Reinforcement Learning (II)

©Dr. Min Chi
mchi@ncsu.edu

AI Academy

Reinforcement learning of optimal policies

- **In the RL framework we do not know the Markov decision process(MDP) model !!!**
- **Goal:** learn the optimal policy

$$\pi^* : S \rightarrow A$$

- **Two basic approaches:**
  - **Model based learning**
    - Learn the MDP model (probabilities, rewards) first
    - Solve the MDP afterwards
  - **Model-free learning**
    - Learn how to act directly
    - No need to learn the parameters of the MDP
  - A number of clones of the two in the literature

Model-based learning

- We need to learn **transition probabilities** and **rewards**
- **Learning of probabilities**
  - ML or Bayesian parameter estimates
  - Use counts

$$\tilde{P}(s'|s,a) = \frac{N_{s,a,s'}}{N_{s,a}} \qquad N_{s,a} = \sum_{s' \in S} N_{s,a,s'}$$

- **Learning rewards**
  - Similar to learning with immediate rewards

$$\tilde{R}(s,a) = \frac{1}{N_{s,a}} \sum_{i=1}^{N_{s,a}} r_i^{s,a}$$

- **Problem: on-line update of the policy**
  - **would require us to solve the MDP after every update !!**

Model-Based learning

- **Motivation:** value function update (value iteration):

$$V(s) \leftarrow \max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} P(s' \mid s,a) V(s') \right]$$

- Let

$$Q(s,a) = R(s,a) + \gamma \sum_{s' \in S} P(s' \mid s,a) V(s')$$

- Then $V(s) \leftarrow \max_{a \in A} Q(s,a)$

- Note that the update can be defined purely in terms of Q-functions

$$Q(s,a) \leftarrow R(s,a) + \gamma \sum_{s' \in S} P(s' \mid s,a) \max_{a'} Q(s',a')$$

Model-Free Q-learning

- **Q-learning** uses the Q-value update idea
  - **But** relies on a stochastic (on-line, sample by sample) update

$$Q(s,a) \leftarrow R(s,a) + \gamma \sum_{s' \in S} P(s' \mid s,a) \max_{a'} Q(s',a')$$

is replaced with

$$\hat{Q}(s,a) \leftarrow (1-\alpha)\hat{Q}(s,a) + \alpha\left(r(s,a) + \gamma \max_{a'} \hat{Q}(s',a')\right)$$

$r(s,a)$ - reward received from the environment after performing an action *a* in state *s*

s' - new state reached after action *a*

α - learning rate, a function of $N_{s,a}$ - a number of times *a* executed at *s*

Q-learning

The on-line update rule is applied repeatedly during direct interaction with an environment

Q-learning

initialize *Q(s,a) =0* for all *s,a* pairs

observe current state *s*

repeat

  select action *a* ; use some exploration/exploitation schedule

  receive reward *r*

  observe next state *s'*

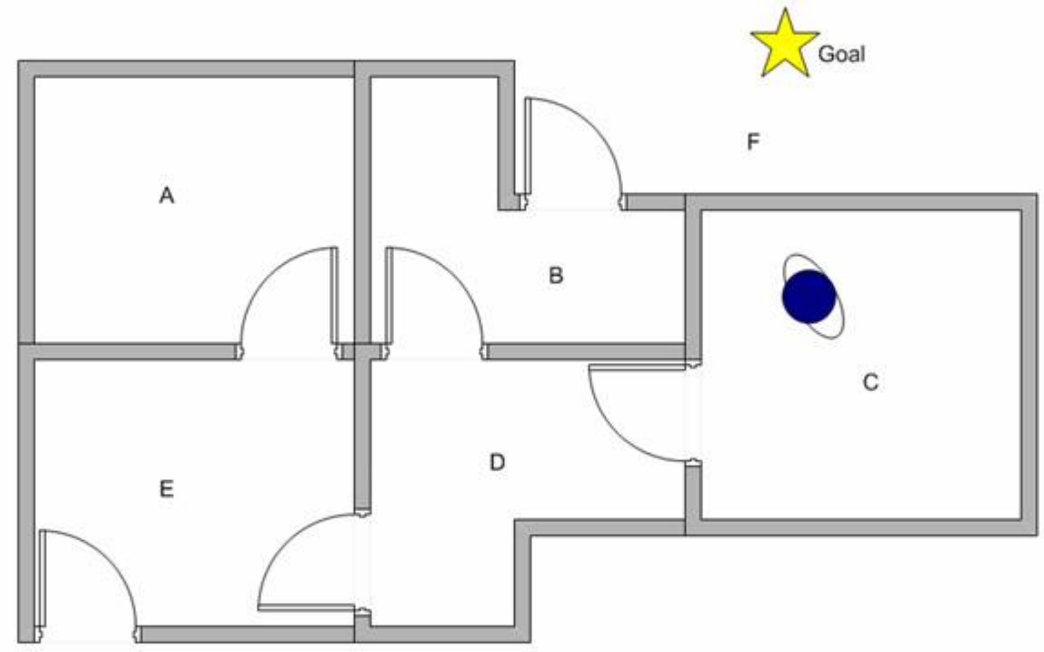
  update $Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q(s',a')\right)$

