Learning to Select Actions

Manuela Veloso
CMU 15-381, Fall 2000

- Machine learning: classification
- Control learning
- Learning in deterministic planning
- Reinforcement learning: Q learning

Different Aspects of "Machine Learning"

- Supervised learning
  - ▷ Classification - concept learning
  - ▷ Learning from labeled data
  - ▷ Function approximation
- Unsupervised learning
  - ▷ Data is not labeled
  - ▷ Data needs to be *grouped*, *clustered*
  - ▷ We need distance metric
- Control and action model learning
  - ▷ Learning to select actions efficiently
  - ▷ Feedback: goal achievement, failure, *reward*
  - ▷ Search control learning, reinforcement learning

Search Control Learning

- Improve *search* efficiency, *plan quality*
- Learn *heuristics*

```
(if (and (goal (enjoy weather))
         (goal (save energy))
         (state (weather fair)))
(then (select action RIDE-BIKE)))
```

Learning Opportunities in Planning

- Learning to improve planning efficiency
- Learning the domain model
- Learning to improve plan quality
- Learning a universal plan

Which action model,
which planning algorithm,
which heuristic control
is the most efficient for a given task?

Reinforcement Learning

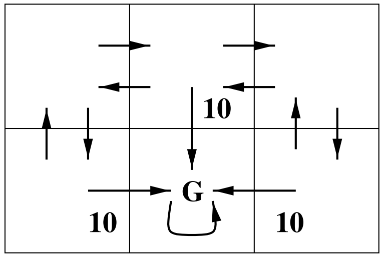
- A variety of algorithms to address:
  - ▷ learning the *model*
  - ▷ converging to the *optimal* plan.

Markov Decision Processes

- Finite set of states, S
- Finite set of actions, A
- Probabilistic state transitions, $\delta(s, a)$
- Reward for each state and action, $R(s, a)$
- Process:
  - ▷ Observe state $s_t \in S$
  - ▷ Choose action $a_t \in A$
  - ▷ Receive immediate reward r_t
  - ▷ State changes to s_{t+1}

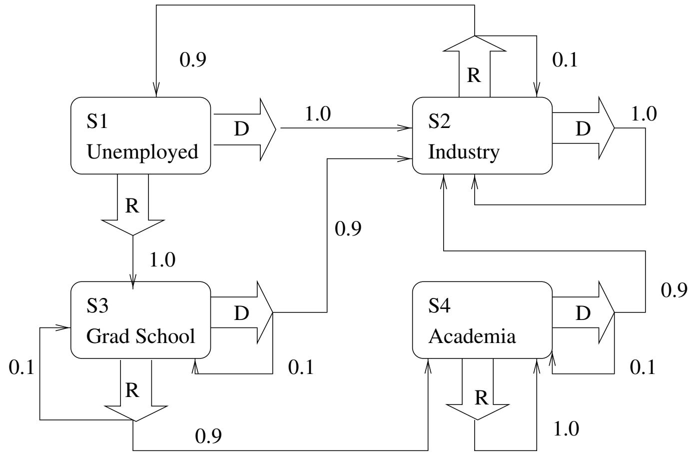
Model: states, actions, probabilistic transitions, rewards

Deterministic Example



(Reward on unlabelled transitions is zero.)

Nondeterministic Example



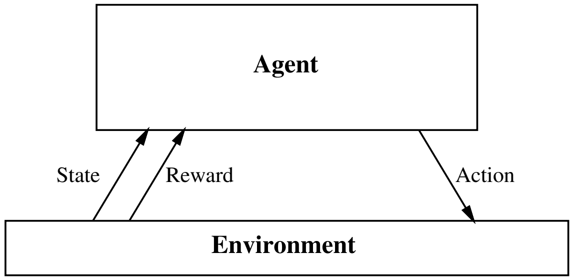
Learning Conditions

- *Markov assumption:*
  New states and rewards are a function only of the current state and action, i.e.,
  - ▷ $s_{t+1} = \delta(s_t, a_t)$
  - ▷ $r_t = r(s_t, a_t)$
- *Unknown and uncertain environment:*
  Functions δ and r may be nondeterministic and are not necessarily known to learner.

