CS 570: Machine Learning Seminar

Fall 2016
Class Information

• Class web page:  
  http://web.cecs.pdx.edu/~mm/mlseminar2016-2017/fall2016/

• Class mailing list: cs570@cs.pdx.edu

• My office hours: T, Th, 2-3pm or by appointment (FAB 115-13)

• Machine Learning mailing list: machinelearning-request@cs.pdx.edu
Class Description

• Course is 1-credit graduate seminar for students who have already taken a course in Artificial Intelligence or Machine Learning.

• **Purpose of the course:** Each term: Deep dive into one recent area of ML. Not a general ML course.

• Students will read and discuss recent papers in the Machine Learning literature.

• Each student will be responsible for presenting 1 or 2 papers during the term.

• Course will be offered each term, and students may take it multiple times.

• CS MS students who take this course for three terms may count it as one of the courses for the "Artificial Intelligence and Machine Learning" masters track requirement.
Course Topics

Previous topics (under 510 number):

Winter 2015: Graphical Models
Spring 2015: Reinforcement Learning
Fall 2015: Neural Network Approaches to Natural Language Processing
Winter 2015: Analogy, Metaphor and Common Sense Reasoning
Spring 2016: AI and Creativity: AI in Art, Music, and Literature

This term’s topic: Deep Reinforcement Learning
Course work

• 1-2 papers will be assigned per week for everyone in the class to read.

• Each week 1 or 2 students will be assigned as discussion leaders for the week's papers.

• I will hand out a “question list” with each paper assignment. You need to complete it and turn it in on the day we cover the paper. (Just to make sure you read the paper).

• **Expected homework load:** ~3-4 hours per week on average
Grading

• Class attendance and participation
  – You may miss up to one class if necessary with no penalty

• Weekly questions
  – You can skip at most one week of this with no penalty

• Discussion leader presentation
Class Introductions
How to present a paper

• Start early

• Use slides

• Plan for 30 minutes or so

• Explain any unfamiliar terminology in the paper (do additional background reading if necessary)

• Don’t go through the paper exhaustively; just talk about most important parts

• Don’t overload slides with text

• Include graphs / images on your slides

• Outside information is encouraged (demos, short videos, explanations)

• Present 1–3 discussion questions for class.

• If you don’t understand something in the paper, let the class help out
Discussion Leader Schedule
Review of Reinforcement Learning
Robby the Robot can learn via reinforcement learning

Sensors:
N,S,E,W,C(urrent)

Actions:
 Move N
 Move S
 Move E
 Move W
 Move random
 Stay put
 Try to pick up can

Rewards/Penalties (points):
 Picks up can: 10
 Tries to pick up can on empty site: -1
 Crashes into wall: -5

"policy" = "strategy"
Supervised / Unsupervised / Reinforcement Learning

- Supervised: Each example has a label. Goal is to predict labels of unseen examples.

- Unsupervised: No labels. Goal is to cluster examples into unlabeled “classes”.

- Reinforcement: No labels. Sparse, time-delayed rewards. Goal is to learn a “policy”: mapping from states to actions.

- Hard problem in reinforcement learning: credit-assignment
Reinforcement learning is typically formalized as a **Markov decision process (MDP)**:

Agent $L$ only knows current state and actions available from that state.

Agent $L$ can:
- perceive a set $S$ of distinct states of an environment
- perform a set $A$ of actions.

Components of Markov Decision Process (MDP):
- Possible set $S$ of states (state space)
- Possible set $A$ of actions (action space)
- State transition function (unknown to learner $L$)
  \[ \delta : S \times A \rightarrow S \quad \delta(s_t,a_t) = s_{t+1} \]
- Reward function (unknown to learner $L$)
  \[ r : S \times A \rightarrow \mathbb{R} \quad r(s_t,a_t) = r_t \]
Formalization

Reinforcement learning is typically formalized as a Markov decision process (MDP):

Agent \( L \) only knows current state and actions available from that state.

Agent \( L \) can:

- perceive a set \( S \) of distinct states of an environment
- perform a set \( A \) of actions.

