Outline

- Goal of Reinforcement Learning
- Mathematical Model (MDP)
- Planning
- Learning
- Current Research issues

Reinforcement Learning

Yishay Mansour

Reinforcement Learning - origins

Artificial Intelligence

Control Theory

Operation Research

Cognitive Science & Psychology

Solid foundations; well established research.

Goal of Reinforcement Learning

Goal oriented learning through interaction

Control of large scale stochastic environments with partial knowledge.

Supervised / Unsupervised Learning
Learn from labeled / unlabeled examples

Contrast with Supervised Learning

The system has a "state".

The algorithm has a great influence on the state.

Tradeoff lower immediate reward for higher future rewards.

GOAL oriented: present versus future rewards

Typical Applications

- Board games (backgammon [T], checkers [S]).
- Elevator control [CB].
- Robo-soccer [SV].
- Dynamic channel allocation [SB].
- Inventory problems.

Mathematical Model - MDP

Markov decision processes

S- set of states

A- set of actions

δ - Transition probability

R - Reward function

Similar to DFA!

Mathematical Model - Motivation

Model of uncertainty:

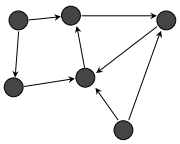
Environment, actions, our knowledge.

Focus on decision making.

Maximize long term reward.

Markov Decision Process (MDP)

MDP model - rewards

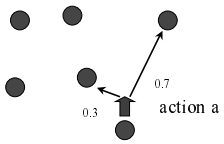


R(s,a) = reward at state ***s***
for doing action ***a***
(a random variable).

Example:
R(s,a) = -1 with probability 0.5
+10 with probability 0.35
+20 with probability 0.15

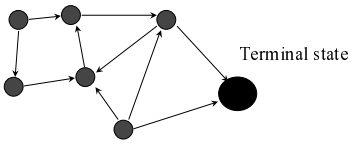
MDP model - states and actions

Environment = states



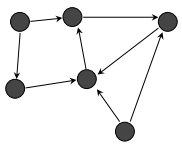
Actions = transitions $\delta(s, a, s')$

MDP model - Episodic tasks



For any policy with probability 1 we reach a terminal state.

MDP model - trajectories



trajectory:

MDP - Return function.

Combining all the immediate rewards to a single value.

Issues:

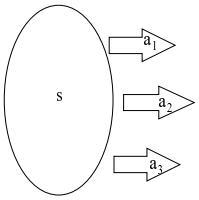
Are early rewards more valuable than later rewards?

Is the system "terminating" or continuous?

Usually the return is linear in the rewards.

Simple example: N- armed bandit

Single state.



Goal: Maximize sum of immediate rewards.

Given the model: Greedy action.

Difficulty: unknown model.

MDP model - action selection

AIM: Maximize the expected return.

Fully Observable - can "see" the "entire" state.

Policy - mapping from states to actions

Optimal policy: optimal from any start state.

THEOREM: There exists a deterministic optimal policy

MDP model - return functions

Finite Horizon - parameter H $return = \sum_{1 \le i \le H} R(s_i, a_i)$

Infinite Horizon

discounted - parameter $\gamma<1$. $return = \sum_{i=0}^{\infty} \gamma^i R(s_i, a_i)$

undiscounted $\frac{1}{N}\sum_{i=0}^{N-1} R(s_i, a_i) \xrightarrow{N \to \infty} return$

Episodic:

Total reward $return = \sum_{i=1}^{t} R(s_i, a_i)$

MDP model - summary

$s \in S$ - set of states, $|S|=n$.

$a \in A$ - set of k actions, $|A|=k$.

$\delta(s_1, a, s_2)$ - transition function.

R(s,a) - immediate reward function.

$\pi : S \to A$ - policy.

$\sum_{i=0}^{\infty} \gamma^i r_i$ - discounted cumulative return.

Contrast with Supervised Learning

Supervised Learning:
Fixed distribution on examples.

Reinforcement Learning:

The distribution is policy dependent!!!

A small local change in the policy can make a huge global change in the return.

Planning - Value Functions

$V^{\pi}(s)$ The expected value starting at state s following π.

$Q^{\pi}(s,a)$ The expected value starting at state s with action a and then following π.

$V^{*}(s)$ and $Q^{*}(s,a)$ are define using an optimal policy π^{*}.

$V^{*}(s) = max_{\pi} V^{\pi}(s)$

Planning - Basic Problems.

