Reinforcement Learning: Part 2

Chris Watkins

Department of Computer Science
Royal Holloway, University of London

July 27, 2015

TD(0) learning

Define the *temporal difference prediction error*

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$$

Agent maintains a V-table, and updates $V(s_t)$ at time $t+1$:

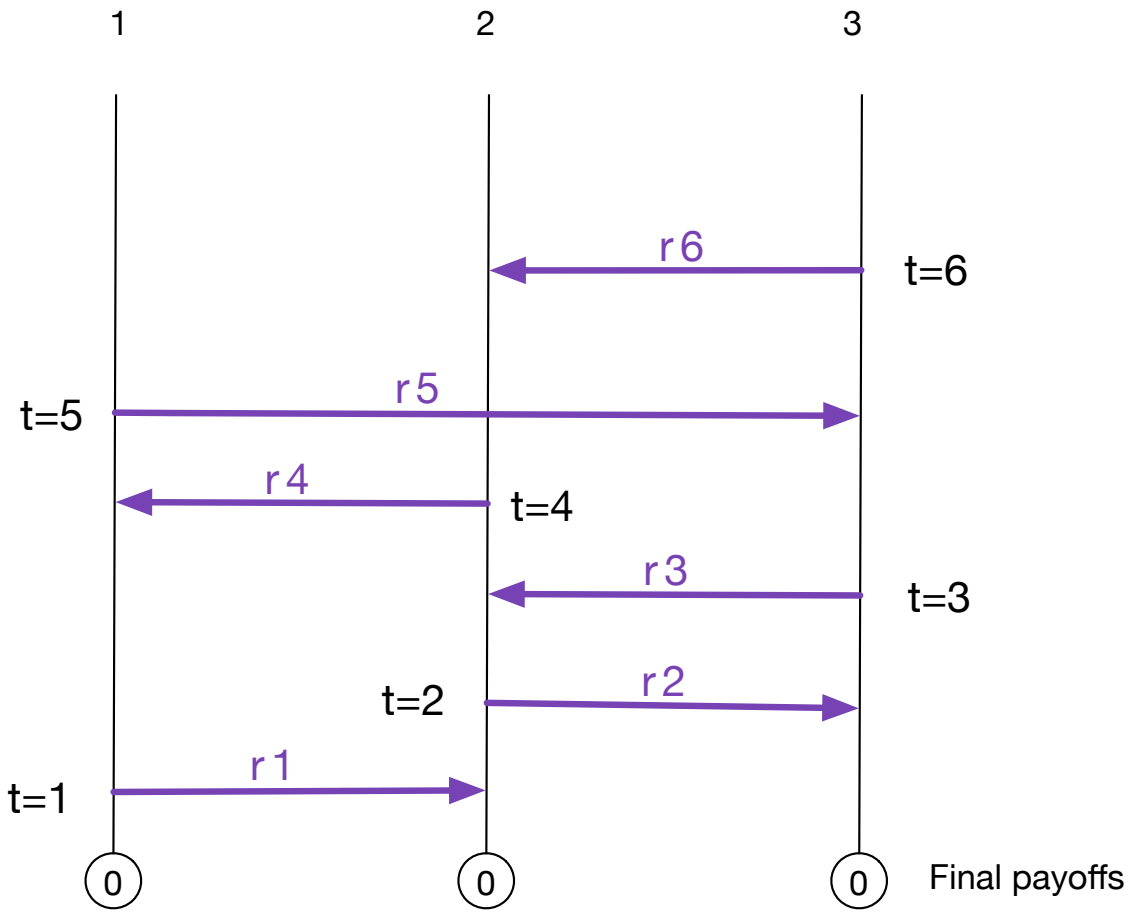
$$V(s_t) \leftarrow V(s_t) + \alpha\delta_t$$

Simple mechanism; solves problem of short segments of experience.

Dopamine neurons seem to compute δ_t !

