrix}$$

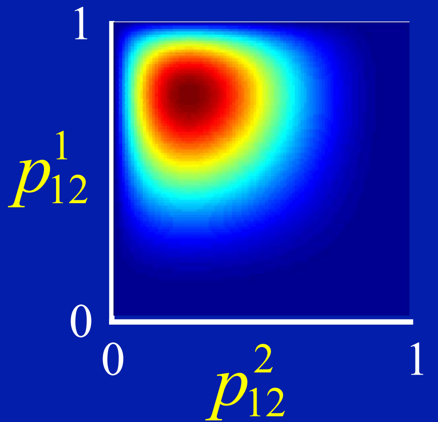


- The density over possible **P** depends on **M**, $f(\mathbf{P}|\mathbf{M})$ and is a product of Dirichlet densities

A density over multinomials

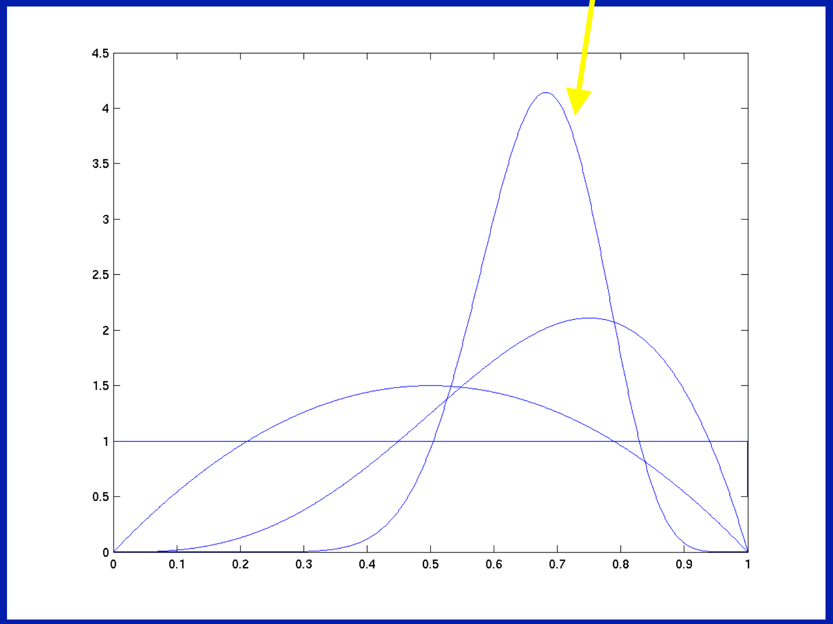
$$\mathbf{m}^{1} = \begin{bmatrix} m_{11}^{1} & m_{12}^{1} \end{bmatrix} \qquad \mathbf{p}^{1} = \begin{bmatrix} \mathrm{p}_{11}^{1} & \mathrm{p}_{12}^{1} \end{bmatrix}$$