Components of Markov Decision Process (MDP):

- Possible set \( S \) of states (state space)
- Possible set \( A \) of actions (action space)
- State transition function (unknown to learner \( L \))
  \[ \delta : S \times A \rightarrow S \]
  \[ \delta(s_t, a_t) = s_{t+1} \]

- Reward function (unknown to learner \( L \))
  \[ r : S \times A \rightarrow \mathbb{R} \]
  \[ r(s_t, a_t) = r_t \]

Note: Both \( \delta \) and \( r \) can be deterministic or probabilistic.

In the Robby the Robot example, they are both deterministic.
Goal is for agent $L$ to learn policy $\pi$, $\pi : S \rightarrow A$, such that $\pi$ maximizes cumulative reward.
Example: Cumulative value of a policy

Sensors:
- N, S, E, W, C(urrent)

Actions:
- Move N
- Move S
- Move E
- Move W
- Move random
- Stay put
- Try to pick up can

Rewards/Penalties (points):
- Picks up can: 10
- Tries to pick up can on empty site: -1
- Crashes into wall: -5

Policy A: Always move east, picking up cans as you go

Policy B: Move up and down the columns of the grid, picking up cans as you go
Value function:
Formal definition of “cumulative value” with discounted rewards

Let $V^\pi(s_t)$ denote the “cumulative value” of $\pi$ starting from initial state $s_t$:

$$V^\pi(s_t) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ...$$

$$= \sum_{i=0}^{\infty} \gamma^i r_{t+i}$$

- where $0 \leq \gamma \leq 1$ is a “discounting” constant that determines the relative value of delayed versus immediate rewards.

- $V^\pi(s_t)$ is called the “value function” for policy $\pi$. It gives the expected value of starting in state $s_t$ and following policy $\pi$ “forever”.
• Note that rewards received $i$ times steps into the future are discounted exponentially (by a factor of $\gamma^i$).

• If $\gamma = 0$, we only care about immediate reward.

• The closer $\gamma$ is to 1, the more we care about future rewards relative to immediate reward.
• Precise specification of learning task:

We require that the agent learn policy $\pi$ that maximizes $V^\pi(s)$ for all states $s$.

We call such a policy an *optimal* policy, and denote it by $\pi^*$:

To simplify notation, let

$$V^*(s) = V^{\pi^*}(s)$$
What should L learn?

- Hard to learn $\pi^*$ directly, since we don’t have training data of the form $(s, a)$

- Only training data available to learner is sequence of rewards:
  
  $$r(s_0, a_0), r(s_1, a_1), \ldots$$

Most often, $r(s_t, a_t) = 0$ !

- So, what should the learner $L$ learn?
Learn evaluation function $V$?

• One possibility: learn evaluation function $V^*(s)$.

• Then $L$ would know what action to take next:

  – $L$ should prefer state $s_1$ over $s_2$ whenever $V^*(s_1) > V^*(s_2)$

  – Optimal action $a$ in state $s$ is the one that maximizes sum of $r(s,a)$ (immediate reward) and $V^*$ of the immediate successor state, discounted by $\gamma$:

    $$\pi^*(s) = \arg\max_a [r(s,a) + \gamma V^*(\delta(s,a))]$$
\[ \pi^*(s) = \arg \max_a \left[ r(s,a) + \gamma V^*(\delta(s,a)) \right] \]
Problem with learning $V^*$

**Problem:** Using $V^*$ to obtain optimal policy $\pi^*$ *requires* perfect knowledge of $\delta$ and $r$, which we earlier said are unknown to $L$. 
Alternative: Learning $Q$

- Alternative: we will estimate the following evaluation function $Q: S \times A \rightarrow \mathbb{R}$, as follows:

$$
\pi^*(s) = \arg\max_a [r(s,a) + \gamma V^*(\delta(s,a))]
$$

$$
= \arg\max_a [Q(s,a)],
$$

where $Q(s,a) = r(s,a) + \gamma V^*(\delta(s,a))$

- Suppose learner L has a good estimate, $\hat{Q}(s,a)$. Then, at each time step, L simply chooses action that maximizes $\hat{Q}(s,a)$. 