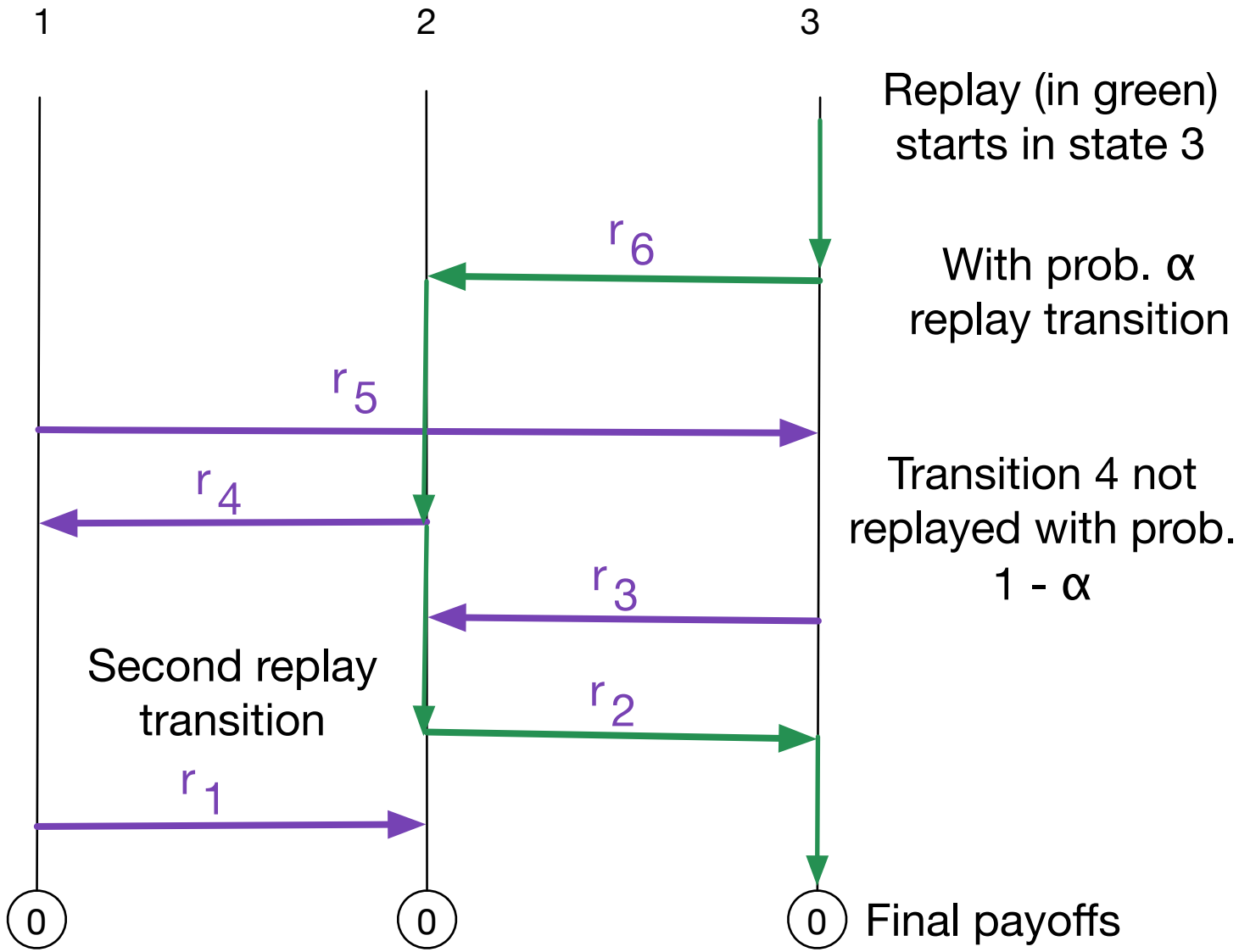
Does TD(0) converge?
Can be proved using results from theory of stochastic approximation, but simpler to consider a visual proof.

