MODELING

Value function
Q-factor
Optimality equation

Diego Klabjan
Professor, Industrial Engineering and Management Sciences

Northwestern | McCORMICK SCHOOL OF ENGINEERING

Outline

- Transition function
- Policies in depth
- Value function
- Q-factor
- Various forms of optimality equations
- Model-free vs model-based

TRANSITION FUNCTION AND POLICIES

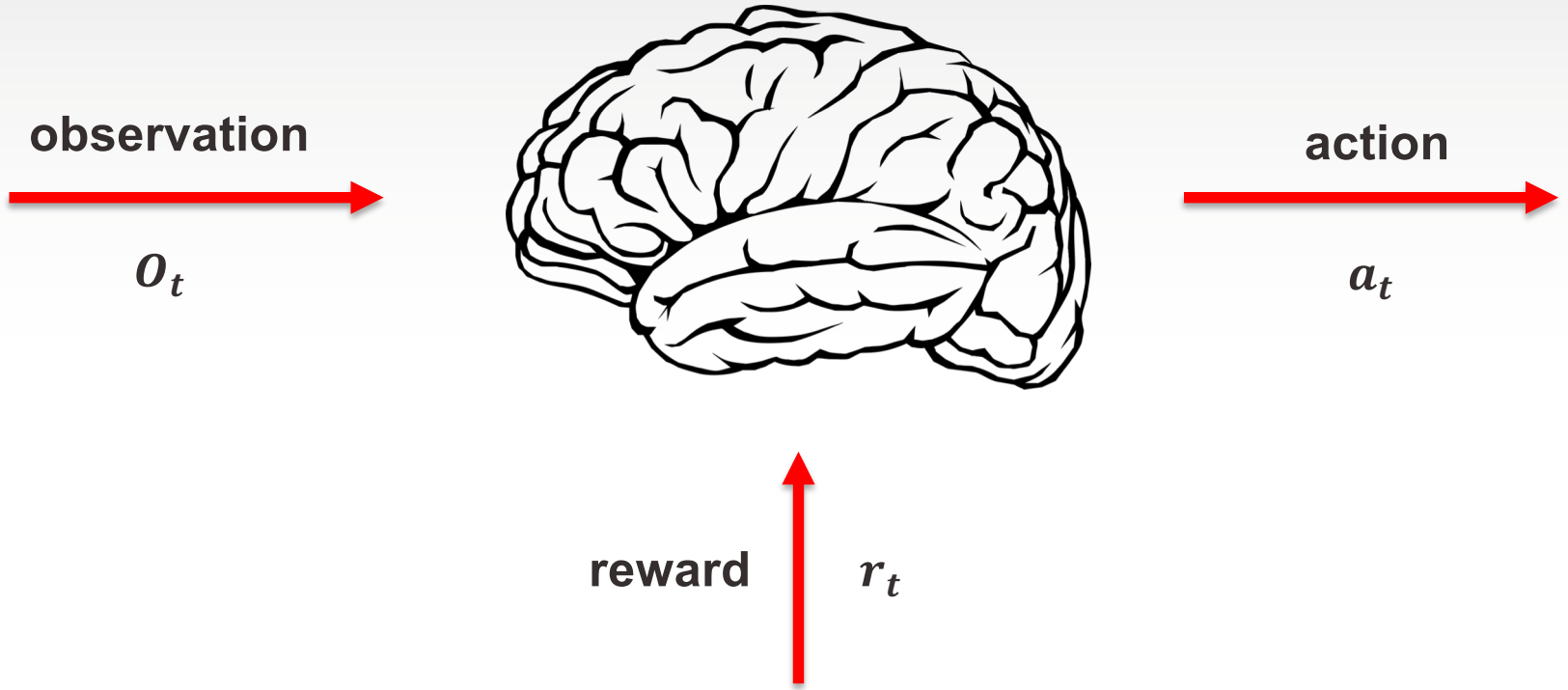
Characteristics of Reinforcement Learning

- What makes reinforcement learning different from other machine learning paradigms?
  - There is no supervisor
    - Reward signal
  - Feedback can be delayed delayed, not instantaneous
    - Lead time after placing an order
  - Time really matters
    - Sequential non i.i.d data
  - Agent's actions affect the subsequent data it receives
    - Customer demand depends on inventory

