Outline

Concepts
States and Actions
Values

Example: Maze
Imagine a position in a tic-tac-toe game (knots and crosses). How do you decide next action?

Which are you most likely to win from?
Imagine a position in a tic-tac-toe game (knots and crosses). How do you decide next action?

Which are you most likely to win from?
- Guess at how likely to win. **definite, likely, maybe**
States and Actions

- Set of possible states, $S$. 

- Can be discrete values ($|S| < \infty$)
  - Tic-Tac-Toe game positions
  - Position in a maze
  - Sequence of steps in a plan

- Can be continuous values ($|S| = \infty$)
  - Joint angles of a robot arm
  - Position and velocity of a race car
  - Parameter values for a network routing strategy

- Set of possible actions, $A$.

- Can be discrete values ($|A| < \infty$)
  - Next moves in Tic-Tac-Toe
  - Directions to step in a maze
  - Rearrangements of a sequence of steps in a plan

- Can be continuous values ($|A| = \infty$)
  - Torques to apply to the joints of a robot arm
  - Fuel rate and turning torque in a race car
  - Settings of parameter values for a network routing strategy
States and Actions

- Set of possible states, \( S \).
- Can be discrete values \(|S| < \infty\)
States and Actions

- Set of possible states, $S$.
  - Can be discrete values ($|S| < \infty$)
  - Tic-Tac-Toe game positions
States and Actions

- Set of possible states, $S$.
  - Can be discrete values ($|S| < \infty$)
    - Tic-Tac-Toe game positions
    - Position in a maze
States and Actions

- Set of possible states, $S$.
  - Can be discrete values ($|S| < \infty$)
    - Tic-Tac-Toe game positions
    - Position in a maze
    - Sequence of steps in a plan
States and Actions

- Set of possible states, $S$.
  - Can be discrete values ($|S| < \infty$)
    - Tic-Tac-Toe game positions
    - Position in a maze
    - Sequence of steps in a plan
  - Can be continuous values ($|S| = \infty$)
    - Joint angles of a robot arm
    - Position and velocity of a race car
    - Parameter values for a network routing strategy
States and Actions

- Set of possible states, $S$.
  - Can be discrete values ($|S| < \infty$)
    - Tic-Tac-Toe game positions
    - Position in a maze
    - Sequence of steps in a plan
  - Can be continuous values ($|S| = \infty$)
    - Joint angles of a robot arm
States and Actions

- Set of possible states, $S$.
  - Can be discrete values ($|S| < \infty$)
    - Tic-Tac-Toe game positions
    - Position in a maze
    - Sequence of steps in a plan
  - Can be continuous values ($|S| = \infty$)
    - Joint angles of a robot arm
    - Position and velocity of a race car
States and Actions

- Set of possible states, $S$.
  - Can be discrete values ($|S| < \infty$)
    - Tic-Tac-Toe game positions
    - Position in a maze
    - Sequence of steps in a plan
  - Can be continuous values ($|S| = \infty$)
    - Joint angles of a robot arm
    - Position and velocity of a race car
    - Parameter values for a network routing strategy
States and Actions

- Set of possible states, $S$.
  - Can be discrete values ($|S| < \infty$)
    - Tic-Tac-Toe game positions
    - Position in a maze
    - Sequence of steps in a plan
  - Can be continuous values ($|S| = \infty$)
    - Joint angles of a robot arm
    - Position and velocity of a race car
    - Parameter values for a network routing strategy

- Set of possible actions, $A$. 

Example: Maze
States and Actions

- Set of possible states, $S$.
  - Can be discrete values ($|S| < \infty$)
    - Tic-Tac-Toe game positions
    - Position in a maze
    - Sequence of steps in a plan
  - Can be continuous values ($|S| = \infty$)
    - Joint angles of a robot arm
    - Position and velocity of a race car
    - Parameter values for a network routing strategy

- Set of possible actions, $A$.
  - Can be discrete values ($|A| < \infty$)
States and Actions