  set *s* to *s'*

end repeat

Problem Description

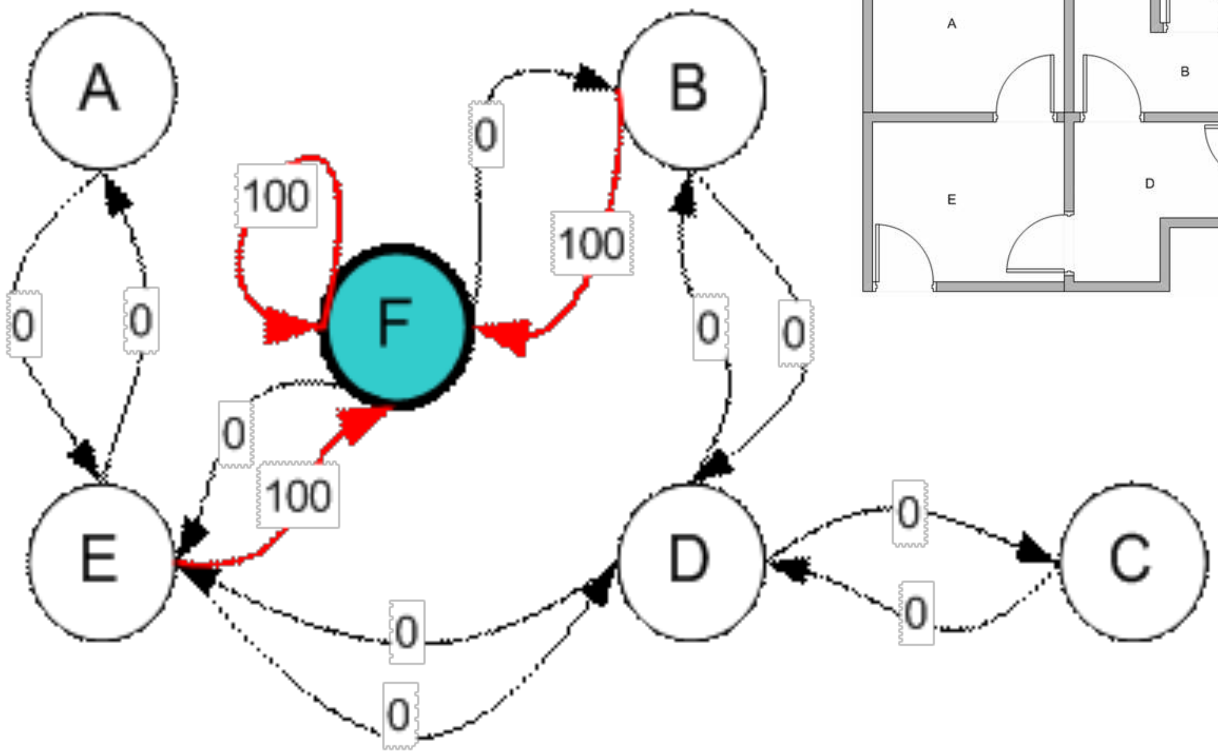
- Environment

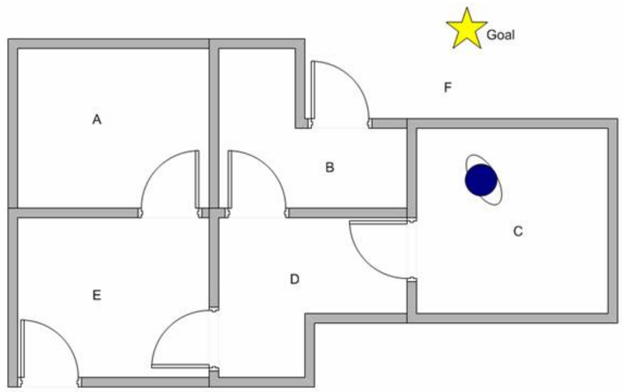


A to E: rooms, F: outside building (target).

