Machine Learning

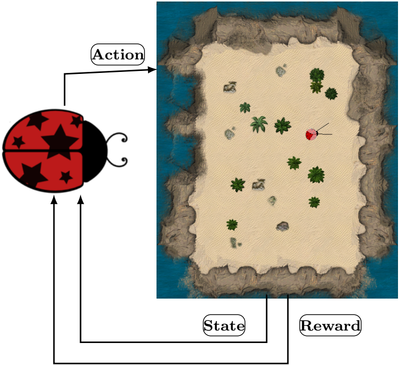
CSE204

Jesse Read

Reinforcement Learning

Introduction: Reinforcement Learning

- No supervisor or labels, only reward signal
- Reward signal is delayed, sparse, and/or weak
- No i.i.d. 'dataset', but rather an environment
- Learning algorithm as part of an agent, mapping observations to actions



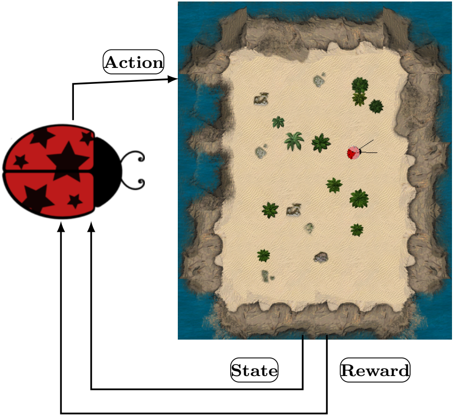
Applications

- Games (GO, Backgammon, Atari games, StarCraft, ...)
- Auto-piloting vehicles, robots, ...
- Manage supply and demand (products, electricity)
- Traffic control
- Trading/manage investment portfolio
- Auto-tune the parameters of a deep neural network
- Farming and Agriculture
- Bidding / advertising
- And many others (Health, Sports, Politics, ...)

▸ AIGym ▸ Walker1 ▸ Walker2 ▸ Robot ▸ Cars ▸ Helicopter
▸ Starcraft-Simple ▸ Starcraft-AlphaStar ▸ Quake Arena ▸ Energy ▸ Politics

The number of real-world applications of reinforcement learning is increasing rapidly.

The Agent and the Environment



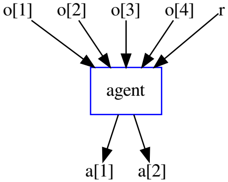
1. Observe the [state of] the environment
2. Perform some action
3. Obtain reward; and repeat

Goal: take actions to maximize future rewards.

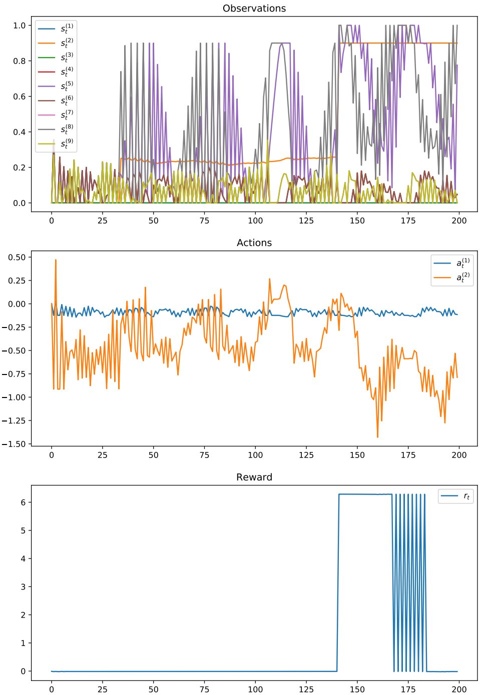
Formulating a Reinforcement Learning Problem

Main considerations:

- What is the state space / what is observed
- What actions are available
- What is the reward function/signal



The agent design (observations, actions, rewards) involves design choices under the constraints of the environment.