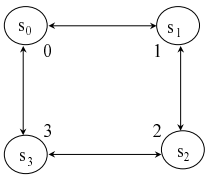
Given a complete model.

Policy evaluation - Given a policy π, estimate its return.

Optimal control - Find an optimal policy π^{*} (maximizes the return from any start state).

Algorithms - Policy Evaluation
Example

A={+1,-1}
$\gamma = 1/2$
$\delta(s_i,a) = s_{i+a}$
π random
$\forall a: R(s_i,a) = i$



$V^{\pi}(s_0) = 0 + \gamma\,(\pi(s0,+1)V^{\pi}(s_1) + \pi(s0,-1)\,V^{\pi}(s_3))$

Planning - Policy Evaluation

Discounted infinite horizon (Bellman Eq.)

$$V^{\pi}(s) = E\,[\,R(s,\pi(s)) + \gamma\, V^{\pi}(s')\,]$$

Rewrite the expectation

$$V^{\pi}(s) = E[R(s,\pi(s))] + \gamma \sum_{s'} \delta(s,\pi(s),s')V^{\pi}(s')$$

Linear system of equations.

Algorithms - optimal control

State-Action Value function:

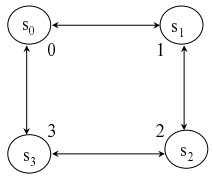
$$Q^{\pi}(s,a) = E\,[\,R(s,a) + \gamma V^{\pi}(s')\,]$$

Note $V^{\pi}(s) = Q^{\pi}(s,\pi(s))$

π is deterministic.

Algorithms -Policy Evaluation
Example

A={+1,-1}
$\gamma = 1/2$
$\delta(s_i,a) = s_{i+a}$
π random
$\forall a: R(s_i,a) = i$



$V^{\pi}(s_0) = 5/3$
$V^{\pi}(s_1) = 7/3$
$V^{\pi}(s_2) = 11/3$
$V^{\pi}(s_3) = 13/3$

$V^{\pi}(s_0) = 0 + (V^{\pi}(s_1) + V^{\pi}(s_3))/4$

Algorithms - optimal control

CLAIM: A policy π is optimal if and only if at each state *s*:

$V^{\pi}(s) = MAX_a \{Q^{\pi}(s,a)\}$ **(Bellman Eq.)**

PROOF: Assume there is a state *s* and action *a* s.t.,

$V^{\pi}(s) < Q^{\pi}(s,a)$.

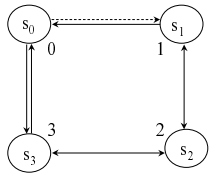
Then the strategy of performing *a* at state *s* (the first time) is better than π.

This is true each time we visit *s*, so the policy that performs action *a* at state *s* is better than π. □

Algorithms -Optimal control
Example

A={+1,-1}
γ = 1/2
δ(s_i,a)= s_{i+a}
π random

R(s_i,a) = i



$Q^{\pi}(s_0,+1) = 5/6$
$Q^{\pi}(s_0,-1) = 13/6$

$Q^{\pi}(s_0,+1) = 0 + \gamma V^{\pi}(s_1)$

Algorithms - optimal control

The greedy policy with respect to $Q^{\pi}(s,a)$ is

$\pi(s) = \text{argmax}_a\{Q^{\pi}(s,a)\}$

The ε-greedy policy with respect to $Q^{\pi}(s,a)$ is

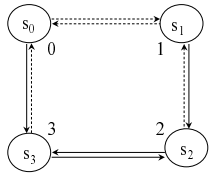
$\pi(s) = \text{argmax}_a\{Q^{\pi}(s,a)\}$ with probability 1-ε, and

$\pi(s) = a$ with probability $\varepsilon/|A|$

Algorithms -optimal control
Example

A={+1,-1}
γ = 1/2
δ(s_i,a)= s_{i+a}
π random

R(s_i,a) = i



Changing the policy using the state-action value function.

Algorithms - Linear Programming

MAXIMIZE $\Sigma_{s\varepsilon S} V(s)$, such that

$\forall a \varepsilon A \; s \varepsilon S$:

$V(s) \geq E[R(s,a)] + \gamma \Sigma_{s' \varepsilon S} \delta(s,a,s') V(s')$

Variables: |S| variables
Inequalities: |S| |A|

Algorithms - computing optimal policy

1. Linear Programming.
2. Value Iteration.
3. Policy Iteration.

Algorithms - Policy Iterations

$Q_i := Q^{\pi_i}(s,a)$
$\pi_i(s) := \text{argmax}_{a \in A} \{Q_{i-1}(s,a)\}$

Convergence:
1. Number of iterations less than Value Iterations.
2. Never considers the same policy twice, $|A|^{|S|}$.