Replay process: exact values of replay process are equal to TD estimates of values of actual process



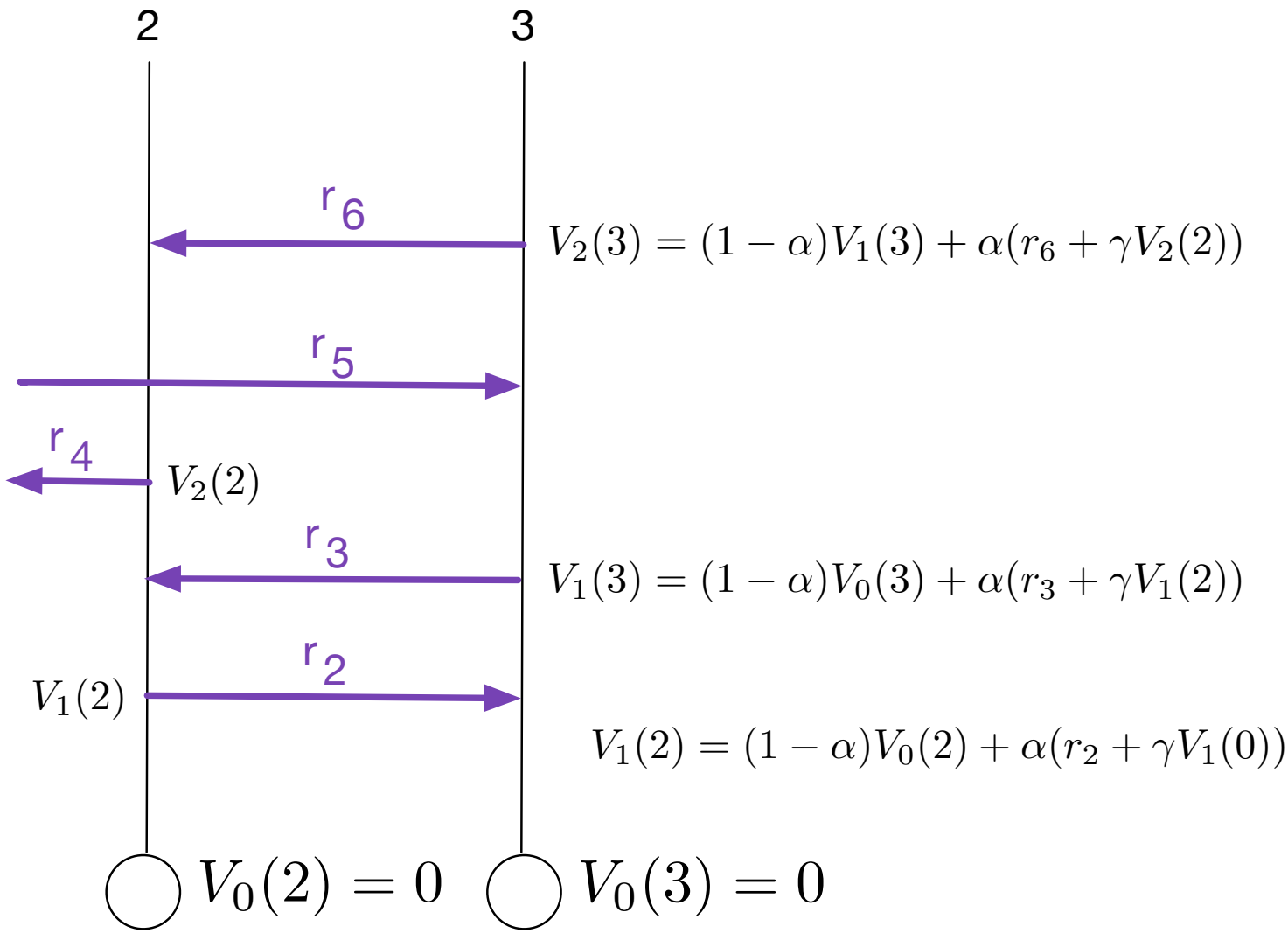
Shows 7 state-transitions and rewards, in a 3 state MDP. Replay process is built from bottom, and replayed from top.

Replay process: example of replay sequence



Return of this replay = $r_6 + \gamma\, r_2$

Values of replay process states



Each stored transition is replayed with probability α

Downward transitions have no discount factor.