Discounted Rewards

- "Reward" today versus future (promised) reward
- \$100K + \$100K + \$100K + ...
  INFINITE!
- Future rewards not worth as much as current.
- Assume reality ...: discount factor , say γ.
- \$100K + γ \$100K + γ^2 \$100K + ...
  CONVERGES!

Reinforcement Learning Problem



$$s_0 \xrightarrow[r_0]{a_0} s_1 \xrightarrow[r_1]{a_1} s_2 \xrightarrow[r_2]{a_2} \ldots$$

Goal: Learn to choose actions that maximize

$r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots$, where $0 \leq \gamma < 1$

Control Learning Task

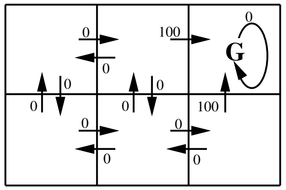
- Execute actions in world,
- Observe state of world,
- Learn action policy $\pi : S \to A$
  - ▷ Maximize expected reward
    $$E[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots]$$
    from any starting state in S.
  - ▷ $0 \leq \gamma < 1$, discount factor for future rewards

Reinforcement Learning

- A variety of successful algorithms
  - ▷ Mitchell's book "Machine Learning": *This lecture is closely based on chapter 13.*
  - ▷ Sutton and Barto's book "Reinforcement Learning"
  - ▷ Kaebling, Moore, Littman: JAIR survey
- If we can do reinforcement learning, then:
  - ▷ outcome: for *every* state, *optimal* action is known
  - ▷ **Universal plan!**

Statement of Learning Problem

- We have a target function to learn
  - ▷ $\pi : S \rightarrow A$
- We have **no** training examples of the form
  - ▷ $\langle s, a \rangle$
- We have training examples of the form
  - ▷ $\langle \langle s, a \rangle, r \rangle$
    (rewards can happily be *any* real number)



immediate reward values $r(s, a)$

Policies

$\boxed{\text{Assume deterministic world}}$

- There are *many possible* policies, of course not necessarily *optimal*, i.e., with maximum expected reward
- There can be also *several OPTIMAL* policies.

Value Function

- For each possible policy π, we can define an *evaluation function over states*

$$V^\pi(s) \equiv r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots$$
$$\equiv \sum_{i=0}^{\infty} \gamma^i r_{t+i}$$

where $r_t, r_{t+1}, \ldots$ are generated by following policy π starting at state s

- Restated learning task: Learn OPTIMAL policy

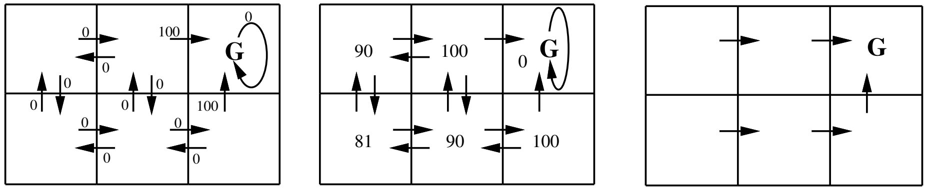
$$\boxed{\pi^* \equiv \mathrm{argmax}_\pi V^\pi(s), (\forall s)}$$

Learn Value Function

We can have agent learn the evaluation function V^{π^*}, which we write as V^*.