RECENT: number of iterations is $O(|A|^{|S|}/|S|)$ [MS, UAI99].

Algorithms - Value Iterations

$V_0(s) := 0$
$V_{t+1}(s) := MAX_{a \in A} \{ E[R(s,a)] + \gamma \Sigma_{s' \in S} \delta(s,a,s') V_t(s') \} \quad \forall s$

Let $d_t = MAX_{s \in S} \{|V_t(s) - V^*(s)|\}$

Claim: $d_t \leq \gamma d_{t+1}$ □

Convergence: After t iterations we have error γ^t.

Learning Algorithms

Given access only to actions perform:
1. policy evaluation.
2. control - find optimal policy.

Two approaches:
1. Model based (Dynamic Programming).
2. Model free (Temporal Differences and Monte Carlo).

Algorithms - Open problems

Is there a strongly polynomial algorithm to compute an optimal policy.

Policy iterations:
1. Non-trivial lower bounds.
2. Better upper bounds.

Relationships to "new" shortest path algorithms (for episodic MDP).

Learning - Model Based

Sample size (optimal policy):

Naive: $O(|S|^2 |A|)$ samples.
(approximates each transition $\delta(s,a,s')$ well.)

Better: $O(|S| |A| \log (|S| |A|))$ samples.
(Sufficient to approximate optimal policy.)
[KS, NIPS'98]

Learning - Model Based

Estimate the model from the observation.
(Both transition probability and rewards.)

Use the estimated model as the true model, and find optimal policy (using dynamic programming).

If we have a "good" estimated model, we should have a "good" estimation.

Monte Carlo - optimal control.

- Start with some policy π_0.
- Use π_i to generate a trajectory t.
- For each (s,a) and trajectory t, let $R'(s,a)$ be the return of the suffix starting with (s,a).
- $Q(s,a)$ = average $R'(s,a)$.
- Let π_i be the ε–greedy policy of $Q(s,a)$.

PROBLEM: $R'(s,a)$ includes returns of various policies.

CONVERGENCE:
If the algorithm converges is converges to the optimal policy.

Open Problem: Does it converge?

Learning - Model free
Monte Carlo - Policy Evaluation

Given a policy π :

Generate m trajectories following π.

For each state s let $T(s)$ include all the suffixes of trajectories starting at s.

Let $\hat{V}^{\pi}(s)$ be the average return of $T(s)$.

$$|V^{\pi}(s) - \hat{V}^{\pi}(s)| \approx 1/\sqrt{|T(s)|}$$

TD(0) - Optimal Control

Learn online the Q function.

$$Q_{t+1}(s_t,a_t) = Q_t(s_t,a_t) + \alpha[r_t + \gamma Q_t(s_{t+1},a_{t+1}) - Q_t(s_t,a_t)]$$

a_{t+1} the ε-greedy policy for Q_t.

CONVERGENCE: GUARANTEED [S,JJS]

sarsa - algorithm

Learning - Model Free
Temporal Differences -Policy evaluation

An online view:
Assume that at state s_t we performed action a_t, received reward r_t and moved to state s_{t+1}.

Our "estimation error" is $r_t + \gamma V(s_{t+1}) - V(s_t)$, we update:

$$V_{t+1}(s_{t+1}) = V_t(s_t) + \alpha[r_t + \gamma V_t(s_{t+1}) - V_t(s_t)]$$

Note that for the correct value function we have:

$$E[r + \gamma V(s') - V(s)] = 0$$

Learning - Open Problems

A systematic understanding of the online updates.

Sample complexity bounds for TD and MC.

TD(λ) - understand better the tradeoff in λ.

Comparing TD and MC

Monte Carlo update: $R_t - V(s)$, where R_t is the return from s.

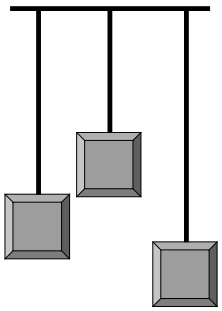
Temporal Differences update: $r_t + \gamma V(s') - V(s)$

Common Belief in RL: TD is superior to MC.

CHALLENGE: Give a formal justification.

TD(λ) - a family of algorithms between MC and TD.

Example - Elevator Control



Learning (alone):
Model the arrival model well.