Replay process: immediate remarks

- The values of states in the replay process are exactly equal to the TD(0) estimated values of corresponding states in the observed process.
- For small enough α, and with sufficiently many TD(0) updates from each state, the values in the replay process will approach the true values of the observed process.
- Observed transitions can be replayed many times: in the limit of many replays, state values converge to the value function of the maximum likelihood MRP, given the observations.
- Rarely visited states should have higher α, or (better) their transitions replayed more often.
- Stored sequences of actions should be replayed in reverse order.
- Off-policy TD(0) estimation by re-weighting observed transitions

Model-free estimation: backward-looking TD(1)

Idea 2: for each state visited, calculate the *return* for a long sequence of observations, and then update the estimated value of the state.

Set $T \gg \frac{1}{1-\gamma}$. For each state s_t visited, and for a learning rate α,

$$V(s_t) \leftarrow (1-\alpha)V(s_t) + \alpha(r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^T r_{t+T})$$

Problems:

- Return estimate only computed after T steps; need to remember last T states visited. Update is late!
- What if process is frequently interrupted, so that only small segments of experience available?
- Estimate is unbiased, but could have high variance. Does not exploit Markov property!

Telescoping of TD errors

$$
\begin{aligned}
TD(1)(s_0) - V(s_0) &= r_0 + \gamma r_1 + \cdots \\
&= -V(s_0) + r_0 + \gamma V(s_1) + \\
&\quad \gamma(r_1 + \gamma V(s_2) - V(s_1)) \\
&\quad \gamma^2(r_2 + \gamma V(s_3) - V(s_2)) \\
&\quad \vdots \\
&= \delta_0 + \gamma\delta_1 + \gamma^2\delta_2 + \cdots
\end{aligned}
$$

Hence the TD(1) error arrives incrementally in the δ_t.

TD(λ)

As a compromise between TD(1) (full reward sequence) and TD(0) (one step) updates, there is a convenient recursion called TD(λ), for $0 \leq \lambda \leq 1$.

The 'accumulating traces' update uses an 'eligibility trace' $z_t(i)$, defined for each state i at each time t. $z_0(i)$ is zero for all i:

$$\begin{aligned} \delta_t &= r_t + \gamma V_t(s_{t+1}) - V_t(s_t) \\ z_t(i) &= [s_t = i] + \gamma\lambda z_{t-1}(i) \\ V_{t+1}(i) &= V_t(i) + \alpha\delta_t z_t(i) \end{aligned}$$

Q-learning of control

An agent in a MDP maintains a table of Q values, which need not (at first) be consistent with any policy.

When agent performs a in state s, and receives r and transitions to s', it is tempting to update $Q(s, a)$ by:

$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha(r + \gamma \max_b Q(s', b))$$

This is a stochastic, partial value-iteration update.

It is possible to prove convergence by stochastic approximation arguments,
but can we devise a suitable replay process which makes convergence obvious?

Replay process for Q-learning

Suppose that Q-learning updates are carried out for a set of $\langle s, a, s', r \rangle$ experiences.

We construct a *replay MDP* using the $\langle s, a, s', r \rangle$ data.

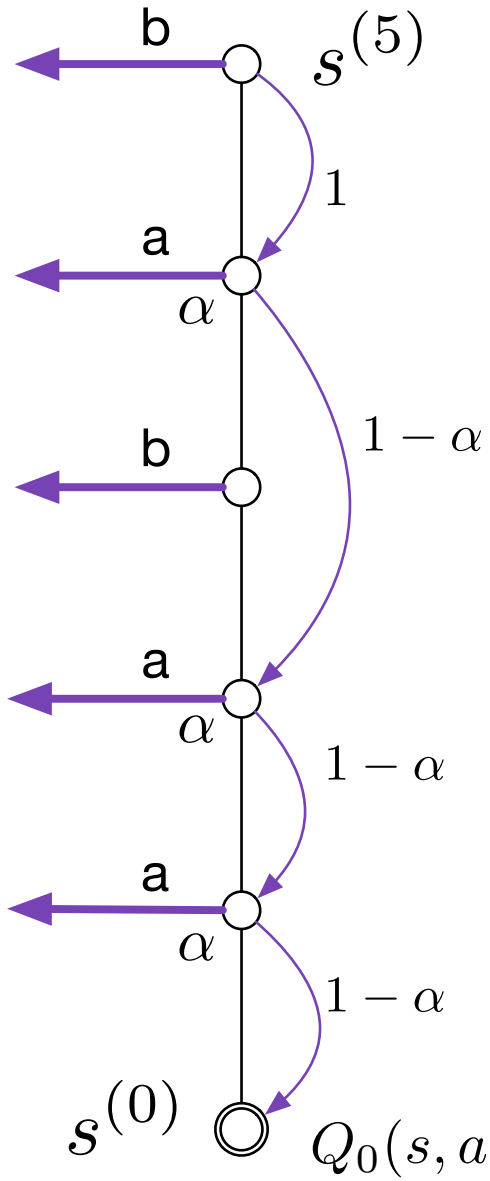
If Q values for s were updated 5 times using the data, the replay MDP contains states $s^{(0)}, s^{(1)}, \ldots, s^{(5)}$.