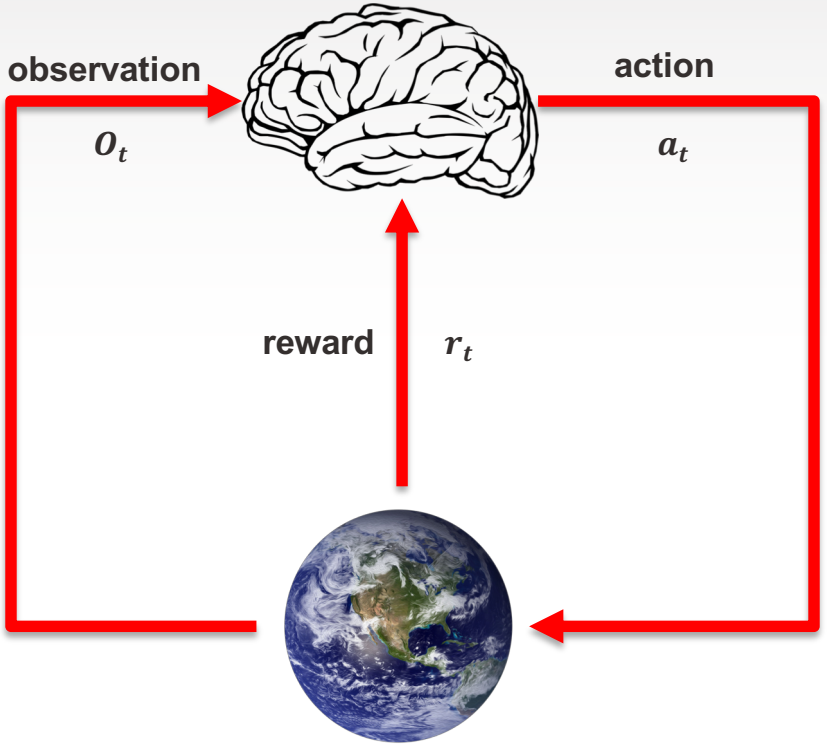
Sequential Decision Making

- Goal
  - Select actions to maximize total future reward
- Actions may have long term consequences
- It may be better to sacrifice immediate reward to gain long-term reward
  - Myopic actions not optimal
- Examples
  - A financial investment
    - May take months to yield P&L
  - Guiding a drone
    - Should prevent a crash happening in several minutes
  - Games
    - Only at the end you know the reward

Agent and Environment



Agent and Environment



- At each step t the agent
  - Executes action a_t
  - Receives observation O_t
  - Receives scalar reward r_t
- The environment
  - Executes action a_t
  - Receives observation O_{t+1}
- t increments

Agent and Environment

- Drone navigation
  - Action = one-hot encoding of direction to follow
  - Observation = position in the space, distance and shapes to obstacles
  - Reward combination of
    - Distance to final destination
    - Penalty for crashing
- Environment = drone
  - Execute action (move drone)
  - Gather next observation

- Portfolio management
  - Action = Number of units to buy/sell of each security in portfolio
  - Observation
    - Current positions
      - Number of units
      - Cash on hand
  - Reward
    - Profit and loss upon sell
- Environment = broker
  - Execute trades
  - Record new portfolio status

History and State

- The history is the sequence of observations, actions, rewards

$$H_t = O_0, r_0, a_0, \ldots, a_{t-1}, O_t, r_t$$

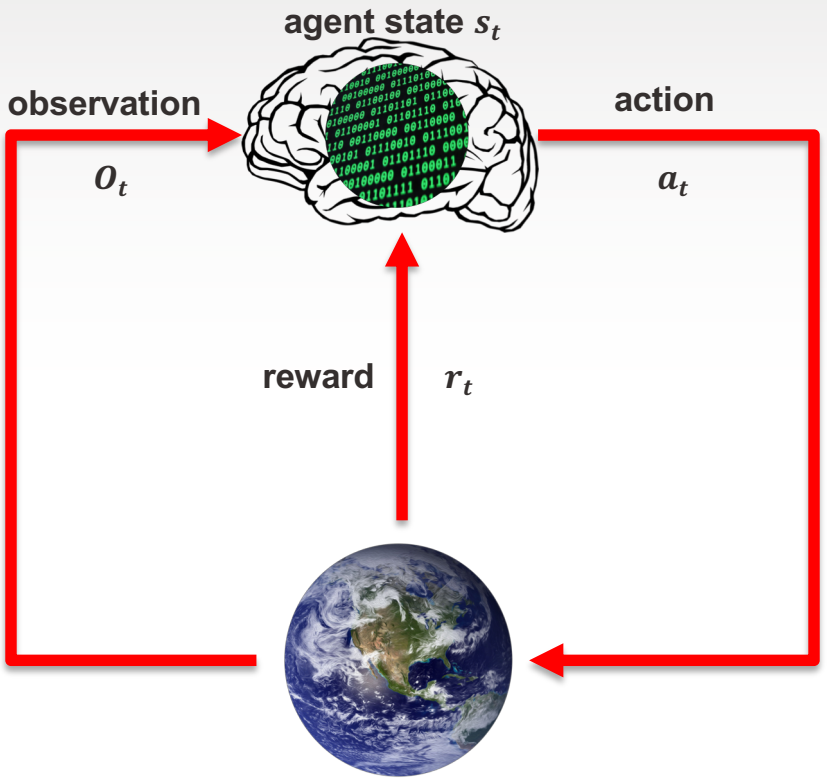
- What happens next depends on the history
  - The agent selects actions
  - The environment selects observations
- State is the information used to determine what happens next
- State a function of the history

$$s_t = f(H_t)$$

History and State

- Why do we need previous actions?
  - Action executed in the past but consequences in the future
- Why do we need previous observations (exogenous unpredicted)?
  - In reward we predict P&L
  - Forecasting methodology depends on previous realizations
- Why do we need previous rewards?
  - Reward might depend on them
  - For tax purposes
- Why state a function of history?
  - Encoding sequences with RNN
  - Encoding images with CNN