How to learn $Q$?
(In theory)

We have:

$$\pi^*(s) = \operatorname{argmax}_{a'} [r(s,a') + \gamma V^*(\delta(s,a'))]$$

$$V^*(s) = \max_a [r(s,a') + \gamma V^*(\delta(s,a'))]$$

$$= \max_{a'} [Q(s,a')]$$

because $Q(s,a') = r(s,a') + \gamma V^*(\delta(s,a'))$, by definition

So,

$$Q(s,a) = r(s,a) + \gamma \max_{a'} [Q(\delta(s,a),a')]$$

We can estimate $Q$ via iterative approximation.
How to learn $Q$ in practice?

- Initialize $\hat{Q}(s,a)$ to small random values for all $s$ and $a$
- Initialize $s$
- Repeat forever (or as long as you have time for):
  - Select action $a$
  - Take action $a$ and receive reward $r$
  - Observe new state $s'$
  - Update $\hat{Q}(s,a)$:
    $$
    \hat{Q}(s,a) \leftarrow \hat{Q}(s,a) + \eta \left( r + \gamma \max_{a'} \hat{Q}(s',a') - \hat{Q}(s,a) \right)
    $$
  - Update $s \leftarrow s'$
**Example**

Let $\gamma = 0.2$

Let $\eta = 1$

**State:** North, South, East, West, Current

$W = \text{Wall}$

$E = \text{Empty}$

$C = \text{Can}$

**Reward:**

<table>
<thead>
<tr>
<th>$\hat{Q}(s,a)$</th>
<th>MoveN</th>
<th>MoveS</th>
<th>MoveE</th>
<th>MoveW</th>
<th>Move Random</th>
<th>Stay Put</th>
<th>Pick Up Can</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
State: North, South, East, West, Current

W = Wall
E = Empty
C = Can

Reward:

<table>
<thead>
<tr>
<th>( \hat{Q}(s,a) )</th>
<th>MoveN</th>
<th>MoveS</th>
<th>MoveE</th>
<th>MoveW</th>
<th>Move Random</th>
<th>Stay Put</th>
<th>Pick Up Can</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Initialize \( \hat{Q}(s,a) \) to small random values (here, 0) for all \( s \) and \( a \)
Initialize $s$

**State:** North, South, East, West, Current

$W =$ Wall
$E =$ Empty
$C =$ Can

**Reward:**

<table>
<thead>
<tr>
<th>$\hat{Q}(s, a)$</th>
<th>MoveN</th>
<th>MoveS</th>
<th>MoveE</th>
<th>MoveW</th>
<th>Move Random</th>
<th>Stay Put</th>
<th>Pick Up Can</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>W, E, C, W, E</strong></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Select action $a$

State: North, South, East, West, Current

W = Wall
E = Empty
C = Can

Trial: 1

Reward:

<table>
<thead>
<tr>
<th>$\hat{Q}(s,a)$</th>
<th>MoveN</th>
<th>MoveS</th>
<th>MoveE</th>
<th>MoveW</th>
<th>Move Random</th>
<th>Stay Put</th>
<th>Pick Up Can</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, C, W, E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Perform action $a$ and receive reward $r$

**State:** North, South, East, West, Current

$W =$ Wall  
$E =$ Empty  
$C =$ Can

**Reward:** $-1$

<table>
<thead>
<tr>
<th>$\hat{Q}(s,a)$</th>
<th>MoveN</th>
<th>MoveS</th>
<th>MoveE</th>
<th>MoveW</th>
<th>Move Random</th>
<th>Stay Put</th>
<th>Pick Up Can</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$W, E, C, W, E$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Observe new state \( s' \)

State: North, South, East, West, Current

\( W \) = Wall  
\( E \) = Empty  
\( C \) = Can

Reward: \(-1\)

<table>
<thead>
<tr>
<th>( \hat{Q}(s,a) )</th>
<th>MoveN</th>
<th>MoveS</th>
<th>MoveE</th>
<th>MoveW</th>
<th>Move Random</th>
<th>Stay Put</th>
<th>Pick Up Can</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, C, W, E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
MoveN  MoveS  MoveE  MoveW  Move  Random  Stay Put  Pick Up Can

State: North, South, East, West, Current

W = Wall
E = Empty
C = Can

Reward: \(-1\)