$f(p_{12}^{1} \mid \mathbf{m}^{1})$

$\mathbf{m}^{1} = [8,16]$

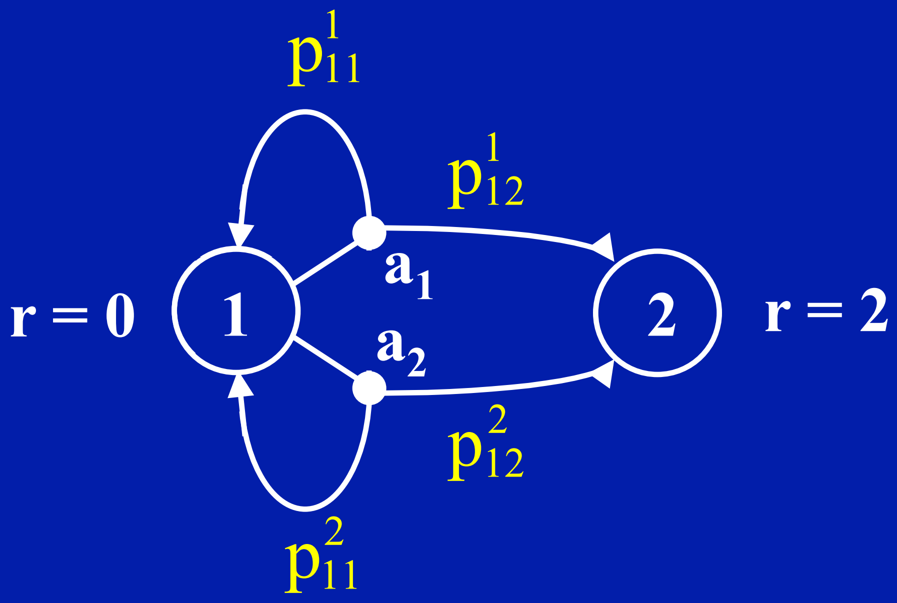


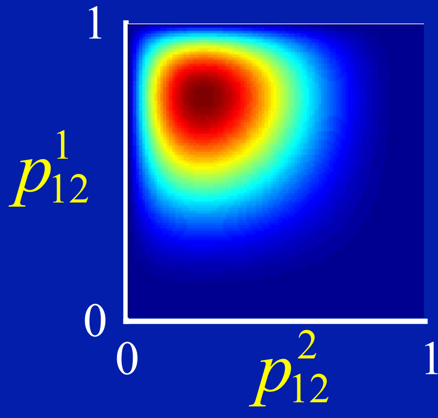
p_{12}^{1}

Optimal learning formally stated

- Given $f(\mathbf{P}|\mathbf{M})$, find p that maximises

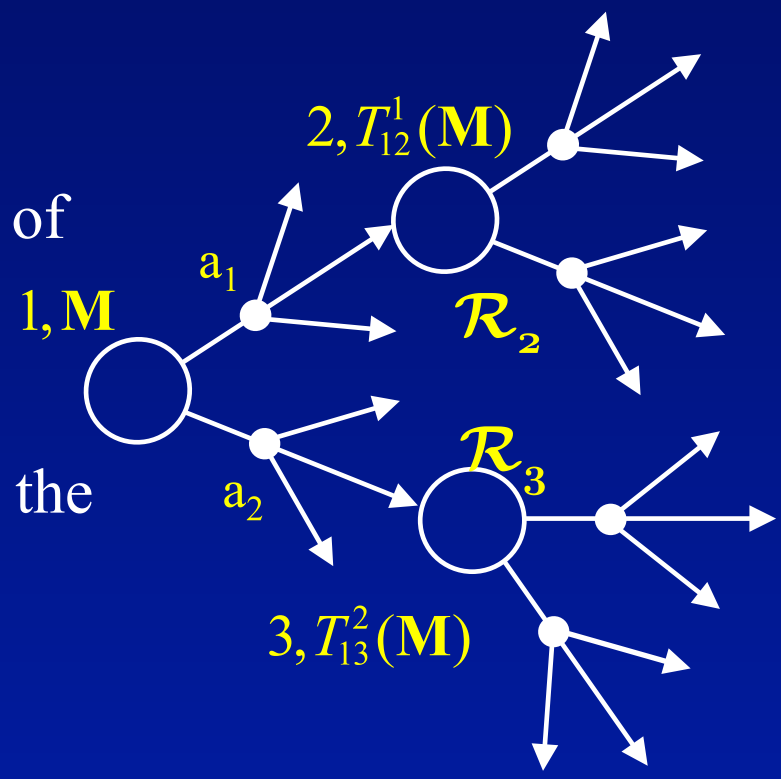
$$Q^*(i,a,\mathbf{M}) = \int_{\mathbf{M}} Q^*(i,a,\mathbf{P}) f(\mathbf{P}\,|\,\mathbf{M})d\mathbf{P}$$





Transforming the problem

- When we evaluate the integral we get another MDP!
- This one is defined over the space of information states
- This space grows exponentially in the depth of the look ahead

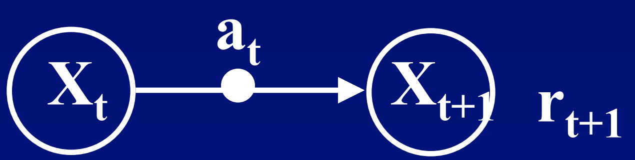


$$Q^*(i,a,\mathbf{M}) = \sum_j \bar{p}^a_{ij}(\mathbf{M})\left\{\mathrm{R}_j + \gamma V^*(j, T^a_{ij}(\mathbf{M}))\right\}$$