- Set of possible states, $S$.
  - Can be discrete values ($|S| < \infty$)
    - Tic-Tac-Toe game positions
    - Position in a maze
    - Sequence of steps in a plan
  - Can be continuous values ($|S| = \infty$)
    - Joint angles of a robot arm
    - Position and velocity of a race car
    - Parameter values for a network routing strategy

- Set of possible actions, $A$.
  - Can be discrete values ($|A| < \infty$)
    - Next moves in Tic-Tac-Toe
  - Can be continuous values ($|A| = \infty$)
    - Torques to apply to the joints of a robot arm
    - Fuel rate and turning torque in a race car
    - Settings of parameter values for a network routing strategy
States and Actions

- Set of possible states, $S$.
  - Can be discrete values ($|S| < \infty$)
    - Tic-Tac-Toe game positions
    - Position in a maze
    - Sequence of steps in a plan
  - Can be continuous values ($|S| = \infty$)
    - Joint angles of a robot arm
    - Position and velocity of a race car
    - Parameter values for a network routing strategy

- Set of possible actions, $A$.
  - Can be discrete values ($|A| < \infty$)
    - Next moves in Tic-Tac-Toe
    - Directions to step in a maze
States and Actions

- Set of possible states, $S$.
  - Can be discrete values ($|S| < \infty$)
    - Tic-Tac-Toe game positions
    - Position in a maze
    - Sequence of steps in a plan
  - Can be continuous values ($|S| = \infty$)
    - Joint angles of a robot arm
    - Position and velocity of a race car
    - Parameter values for a network routing strategy

- Set of possible actions, $A$.
  - Can be discrete values ($|A| < \infty$)
    - Next moves in Tic-Tac-Toe
    - Directions to step in a maze
    - Rearrangements of a sequence of steps in a plan
States and Actions

- Set of possible states, $S$.
  - Can be discrete values ($|S| < \infty$)
    - Tic-Tac-Toe game positions
    - Position in a maze
    - Sequence of steps in a plan
  - Can be continuous values ($|S| = \infty$)
    - Joint angles of a robot arm
    - Position and velocity of a race car
    - Parameter values for a network routing strategy

- Set of possible actions, $A$.
  - Can be discrete values ($|A| < \infty$)
    - Next moves in Tic-Tac-Toe
    - Directions to step in a maze
    - Rearrangements of a sequence of steps in a plan
  - Can be continuous values ($|A| = \infty$)
    - Torques to apply to the joints of a robot arm
    - Fuel rate and turning torque in a race car
    - Settings of parameter values for a network routing strategy
States and Actions

- Set of possible states, $S$.
  - Can be discrete values ($|S| < \infty$)
    - Tic-Tac-Toe game positions
    - Position in a maze
    - Sequence of steps in a plan
  - Can be continuous values ($|S| = \infty$)
    - Joint angles of a robot arm
    - Position and velocity of a race car
    - Parameter values for a network routing strategy

- Set of possible actions, $A$.
  - Can be discrete values ($|A| < \infty$)
    - Next moves in Tic-Tac-Toe
    - Directions to step in a maze
    - Rearrangements of a sequence of steps in a plan
  - Can be continuous values ($|A| = \infty$)
    - Torques to apply to the joints of a robot arm
States and Actions

- Set of possible states, $S$.
  - Can be discrete values ($|S| < \infty$)
    - Tic-Tac-Toe game positions
    - Position in a maze
    - Sequence of steps in a plan
  - Can be continuous values ($|S| = \infty$)
    - Joint angles of a robot arm
    - Position and velocity of a race car
    - Parameter values for a network routing strategy

- Set of possible actions, $A$.
  - Can be discrete values ($|A| < \infty$)
    - Next moves in Tic-Tac-Toe
    - Directions to step in a maze
    - Rearrangements of a sequence of steps in a plan
  - Can be continuous values ($|A| = \infty$)
    - Torques to apply to the joints of a robot arm
    - Fuel rate and turning torque in a race car
States and Actions

- Set of possible states, $S$.
  - Can be discrete values ($|S| < \infty$)
    - Tic-Tac-Toe game positions
    - Position in a maze
    - Sequence of steps in a plan
  - Can be continuous values ($|S| = \infty$)
    - Joint angles of a robot arm
    - Position and velocity of a race car
    - Parameter values for a network routing strategy