The *optimal* Q values of $s^{(k)}$ in the replay MDP are **equal** to the *estimated* Q values of the learner after the kth Q learning update in the real MDP.

$$Q_{Real} \;=\; Q^*_{Replay} \;\approx\; Q^*_{Real}$$

$Q^*_{Replay} \approx Q^*_{Real}$ if there are sufficiently many Q updates of all state-action pairs in the MDP, with sufficiently small learning factors α.

Replay process for Q-learning



To perform action a in state s(5):

Transition (with no discount) to most recent performance of a in s;

REPEAT

With probability α replay this performance, else transition with no discount to next most recent performance.

UNTIL a replay is made, or final payoff reached.

Some properties of Q-learning

- Both TD(0) and Q-learning have low computational requirements: are they 'entry-level' associative learning for simple organisms?
- In principle, needs event-memory only for one time-step, but can optimise behaviour for a time-horizon of $\frac{1}{1-\gamma}$
- Constructs no world-model: it samples the world instead.
- Can use replay-memory: a store of past episodes, not ordered in time.
- Off-policy: allows construction of optimal policy while exploring with sub-optimal actions.
- Works better for frequently visited states than for rarely visited states: learning to approach good states may work better than learning to avoid bad states.
- Large-scale implementation possible with a large collection of stored episodes.

What has been achieved?

For *finite* state-spaces and short time horizons, we have:

- solved the problem of preparatory actions
- developed a range of tabular associative learning methods for finding a policy with optimal return
  - Model-based methods based on learning $P(a)$, and several possible modes of calculation.
  - Model-free methods for learning V^*, π^*, and/or Q^* directly from experience.

Computational model of operant reinforcement learning that is more coherent than the previous theory.
General methods of associative learning and control for *small* problems.

The curse of dimensionality

Tabular algorithms feasible only for **very small** problems.

In most practical cases, size of state space is given as number of dimensions, or number of features; the **number of states** is then **exponential** in the number of dimensions/features.

Exact dynamic programming using tables of V or Q values is computationally impractical except for low dimensional problems, or problems with special structure.

A research programme: scaling up

Tables of discrete state values are infeasible for large problems.

Idea: **use supervised learning** to approximate some or all of:

- dynamics (state transitions)
- expected rewards
- policy
- value function
- Q, or the *action advantages* $Q - V$

Use RL, modfiying supervised learning function approximators instead of tables of values.

Some major successes

- Backgammon (TDGammon, by Tesauro, 1995)
- Helicopter manoeuvres (Ng et al, 2006)
- Chess (Knightcap, by Bartlett et al, 2000)
- Multiple arcade games (Mnih et al, 2015)

Also applications in robotics...

Challenges in using function approximation

Standard challenges of non-stationary supervised learning, and then in addition:

1. Formulation of reward function
2. Finding an initial policy
3. Exploration
4. Approximating π, Q, and V
5. Max-norm, stability, and extrapolation
6. Local maxima in policy-space
7. Hierarchy

Finding an initial policy

In a vast state-space, this may be hard! Human demonstration only gives paths, not a policy.

1. supervised learning of initial policy from human instructor
2. Inverse RL and apprenticeship learning (Ng and Russell 2000, Abbeel and Ng, 2004)
   Induce or learn reward functions that reinforce a learning agent for performance similar to that of a human expert.
3. 'Shaping' with a potential function (Ng 1999)

Shaping with a potential function

In a given MDP, what transformations of the reward function will leave the optimal policy unchanged? [1]

Consider a finite horizon MDP. Define a *potential function* Φ over states, with all terminal states having same potential. Define an artificial reward

$$\phi(s, s') = \Phi(s') - \Phi(s)$$

Adjust the MDP so that $\langle s, a, s', r \rangle$ becomes $\langle s, a, s', r + \phi(s, s') \rangle$.