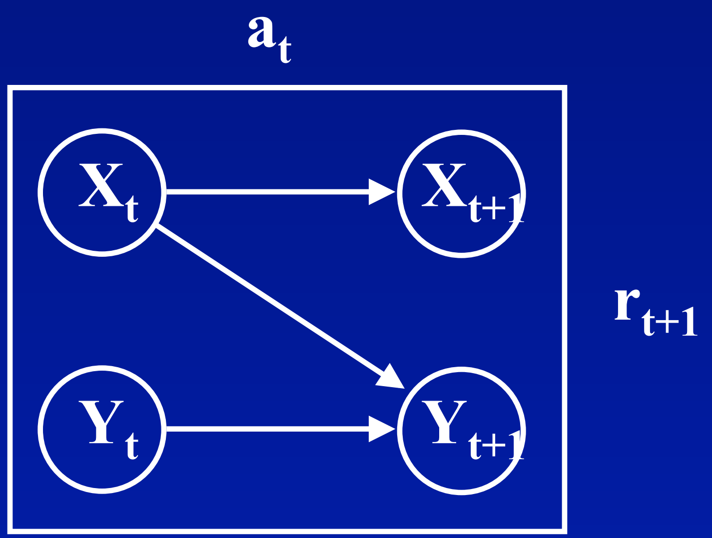
Extras: 3 Structured Representations

RL in structured environments

- A Markov Decision Process describes the evolution of one variable



- We can extend this to several variables whose evolution is coupled

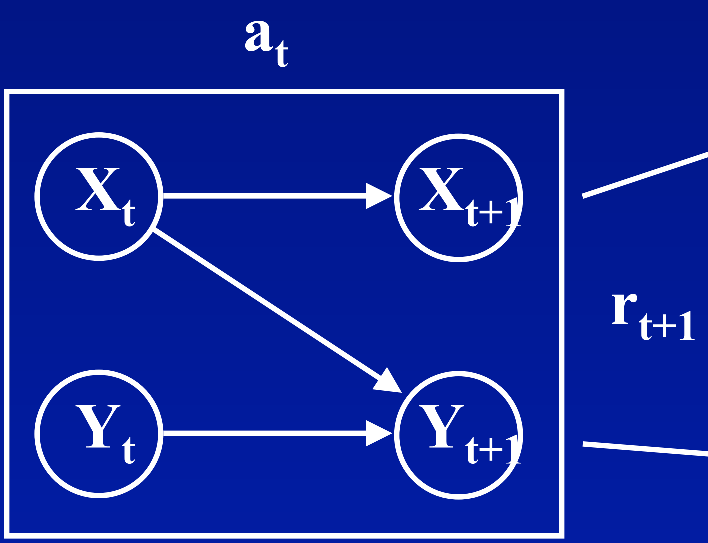


