Ch 1: Introduction

Reinforcement Learning

- learn what to do to maximize a numerical reward signal
+ helps map situation → action
- situation outcomes and subsequent rewards can stack
- formalize the problem using language from dynamical systems theory
+ incompletely known Markov Decision processes
* sense state of environment, take actions that affect state, have a goal
- supervised learning not good enough, must be able to learn from new interactions to adapt
- unsupervised learning - finding structure in unlabeled data
+ goal is confused in this case
- challenges of RL: exploration vs exploitation
+ to get reward, prefer what has already gotten it the reward: exploit
+ but to discover said action, or better ones, must explore
* both required for optimal outcomes, but stochastically challenging to represent
- RL focuses on whole problem immediately w/o concern for subproblems
+ thus a complete, interactive, goal-seeking agent balances solutions to subproblems with their desire for the main goal
- RL affects many engineering disciplines, influenced psychology and neuroscience

Examples

- adaptive controller adjusting parameters of a petroleum refinery in real-time
+ optimizes yield / cost / quality without sticking to set points, allowing nuanced control in unfamiliar scenarios
- gazelle, once born, struggles to walk. 30 min later, can take off at 20 mph

Elements of RL

- 4 main subelements: a policy, a reward signal, a value function, and a model of env.
+ policy - how to behave given a perceived state
* can be a lookup table or extensive search function
* stochastic policies specify probabilities for each action
+ reward signal - goal of RL problem, to maximize in the long run
* analogous to pleasure / pain
* stochastic functions of the state of env and actions taken
+ value function - what's good in the long run vs reward (what's good immediately)
* associated with a state's intrinsic desirability
* far-sighted judgment
* without rewards there are no values, but it is value that brings about the most success
* efficiently estimating value is the most important component of RL
+ model of env - inferences about how env will behave
* Agents can be modeled or model-free
* modeled engage in planning

Limitations And Scope

- heavy reliance on concept of state
+ don't concern ourselves too heavily with the construction, changing, or learning of state
- book focuses on learning methods with a value function, not evolutionary techniques
+ e.g. genetic programming, simulated annealing, etc.
* never understand states implicitly, learn by passing good performers into next generation, requiring many trials with env
+ we want interaction and evaluation, take advantage of info evolutionary methods throw away
* e.g. state links, structured feedback, more efficient search

Example: Tic-Tac-Toe

How to solve?

X	O	O
O	X	X
		X

3 X's, win state = 1

3 O's or draw, loss state = 0

- "min max" assumes how a player may play, won't work
- classical optimization technique require complete specification of opp.
- evolutionary methods would search space, searching for states of high win probability, leading to finding best next state policy
+ or genetically maintain population of high performing policies
- Value function approach:

1) value function is a lookup table of each state's estimated value
2) all unknown state win % are 0.5, known win = 1, known loss = 0

+ play many games now, moving greedily for the most value
* on occasion we move exploratorily to see if higher value exists elsewhere
+ value of previous state is updated to be closer to the state it led too
* $V(S_t) = V(S_t) + \alpha[V(S_{t+1}) - V(S_t)]$

  S_t - state before greedy move

  S_{t+1} - after move

  $V(x)$ - value of state

  α - step-size param influences rate of learning
* known as temporal-difference learning

- function does very well, producing optimal state probabilities if α converges to zero over time
+ if not, does well against player that changes strategy over time
- evolutionary method, in this instance, gives credit to "all states" it accessed when winning, even if sub-optimal
- Tic-tac-toe has a finite set of states
+ Gerry Tesauro combined algo with neural net to play backgammon, with 10^{20} states
* can only ever experience a fraction of all states
* NN helps generalize experience between similar states
- a-priori info can be injected for more efficient learning
- we used a model for state lookahead, but model-free is possible too
+ remove complexity based on model resolution

Summary

- interactions with the environment serious step
- use Markov decision process to formalize interaction
+ cause & effect, uncertainty / non determinism, existence of explicit goals
- value functions for efficient search of policy space

Early History of RL skipped
refer to textbook for info