Observations
$s_t^{(1)}$
$s_t^{(2)}$
$s_t^{(3)}$
$s_t^{(4)}$
$s_t^{(5)}$
$s_t^{(6)}$
$s_t^{(7)}$
$s_t^{(8)}$
$s_t^{(9)}$
Actions
$a_t^{(1)}$
$a_t^{(2)}$
Reward
r_t

The Reward Signal

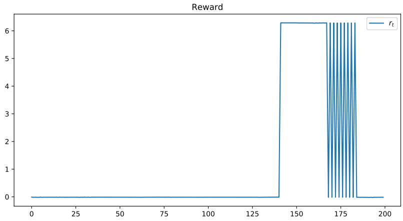
We observe state s_t and obtain reward r_t, and take action a_t at time t. Over an episode, we see

$$s_1, a_1, r_1, \quad s_2, a_2, r_2, \quad \ldots \quad , s_T, a_T, r_T$$

Problems:

- Reward may be delayed
- Reward signal may be sparse, and/or weak

i.e., temporal credit assignment problem



Fully Observed Environments

If fully observed, we see the whole state[1]. Thus

- $\mathcal{S}$ state space (e.g., $\mathcal{S} = \{A, B, C, D, E\}$)
- $\mathcal{A}$ action space (e.g., $\mathcal{A} = \{1, 2\}$)
- $\mathcal{R}$ reward space (e.g., $\mathcal{R} = \mathbb{R}$)

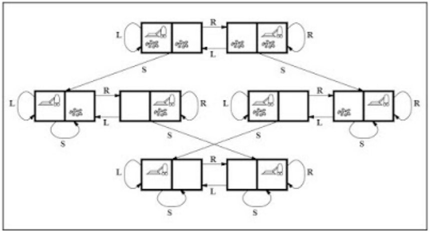
[1] In the partially observed case, we have only an observation of a state

Fully Observed Environments

If fully observed, we see the whole state[1]. Thus

- $\mathcal{S}$ state space (e.g., $\mathcal{S} = \{A, B, C, D, E\}$)
- $\mathcal{A}$ action space (e.g., $\mathcal{A} = \{1, 2\}$)
- $\mathcal{R}$ reward space (e.g., $\mathcal{R} = \mathbb{R}$)

For example, Vaccuum Cleaner World; There are 8 possible states, 3 possible actions (L,R,S), reward 1 iff cleaned (at end state):



Question: Describe a possible episode in this environment.

[1] In the partially observed case, we have only an observation of a state

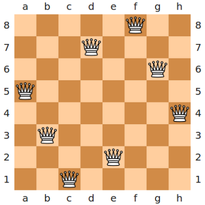
Deterministic Environments

In deterministic environments, given an action the next state is known for certain.

For example,

- Noughts & crosses (tic tac toe)
- Chess
- 8-puzzle

×		○
	○	×
×		○



1	3	5
7	2	4
8		6

For example, the `8-puzzle`,

1	3	5
7	2	4
8		6

$\Rightarrow$

1	2	3
4	5	6
7	8	

It's just a search (each node is a state):

1. Generate the search tree to goal state
2. Apply payoff to leaf
3. Backup
4. Choose the best branch

For example, the `8-puzzle`,

1	3	5
7	2	4
8		6

$\Rightarrow$

1	2	3
4	5	6
7	8	

It's just a search (each node is a state):

1. Generate the search tree to goal state
2. Apply payoff to leaf
3. Backup
4. Choose the best branch

But prohibitive if search space is huge/long route to goal (reward)!

Stochastic Environments

In stochastic environments, the agent takes action a from state s and the next state is $s_{t+1} \sim P(\cdot|s_t = s, a_t = a)$.

For example, in the `Frozen Lake` environment, the ice is slippery and the agent does not always move in the intended direction.

S			
	H		H
			H
H			G

*S*tart, *G*oal and *H*ole in the ice

Not a search path from start to goal!