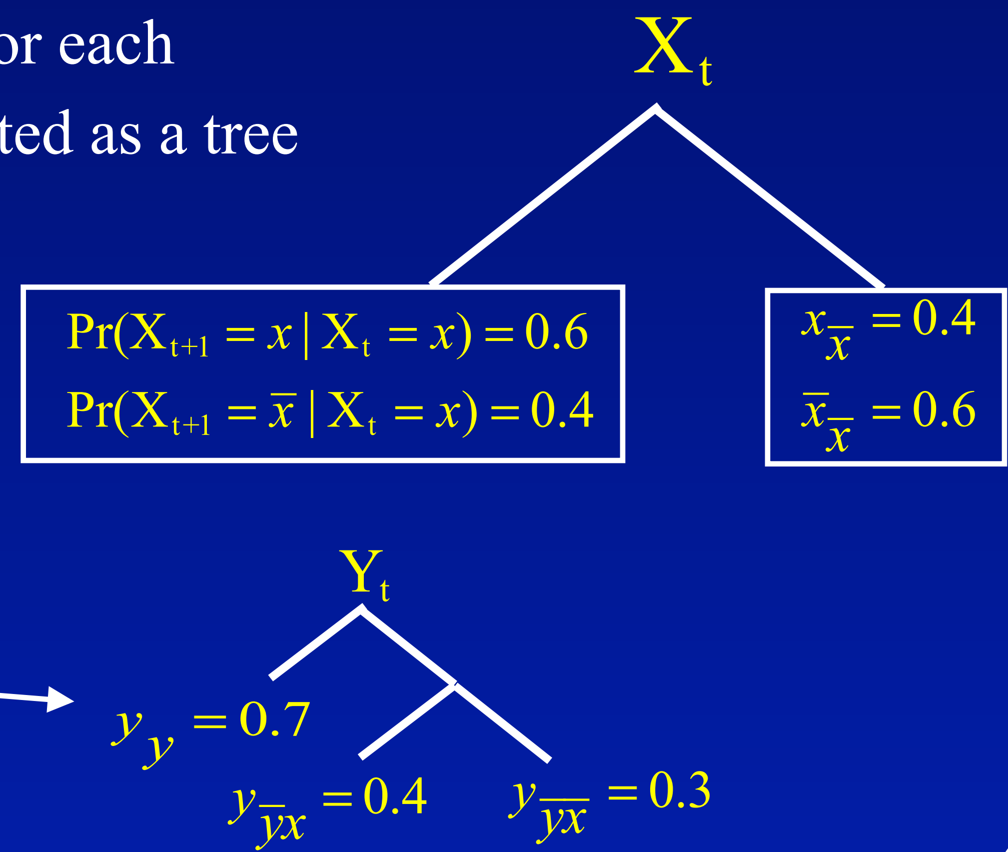
RL in structured environments

- Imagine that each variable here only has two states

$$X_t = x$$
$$X_t = \bar{x}$$

- The transition function for each variable can be represented as a tree

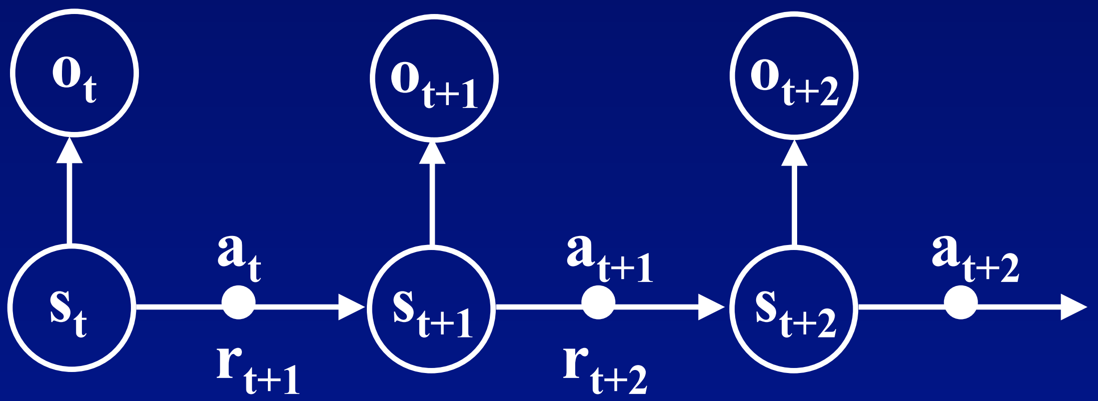




Extras: 4 Hidden State

Challenge: Learning with Hidden State