Agent State



- Agent state s_t is the agent's internal representation
  - Whatever information the agent uses to pick the next action
  - Or/and to calculate reward
- Information used by reinforcement learning algorithms

Rewards

- Reward r_t is a scalar feedback signal
- Indicates how well agent is doing at step t
- The agent's job is to maximize cumulative reward
- All goals described by the maximization of expected cumulative reward

Examples of Rewards

- P&L in portfolio optimization
- Profit in inventory management
- Drone: positive for following a trajectory, negative for crashing
- Robot walking: positive to move forward, negative to fall
- Positive for winning a game, negative for losing the game
- Playing basketball: positive for scoring, negative for opponent scoring
- Shortest path: distance of traversed arc

State

- Information state contains all useful information from the history
  - Markov state
- State s_t is Markov if and only if

$$\mathbb{P}[s_{t+1} \mid s_t] = \mathbb{P}[s_{t+1} \mid s_0, \ldots, s_t]$$

  - "The future is independent of the past given the present"
- The history H_t is Markov
  - The state is a sufficient statistic of the future

State Space

- State space $\mathcal{S}$
  - Set of all possible states
- Discrete or continuous
  - Usually discrete and finite
  - Continuous
    - Discretize
- Challenge
  - In practice extremely high cardinality

- Inventory management: 0, 1, 2, 3, …
  - 3 SKU's
    - $\mathbb{Z} \; x \; \mathbb{Z} \; x \; \mathbb{Z}$
- Drone: All possible x,y,z coordinates
  - Continuous but discretized in practice
- Tic-tac-toe: $\{-1,0,1\}^9$

Actions

- $\mathcal{A}$ – action space
- Usually discrete and finite
  - If continuous, discretize
- $a_t \in \mathcal{A}$
- Goal of RL: find a_t for each time t
- Depends on state
  - $a_t = a_t(s_t)$

- Inventory
  - $\mathcal{A}$ = {0,1,2,…,M}
- Drone
  - $\mathcal{A}$ = {stay, up, down, left, right, …}
  - 27 options
- Tic-tac-toe
  - $\mathcal{A}$ = {1,2,…,9}

Markov Chains

- s_{t+1} depends on s_t
- Key to identify $p(s_{t+1}|s_t)$
- Inventory
  - $p(s_{t+1} = 10|s_t = 5)???$
  - Ordered 5, demand = 0
  - Ordered 8, demand = 3
  - If we know action, we can calculate the transition probability

$$p(s_{t+1}|s_t, a_t)$$

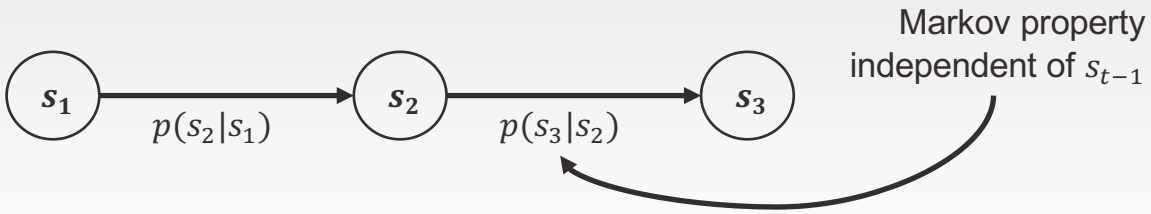
  - $p(s_{t+1} = 10|s_t = 5, a_t = 8) = p(D = 3)$

- Markov chains *P*
- Transition matrix in RL
  - P^a
- Every action function gives a transition function

Transitions

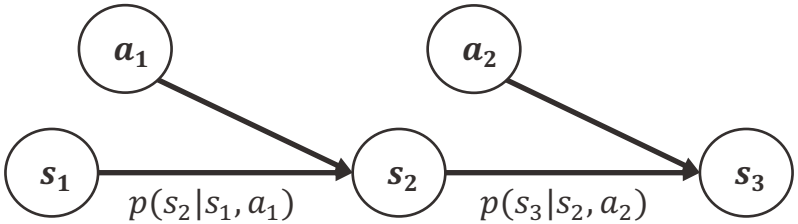
$$P_{ij} = p(s_{t+1} = i | s_t = j)$$

matrix



$$P_{ij}^{k} = P_{ijk} = p(s_{t+1} = i | s_t = j, a_t = k)$$

tensor



Policy

- Permissible family of action functions
  - Denoted by π
  - Policy again function of state
- Restrict action functions
  - Computational tractability
  - Business and operational requirements
    - Order only 0 or Q
- Policy usually parametric π_θ
  - Goal to find best set of parameters

