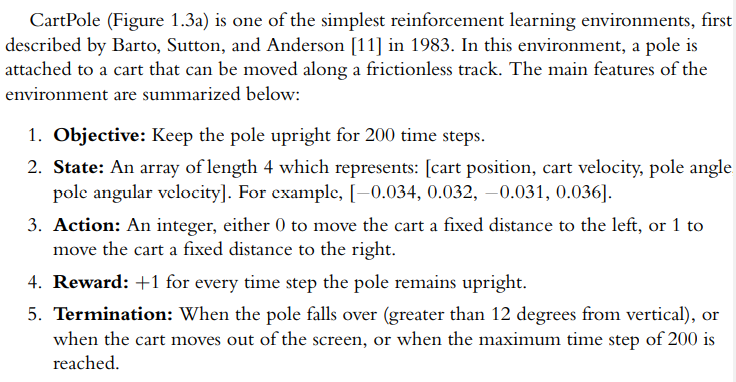
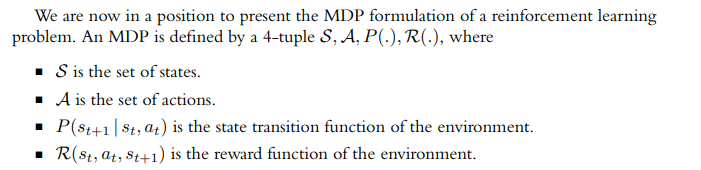
**Intro to RL**

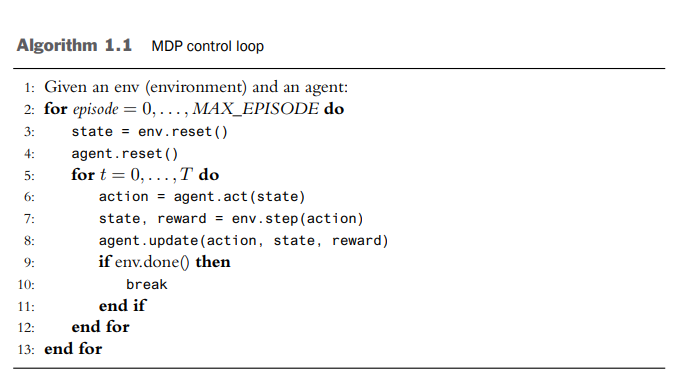
**1/ RL**

* RL is concerned with solving sequential decision-making problems
* The (st, at, rt) tuple is called an experience.
* The time horizon from t = 0 to when the environment terminates is called an episode.
* An agent needs many episodes to learn a good policy ranging from hundred to million



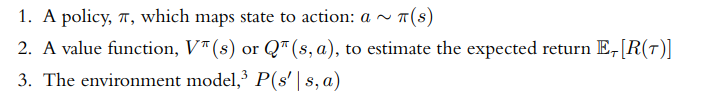
**2/ RL as MDP = Markov Decision Process**

* The consideration of how the environment transition from 1 state to the next using what is known as the transition function.
* In RL, transition function is formulated as MDP
* 
* Markov property implies that the current state and action at time step t contain sufficient info to fully determine the transition probability fr the next state at t+1
* 
* The agents do not have access to the transition function or reward function. The only way an agent can get info about these function is thru the state, actions, and reward is thru experiences in the environment



* This algorithm expresses the interactions between an agent and env over many episodes and time steps.
* At the beginning of each episode, the env and agent are reset.
* On reset, the environment produces an initial state. Then the agent and environment begin to interact meaning that the agent produce action (line 6) and then env produce next state and reward (line 7)

**3/ Learnable Functions In RL**

* With RL formulate as MDP, what should the agent learn? **We see that the agent can learn an action producing function known as polic**y. However, there are other properties of environment that can be useful to agent. Specifically there are 3 primary function to learn in RL
* 
* V function measures the expected return from being in state s, assuming the agent continues to act according to its current policy pi
* Q function evauate how good or bad a state-action pair is
* The transition function P(s 0 | s, a) provides information about the environment. If an agent learns this function, it is able to predict the next state s 0 that the environment will transition into after taking action a in state s. By applying the learned transition function, an agent can “imagine” the consequences of its actions without actually touching the environment. It can then use this information to plan good actions.

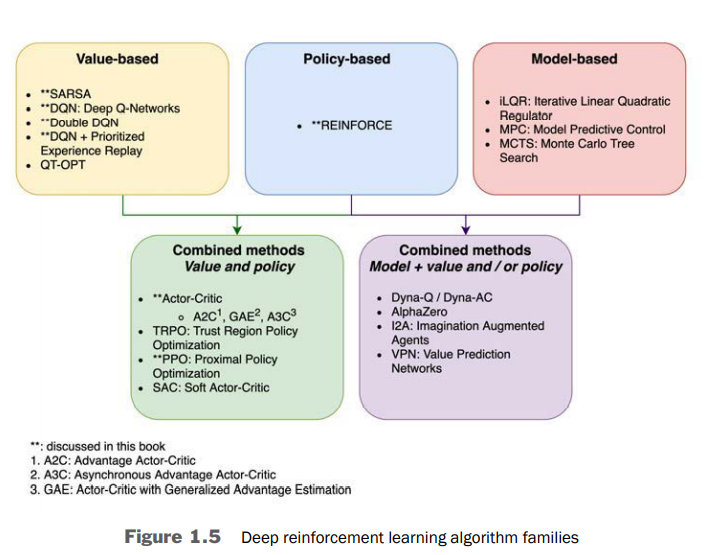
**4/ DRL Algorithms**

* In RL, an agent leans function to help it act and maximize the objective.
* DRL – use deep neural networks as function approximation method
* There are 3 RL algorithms

