Reinforcement Learning
Lecture 18

Gillian Hayes

7th March 2007

School of informatics

Planning or Model-Based Learning

- Using models of the environment: what is it?
- Procedure for using models
- The DYNA model
- DYNA-Q
- What happens if the model is wrong?

Using Models of the Environment

- Dynamic Programming needs a model of the environment, $P^a_{ss'}$ and $R^a_{ss'}$.
- Monte Carlo and Temporal Difference methods do not.
- Model-based vs. model-free learning
- Planning vs. learning – planning uses a model
- Model: agent can use this to predict what will happen in the environment when it tries an action – which state will it end up in? what reward will it get?
- Stochastic models: given state and action, there are many possible next states and next rewards.

- Distribution models: give full probabilities of all possibilities. DP assumes you have this.

- Sample models: produce just one sample from all the possibilities (sampled according to the underlying probability distribution, of course) cf. blackjack.

- Why use? Simulate environment. Agent can use simulations of what will happen in learning procedure

- If model available can make use of scarce learning experiences

Using Planning/Models

Model $\rightarrow$ simulated experience $\rightarrow_{\text{backups}}$ values $\rightarrow$ policy

Example algorithm: random-sample one-step tabular Q-learning with a model

Do forever:

Select state s and an action a at random
Input s, a to a model and get s', r
Do one-step tabular Q-learning:

$$Q(s,a) = Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$

Converges to the optimal policy for the model (subject to usual conditions)

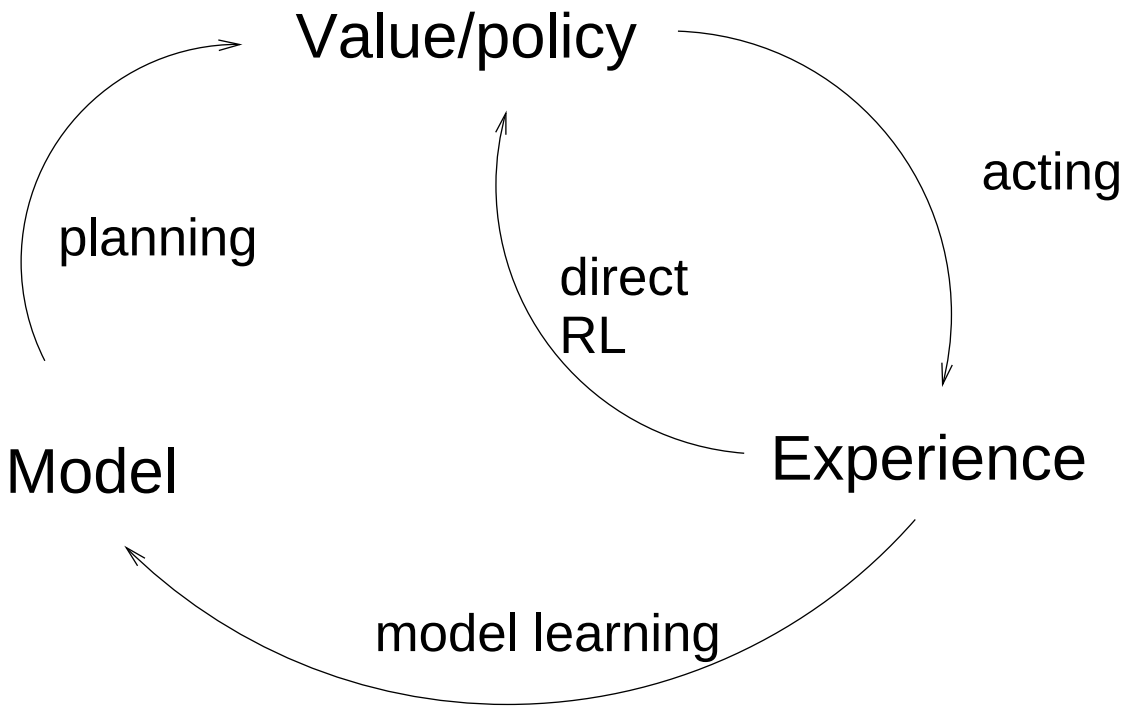
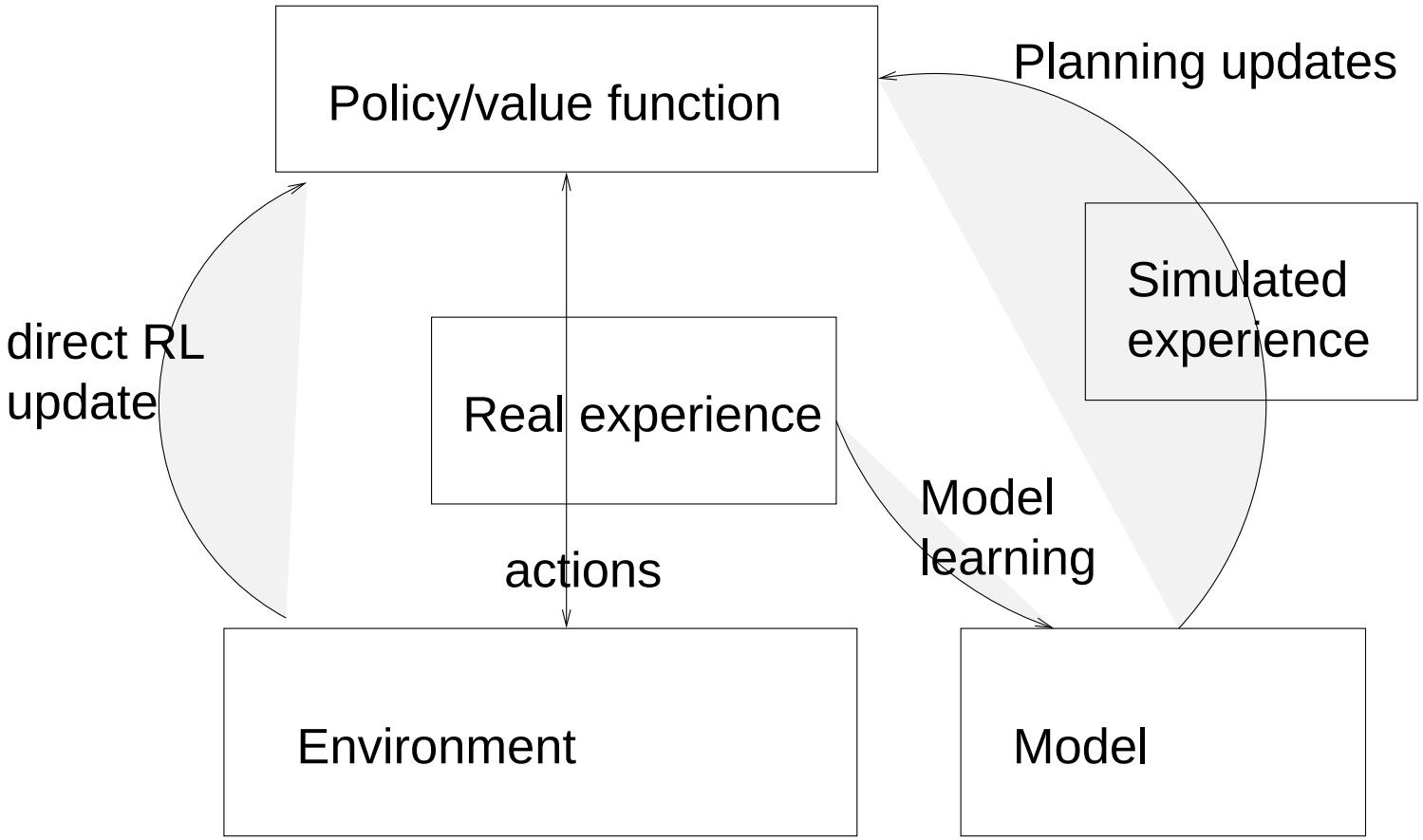