$\pi_\theta(s_t) = \theta \cdot s_t$

Better: neural network

Stochastic Policies

- Flip a coin based on certain probability to select an action
- Find probability distribution over all possible actions
- Drone
  - 27 dimensional probability vector
  - Neural networks
    - Input state vector
    - Output 27 dimensional probability
      - Obtained through softmax
- Continuous actions
  - Normal distribution with mean and standard deviations depending on state

Stochastic Policies

- Deterministic/pure policy special stochastic policy
- Stochastic policies more general
- Algorithmically much better to handle
  - Continuous in parameters
  - Differentiable in parametes
- Can be mathematically shown
  - Given a stochastic policy there is a deterministic policy of the same reward
- On the paper stochastic policies do not yield anything new

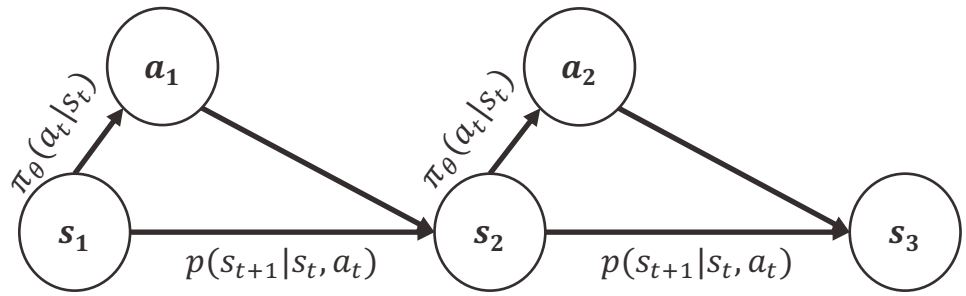
Objective Function

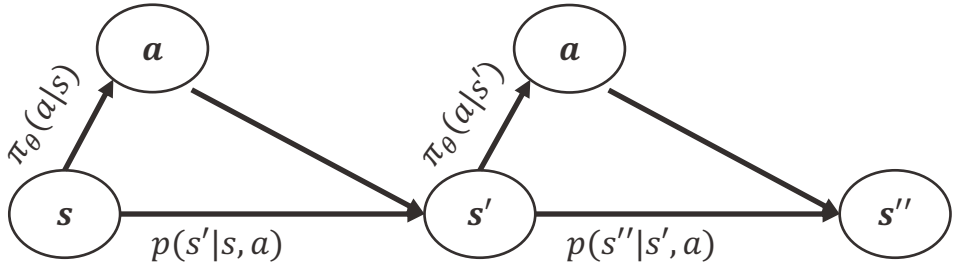
- Find policy that maximizes total reward
- $\max_{\pi} E[\sum_{t=0}^{T} \gamma^t \cdot r(s_t, a_t)]$
- Looking for a function
- Future reward is discounted
  - Future money is discounted
- $0 \leq \gamma < 1$
  - Well defined also for $T = \infty$

$$s_0, \pi(s_o), r_0, s_1, \pi(s_1), r_1, s_2, \pi(s_2), r_2, s_3, \pi(s_3), r_3, \ldots$$

Infinite case

- Drop t
  - Everything else the same
- s, a, $r(s,a)$, and $p(s'|s,a)$ define Markov decision process
  - Infinite version of Markov chains with actions





Objective Function

- $\tau = s_0, a_0, s_1, a_1, s_2, a_2, s_3, a_3, \ldots$
  - Trajectory, sample path, episode

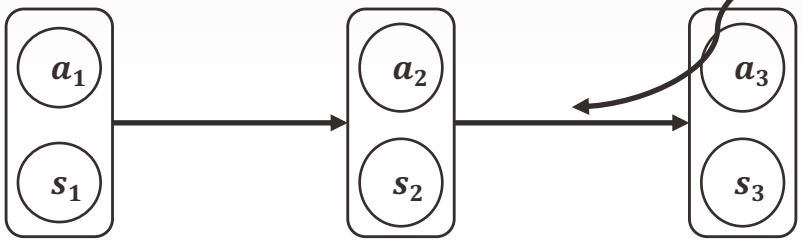
$$p_\theta(\tau) = p_\theta(s_0, a_0, \ldots, s_T, a_T) = p(s_0) \prod_{t=0}^{T} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t)$$

  - Policy/actions parametric with respect to θ
- Inventory
  - P(10,3; 5,7; 1,0; 0,10) = p(state = 10) π_θ(3|10) p(D=8) π_θ(7|5) p(D=11) π_θ(1|0) p(D=1) π_θ(10|0)

Objective Function

- (state, action) representation

$$p_\theta((s_{t+1}, a_{t+1})|(s_t, a_t)) = p(s_{t+1}|s_t, a_t)\pi_\theta(a_{t+1}|s_{t+1})$$



$$p_\theta((s_{t+1}, a_{t+1})|(s_t, a_t)) = p(s_{t+1}|s_t, a_t)\pi_\theta(a_{t+1}|s_{t+1})$$