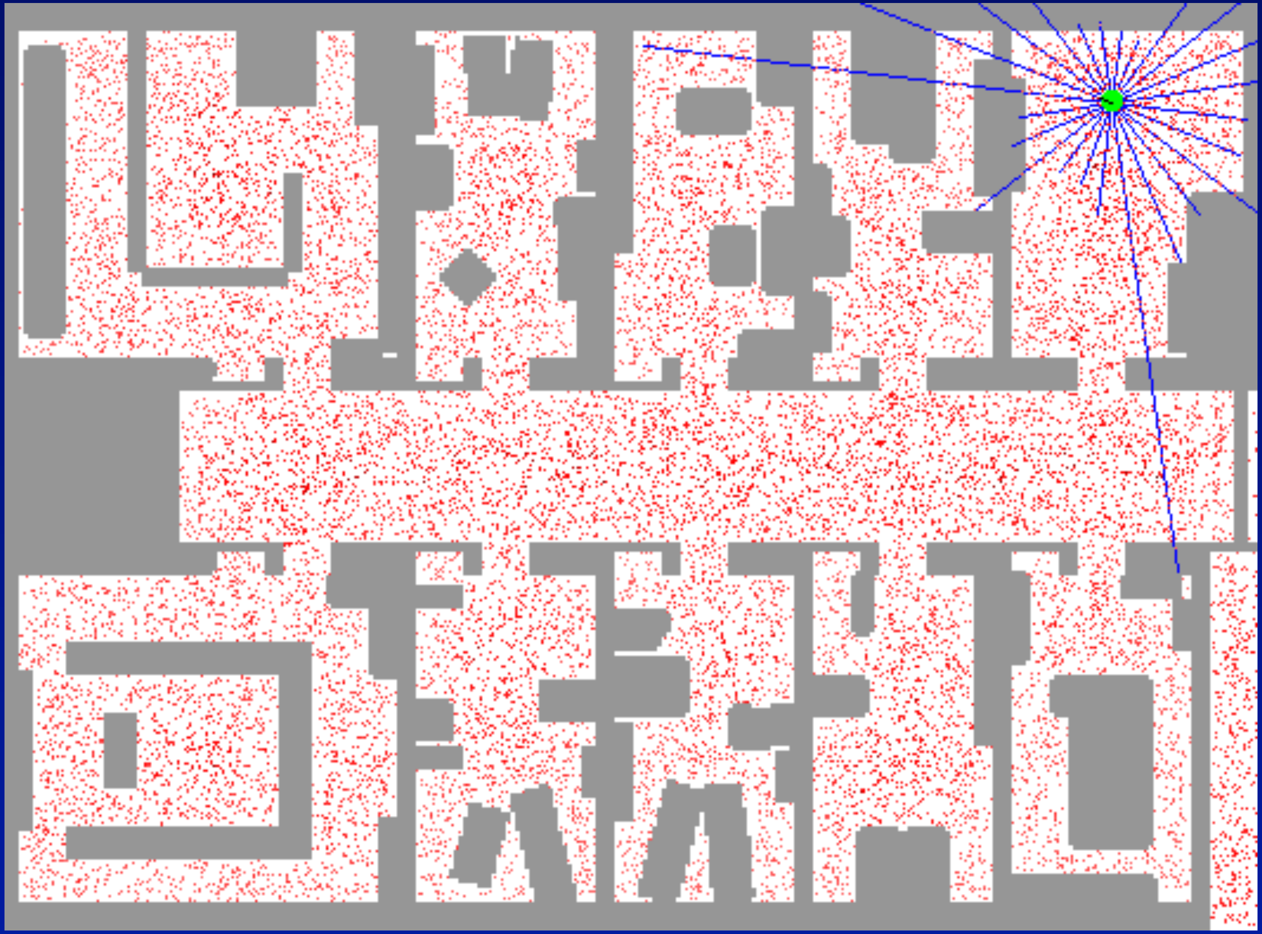
- Learning in a POMDP, or k-Markov environment



- Planning in POMDPs is intractable

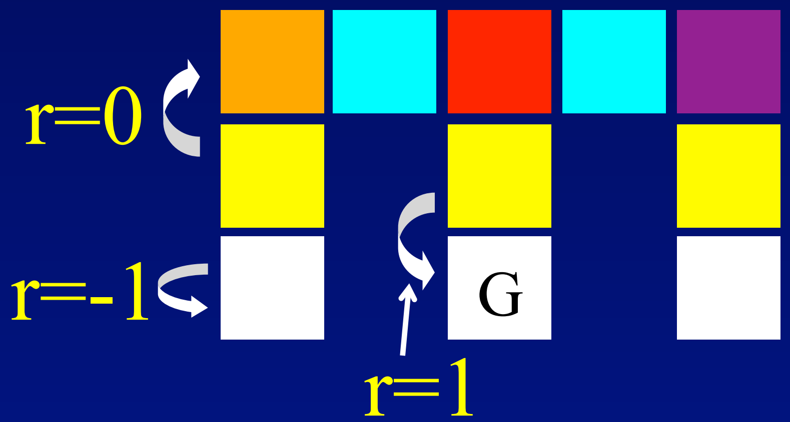
- State estimation isn't ⇒ POMDPs are the basis of many localisation, navigation and mapping algorithms in mobile robotics