- Set of possible actions, $A$.
  - Can be discrete values ($|A| < \infty$)
    - Next moves in Tic-Tac-Toe
    - Directions to step in a maze
    - Rearrangements of a sequence of steps in a plan
  - Can be continuous values ($|A| = \infty$)
    - Torques to apply to the joints of a robot arm
    - Fuel rate and turning torque in a race car
    - Settings of parameter values for a network routing strategy
Value of an Action from a State

- Want to choose action that we predict will result in the best possible future from the current state. Need a value that represents future outcome.
Value of an Action from a State

- Want to choose action that we predict will result in the best possible future from the current state. Need a value that represents future outcome.
- What should the value represent?

Tic-Tac-Toe: Likelihood of winning from a game position.
Maze: Number of steps to reach the goal.
Plan: Efficiency in time and cost of accomplishing the objective for particular rearrangement of steps in a plan.
Robot: Energy required to move the gripper on a robot arm to a destination.
Race car: Time to reach the finish line.
Network routing: Throughput.

With correct values, multi-step decision problems are reduced to single-step decision problems. Just pick action with best value. Guaranteed to find optimal multi-step solution (dynamic programming).
Value of an Action from a State

- Want to choose action that we predict will result in the best possible future from the current state. Need a value that represents future outcome.
- What should the value represent?
  - Tic-Tac-Toe: Likelihood of winning from a game position.
Value of an Action from a State

- Want to choose action that we predict will result in the best possible future from the current state. Need a value that represents future outcome.
- What should the value represent?
  - Tic-Tac-Toe: Likelihood of winning from a game position.
  - Maze: Number of steps to reach the goal.
Value of an Action from a State

- Want to choose action that we predict will result in the best possible future from the current state. Need a value that represents future outcome.
- What should the value represent?
  - Tic-Tac-Toe: Likelihood of winning from a game position.
  - Maze: Number of steps to reach the goal.
  - Plan: Efficiency in time and cost of accomplishing the objective for particular rearrangement of steps in a plan.
Value of an Action from a State

- Want to choose action that we predict will result in the best possible future from the current state. Need a value that represents future outcome.

- What should the value represent?
  - Tic-Tac-Toe: Likelihood of winning from a game position.
  - Maze: Number of steps to reach the goal.
  - Plan: Efficiency in time and cost of accomplishing the objective for particular rearrangement of steps in a plan.
  - Robot: Energy required to move the gripper on a robot arm to a destination.
Value of an Action from a State

- Want to choose action that we predict will result in the best possible future from the current state. Need a value that represents future outcome.
- What should the value represent?
  - Tic-Tac-Toe: Likelihood of winning from a game position.
  - Maze: Number of steps to reach the goal.
  - Plan: Efficiency in time and cost of accomplishing the objective for particular rearrangement of steps in a plan.
  - Robot: Energy required to move the gripper on a robot arm to a destination.
  - Race car: Time to reach the finish line.
Value of an Action from a State

- Want to choose action that we predict will result in the best possible future from the current state. Need a value that represents future outcome.
- What should the value represent?
  - Tic-Tac-Toe: Likelihood of winning from a game position.
  - Maze: Number of steps to reach the goal.
  - Plan: Efficiency in time and cost of accomplishing the objective for particular rearrangement of steps in a plan.
  - Robot: Energy required to move the gripper on a robot arm to a destination.
  - Race car: Time to reach the finish line.
  - Network routing: Throughput.
Value of an Action from a State

- Want to choose action that we predict will result in the best possible future from the current state. Need a value that represents future outcome.

- What should the value represent?
  - Tic-Tac-Toe: Likelihood of winning from a game position.
  - Maze: Number of steps to reach the goal.
  - Plan: Efficiency in time and cost of accomplishing the objective for particular rearrangement of steps in a plan.
  - Robot: Energy required to move the gripper on a robot arm to a destination.
  - Race car: Time to reach the finish line.
  - Network routing: Throughput.