+ policy-based – learn policy

+ value-based – learn the value function

+ model-based – learn about models.



a/ Policy-based Algorithms – learn to optimized mapping of STATE & ACTION:

* Objective: learn the policy π.
* Good policies should generate action wich produce trajectories that maximize the agent’s objective
* This is intuitive – if an agent needs to act in environment, it make sense to learn a policy.
* **What constitutes a good action at a given moment depends on the state, so a policy function π takes a state s as input to produce an action a ∼ π(s)**
* Advantage:

+ optimization methods meaning this method can apply with any type of actions – discrete, continuous or mix – mltiaction. Also the agent care most about the objective

+ this class of methods isptg31266351 1.4 Deep Reinforcement Learning Algorithms 13 guaranteed to converge to a locally4 optimal policy, as proven by Sutton et al. with the Policy Gradient Theorem [133].

* Disadvantage:

+ Methods have high variance and sample-inefficient

b/ Value-based Algorithms – evaluate the cost of state & action and pick the highest value

* An agent learns either V π (s) or Qπ (s, a).
* It uses the learned value function to evaluate (s,a) pairs and generate a policy.
* **For example, an agent’s policy could be to always select action a in state s with the highest estimated Q(s,a)**
* Q function is more common than V since it is easier to convert Q to policy
* SARSA is not commonly used due to high variance and sample inefficiency during training
* DQN, DDQN, DQN with PER (Prioritized Experience Replay)
* Value-based algo are more sample-efficient than policy-based algo since they have lower variance and make better use of data gather from the environment. However, there are no guarantees that these algorithms will converge to an optimum.

+ most advance – QT-OPT – applied to env with continuous space

c/ Model-based Algorithms

* Algo learn a model of an environment’s transition dynamics make use of a known dynamic model.
* Once an agent has a model of the environment P(s 0 | s, a), it can imagine what will happen in the future by predicting the trajectory for a few time step.
* **If the environment is in state s, an agent can estimate how the state will change if it makes a sequence of actions a1, a2, . . . , an by repeatedly applying P(s 0 | s, a), all without actually producing an action to change the environment.**
* Purely model-based is commonly applied to games with target state

+ Monte Carlo Tree Search

+ iLQR = iterative Linear Quadratic Regulators

+ MPC – model predictive control involve learning the transition dynamics, often under quite restrictive assumption

* Model-based algo is very appealing since a perfect model endows an agent with foresight-it can play out scenarios and understand the consequences of its action without having to actually act in an environment
* Advantage: save time in not having to work on scenarios in robotics. Also require less data to learn good policies since the model enables an agent to supplement its actual experience with imagined ones
* Disadvantages: environment is large state space and action space which is difficult to model => Hard to track. Also, the models are useful when they can accurately predict the transition of the environment many steps into the futures
* This distinction between model-based and model-free
* Model-based = algo that make use of the transition dynamics of an environment whether learned or known in advance.
* Model-free = algo that don’t explicitly make use of environment transition dynamics

d/ Combined Methods:

* A2C = the policy acts and the value function critiques the action

+ The key idea is that during training, a learned value function can provide a more informative feedback signal to a policy than the sequence of rewards available from the environment.

+ The policy learns using info provided by the learned value function

+ The policy is then used to generate actions, as in policy-based methods

* Actor-Critic algorithms are an active area of research and there have been many interesting developments in recent years—Trust Region Policy Optimization (TRPO) [122], Proximal Policy Optimization (PPO) [124], Deep Deterministic Policy Gradients (DDPG) [81], and Soft Actor-Critic (SAC) [47], to name a few. Of these, PPO is currently the most widely used; we discuss it in Chapter 7.
* Algorithms may also use a model of the environment transition dynamics in combination with a value function and/or a policy. In 2016, researchers from DeepMind developed AlphaGo, which combined MCTS with learning V π and a policy π to master the game of Go [125]. Dyna-Q [130] is another well-known algorithm which iteratively learns a model using real data from the environment, then uses the imagined data generated by a learned model to learn the Q-function.

e/ Algorithms Covered in this book:

This book focuses on methods that are policy-based, value-based, and a combination of the two. We cover REINFORCE (Chapter 2), SARSA (Chapter 3), DQN (Chapter 4) and its extensions (Chapter 5), Actor-Critic (Chapter 6), and PPO (Chapter 7**). Chapter 8 covers parallelization methods that are applicable to all of them.**

**We do not cover model-based algorithms. This book aims to be a practical guide; model-free methods are more well developed and more applicable to a wider range of problems because of their generality.**

f/ On-Policy & Off-Policy Algo:

* An Algorithm is considered to be on-policy if it learned on policy – that is, training can only utilize data generated from the current policy pi. This implies that as training iterates thru version of policies, each training iteration only uses the current policy at that time to generate training data. As a result, all data must be discarded after training, since it becomes unusable.
* . The on-policy methods discussed in this book are REINFORCE (Chapter 2), SARSA (Chapter 3), and the combined methods Actor-Critic (Chapter 6) and PPO (Chapter 7).

**5/ RL & Supervised Learning:**

* At the core of deep RL is function approximation – shared with SL.
* Three main difference between SL and RL

+ Lack of an oracle

+ Sparsity of feedback

+ Data generate during training

a/ Lack of an Oracle:

* For each RL problems, the correct answer for each model nput is not available