Starting from state s, the same total potential difference is added along all possible paths to a terminal state.
The optimal policy is unchanged.

[1] Ng, Harada, Russell, *Policy invariance under reward transformations*, ICML 1999

Exploration

Only a tiny region of state-space is ever visited; an even small fraction of *paths* are taken, or *policies* attempted.

- Inducing exploration with over-optimistic initial value estimates is totally infeasible.
- Naive exploration with ϵ-greedy or softmax action choice may produce poor results.
- Need an exploration plan

Some environments may enforce sufficient exploration: games with a chance (backgammon), and adversarial games (backgammon, chess) may force agent to visit sufficiently diverse parts of the state space.

Approximating π, Q, and V

P may be a 'natural' function, derived from a physical system.
R specified by the modeller; may be simple function of dynamics.

π, Q, and V are **derived** from P and R by an RL operator that involves maximisation and recursion. *Not* 'natural' functions.

Policy is typically both discontinuous and multi-valued.

Value may be discontinuous, and typically has discontinuous gradient.
Either side of a gradient discontinuity, value is achieved by different strategies, so may be heterogeneous.

Q, or 'advantages' $Q - V$, are typically discontinuous and poorly scaled.

Supervised learning of π, V, Q may be challenging.

Max-norm, stability, and extrapolation

Supervised learning algorithms do not usually have max-norm guarantees.

Distribution of states visited depends sensitively on current policy, which depends sensitively on current estimated V or Q.

Many possibilities for instability.

Estimation of V by local averaging is stable (though possibly not accurate). (Gordon 1995)

Local maxima in 'policy-space'

According to the policy improvement lemma, there are no 'local optima' in policy space.
If a policy is sub-optimal, then there is always some state where the policy action can be improved, according to the value function.

Unfortunately, in a large problem, we may never visit those interesting states where the policy could be improved !

'Locally optimal' policies are all too real....

Hierarchy

Three types of hierarchy:

1. Options (macro-operators).
2. Fixed hierarchies (lose optimality)
3. Feudal hierarchies

How state-spaces become large

1. **Complex dynamics**: even a simple robot arm has 7 degrees of freedom. Any complex system has many more, and each degree of freedom adds a dimension to the state-space.
2. A robot arm also has a high-dimension **action-space**. This complicates modelling Q, and finding the action with maximal Q. Finding $\arg\max_a Q(s, a)$ may be a hard optimisation problem even if Q is known.
3. **Zealous modelling**: in practice, it is usually better to work with a highly simplified state-space than to attempt to include all information that could possibly be relevant.

How state-spaces become large (2)

4. **Belief state**: Even if the state-space is small, the agent may not know what the current state is. The agent's actual state is then properly described as a **probability distribution over possible states**. The set of possible **states of belief** can be large.

5. **Goal state**: suppose we wish the system to achieve any of a number of goals: one way to tackle this is to regard the goal as part of the state, so that the new state space is the cartesian product **state-space** $\times$ **goal-space**. Few or rare transitions between different goals: goal is effectively a parameter of the policy.

6. **Reward state**: even in a small system with simple dynamics, the rewards may depend on the history in a complex way. Expansion of reward state happens when an agent is trying to accomplish complex goals, even in a simple system.

Example: Asymmetric Travelling Salesman Problem

Given: distances $d(i,j)$ for K cities; asymmetric so $d(i,j) \neq d(j,i)$.

To find: a permutation σ of $1:K$ such that
$d(\sigma_1,\sigma_2)+\cdots+d(\sigma_{K-1},\sigma_K)+d(\sigma_K,\sigma_1)$ is minimal.