- With correct values, multi-step decision problems are reduced to single-step decision problems. Just pick action with best value. Guaranteed to find optimal multi-step solution (dynamic programming).
The utility or cost of an action taken from a state is the *reinforcement* for that action from that state. The value of that state-action is the expected value of the full *return* or the sum of reinforcements that will follow when that action is taken.

![Diagram of reinforcement and return values](image)

The utility or cost of an action taken from a state is the *reinforcement* for that action from that state. The value of that state-action is the expected value of the full *return* or the sum of reinforcements that will follow when that action is taken.
The utility or cost of an action taken from a state is the \textit{reinforcement} for that action from that state. The value of that state-action is the expected value of the full \textit{return} or the sum of reinforcements that will follow when that action is taken.

- Say we are in state $s_t$ at time $t$. Upon taking action $a_t$ from that state we observe the one step reinforcement $r_{t+1}$, and the next state $s_{t+1}$. 

![Diagram of state transitions] 

- Reinforcements: $r=0.2$ to $R=0.9$, $r=0.3$ to $R=0.7$, $r=0.1$ to $R=0.4$, $r=0.2$ to $R=0.3$, $r=0.1$ to $R=0.1$
The utility or cost of an action taken from a state is the *reinforcement* for that action from that state. The value of that state-action is the expected value of the full *return* or the sum of reinforcements that will follow when that action is taken.

![Diagram of state transitions with reinforcements and returns](image)

- Say we are in state $s_t$ at time $t$. Upon taking action $a_t$ from that state we observe the one step reinforcement $r_{t+1}$, and the next state $s_{t+1}$.
- Say this continues until we reach a goal state, $K$ steps later. What is the return $R_t$ from state $s_t$?
The utility or cost of an action taken from a state is the \textit{reinforcement} for that action from that state. The value of that state-action is the expected value of the full \textit{return} or the sum of reinforcements that will follow when that action is taken.

![Diagram of states and transitions with reinforcements and rewards](image)

- Say we are in state $s_t$ at time $t$. Upon taking action $a_t$ from that state we observe the one step reinforcement $r_{t+1}$, and the next state $s_{t+1}$.
- Say this continues until we reach a goal state, $K$ steps later. What is the return $R_t$ from state $s_t$?
The utility or cost of an action taken from a state is the \textit{reinforcement} for that action from that state. The value of that state-action is the expected value of the full \textit{return} or the sum of reinforcements that will follow when that action is taken.

Say we are in state $s_t$ at time $t$. Upon taking action $a_t$ from that state we observe the one step reinforcement $r_{t+1}$, and the next state $s_{t+1}$.

Say this continues until we reach a goal state, $K$ steps later. What is the return $R_t$ from state $s_t$?

$$R_t = \sum_{k=0}^{K} r_{t+k+1}$$
Use the returns to choose best action.

- $a_1$: $r=0.2$, $R=0.9$
- $a_2$: $r=0.3$, $R=0.7$
- $a_3$: $r=0.1$, $R=0.4$

- $a_1$: $r=0.2$, $R=0.3$
- $a_2$: $r=0.1$, $R=0.4$
- $a_3$: $r=0.2$, $R=0.6$

- $a_1$: $r=0.1$, $R=0.3$
- $a_2$: $r=0.3$, $R=0.7$
- $a_3$: $r=0.1$, $R=0.4$

- $a_1$: $r=0.1$, $R=0.2$
- $a_2$: $r=0.2$, $R=0.3$
- $a_3$: $r=0.0$, $R=0.4$

- $a_1$: $r=0.1$, $R=0.1$
- $a_2$: $r=0.1$, $R=0.2$
- $a_3$: $r=0.1$, $R=0.3$

- $a_1$: $r=0.1$, $R=0.1$
- $a_2$: $r=0.1$, $R=0.2$
- $a_3$: $r=0.1$, $R=0.3$

- $a_1$: $r=0.1$, $R=0.1$
- $a_2$: $r=0.1$, $R=0.2$
- $a_3$: $r=0.1$, $R=0.3$

- $a_1$: $r=0.1$, $R=0.1$
- $a_2$: $r=0.1$, $R=0.2$
- $a_3$: $r=0.1$, $R=0.3$

- $a_1$: $r=0.1$, $R=0.1$
- $a_2$: $r=0.1$, $R=0.2$
- $a_3$: $r=0.1$, $R=0.3$

- $a_1$: $r=0.1$, $R=0.1$
- $a_2$: $r=0.1$, $R=0.2$
- $a_3$: $r=0.1$, $R=0.3$
Use the returns to choose best action.

