**Concepts**

- 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?

**Outline**

- Concepts
- States and Actions
- Values
- Example: Maze

**Example: Maze**

- Guess at how likely to win. definite, likely, maybe

**Which are you most likely to win from?**
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

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.

reinforcements

\[ r_{0.2} \rightarrow r_{0.3} \rightarrow r_{0.1} \rightarrow r_{0.2} \rightarrow r_{0.1} \]

Returns

- 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 $s_t$?

\[ R_t = \sum_{k=0}^{K} r_{t+k+1} \]

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 rearrangements 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).

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 $a_3$?

- $a_3$
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: 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
    \[
    \text{value}(s_t, a_t) \approx r_{t+1} + \text{value}(s_{t+1}, a_{t+1})
    \]

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 \(1 + \text{estimated value from next state}\).
- 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,]
```