Then the agent can select the optimal action from any state s, i.e., have an optimal policy, by using V^* with one step lookahead:

$$\pi^*(s) = \underset{a}{\operatorname{argmax}}[r(s,a) + \gamma V^*(\delta(s,a))]$$



rewards $V^*(s)$ values **ONE** optimal policy

Optimal Value to Optimal Policy

$$\boxed{\pi^*(s) = \operatorname{argmax}_a[r(s,a) + \gamma V^*(\delta(s,a))]}$$

A problem:

- This works well if agent knows $\delta : S \times A \rightarrow S$, and $r : S \times A \rightarrow \Re$
- When it doesn't, it can't choose actions this way

Q Function

- Define new function very similar to V^*

$$Q(s,a) \equiv r(s,a) + \gamma V^*(\delta(s,a))$$

Learn Q function - Q-learning

- If agent learns Q, it can choose optimal action even without knowing δ or r.

$$\pi^*(s) = \underset{a}{\operatorname{argmax}}[r(s,a) + \gamma V^*(\delta(s,a))]$$

$$\pi^*(s) = \underset{a}{\operatorname{argmax}}\, Q(s,a)$$

Q-Learning

Note that Q and V^* are closely related:

$$V^*(s) = \max_{a'} Q(s,a')$$

Which allows us to write Q recursively as

$$\begin{aligned} Q(s_t,a_t) &= r(s_t,a_t) + \gamma V^*(\delta(s_t,a_t))) \\ &= r(s_t,a_t) + \gamma \max_{a'} Q(s_{t+1},a') \end{aligned}$$

Q-learning actively generates examples.
It "processes" examples by updating its Q values.
While learning, Q values are approximations.

Training Rule to Learn Q

Let $\hat{Q}$ denote current approximation to Q.

Then Q-learning uses the following **training rule**:

$$\boxed{\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a')}$$

where s' is the state resulting from applying action a in state s

Q Learning for Deterministic Worlds

For each s, a initialize table entry $\hat{Q}(s,a) \leftarrow 0$

Observe current state s

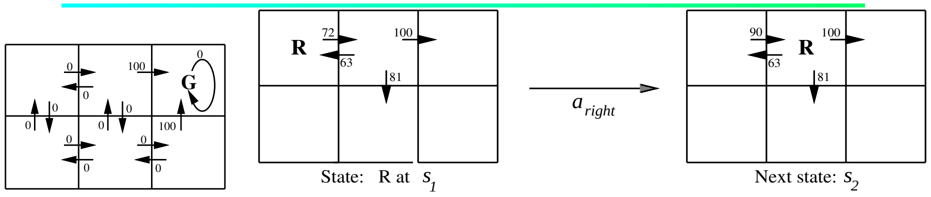
Do forever:

- Select an action a and execute it
- Receive immediate reward r
- Observe the new state s'
- Update the table entry for $\hat{Q}(s,a)$ as follows:

$$\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a')$$

- $s \leftarrow s'$

Example - Updating $\hat{Q}$



$$\begin{aligned}
\hat{Q}(s_1, a_{right}) &\leftarrow r(s_1, a_{right}) + \gamma \max_{a'} \hat{Q}(s_2, a') \\
&\leftarrow 0 + 0.9 \ \max\{63, 81, 100\} \\
&\leftarrow 90
\end{aligned}$$

Notice that if rewards are non-negative, then

$$(\forall s, a, n) \ \ \hat{Q}_{n+1}(s,a) \geq \hat{Q}_n(s,a)$$

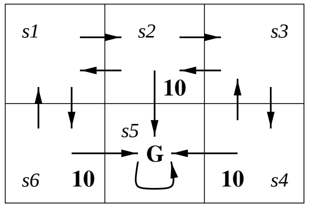
and

$$(\forall s, a, n) \ \ 0 \leq \hat{Q}_n(s,a) \leq Q(s,a)$$

Q Learning Iterations

Starts at bottom left corner - moves clockwise around perimeter; Initially $Q(s,a) = 0$; $\gamma = 0.8$

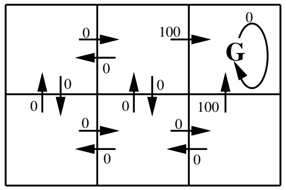
$$\boxed{\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a')}$$

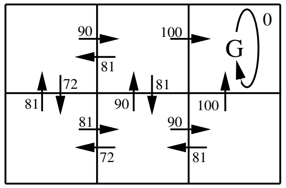


(part of the total Q table - only values that will change in given path)