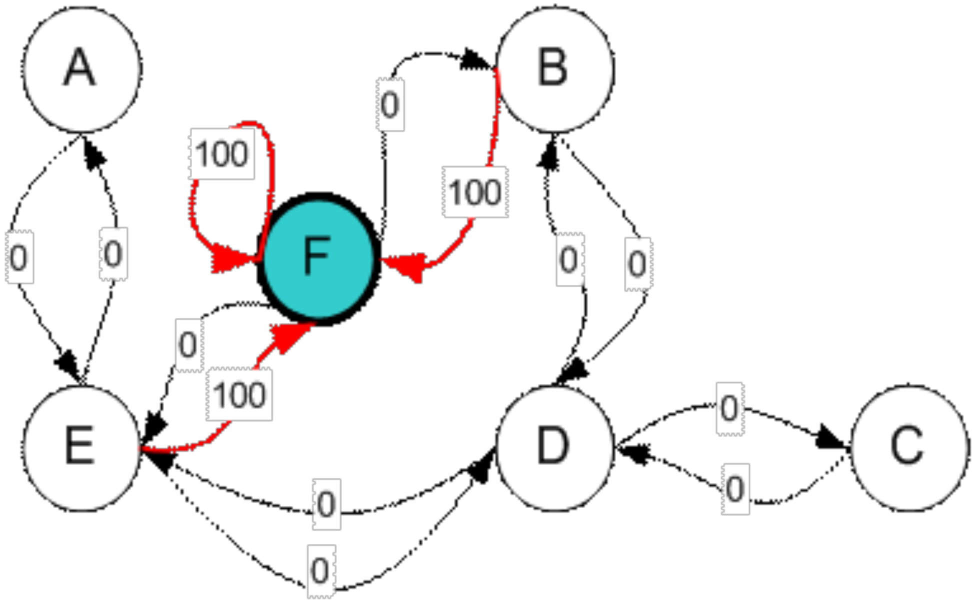
The aim is that an agent to learn to get out of building from any of rooms in an optimal way.

Modeling of the environment





Reward Matrix



$$
\mathbf{R} = \begin{array}{c} \textit{state} \setminus \textit{action} \\ A \\ B \\ C \\ D \\ E \\ F \end{array}
\begin{array}{c}
\begin{array}{cccccc} A & B & C & D & E & F \end{array} \\
\begin{bmatrix}
- & - & - & - & 0 & - \\
- & - & - & 0 & - & 100 \\
- & - & - & 0 & - & - \\
- & 0 & 0 & - & 0 & - \\
0 & - & - & 0 & - & 100 \\
- & 0 & - & - & 0 & 100
\end{bmatrix}
\end{array}
$$

Q-table and the update rule

$$
\mathbf{Q} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array}
\begin{array}{c}
\begin{array}{cccccc} A & B & C & D & E & F \end{array} \\
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\end{array}
$$

Q table update rule:

$$\mathbf{Q}(state, action) = \mathbf{R}(state, action) + \gamma \cdot Max\left[\mathbf{Q}(next\ state, all\ actions)\right]$$

$0 \leq \gamma < 1$: learning parameter

Numerical Example

Let us set the value of learning parameter and initial state as room B.

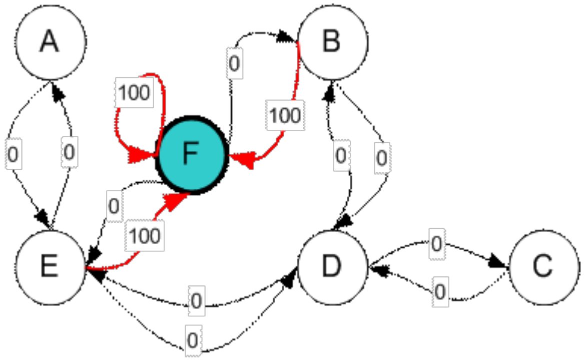
$$
\mathbf{Q} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array}
\begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\
\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}
\qquad
\mathbf{R} = \begin{array}{c} state \setminus action \\ A \\ B \\ C \\ D \\ E \\ F \end{array}
\begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\
\begin{bmatrix} - & - & - & - & 0 & - \\ - & - & - & 0 & - & 100 \\ - & - & - & 0 & - & - \\ - & 0 & 0 & - & 0 & - \\ 0 & - & - & 0 & - & 100 \\ - & 0 & - & - & 0 & 100 \end{bmatrix} \end{array}
$$

