**1/31/18**

**11-MDPS.key**

**Value Iterations: Specific alg we focus on**

**Grid World Idea**

* agent moving through simulated environment
* has rewards and dangers
  + positive + negative rewards
* no min player in this environment
* state space is kind of like a maze
  + can try to move
    - fails when it is either on the edge of the space -> wall
    - or the environment causes the player’s move to fail
  + attempts to move = **actions**
    - actions are non-deterministic
  + rule:
    - if you try to move into a wall you can’t go
    - even if there is space in front of you, there is noise so there is a prob that something will happen
      * ex.: 10% prob that when you go straight you left.
* agent gets a small reward each time step
  + staying alive
  + big rewards come at the end (good or bad)
* goal: maximize sum of rewards

**Grid World Actions**

* general case: can end up in any state
* stochastic grid world: end up in 3 options (go straight, chance to go right, chance to go left)

**Markov Decision Processes**

* mathematical structure with different components
* An MDP is defined by
  + a **set of states s in S**
  + a **set of actions a in A**
  + a **transition function T(s, a, s’)**
    - probability that **a from s leads to s’**
      * P(s’ | s, a)
    - also called the model or the dynamics
    - can represent as a giant table of values (each tile has different probability to arrive in)
  + A **reward function R(s, a, s’)**
    - can be represented as a giant table of values (each tile has a different reward amount)
    - sometimes can be represented as just R(s) or R(s’)
* MDPs are non-deterministic search problems
  + One way to solve them is with expectimax search

**Markov**

* Markov generally means that given the present state, future and past are independent
* For MDPs:
  + Markov means action outcomes depend only on the current state
  + This is just like search, where the successor function could only depend on the current state (not the history)

**Policies**

* a function that maps the state space to the action space
* For MDPs, we want an optimal policy
  + π\*: S → A
  + \* is the optimal (best) action - action that maximizes the expected utility
    - utility: sum of discounted future rewards
    - An optimal policy is one that maximizes expected utility if followed
* A policy π gives an action for each state
* An explicit policy defines a reflex agent