<table>
<thead>
<tr>
<th>State</th>
<th>(\hat{Q}(s,a))</th>
</tr>
</thead>
<tbody>
<tr>
<td>…</td>
<td>MoveN</td>
</tr>
<tr>
<td>…</td>
<td>0</td>
</tr>
<tr>
<td>…</td>
<td>0</td>
</tr>
<tr>
<td>W, E, C, W, E</td>
<td>0</td>
</tr>
<tr>
<td>…</td>
<td>0</td>
</tr>
</tbody>
</table>

Update

\[
\hat{Q}(s,a) \leftarrow \hat{Q}(s,a) + \eta \left( r + \gamma \max_{a'} \hat{Q}(s',a') - \hat{Q}(s,a) \right)
\]

Trial: 1
Update

\[ \hat{Q}(s,a) \leftarrow \hat{Q}(s,a) + \eta \left( r + \gamma \max_{a'} \hat{Q}(s',a') - \hat{Q}(s,a) \right) \]

State: North, South, East, West, Current

W = Wall
E = Empty
C = Can

Trial: 1

Reward: −1

<table>
<thead>
<tr>
<th>(\hat{Q}(s,a))</th>
<th>MoveN</th>
<th>MoveS</th>
<th>MoveE</th>
<th>MoveW</th>
<th>Move Random</th>
<th>Stay Put</th>
<th>Pick Up Can</th>
</tr>
</thead>
<tbody>
<tr>
<td>…</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, C, W, E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>…</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
State: North, South, East, West, Current

W = Wall
E = Empty
C = Can

Reward:

<table>
<thead>
<tr>
<th>$\hat{Q}(s,a)$</th>
<th>MoveN</th>
<th>MoveS</th>
<th>MoveE</th>
<th>MoveW</th>
<th>Move Random</th>
<th>Stay Put</th>
<th>Pick Up Can</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, C, W, E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Select action $a$

**State:** North, South, East, West, Current

$W =$ Wall  
$E =$ Empty  
$C =$ Can

**Reward:**

<table>
<thead>
<tr>
<th>$\hat{Q}(s,a)$</th>
<th>MoveN</th>
<th>MoveS</th>
<th>MoveE</th>
<th>MoveW</th>
<th>Move Random</th>
<th>Stay Put</th>
<th>Pick Up Can</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$W, E, C, W, E$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Perform action $a$ and receive reward $r$

State: North, South, East, West, Current

$W =$ Wall
$E =$ Empty
$C =$ Can

Trial: 2

Reward: 0

<table>
<thead>
<tr>
<th>$\hat{Q}(s,a)$</th>
<th>MoveN</th>
<th>MoveS</th>
<th>MoveE</th>
<th>MoveW</th>
<th>Move Random</th>
<th>Stay Put</th>
<th>Pick Up Can</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$W, E, C, W, E$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Observe new state $s'$

State: North, South, East, West, Current

$W = \text{Wall}$
$E = \text{Empty}$
$C = \text{Can}$

Reward: 0

<table>
<thead>
<tr>
<th>$\hat{Q}(s, a)$</th>
<th>MoveN</th>
<th>MoveS</th>
<th>MoveE</th>
<th>MoveW</th>
<th>Move Random</th>
<th>Stay Put</th>
<th>Pick Up Can</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$W, E, C, W, E$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$W, E, E, E, C$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Update

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

State: North, South, East, West, Current

W = Wall
E = Empty
C = Can

Reward: 0

<table>
<thead>
<tr>
<th>State</th>
<th>MoveN</th>
<th>MoveS</th>
<th>MoveE</th>
<th>MoveW</th>
<th>Move Random</th>
<th>Stay Put</th>
<th>Pick Up Can</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, C, W, E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, E, E, C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Trial: 2
Update
\[
\hat{Q}(s,a) \leftarrow \hat{Q}(s,a) + \eta \left( r + \gamma \max_{a'} \hat{Q}(s',a') - \hat{Q}(s,a) \right)
\]

State: North, South, East, West, Current

W = Wall
E = Empty
C = Can

Reward: 0

<table>
<thead>
<tr>
<th>( \hat{Q}(s,a) )</th>
<th>MoveN</th>
<th>MoveS</th>
<th>MoveE</th>
<th>MoveW</th>
<th>Move Random</th>
<th>Stay Put</th>
<th>Pick Up Can</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, C, W, E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, E, E, C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
State: North, South, East, West, Current