Right...are we maximizing or minimizing? What does the reinforcement represent? Let’s say it is energy used that we want to minimize. $a_1$, $a_2$, or $a_3$?
Use the returns to choose best action.

Right...are we maximizing or minimizing? What does the reinforcement represent? Let's say it is energy used that we want to minimize. $a_1$, $a_2$, or $a_3$?
How to Acquire the Values

- Write the code to calculate them.

Usually not possible. If you can do this for your problem, why are you considering machine learning?

Might be able to do this for Tic-Tac-Toe. Use dynamic programming.

Usually not possible. Requires knowledge of the probabilities of transitions between all states for all actions.

Learn from examples, lots of examples. Lots of 5-tuples: \((s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})\).

Monte Carlo:
Assign to each state-action pair an average of the observed returns.

\[
\text{value}(s_t, a_t) \approx \text{mean of } R(s_t, a_t)
\]

Temporal Difference (TD):
Using \(\text{value}(s_{t+1}, a_{t+1})\) as estimate of return from next state, update current state-action value as

\[
\text{value}(s_t, a_t) \approx r_{t+1} + \text{value}(s_{t+1}, a_{t+1})
\]
How to Acquire the Values

- Write the code to calculate them.
- Usually not possible. If you can do this for your problem, why are you considering machine learning? Might be able to do this for Tic-Tac-Toe.

Monte Carlo:
Assign to each state-action pair an average of the observed returns.

\[ \text{value}(s_t, a_t) \approx \text{mean of } R(s_t, a_t) \]

Temporal Difference (TD):
Using \( \text{value}(s_{t+1}, a_{t+1}) \) as estimate of return from next state, update current state-action value as

\[ \text{value}(s_t, a_t) \approx r_{t+1} + \text{value}(s_{t+1}, a_{t+1}) \]
How to Acquire the Values

- Write the code to calculate them.
  - Usually not possible. If you can do this for your problem, why are you considering machine learning? Might be able to do this for Tic-Tac-Toe.
- Use dynamic programming.
How to Acquire the Values

- Write the code to calculate them.
  - Usually not possible. If you can do this for your problem, why are you considering machine learning? Might be able to do this for Tic-Tac-Toe.
- Use dynamic programming.
  - Usually not possible. Requires knowledge of the probabilities of transitions between all states for all actions.
How to Acquire the Values

- Write the code to calculate them.
  - Usually not possible. If you can do this for your problem, why are you considering machine learning? Might be able to do this for Tic-Tac-Toe.
- Use dynamic programming.
  - Usually not possible. Requires knowledge of the probabilities of transitions between all states for all actions.
- Learn from examples, lots of examples. Lots of 5-tuples: state, action, reinforcement, next state, next action \((s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})\).
How to Acquire the Values

- Write the code to calculate them.
  - Usually not possible. If you can do this for your problem, why are you considering machine learning? Might be able to do this for Tic-Tac-Toe.
- Use dynamic programming.
  - Usually not possible. Requires knowledge of the probabilities of transitions between all states for all actions.
- Learn from examples, lots of examples. Lots of 5-tuples: state, action, reinforcement, next state, next action $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$.
  - **Monte Carlo**: Assign to each state-action pair an average of the observed returns.
    \[
    \text{value}(s_t, a_t) \approx \text{mean of } R(s_t, a_t)
    \]
How to Acquire the Values

- Write the code to calculate them.
  - Usually not possible. If you can do this for your problem, why are you considering machine learning? Might be able to do this for Tic-Tac-Toe.

- Use dynamic programming.
  - Usually not possible. Requires knowledge of the probabilities of transitions between all states for all actions.