- Markov chain with expended states
  - State = (state, action)

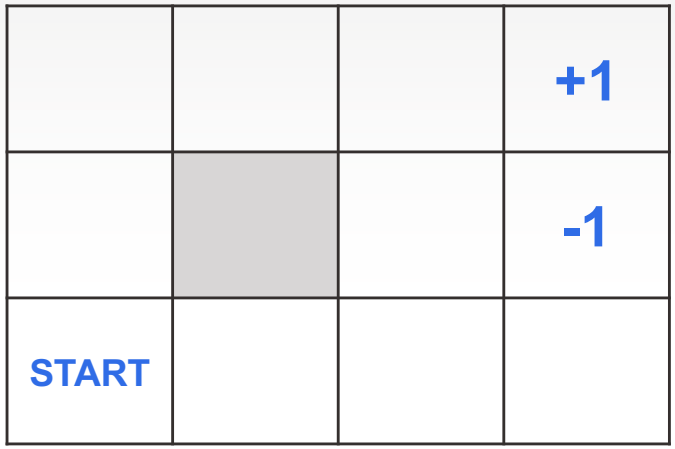
Objective Function

$$\theta^* = \arg\max_\theta E_{\mathcal{T}\sim p_\theta(\mathcal{T})}\left[\sum_{t=0} r(s_t, a_t)\right]$$

$$= \arg\max_\theta \sum_{t=0}^{T} E_{(s_t,a_t)\sim p_\theta(s_t,a_t)}[r(s_t, a_t)]$$

- Each sample path has probability
  - Distribution over sample paths
- Expectation over all sample paths

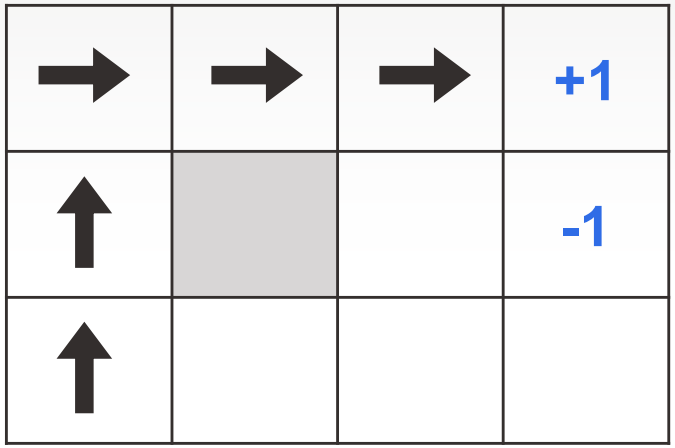
Robot in a Room



actions: UP, DOWN, LEFT, RIGHT

- Reward +1 at [4,3], -1 at [4,2]
- Reward -0.04 for each step
  - Reward does not depend on action and state

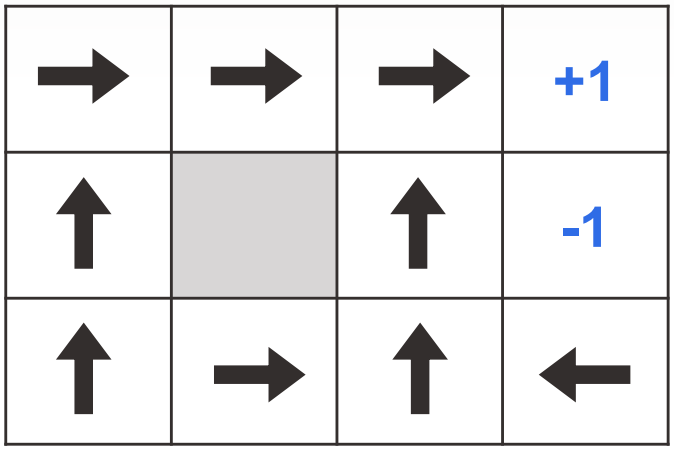
Total Reward



- Total reward of 5 * 0.04 + 1

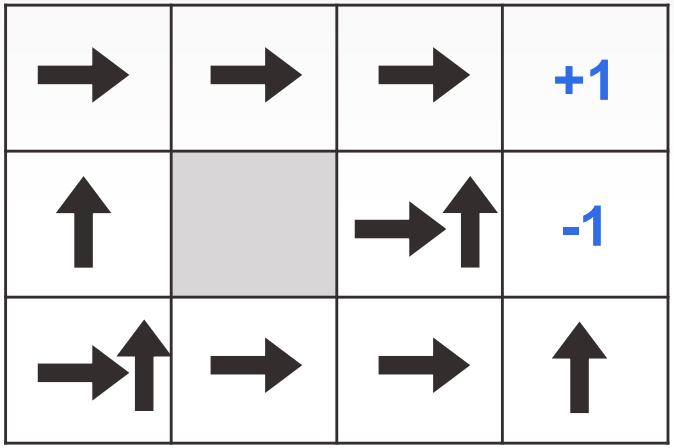
Total Reward

- We start with stochastic policy
  - 80% up, 10% right, 10% left
- We reach every state with positive probability
- Need action for every state
- Optimal actions are depicted