W = Wall
E = Empty
C = Can

Reward: 0

<table>
<thead>
<tr>
<th>$\hat{Q}(s,a)$</th>
<th>MoveN</th>
<th>MoveS</th>
<th>MoveE</th>
<th>MoveW</th>
<th>Move Random</th>
<th>Stay Put</th>
<th>Pick Up Can</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, C, W, E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, E, E, C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Update $s \leftarrow s'$
Select action $a$

State: North, South, East, West, Current

$W = \text{Wall}$

$E = \text{Empty}$

$C = \text{Can}$

Reward:

<table>
<thead>
<tr>
<th>$\hat{Q}(s,a)$</th>
<th>MoveN</th>
<th>MoveS</th>
<th>MoveE</th>
<th>MoveW</th>
<th>Move Random</th>
<th>Stay Put</th>
<th>Pick Up Can</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$W, E, C, W, E$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$W, E, E, E, C$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Perform action $a$ and receive reward $r$

State: North, South, East, West, Current

$W =$ Wall  
$E =$ Empty  
$C =$ Can  

Reward: 10

<table>
<thead>
<tr>
<th>$\hat{Q}(s,a)$</th>
<th>MoveN</th>
<th>MoveS</th>
<th>MoveE</th>
<th>MoveW</th>
<th>Move Random</th>
<th>Stay Put</th>
<th>Pick Up Can</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$W, E, C, W, E$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$W, E, E, E, C$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Observe new state $s'$

State: North, South, East, West, Current

$W = \text{Wall}$
$E = \text{Empty}$
$C = \text{Can}$

Reward: 10

<table>
<thead>
<tr>
<th>$\hat{Q}(s,a)$</th>
<th>MoveN</th>
<th>MoveS</th>
<th>MoveE</th>
<th>MoveW</th>
<th>Move Random</th>
<th>Stay Put</th>
<th>Pick Up Can</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, E, E, E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, C, W, E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, E, E, C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
State: North, South, East, West, Current

W = Wall
E = Empty
C = Can

Reward: 10

Trial: 3

Update

\[ \hat{Q}(s,a) \leftarrow \hat{Q}(s,a) + \eta \left( r + \gamma \max_{a'} \hat{Q}(s',a') - \hat{Q}(s,a) \right) \]

State: North, South, East, West, Current

W = Wall
E = Empty
C = Can

Reward: 10

<table>
<thead>
<tr>
<th>( \hat{Q}(s,a) )</th>
<th>MoveN</th>
<th>MoveS</th>
<th>MoveE</th>
<th>MoveW</th>
<th>Move Random</th>
<th>Stay Put</th>
<th>Pick Up Can</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>W, E, E, E, E</strong></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>W, E, C, W, E</strong></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>W, E, E, E, C</strong></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
State: North, South, East, West, Current

W = Wall
E = Empty
C = Can

Reward: 10

Trial: 3

\[
\hat{Q}(s,a) \leftarrow \hat{Q}(s,a) + \eta \left( r + \gamma \max_{a'} \hat{Q}(s',a') - \hat{Q}(s,a) \right)
\]

Update
State: North, South, East, West, Current

W = Wall
E = Empty
C = Can

Reward: 10

Trial: 3

Update $s \leftarrow s'$

<table>
<thead>
<tr>
<th>$\hat{Q}(s,a)$</th>
<th>MoveN</th>
<th>MoveS</th>
<th>MoveE</th>
<th>MoveW</th>
<th>Move Random</th>
<th>Stay Put</th>
<th>Pick Up Can</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, E, E, E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, C, W, E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, E, E, C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
## Skipping ahead...