- **Learn from examples**, lots of examples. Lots of 5-tuples: state, action, reinforcement, next state, next action \((s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})\).

  - **Monte Carlo**: Assign to each state-action pair an average of the observed returns.
    \[
    \text{value}(s_t, a_t) \approx \text{mean of } R(s_t, a_t)
    \]

  - **Temporal Difference (TD)**: Using value \((s_{t+1}, a_{t+1})\) as estimate of return from next state, update current state-action value as
    \[
    \text{value}(s_t, a_t) \approx r_{t+1} + \text{value}(s_{t+1}, a_{t+1})
    \]
When would TD be better?

- What is estimate of the return $R$ from state B?

Examples:
1: A C L
2: A C L
... 
100: A C L
101: A C W
102: B C W
When would TD be better?

- What is estimate of the return $R$ from state B?

- Monte Carlo: mean of $R(s_t, a_t) = ?$

Examples:

1: A C L
2: A C L
   .
   .
100: A C L
101: A C W
102: B C W

Monte Carlo: mean of $R(s_t, a_t) = ?$
When would TD be better?

- What is estimate of the return $R$ from state B?

  Monte Carlo: mean of $R(s_t, a_t) = ?$

  1, a prediction of a win
When would TD be better?

- What is estimate of the return $R$ from state B?

  - Monte Carlo: mean of $R(s_t, a_t) = \ ?$
    1, a prediction of a win
  - Temporal Difference (TD): $r_{t+1} + \text{value}(s_{t+1}, a_{t+1}) = \ ?$

Examples:

1: A C L
2: A C L
   ...
100: A C L
101: A C W
102: B C W
When would TD be better?

- **What is estimate of the return $R$ from state B?**

  - Monte Carlo: mean of $R(s_t, a_t) = ?$
    
    1, a prediction of a win

  - Temporal Difference (TD): $r_{t+1} + \text{value}(s_{t+1}, a_{t+1}) = ?$
    
    $0 + (100(-1) + 1(1))/100 = -0.99$, a very likely loss

![Diagram of a maze with states A, B, C, and W, L, showing transitions and rewards.]

**Examples:**

1: A C L
2: A C L
.
.
.
100: A C L
101: A C W
102: B C W

What do you do? The green pill or the red pill?

TD takes advantage of the cached experience given in the value learned for State C.
When would TD be better?

- What is estimate of the return $R$ from state B?
  - Monte Carlo: mean of $R(s_t, a_t) = ?$
    - 1, a prediction of a win
  - Temporal Difference (TD): $r_{t+1} + \text{value}(s_{t+1}, a_{t+1}) = ?$
    - $0 + (100(-1) + 1(1))/100 = -0.99$, a very likely loss
  - What do you do? The green pill or the red pill?
When would TD be better?

- What is estimate of the return $R$ from state B?

- Monte Carlo: mean of $R(s_t, a_t) =$ ?
  
  - 1, a prediction of a win

- Temporal Difference (TD): $r_{t+1} + \text{value}(s_{t+1}, a_{t+1}) =$ ?
  
  - $0 + (100(-1) + 1(1))/100 = -0.99$, a very likely loss

- What do you do? The green pill or the red pill?

- TD takes advantage of the cached experience given in the value learned for State C.

Examples:

1: A C L
2: A C L
   ...
100: A C L
101: A C W
102: B C W
Outline

Concepts
States and Actions
Values

Example: Maze
Maze Example

Here is a simple maze.

From any position, how do you decide whether to move up, right, down, or left?

Right. Need an estimate of the number of steps to reach the goal. This will be the return $R$. How to formulate this in terms of reinforcements?Yep.

$r_t = 1$ for every step. Then $R_t = \sum_{k=0}^{K} r_t + k + 1$ will sum of those 1's to produce the number of steps to goal from each state.

Monte-carlo way will assign value as average of number of steps to goal from each starting state tried. TD will update value based on $1 + \text{estimated value from next state}$. 

G
Maze Example

Here is a simple maze.
From any position, how do you decide whether to move up, right, down, or left?
Maze Example

- Here is a simple maze.
- From any position, how do you decide whether to move up, right, down, or left?
- Right. Need an estimate of the number of steps to reach the goal. This will be the return $R$. How to formulate this in terms of reinforcements?
Maze Example

Here is a simple maze.