Planning (alone) :
Given arrival model build schedule

Real objective: Construct a schedule while updating model

Planning versus Learning

Tightly coupled in Reinforcement Learning

Like to maximize return while learning.

Large Scale MDP

Previous Methods: Tabular. (small state space).

Large scale $\Leftrightarrow$ Exponential size.

Similar to COLT view of learning.

Current Research Efforts

- Large scale systems
- Partially Observable MDP

Large scale MDP - Restricted Value Function

(Value) Function Approximation:
Use a limited class of functions to estimate the value function.

Given a good approximation of the value function, we can estimate the optimal policy.

Vague Idea: reduce to a supervised learning.

applications: most of the work (TD-gammon, etc.)

Large scale MDP

Approaches:

1. Restricted Value function.
2. Restricted policy class.
3. Restricted model of MDP.
4. (new) use a different MDP representation.

Function Approximation - basics

$\boxed{Q(s,a) = f_w(s,a)}$ f_w: a parametric function class

Adjusting w using gradient descent:

$$w := w + \alpha [r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \nabla_w f_w(s,a)$$

sampled value; estimated value; gradient

TD-gammon

Linear Function Approximation
policy evaluation

Monte Carlo: converges to minimal MSE

$\| Q'_{MC} - Q^\pi \|_\pi = \min_w \| Q'_w - Q^\pi \|_\pi \div \varepsilon$

TD(0) converges close to MSE

$\| Q'_{TD} - Q^\pi \|_\pi = \varepsilon / (1-\gamma)$ [TV]

DP may diverge

There exists counter examples [B,TV]

Function Approximation - Linear

For every (s,a) there is a feature vector $\rho(s,a)$

Let $Q'(s,a) = f_w(s,a) = < \rho(s,a) , w >$

Gradient-descent:

$\nabla_w f_w(s,a) = \rho(s,a)$

$$w := w + \alpha [r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \rho(s,a)$$

Function Approximation - Conclusion

Algorithms do not always carry from tabular to "function approximation".

Most results use L_2 norm.
(leads to linear function approximation)

Linear Function Approximation
optimal control

Q-Learning with LFA may diverge. [B,G]

Sarsa with LFA converges [NR,S].

Monte Carlo with LFA converges.

Large scale MDP - Restricted models

Assume specific type of MDP:

1. decomposable MDPs [BDH].

2. Sparse representation (Bayes Net) [KP].

CHALLENGE: Find realistic models with tractable evaluation and control

Large scale MDP - Restricted policies

Assume specific class for the policy representation.

Actor-Critic Model

New results about general classes and connection to VC theory [KMN99,M99].

Large Scale MDPs- research

Sample complexity.

Function approximation not under L_2 (L_1 or L_∞)

Other algorithms using the generative model [MS].

Approximation algorithms.

Large Scale MDP - Generative Model.

Clearly we can not use matrix representation.

Generator representation:

Algorithm for estimating optimal policy (for discounted infinite horizon) in time independent of the number of states [KMN, IJCAI99].

POMDP - Belief State Algorithm

Given a history of actions and observable value we compute a posterior distribution for the state we are in (belief state).

The belief-state MDP:

States: distribution over S (states of the POMDP).
actions: as in the POMDP.
Transition: the posterior distribution (given the observation)

We can perform the planning and learning on the belief-state MDP.

Partially Observable MDP

Rather than observing the state we observe some function of the state.

Ob - Observable function.
a random variable for each states.

Example: (1) Ob(s) = s+noise. (2) Ob(s) = first bit of s.

Problem: different states may "look" similar.

The optimal strategy might need to consider the history.

Summary

Markov Decision Process :
- Model.
- Planning Algorithms.
- Learning Algorithms.

Advanced Topics:
- Large scale MDP.
- POMDP.

Interesting open problems !

POMDP - Hard problems.

Computing an infinite (polynomial) horizon undiscounted optimal strategy for a deterministic POMDP is P-space-hard (NP-complete) [PT,L].

Computing an infinite (polynomial) horizon undiscounted optimal strategy for a stochastic POMDP is EXPTIME-hard (P-space-complete) [PT,L].

Computing an infinite (polynomial) horizon undiscounted optimal policy for an MDP is P-complete [PT].

Exploration vs. Exploitation.

Resources

- Reinforcement Learning (an introduction) [Sutton & Barto]
- Markov Decision Processes [Puterman]
- Dynamic Programming and Optimal Control [Bertsekas]
- Neuro-Dynamic Programming [Bertsekas & Tsitsiklis]
- Ph. D. thesis - Michael Littman