### State: North, South, East, West, Current
- **W** = Wall
- **E** = Empty
- **C** = Can

### Trial: $m$

#### Reward:

<table>
<thead>
<tr>
<th>$\hat{Q}(s,a)$</th>
<th>MoveN</th>
<th>MoveS</th>
<th>MoveE</th>
<th>MoveW</th>
<th>Move Random</th>
<th>Stay Put</th>
<th>Pick Up Can</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, E, E, E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, C, W, E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, E, E, C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Select action $a$

State: North, South, East, West, Current

$W =$ Wall  
$E =$ Empty  
$C =$ Can

Trial: $m$

Reward:

<table>
<thead>
<tr>
<th>$\hat{Q}(s,a)$</th>
<th>MoveN</th>
<th>MoveS</th>
<th>MoveE</th>
<th>MoveW</th>
<th>Move Random</th>
<th>Stay Put</th>
<th>Pick Up Can</th>
</tr>
</thead>
<tbody>
<tr>
<td>…</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, E, E, E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, C, W, E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, E, E, C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>…</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Perform action $a$ and receive reward $r$

**State: North, South, East, West, Current**

W = Wall  
E = Empty  
C = Can

**Reward: 10**

<table>
<thead>
<tr>
<th>$\hat{Q}(s,a)$</th>
<th>MoveN</th>
<th>MoveS</th>
<th>MoveE</th>
<th>MoveW</th>
<th>Move Random</th>
<th>Stay Put</th>
<th>Pick Up Can</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, E, E, E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, C, W, E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, E, E, C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Skipping ahead...

State: North, South, East, West, Current

W = Wall
E = Empty
C = Can

Trial: m

Reward: 10

<table>
<thead>
<tr>
<th>( \hat{Q}(s,a) )</th>
<th>MoveN</th>
<th>MoveS</th>
<th>MoveE</th>
<th>MoveW</th>
<th>Move Random</th>
<th>Stay Put</th>
<th>Pick Up Can</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, E, E, E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, C, W, E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, E, E, C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Observe new state \( s' \)
Skipping ahead...

Update

\[ \hat{Q}(s,a) \leftarrow \hat{Q}(s,a) + \eta \left( r + \gamma \max_{a'} \hat{Q}(s',a') - \hat{Q}(s,a) \right) \]

State: North, South, East, West, Current

W = Wall
E = Empty
C = Can

Reward: 10

<table>
<thead>
<tr>
<th>( \hat{Q}(s,a) )</th>
<th>MoveN</th>
<th>MoveS</th>
<th>MoveE</th>
<th>MoveW</th>
<th>Move Random</th>
<th>Stay Put</th>
<th>Pick Up Can</th>
</tr>
</thead>
<tbody>
<tr>
<td>\ldots</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, E, E, E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, C, W, E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, E, E, C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>\ldots</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Skipping ahead...

State:
- North, South, East, West, Current

W = Wall
E = Empty
C = Can

Reward: 10

Trial: \( m \)

Update

\[
\hat{Q}(s,a) \leftarrow \hat{Q}(s,a) + \eta \left( r + \gamma \max_{a'} \hat{Q}(s',a') - \hat{Q}(s,a) \right)
\]

\begin{tabular}{|c|c|c|c|c|c|c|c|}
\hline
\( \hat{Q}(s,a) \) & MoveN & MoveS & MoveE & MoveW & Move Random & Stay Put & Pick Up Can \\
\hline
\ldots & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
W, E, E, E, E & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
W, E, C, W, E & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
W, E, E, E, C & 0 & 0 & 0 & 0 & 0 & 0 & 10 \\
\ldots & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
\end{tabular}
Now on a new environment...

State: North, South, East, West, Current

W = Wall
E = Empty
C = Can

Reward:

$\hat{Q}(s, a)$ | MoveN | MoveS | MoveE | MoveW | Move Random | Stay Put | Pick Up Can
---|---|---|---|---|---|---|---
... | 0 | 0 | 0 | 0 | 0 | 0 | 0
W, E, E, E, E | 0 | 0 | 0 | 0 | 0 | 0 | 0
W, E, C, W, E | 0 | 0 | 0 | -1 | 0 | 0 | 0
W, E, E, E, C | 0 | 0 | 0 | 0 | 0 | 0 | 10
... | 0 | 0 | 0 | 0 | 0 | 0 | 0

Select action $a$
Now on a new environment...

Perform action $a$ and receive reward $r$

State: North, South, East, West, Current

$W =$ Wall  
$E =$ Empty  
$C =$ Can

Reward: 0

<table>
<thead>
<tr>
<th>$\hat{Q}(s,a)$</th>
<th>MoveN</th>
<th>MoveS</th>
<th>MoveE</th>
<th>MoveW</th>
<th>Move Random</th>
<th>Stay Put</th>
<th>Pick Up Can</th>
</tr>
</thead>
<tbody>
<tr>
<td>…</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, E, E, E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, C, W, E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, E, E, C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>…</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Now on a new environment...