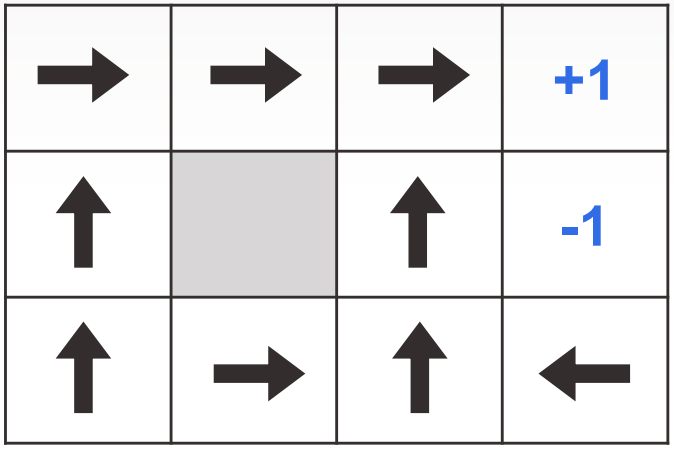
Total Reward

- Reward altered to -2
- There are ties



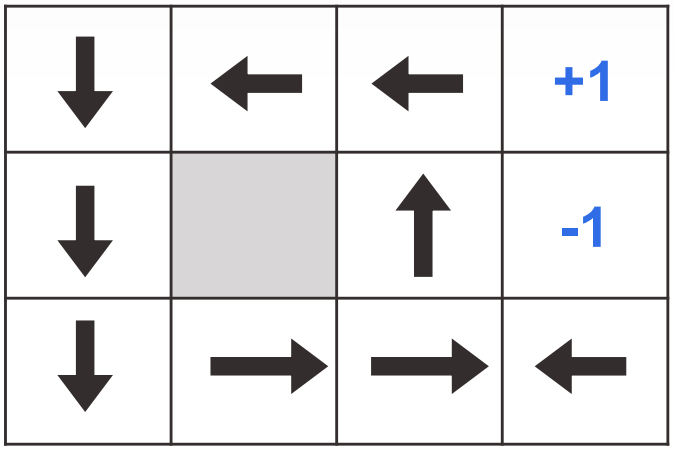
Total Reward

- Reward now -0.01
- Same as reward -0.04



Total Reward for each step: +0.01

- Reward = 0.01
- Encourage long paths
- Infinite total reward
  - Unless it is discounted



Putting it Together

- State and action spaces
- State, action (policy)
- Transition function
- Reward
- Discount factor
- Initial state (or distribution)

- Stochastic policies computationally tractable
  - But not needed on the paper

- Given policy
  - Markov chain
- Transition probabilities can be viewed as tensor
- Another view
  - Markov chain with states (state,action)

VALUE FUNCTION AND Q-FACTOR

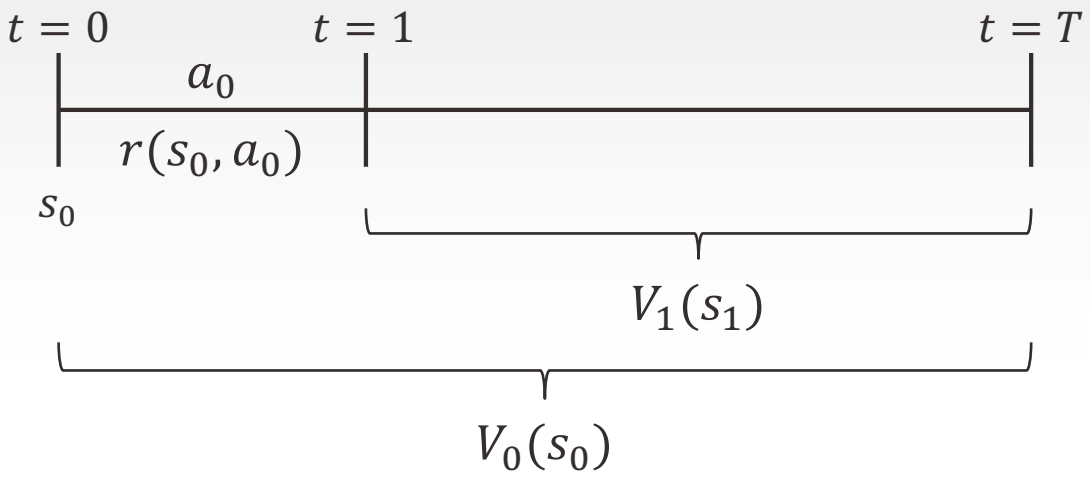
Total Reward

- Initial state $E_{s_0 \sim p(s_0)}$
- $\sum_{t=0}^{T} \gamma^t \cdot r(s_t, a_t) = r(s_0, a_0) + \gamma \underbrace{\sum_{t=1}^{T} \gamma^{t-1} \cdot r(s_t, a_t)}$
  - Same as original
  - Start in a different state
- Let $V_t(s_t) =$ maximum total reward from time *t* to the end given that at time *t* we are in state s_t
- Value function