Figure 1: Relationship between learning and planning

DYNA


Policy/value function
Planning updates
Simulated experience
direct RL update
Real experience
Model learning
actions
Environment
Model

- Carry out actions in environment and do regular RL learning to learn value function and policy
- Use real experience to build up model of environment
- Carry out several planning updates, using model to see what would happen if an action is carried out in a state – "hallucinating" – and use this simulated experience to update policy/value function
- Planning, acting and model learning can happen in parallel
- DYNA-Q algorithm

The DYNA-Q Algorithm

Initialise $Q(s,a)$ and $\mathrm{Model}(s,a)$

Loop

s is current state
Choose action a based on ϵ-greedy(s,Q)
Do a, get s' and r (real experience)
$Q(s,a) = Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
Update $\mathrm{Model}(s,a)$ with s', r
Planning (simulated experience): repeat N times:
s is a randomly allocated previously observed state
a is a random action previously carried out in s
Produce simulated experience: $\mathrm{Model}(s,a) \rightarrow s', r$
$Q(s,a) = Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$

Good Points

Example: Dyna maze (see S+B section 9.2)

- Agent can make use of model to do backups: so knowledge based on real reward can be used in simulated experiences to greatly increase speed of learning (tho' more computation is needed).

- Can do planning while, e.g. robot is moving – it takes a long time (on a CPU timescale) to carry out the action so use this time when the processor is not needed to do real-experience RL to do planning.

Wrong Models

- What if the model is wrong?
- Environment could be stochastic. We may not have observed enough real transitions to estimate well the transition probabilities/rewards. Or we used function approximation to learn the model and it is not yet generalising correctly.
- We may not have observed a transition at all – model starts out empty and we can only update value functions/policies for transitions that have been observed.
- Environment may have changed and we may not have observed its new behaviour.

- We will then learn a suboptimal policy. If the model is optimistic, i.e. predicts greater reward than we actually receive, then we'll discover it's wrong – we'll do worse than it predicted and go on to explore elsewhere, thus finding out new behaviour of environment.

Example: Blocking maze (see S+B section 9.3).

Counterexample: See shortcut maze (S+B section 9.3): environment gets **better** and we may not discover that – exploration vs. exploitation. Keep track of how long ago we experienced a **real** s-a-s' transition and give a bonus reward for trying out these transitions: DYNA-Q+.