Episode 1

Look at the second row (state B) of matrix **R**. There are two possible actions for the current state B, that is to go to state D, or go to state F. By random selection, we select to go to F as our action.

$$\mathbf{Q}(state, action) = \mathbf{R}(state, action) + \gamma \cdot Max\left[\mathbf{Q}(next\ state, all\ actions)\right]$$

$$\mathbf{Q}(B,F) = \mathbf{R}(B,F) + 0.8 \cdot Max\{\mathbf{Q}(F,B), \mathbf{Q}(F,E), \mathbf{Q}(F,F)\} = 100 + 0.8 \cdot 0 = 100$$



$$\mathbf{Q} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

$$\mathbf{R} = \begin{array}{c} state \backslash action \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} - & - & - & - & 0 & - \\ - & - & - & 0 & - & 100 \\ - & - & - & 0 & - & - \\ - & 0 & 0 & - & 0 & - \\ 0 & - & - & 0 & - & 100 \\ - & 0 & - & - & 0 & 100 \end{bmatrix} \end{array}$$

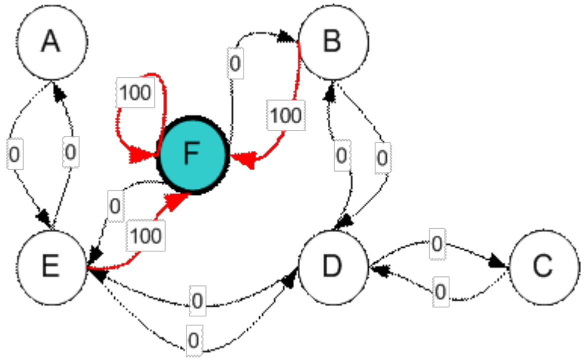
$$\mathbf{Q} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Episode 2

This time for instance we randomly have state D as our initial state. From **D;** it has 3 possible actions, B, C and E. We randomly select to go to state B as our action.

$$\mathbf{Q}(state, action) = \mathbf{R}(state, action) + \gamma \cdot Max\left[\mathbf{Q}(next\ state, all\ actions)\right]$$

$$\mathbf{Q}(D,B) = \mathbf{R}(D,B) + 0.8 \cdot Max\{\mathbf{Q}(B,D), \mathbf{Q}(B,F)\} = 0 + 0.8 \cdot Max\{0, 100\} = 80$$



$$\mathbf{Q} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

$$\mathbf{R} = \begin{array}{c} state \setminus action \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} - & - & - & - & 0 & - \\ - & - & - & 0 & - & 100 \\ - & - & - & 0 & - & - \\ - & 0 & 0 & - & 0 & - \\ 0 & - & - & 0 & - & 100 \\ - & 0 & - & - & 0 & 100 \end{bmatrix} \end{array}$$

$$\mathbf{Q} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

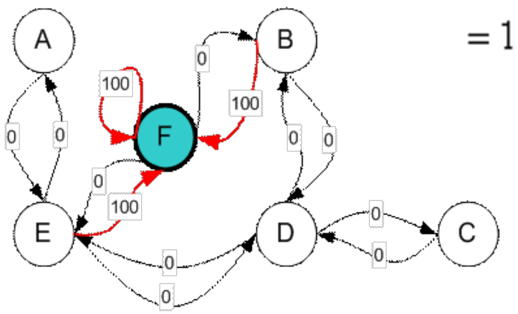
Episode 2 (cont'd)

The next state is B, now become the current state. We repeat the inner loop in Q learning algorithm because state B is not the goal state. There are two possible actions from the current state B, that is to go to state D, or go to state F. By lucky draw, our action selected is state F.

$$\mathbf{Q}(state, action) = \mathbf{R}(state, action) + \gamma \cdot Max\left[\mathbf{Q}(next\ state, all\ actions)\right]$$

$$\mathbf{Q}(B,F) = \mathbf{R}(B,F) + 0.8 \cdot Max\left\{\mathbf{Q}(F,B), \mathbf{Q}(F,E), \mathbf{Q}(F,F)\right\}$$

$$= 100 + 0.8 \cdot Max\left\{0,0,0\right\} = 100$$



$$\mathbf{Q} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

$$\mathbf{R} = \begin{array}{c} state \backslash action \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} - & - & - & - & 0 & - \\ - & - & - & 0 & - & 100 \\ - & - & - & 0 & - & - \\ - & 0 & 0 & - & 0 & - \\ 0 & - & - & 0 & - & 100 \\ - & 0 & - & - & 0 & 100 \end{bmatrix} \end{array}$$