A Markov Decision Process / Model of the Environment

Most reinforcement learning problems can be framed as an MDP:

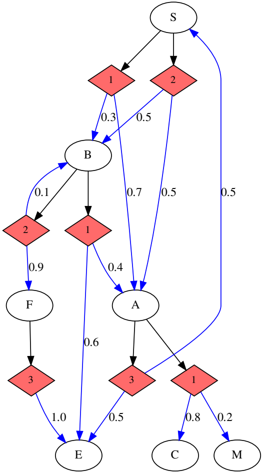
- $\mathcal{S}$ state space (e.g., {S,B,F,A,E,C,M})
- $\mathcal{A}$ action space (e.g., {1,2})
- $\mathcal{P}(s'|s,a)$ transition function
- $\mathcal{R}(s,a,s')$ reward function

where $\mathcal{P}$ embodies the Markov property.

We do not necessarily know this model.

How to approach the problem?

We want a policy, of which action is best to take from a given state.



$\mathcal{P}$ shown on edges;
$\mathcal{R}(\mathtt{A},\mathtt{1},\mathtt{M}) = 1$ and 0 elsewhere

Markov Property

MDPs provide a framework for modeling sequential decision making in stochastic environments.

MDP = Markov Chain + One-step Decision Theory

Markov property: The effects of action a_t from state s_t depend only on those values;

$$P(s_{t+1}|s_t, a_t) = P(s_{t+1}|s_1, \ldots, s_t, a_1, \ldots, a_t)$$

If $P(s_{t+1}|s_t, a_t) = P(s_{t+1}|s_t)$ we reduce to a Markov chain.

The Policy

A policy

$$\pi : \mathcal{S} \mapsto \mathcal{A}$$

indicates the action to take in a given state/under the current observation; i.e., the agent observes s_t and takes action $a_t = \pi(s_t)$.

Thus, a policy defines the behaviour of the agent.

The Policy

A policy

$$\pi : \mathcal{S} \mapsto \mathcal{A}$$

indicates the action to take in a given state/under the current observation; i.e., the agent observes s_t and takes action $a_t = \pi(s_t)$.

Thus, a policy defines the behaviour of the agent.
Which policy is best?

The Policy

A policy

$$\pi : \mathcal{S} \mapsto \mathcal{A}$$

indicates the action to take in a given state/under the current observation; i.e., the agent observes s_t and takes action $a_t = \pi(s_t)$.

Thus, a policy defines the behaviour of the agent.
Which policy is best?

The best policy should take the *best* action from the current state.

$$a_t^* = \pi(s_t)$$

i.e., action a_t to to optimize ...

The Policy

A policy

$$\pi : \mathcal{S} \mapsto \mathcal{A}$$

indicates the action to take in a given state/under the current observation; i.e., the agent observes s_t and takes action $a_t = \pi(s_t)$.

Thus, a policy defines the behaviour of the agent.
Which policy is best?

The best policy should take the *best* action from the current state.

$$a_t^* = \pi(s_t)$$

i.e., action a_t to to optimize . . . the reward r_t? *No*, consider –

- what if the reward is only given at the end (r_T)
- what if a_t has nothing to do with r_{t+1}

We need a payoff function / evaluation metric!

The Return/Gain (Finite Scenario)

The return (aka gain) (at step t) is thus

$$G_t = \sum_{i=t}^{T} r_i$$
$$= r_t + r_{t+1} + r_{t+2} + \ldots + r_T$$

i.e., the sum of rewards until the end of the episode.

The return indicates the value of the current state.

It is our 'loss function'; an important consideration ...

- Should it be that $r_{t+1} = r_T$ (as above)?
  (or should closer rewards be more important?)
- What is T? Perhaps $T = \infty$?
  (agent may not survive until then)

The Return (Infinite Scenario)

The return (at step t) is thus

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{k+t+1} \tag{1}$$

$$= r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots \tag{2}$$

for discount factor $\gamma \in (0, 1)$ which indicates the relative value of closer rewards.