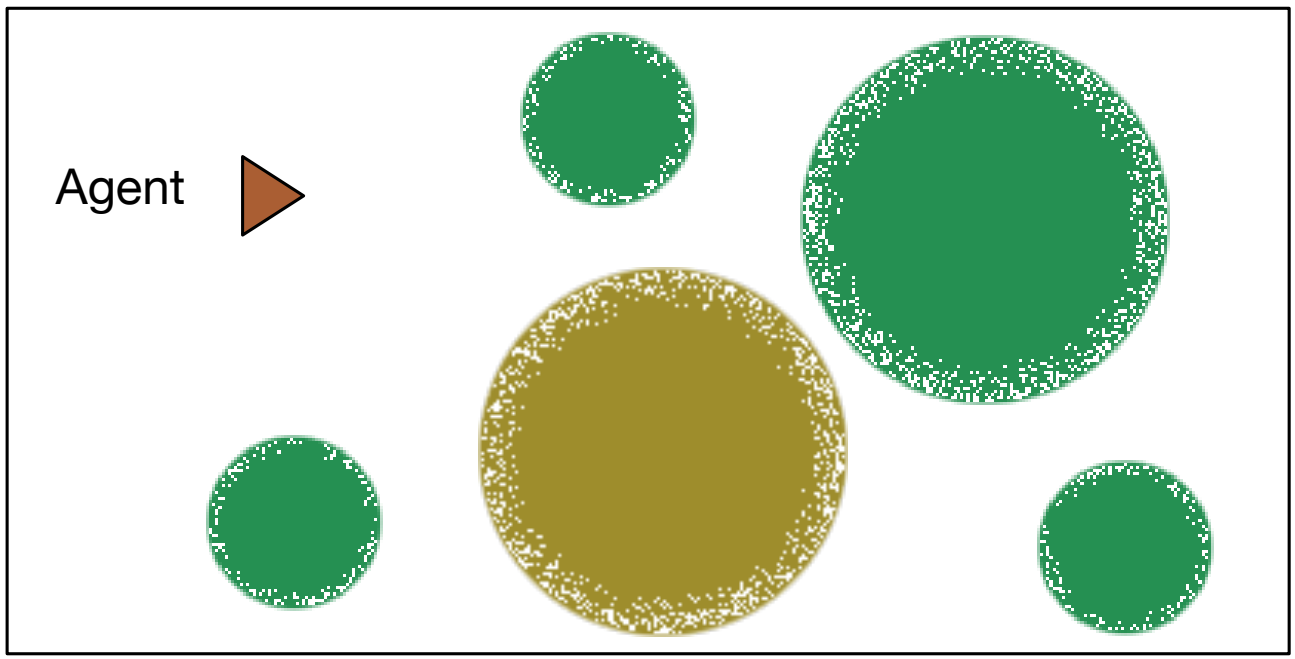
RL formulation as a finite horizon problem:

- w.l.o.g. select city 1 as start state.
- **state** is $\langle$current city, set of cities already visited$\rangle$. Number of states is:
$$N = 1 + (K-1)2^{K-2}$$
- **actions**: In state $\langle i, S\rangle$, agent can move from i to any state not yet visited.
- **rewards**: In moving from i to j, agent receives $d(i,j)$.
In the $K-1$ states where all cities have been visited, and agent is at $j \neq 1$, final payoff is $d(j,1)$.

Although TSP can be formulated as RL, no gain in doing so!

Example: Searching an Area

An agent searches a field for mushrooms: it finds a mushroom only if close to it. What is the state-space?



State includes:

- area already searched: can be a complex shape.
- estimates of mushroom abundance in green and brown areas
- time remaining; level of hunger; distance from home...

Optimisation of Subjective Return?

In RL, the theory we have is for how to optimise expected return from a sequence of immediate rewards.
In some control applications, this is the true aim of the system design: the control costs and payoffs can be adequately expressed as immediate rewards. The RL formalisation then really does describe the problem as it really is.

From point of view of psychology, continual optimisation of a stream of subjective immediate rewards is a strong and implausible theory.
No evidence for this at all !!

A bigger question: where do subjective rewards come from?

Where next?

1. New models: policy optimisation as probabilistic inference, including path integral methods (Kappen, Todorov)
2. ?? New compositional models needed for accumulating knowledge through exploration.
3. Simpler approaches: parametric policy optimisation, cross-entropy method
4. Different models of learning and evolution.