Equivalent Expressions for Expectation

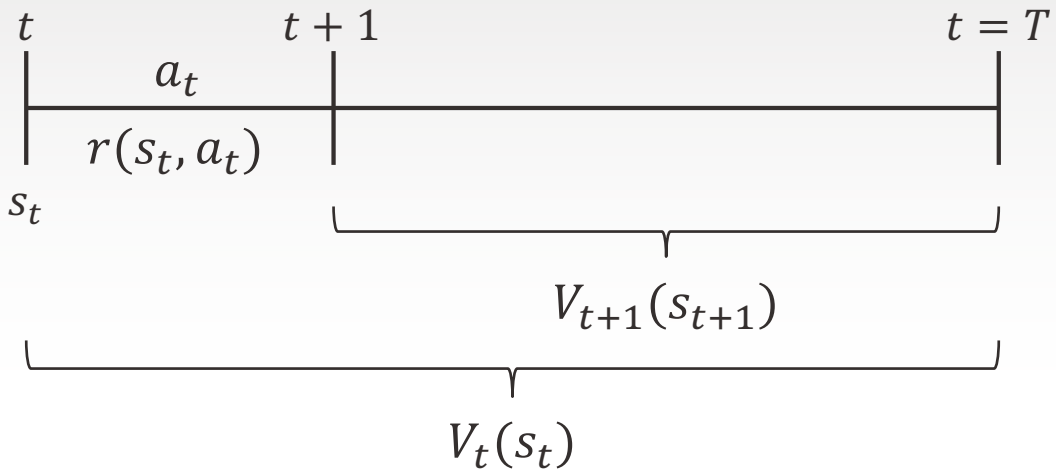
- Given s_t and policy π
- $a_t \sim \pi(a_t|s_t)$ - sampled action based on policy
- Next state
  - $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$
- Entire process
  - $a_t \sim \pi_\theta(a_t|s_t)$, $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$ - sampled next state based on sampled action
  - Equivalently $(s_{t+1}, a_{t+1}) \sim p_\theta(s_{t+1}, a_{t+1}|s_t, a_t)$

Optimality Equation



$$V_0(s_0) = \max_{a_0 \in \mathcal{A}} [r(s_0, a_0) + \gamma E_{s_1 \sim p(s_1|s_0,a_0)} V_1(s_1|s_0)]$$

Optimality Equation



$$V_t(s_t) = \max_{a_t \in \mathcal{A}} [r(s_t, a_t) + \gamma E_{s_{t+1} \sim p(s_{t+1}|s_t, a_t)} V_{t+1}(s_{t+1}|s_t)]$$

Value of Policy

- Let $V_t^{\pi}(s_t) =$ total reward of policy π from time t to the end given that at time t we are in state s_t

$$V_t^{\pi}(s_t) = \sum_{t'=t}^{T} \gamma^{t'} E_{(s_{t'}, a_{t'}) \sim p_\theta(s_{t'}, a_{t'} | s_{t'-1}, a_{t'-1})} [r(s_{t'}, a_{t'}) | s_t]$$

$$V_t^{\pi}(s_t) = E_{a_t \sim \pi(a_t|s_t), s_{t+1} \sim p(s_{t+1}|s_t, a_t)} [r(s_t, a_t) + \gamma V_{t+1}^{\pi}(s_{t+1}|s_t)]$$

- s_{t+1} function of a_t, s_t and stochastic environment

$$V_t^{\pi}(s_t) = E_{\substack{a_t \sim \pi(a_t|s_t) \\ s_{t+1} \sim p(s_{t+1}|s_t, a_t)}} [r(s_t, a_t) + \gamma V_t^{\pi}(s_{t+1}|s_t)]$$

Everywhere $\pi = \pi_\theta$

Stationarity

- An optimal policy
  - Depends on t (π_t or π_{θ_t})
- Optimal value function depends on t
  - V_t
- In practice for computational tractability
  - Policy does not depend on t
  - Value function and other subsequent functions do not depend on t
  - Our standard assumption