From any position, how do you decide whether to move up, right, down, or left?

Right. Need an estimate of the number of steps to reach the goal. This will be the return $R$. How to formulate this in terms of reinforcements?

Yep. $r_t = 1$ for every step. Then $R_t = \sum_{k=0}^{K} r_{t+k+1}$ will sum of those 1’s to produce the number of steps to goal from each state.
Maze Example

- Here is a simple maze.
- From any position, how do you decide whether to move up, right, down, or left?
- Right. Need an estimate of the number of steps to reach the goal. This will be the return $R$. How to formulate this in terms of reinforcements?
- Yep. $r_t = 1$ for every step. Then $R_t = \sum_{k=0}^{K} r_{t+k+1}$ will sum of those 1’s to produce the number of steps to goal from each state.
- Monte-carlo way will assign value as average of number of steps to goal from each starting state tried.
Maze Example

- Here is a simple maze.
- From any position, how do you decide whether to move up, right, down, or left?
- Right. Need an estimate of the number of steps to reach the goal. This will be the return $R$. How to formulate this in terms of reinforcements?
- Yep. $r_t = 1$ for every step. Then $R_t = \sum_{k=0}^{K} r_{t+k+1}$ will sum of those 1’s to produce the number of steps to goal from each state.
- Monte-carlo way will assign value as average of number of steps to goal from each starting state tried.
- TD will update value based on $1 +$ estimated value from next state.
How shall we store the values?

- Can only be at discrete positions, and only 4 actions. 
- So make a table of values. How many dimensions to this table?
- Need dimensions for \( m < -10 \) and \( n < -10 \).
- State is two-dimensional. Actions, up, right, down, left, will be stored as changes to \( x \) and \( y \).
- \( Q \) is an array \((0, c(m,n,4))\).
- \( \text{actions} \) is \( rbind(\{0,1\}, \{1,0\}, \{0,-1\}, \{-1,0\}) \).
- To choose best action for state \((x, y)\): 
  \[ a \leftarrow \text{which.min}(Q[x,y]) \]
  \[ \text{act} \leftarrow \text{actions}[a,] \]
- How shall we store the values?
- Can only be at discrete positions, and only 4 actions. So make a table of values. How many dimensions to this table?
How shall we store the values?

Can only be at discrete positions, and only 4 actions. So make a table of values. How many dimensions to this table?

Need dimensions for \( x \), \( y \), and action. State is two-dimensional. Actions, up, right, down, left, will be stored as changes to \( x \) and \( y \).

\[
m \leftarrow 10 \\
n \leftarrow 10 \\
Q \leftarrow \text{array}(0,\text{c}(m,n,4)) \\
\text{actions} \leftarrow \text{rbind(} \text{c}(0,1), \text{c}(1,0), \text{c}(0,-1), \text{c}(-1,0))
\]
How shall we store the values?

Can only be at discrete positions, and only 4 actions. So make a table of values. How many dimensions to this table?

Need dimensions for $x$, $y$, and action. State is two-dimensional. Actions, up, right, down, left, will be stored as changes to $x$ and $y$.

```r
m <- 10
n <- 10
Q <- array(0,c(m,n,4))
actions <- rbind(c(0,1),c(1,0),c(0,-1),c(-1,0))
```

To choose best action for state $(x, y)$

```r
a <- which.min(Q[x,y,])
act <- actions[a,]
```
TD versus Monte Carlo

- **Q Policy**
- **Monte Carlo Q Policy**
- **Most Recent Trial**

**Example: Maze**

- TD Min 0 Max 17.2
- Monte Carlo Min 0 Max 17.2

**Steps to Goal**

- TD:
  - 0.0 0.2 0.4 0.6 0.8 1.0
- Monte Carlo:
  - 0.0 0.2 0.4 0.6 0.8 1.0

**Trial**

- TD:
  - 0 10000 20000 30000 40000 50000
- Monte Carlo:
  - 0 100 200 300 400 500 600