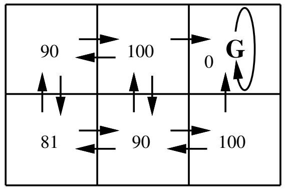
Q(s1,E)	Q(s2,E)	Q(s3,S)	Q(s4,W)
0	0	0	r + γ max{Q(s5,loop)}= 10 + 0.8 . 0 = 10
0	0	r + γ max{Q(s4,W),Q(s4,N)}= 0 + 0.8 max{10,0}= 8	10
0	r + γ max{Q(s3,W),Q(s3,S)}= 0 + 0.8 max{0,8}= 6.4	8	10

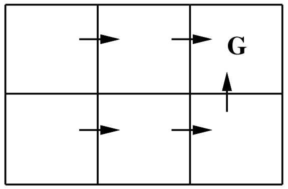
A Simple Deterministic Example of Q-learning





$r(s, a)$ values $\quad$ $Q(s, a)$ values





$V^*(s)$ values $\quad$ One optimal policy

Nondeterministic Case

What if reward and next state are non-deterministic?

We redefine V, Q by taking expected values

$$\begin{aligned} V^\pi(s) &\equiv E[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots] \\ &\equiv E[\sum_{i=0}^{\infty} \gamma^i r_{t+i}] \end{aligned}$$

$$Q(s, a) \equiv E[r(s, a) + \gamma V^*(\delta(s, a))]$$

Nondeterministic Case

Q learning generalizes to nondeterministic worlds

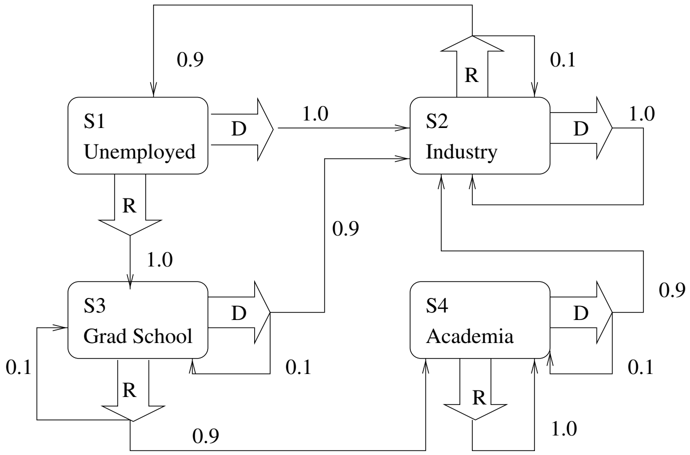
Alter training rule to

$$\begin{aligned} \hat{Q}_n(s, a) \leftarrow \; & (1 - \alpha_n)\hat{Q}_{n-1}(s, a) + \\ & \alpha_n[r + \gamma \max_{a'} \hat{Q}_{n-1}(s', a')], \end{aligned}$$

where $\alpha_n = \frac{1}{1 + visits_n(s,a)}$, and $s' = \delta(s, a)$.

Can still prove convergence of $\hat{Q}$ [Watkins and Dayan, 1992]

Nondeterministic Example



Discussion

- How should the learning agent use the *intermediate* Q values?
- Exploration
- Exploitation
- Scaling up in the size of the state space
  - ▷ Function approximator (neural net instead of table)
  - ▷ Generalization
  - ▷ Reuse, use of macros
  - ▷ Abstraction, learning substructure

Ongoing Research

- Handle case where state only partially observable
- Design optimal exploration strategies
- Extend to continuous action, state
- Learn and use $\hat{\delta} : S \times A \rightarrow S$
- Relationship to dynamic programming
- Multiple learners - Multi-agent reinforcement learning

Summary

- Active learning: no labelled data, action, state changes, reward
- Concept of feedback through delayed reward
- Markov decision processes
- Reinforcement learning
- Q-Learning
- Discussion