The Return (Infinite Scenario)

The return (at step t) is thus

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{k+t+1} \tag{1}$$

$$= r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots \tag{2}$$

for discount factor $\gamma \in (0, 1)$ which indicates the relative value of closer rewards.

Issue: G_t is from the future (and may include up to r_∞)!
Instead, consider the expected return $\mathbb{E}[G_t]$.

The Value Function

The value function (or state-value function),

$$V^{\pi}(s) = \mathbb{E}[G_t | s_t = s] \tag{3}$$

maps a state to a value.

The value of a state s is the expected return from that state following policy π.

We may think of a vector of $|\mathcal{S}|$ values;
e.g., if $\mathcal{S} = \{s_1, s_2, s_3, s_4\}$:

	s_1	s_2	s_3	s_4
V				

The policy is implicit: move to states with high value.

The Action-Value Function

The action-value function,

$$Q^{\pi}(s, a) = \mathbb{E}[G_t | s_t = s, a_t = a]$$

maps a state *and action* to a value.

We may think of a table of $|\mathcal{S}| \times |\mathcal{A}|$ values;
e.g., with $\mathcal{A} = \{a_1, a_2\}$:

Q	a_1	a_2
s_1		
s_2		
s_3		
s_4		

The greedy policy is explicit:

$$a_t = \pi(s_t) = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\, Q(s_t, a)$$

Given Q (or V) we can construct an optimal policy using dynamic programming; solved!

What is the problem? We don't usually have Q or the MDP (observed environment) which can produce it!

That's where reinforcement learning comes in: Estimate Q (hence, e.g., Q-Learning!)

Exploration vs Exploitation

We need to learn the system dynamics through interaction.

One possibility: Use a behaviour policy (possibly random) μ:

$$a_t \sim \mu$$

(this is an off-policy method).

1. Play many episodes with policy μ; record G_t for each (s_t, a_t)
2. Use these samples to approximate the expectation:

$$Q^{\mu}(s, a) = \frac{1}{n} \sum_{i=1}^{n} G_t^{(i)} \approx \mathbb{E}[G_t | s, a]$$

3. Employ the greedy policy π on Q.

Trade-off: Exploration vs exploitation.

Another Way: Stochastic Optimization

We can perform a search directly in policy space.

Imagine:

- Neural network as π.
- Try many $\pi_1, \ldots, \pi_M$ in the environment
- Find the one that performs best

You won't find this covered extensively in reinforcement learning books, because it's already covered in stochastic optimization, including evolutionary methods.

Can be "*embarassingly effective*" for a small number of parameters (< 100), especially for continuous state spaces, in particular domains such as robotics; but often severely limited on larger problems.

Types of Reinforcement Learning Agents

Value-Based or Policy-Based ?

- Value Based (value function; policy is implicit)
- Policy Based (no value function)
- Actor Critic (both value function + policy)

Model Free vs Model Based: We can build a model of the environment, or do without it.

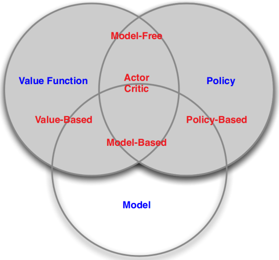


Image from [1]

Summary: Important Concepts

- Agent and Environment
- States, Actions, Reward
- Model of the environment (MDP)
  - Markov property
  - Deterministic vs stochastic
  - Unknown vs fully observable
- Return/Gain (finite horizon vs infinite horizon; γ)
- Policy
- State Value and Action-Value functions
- Exploration vs Exploitation trade-off

Reinforcement learning is difficult (compared to supervised learning), but can be very useful – there are many applications where supervised learning does not apply.

Machine Learning
CSE204

Jesse Read

Reinforcement Learning