State: North, South, East, West, Current

W = Wall
E = Empty
C = Can

Reward: 0

### Table: \( \hat{Q}(s,a) \)

<table>
<thead>
<tr>
<th>( \hat{Q}(s,a) )</th>
<th>MoveN</th>
<th>MoveS</th>
<th>MoveE</th>
<th>MoveW</th>
<th>Move Random</th>
<th>Stay Put</th>
<th>Pick Up Can</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, E, E, E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, C, W, E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, E, E, C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Observe new state \( s' \)
Now on a new environment...

Update

\[
\hat{Q}(s,a) \leftarrow \hat{Q}(s,a) + \eta \left( r + \gamma \max_{a'} \hat{Q}(s',a') - \hat{Q}(s,a) \right)
\]

State: North, South, East, West, Current

W = Wall
E = Empty
C = Can

Reward: 0

<table>
<thead>
<tr>
<th>( \hat{Q}(s,a) )</th>
<th>MoveN</th>
<th>MoveS</th>
<th>MoveE</th>
<th>MoveW</th>
<th>Move Random</th>
<th>Stay Put</th>
<th>Pick Up Can</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, E, E, E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, C, W, E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, E, E, C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Now on a new environment...

Update

\[ \hat{Q}(s,a) \leftarrow \hat{Q}(s,a) + \eta \left( r + \gamma \max_{a'} \hat{Q}(s',a') - \hat{Q}(s,a) \right) \]

State: North, South, East, West, Current

W = Wall
E = Empty
C = Can

Reward: 0

Trial: \( n \)

<table>
<thead>
<tr>
<th>( \hat{Q}(s,a) )</th>
<th>MoveN</th>
<th>MoveS</th>
<th>MoveE</th>
<th>MoveW</th>
<th>Move Random</th>
<th>Stay Put</th>
<th>Pick Up Can</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, E, E, E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, C, W, E</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W, E, E, E, C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
How to choose actions?
How to choose actions?

• Naïve strategy: at each time step, choose action that maximizes $\hat{Q}(s,a)$
How to choose actions?

• Naïve strategy: at each time step, choose action that maximizes $\hat{Q}(s,a)$

• This exploits current $\hat{Q}(s,a)$ but doesn’t further explore the state-action space (in case $\hat{Q}(s,a)$ is way off)
How to choose actions?

- Naïve strategy: at each time step, choose action that maximizes $\hat{Q}(s, a)$

- This exploits current $\hat{Q}(s, a)$ but doesn’t further explore the state-action space (in case $\hat{Q}(s, a)$ is way off)

- Common approach in $Q$ learning:

$$P(a_i \mid s) = \frac{e^{\hat{Q}(s, a_i)/T}}{\sum_j e^{\hat{Q}(s, a_j)/T}}, \quad T > 0$$
Another common approach: “epsilon-greedy: With probability epsilon, choose random action; otherwise choose action with highest $\hat{Q}(s,a)$ value
• This balances exploitation and exploration in a tunable way:
  – high $T$: more exploration (more random)
  – low $T$: more exploitation (more deterministic)

• Can start with high $T$, and decrease it as improves
Representation of $\hat{Q}(s, a)$

- Note that in all of the above discussion, $\hat{Q}(s, a)$ was assumed to be a look-up table, with a distinct table entry for each distinct $(s, a)$ pair.

- More commonly, $\hat{Q}(s, a)$ is represented as a function (e.g., as a neural network), and the function is estimated (e.g., through back-propagation).
  - **Input:** Representation of state $s$ and action $a$
  - **Output:** $\hat{Q}(s, a)$ value

- To speed things up, if number of possible actions is small:
  - **Input:** Representation of state $s$
  - **Output:** $\hat{Q}(s, a)$ value for each possible action $a$
Q-Learning Algorithm for Neural Networks

Given observed transition $< s, a, r, s' >$:

1. Do feedforward pass for the current state $s$ to get predicted Q-values for all possible actions.

2. Do feedforward pass for next state $s'$ and calculate maximum over network outputs:
   \[ \max_{a'} \hat{Q}(s', a') \]
   Let $a^*$ denote the action with maximum $\hat{Q}(s', a')$.