No change

$$\mathbf{Q} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

AI Academy

NC STATE

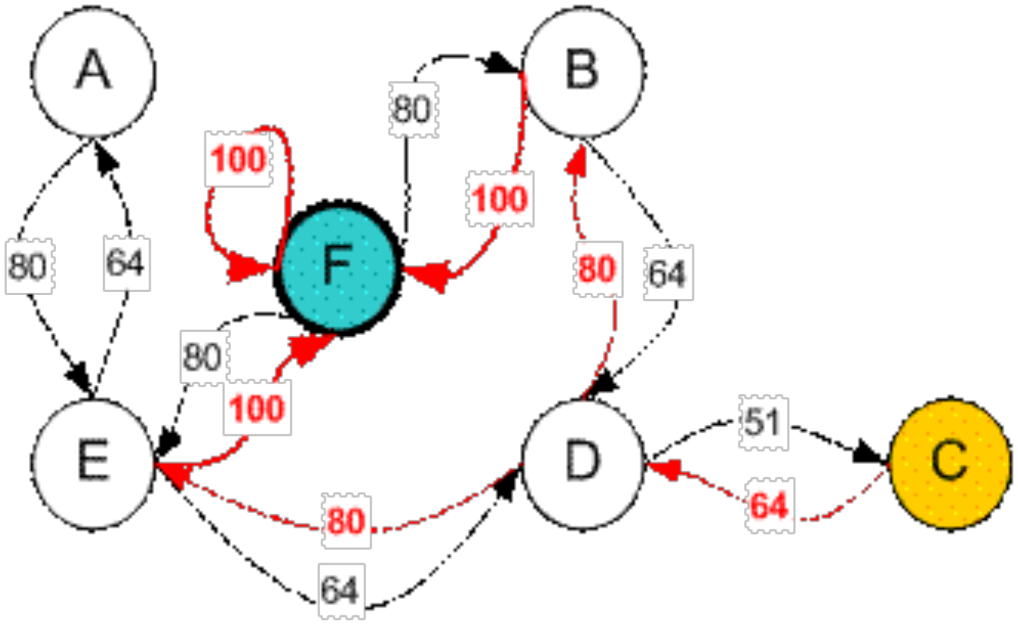
After Many Episodes

If our agent learns more and more experience through many episodes, it will finally reach convergence values of Q matrix as

$$\mathbf{Q} = \begin{array}{c} \textit{state} \setminus \textit{action} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \left[\begin{array}{cccccc} - & - & - & - & 400 & - \\ - & - & - & 320 & - & 500 \\ - & - & - & 320 & - & - \\ - & 400 & 256 & - & 400 & - \\ 320 & - & - & 320 & - & 500 \\ - & 400 & - & - & 400 & 500 \end{array}\right] \end{array}$$