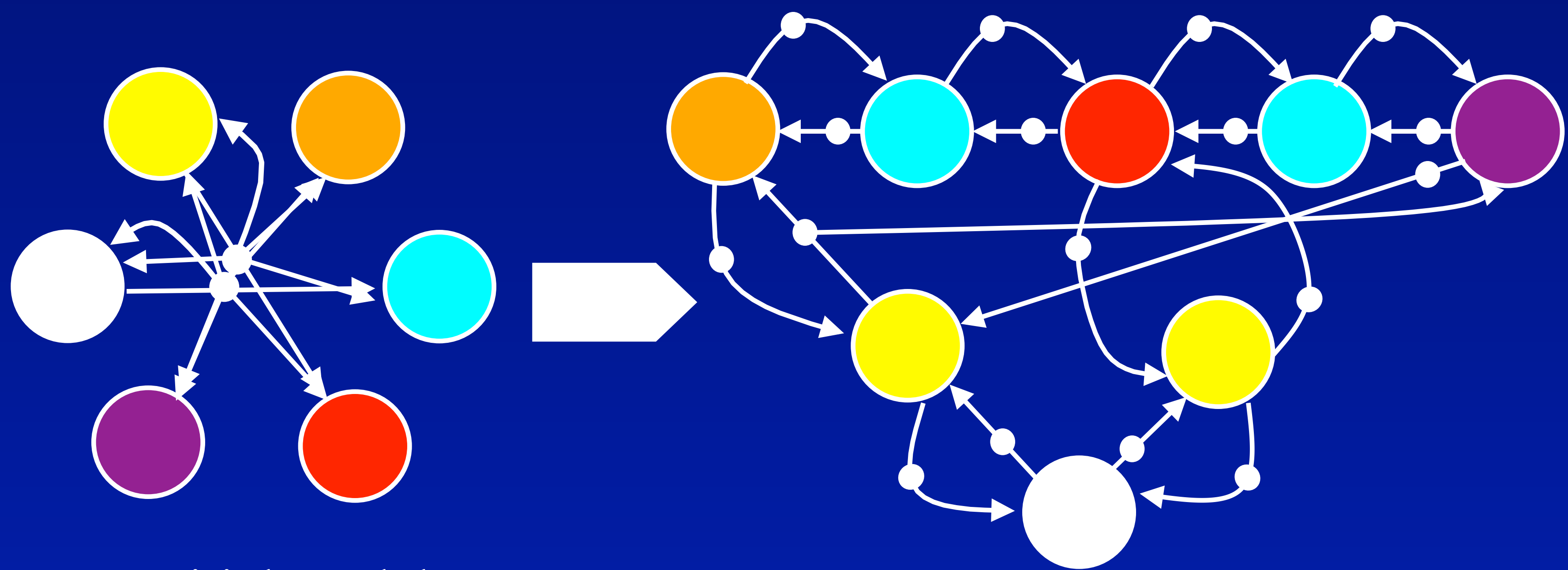
POMDPs in robot localisation



RL in POMDPs is hard

- Learn a POMDP model
- Learn/infer p*
- Same colour states look the same



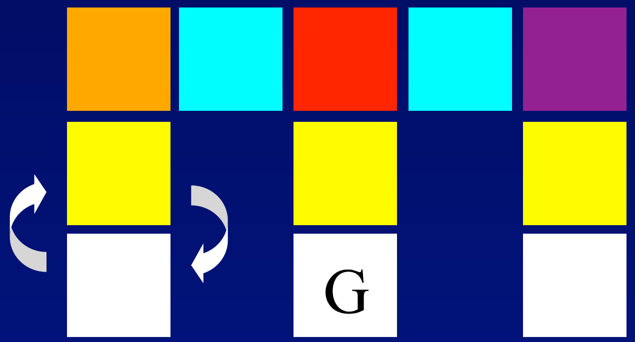


Initial model

A good enough model

Poor Exploration → Amnesia

- Select the best looking (greedy) action with Pr = .95



- Otherwise select a random action
- Agent gets confused about where it is and spends its time oscillating between states
- This causes the learner to fail to find the right model or p*
- It can even find p* and then forget it again

Good exploration avoids instability

- Which state am I most likely to be in?
- Pick the least tried action in that state
- Belief Counter exploration
- Improves learning performance