3. Set $a^*$ output node’s target to $r + \gamma \max_{a'} \hat{Q}(s', a')$. Set target for other actions’ output nodes to predicted Q-value from step 1 (so error is zero).

4. Update weights using backpropagation.
Example: Learning to play backgammon

Rules of backgammon
Complexity of Backgammon

• Over $10^{20}$ possible states.

• At each ply, 21 dice combinations, with average of about 20 legal moves per dice combination. Result is branching ratio of several hundred per ply.

• Chess has branching ratio of about 30-40 per ply.

• Brute-force look-ahead search is not practical!
Neurogammon
(G. Tesauro, 1989)

- Used supervised learning approach: multilayer NN trained by back-propagation on data base of recorded expert games.

- Input: raw board information (number of pieces at each location), and a few hand-crafted features that encoded important expert concepts.

- Neurogammon achieved strong intermediate level of play.

TD-Gammon
(G. Tesauro, 1994)

- Program had two main parts:
  - **Move Generator**: Program that generates all legal moves from current board configuration.
  - **Predictor network**: Multi-layer NN that predicts $Q(s,a)$: probability of winning the game from the current board configuration.

- Predictor network scores all legal moves. Highest scoring move is chosen.

- **Rewards**: Zero for all time steps except those on which game is won or lost.
Network Overview

Input Layer: 198 - 50 - 1, feedforward, fully connected
10,001 independent weights
Trained via TD(\(\lambda\)) and standard backpropagation

Hidden Layer: 50 nodes

Output Layer: 1 node
• Input: 198 units
  – 24 positions, 8 input units for each position (192 input units)
    • First 4 input units of each group of 8 represent # white pieces at that position,
    • Second 4 represent # black units at that position
  – Two inputs represent who is moving (white or black)
  – Two inputs represent pieces on the bar
  – Two inputs represent number of pieces borne off by each player.
• 50 hidden units

• 1 output unit (activation represents probability that white will win from given board configuration)
Program plays against itself.

On each turn:

- Use network to evaluate all possible moves from current board configuration. Choose the move with the highest (lowest as black) evaluation. This produces a new board configuration.

- If this is end of game, run back-propagation, with target output activation of 1 or 0 depending on whether white won or lost.

- Else evaluate new board configuration with neural network. Calculate difference between current evaluation and previous evaluation.

- Run back-propagation, using the current evaluation as desired output, and the board position previous to the current move as the input.
<table>
<thead>
<tr>
<th>Program</th>
<th>Training Games</th>
<th>Opponents</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDG 1.0</td>
<td>300,000</td>
<td>Robertie, Davis, Magriel</td>
<td>-13 pts/51 games (-0.25 ppg)</td>
</tr>
<tr>
<td>TDG 2.0</td>
<td>800,000</td>
<td>Goulding, Woolsey, Snellings, Russell, Sylvester</td>
<td>-7 pts/38 games (-0.18 ppg)</td>
</tr>
<tr>
<td>TDG 2.1</td>
<td>1,500,000</td>
<td>Robertie</td>
<td>-1 pt/40 games (-0.02 ppg)</td>
</tr>
</tbody>
</table>

**Table 1.** Results of testing TD-Gammon in play against world-class human opponents. Version 1.0 used 1-play search for move selection; versions 2.0 and 2.1 used 2-ply search. Version 2.0 had 40 hidden units; versions 1.0 and 2.1 had 80 hidden units.
From Sutton & Barto, *Reinforcement Learning: An Introduction*:

“After playing about 300,000 games against itself, TD-Gammon 0.0 as described above learned to play approximately as well as the best previous backgammon computer programs.”

“TD-Gammon 3.0 appears to be at, or very near, the playing strength of the best human players in the world. It may already be the world champion. These programs have already changed the way the best human players play the game. For example, TD-Gammon learned to play certain opening positions differently than was the convention among the best human players. Based on TD-Gammon's success and further analysis, the best human players now play these positions as TD-Gammon does (Tesauro, 1995).”
Convolutional Neural Networks

http://cs231n.github.io/convolutional-networks/