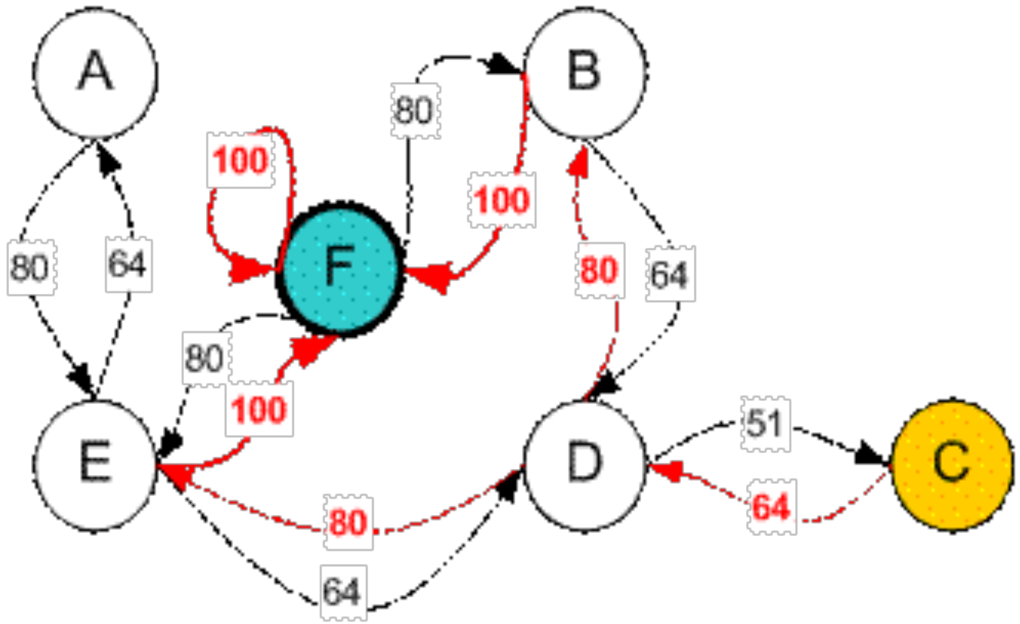
Normalized to percentage

$$\hat{\mathbf{Q}} = \begin{array}{c} \textit{state} \setminus \textit{action} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \left[\begin{array}{cccccc} - & - & - & - & 80 & - \\ - & - & - & 64 & - & 100 \\ - & - & - & 64 & - & - \\ - & 80 & 51 & - & 80 & - \\ 64 & - & - & 64 & - & 100 \\ - & 80 & - & - & 80 & 100 \end{array}\right] \end{array}$$



$$
\tilde{\mathbf{Q}} = \begin{array}{c} state \backslash action \\ A \\ B \\ C \\ D \\ E \\ F \end{array}
\begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\
\begin{bmatrix}
- & - & - & - & 80 & - \\
- & - & - & 64 & - & 100 \\
- & - & - & 64 & - & - \\
- & 80 & 51 & - & 80 & - \\
64 & - & - & 64 & - & 100 \\
- & 80 & - & - & 80 & 100
\end{bmatrix} \end{array}
$$

Once the Q matrix reaches almost the convergence value, our agent can reach the goal in an optimum way. To trace the sequence of states, it can easily compute by finding action that makes maximum Q for this state.



For example from initial State C, using the Q matrix, we can have the sequences C – D – B – F or C-D-E-F

AI Academy

NC STATE

Q-learning convergence

The **Q-learning is guaranteed to converge** to the optimal Q-values under the following conditions:

- Every state is visited and every action in that state is tried infinite number of times
  - This is assured via exploration/exploitation schedule
- The sequence of learning rates for each *Q(s,a)* satisfies:

1. $$\sum_{i=1}^{\infty} \alpha(i) = \infty$$

2. $$\sum_{i=1}^{\infty} \alpha(i)^2 < \infty$$

$\alpha(n(s,a))$ - Is the learning rate for the *n*th trial of (*s*,*a*)

Exploration vs. Exploitation

- In the RL with the delayed rewards
  - At any point in time the learner has an estimate of $\hat{Q}(\mathbf{x}, a)$ for any state action pair
- **Dilemma:**
  - Should the learner use the current best choice of action (exploitation)

$$\hat{\pi}(\mathbf{x}) = \arg\max_{a \in A} \hat{Q}(\mathbf{x}, a)$$

  - Or choose other action *a* and further improve its estimate of $\hat{Q}(\mathbf{x}, a)$ (exploration)
- **Exploration/exploitation strategies**
  - **Uniform exploration**
  - **Boltzman exploration**

Exploration vs. Exploitation

- **Uniform exploration**
  - Choose the "current" best choice with probability $1-\varepsilon$ $\hat{\pi}(\mathbf{x}) = \arg\max_{a \in A} \tilde{R}(\mathbf{x}, a)$
  - All other choices are selected with a uniform probability

$$p(a \mid x) = \frac{\varepsilon}{|A|-1}$$

- **Boltzman exploration**
  - The action is chosen randomly but proportionally to its current expected reward estimate

Summary

- Both value iteration and policy iteration are standard algorithms for solving MDPs, and there isn't currently universal agreement over which algorithm is better.

- For small MDPs, value iteration is ofte very fast and converges with very few iterations. However, for MDPs with large state spaces, solving for V explicitly would involve solving a large system of linear equations, and could be difficult.

- In these problems, policy iteration may be preferred. In practice value iteration seems to be used more often than policy iteration.

- Q-learning is model-free, and explore the temporal difference

Stay Connected

Dr. Min Chi

Associate Professor
mchi@ncsu.edu
(919) 515-7825

AI Academy

AI Academy

go.ncsu.edu/aiacademy

NC STATE UNIVERSITY