Optimal Policy

- $V(s_t) = \max_{\pi} V^{\pi}(s_t)$
- RL is about

$$\arg \ max_{\theta} V^{\pi_\theta}(s_o)$$

$$\arg \ max_{\theta} E_{s_o \sim p(s_o)} V^{\pi_\theta}(s_o)$$

- Omitting discounting factor from now on
- Omitting t in V_t and Q_t

Example

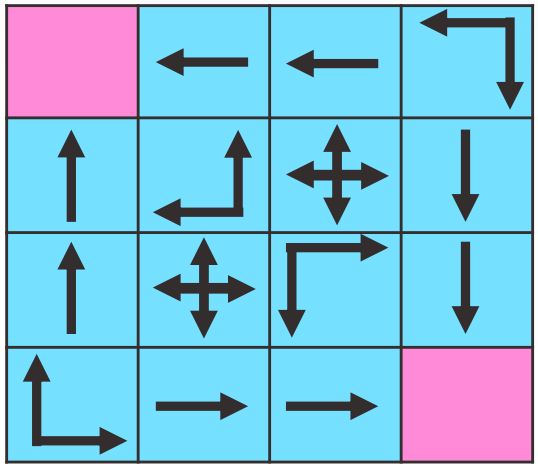
- Reward of -1, 4 actions, reward of 0 at corner points

Random policy

0	-14	-20	-22
-14	-18	-22	-20
-20	-22	-18	-14
-22	-20	-14	0

Optimal policy

0	-1	-2	-3
-1	-2	-3	-2
-2	-3	-2	-1
-3	-2	-1	0



M. Harmon, S. Harmon: Reinforcement Learning: a Tutorial

Q-factor

- $\sum_{t=0}^{T} E_{(s_t,a_t)\sim p_\theta(s_t,a_t)}[r(s_t,a_t)]$
- $Q^\pi(s_t,a_t) = \sum_{t'=t}^{T} E_{(s_{t'},a_{t'})\sim p_\theta(s_{t'},a_{t'})}[r(s_{t'},a_{t'})]$

$$Q^\pi(s_0,a_0) = r(s_0,a_0) + E_{\substack{a_1\sim\pi(a_1|s_1)\\ s_1\sim p(s_1|s_0,a_0)}}[[r(s_1,a_1) + \ \ldots|s_1]\,|s_0,a_0]$$

$$Q^\pi(s_t,a_t) = \sum_{t'=t}^{T} E_{(s_{t'},a_{t'})\sim p_\theta(s_{t'},a_{t'})} r(s_{t'},a_{t'})$$

Q and V

$$Q(s_t, a_t) = \max_{\pi} Q^{\pi}(s_t, a_t)$$

- Recursive definition of Q-factor

$$Q^{\pi}(s_t, a_t) = r(s_t, a_t) + E_{(s_{t+1}, a_{t+1}) \sim p_{\theta}(s_t, a_t)} Q^{\pi}(s_{t+1}, a_{t+1})$$

$$Q(s_t, a_t) = r(s_t, a_t) + \max_{a_{t+1}} E_{s_{t+1} \sim p(s_t, a_t)} Q(s_{t+1}, a_{t+1})$$

- Connection

$$V(s_t) = \max_{a_t} Q(s_t, a_t)$$

$$Q(s_t, a_t) = r(s_t, a_t) + E_{s_{t+1} \sim p(s_{t+1} | s_t, a_t)} V(s_{t+1})$$

Key Algorithmic Ideas

- Incumbent policy π and we know Q^{π}
  - We are at state s
  - Find $a' = \arg\max_{a} Q^{\pi}(s, a)$
  - Set $\pi(a'|s) = 1$
  - Sample new state $s \sim p(s|s, a')$
  - Caveat: We need to update the Q-factor

Key Algorithmic Ideas

- If $Q^{\pi}(s,a) > V^{\pi}(s)$
  - a is better than average under $\pi(a|s)$
  - Recall $V^{\pi}(s) = E[Q^{\pi}(s,a)]$
- Modify $\pi(a|s)$ to increase probability of a if $Q^{\pi}(s,a) > V^{\pi}(s)$
  - Use gradient
  - How?

MODEL FREE VARIANT

Model-based RL

- State and action spaces are given
- Reward and transition functions are also given
- Can compute best policy
  - Planning
- Use policy in actual deployment (execution)
- Example
  - Inventory problem
  - Drone navigation
    - Or is it?

Model-free RL

- Chess
  - Reward if you lose or win at the end
  - During game no clear notion of reward
  - Next state function also not explicitly given
    - Depends on the move of the opponent
- Applies to all games with lose/win
  - Includes sport
- Robot control
  - Environment not given
- Autonomous driving
  - What is reward?
  - Know that final reward to get to the destination without crash

Model-free RL

- Agent must take action without knowing
  - Immediate reward to be received
  - Next state
- Drone
  - Take action
  - If crash to a wall, receive negative reward
- Observe trajectories
  - Sample path
  - Most commonly called episodes

$$\tau_1 = s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, s_3, a_3, r_3, \ldots$$
$$\tau_2 = s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, s_3, a_3, r_3, \ldots$$
$$\tau_3 = s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, s_3, a_3, r_3, \ldots$$

Model-free RL

- Naïve algorithm
  - Randomly select states and actions
    - Observe reward and transition
  - Have supervised training dataset
    - *(s,a)*; label *r*
    - *(s,a)*; label s'
  - Use a classifier to model reward and transitions
  - Use model-based RL algorithm
- Not efficient in practice
  - Many algorithms that work directly with episodes
  - Imitation learning when reward not available in episodes