**Artificial Intelligence in Games**

**Session 6**

1. **Reinforcement Learning**:
   1. Goals are defined through reward mechanisms.  
        
      Fig:  
      Agent performs an action A\_{t} in an Environment which sends back a reward R\_{t} and the state S\_{t} back to the agent. This continues in a loop
   2. An agent interacts with an environment during a sequence of discrete time steps
   3. At each time step t >= 0, the agent receives some representation of the state s\_{t} contained within S
   4. The agent then selects an action a\_{t} contained within A(s\_{t})
   5. One time step later, the agent receives a reward r{t+1} contained within R, and a new state s\_{t+1} contained within S
   6. A policy pi: S × A → [0, 1] is a function such that pi(s, a) represents the probability that a\_{t} = a given that s\_{t} = s
   7. The discounted return u\_{t} after time step t is given by:  
        
      u\_{t} = summation of gamma^{k} \* r\_{t+1+k}  
        
      where,  
      0 <= gamma < 1 is the discount factor
   8. A reward received k time steps into the future is only worth gamma^{k−1} times what it would be worth if it were received on the next step
   9. If necessary, a state can transition only to itself and yield no rewards
   10. The objective of the agent is to maximize the discounted return
   11. Example: grid world
       1. A 3x4 grid as

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | +1 |
|  | 0 |  | -1 |
| Start |  | Robot can go left, right, up on the grid |  |

* + 1. States S = {1, 2, . . . , 12}, actions A = {1, 2, 3, 4}
    2. Reward +1 on action at goal, reward −1 on action at trap, and reward 0 on action at other states
    3. Actions at goal and trap transition to an absorbing state
    4. Discount factor γ = 0.9
  1. Practical: logistics, finance, and marketing
  2. Theoretical: every task with a computable description can be formulated as a reinforcement learning problem [Hutter, 2004]
  3. Curse of generality: every well-defined problem is a reinforcement learning problem, but most reinforcement learning problems cannot be solved efficiently

1. **Markov Decision Process**:
   1. A state that summarizes the entire past with all that is relevant for decision making has the Markov property
   2. In a Markov decision process, for any sequence of states, action and rewards (s\_{t}, a\_{t}, r\_{t}, …, r1, s0, a0) (history) and all s’ contained in S, r’ contained in R,  
        
      P(S\_{t+1} = s’, R\_{t+1}= r’ given s\_{t}, a\_{t}, r\_{t}, …, r1, s0, a0)) = P(S\_{t+1} = s’, R\_{t+1}= r’ given s\_{t}, a\_{t}).
   3. The conditional joint probability distribution over states and rewards on the right side defines the one-step dynamics of the problem
2. **Environment model:**
   1. Probability distribution over next state given each state and action
   2. Expected reward given each state and action.
   3. P^{a}{\_{ss’}}
      1. The probability P^{a}{\_{ss’}} of transitioning from state s to s’ given action a is given by  
           
         P^{a}{\_{ss’}} = P(S\_{t+1} = s’ given S\_{t}=s, A\_{t}=a) = summation of P(S\_{t+1} = s’, R\_{t+1} = r’ given S\_{t}=s, A\_{t}=a)
      2. Note that P^{a}{\_{ss’}} is independent of the current time step
   4. R^{a}{\_{ss’}}
      1. The expected reward on transitioning from state s to state s 0 given the action a is given by  
           
         R^{a}{\_{ss’}} = Expectation of (R\_{t+1} given S\_{t}=s, A\_{t}=a, S\_{t+1} = s’)  
           
         = (1/ P^{a}{\_{ss’}}) \* (summation of (r’ \* (S\_{t+1} = s’, R\_{t+1} = r’ given S\_{t}=s, A\_{t}=a)))
      2. Note that R^{a}{\_{ss’}} is independent of the current time step
3. **Value function** V^{pi}:
   1. The value Vpi(s) of a state s contained within S is the expected (discounted) return of starting in state s and following the policy pi:  
        
      V^{pi} = Expectation\_{pi} (U\_t given S\_t=s) = Expectation\_{pi} (sum of gamma^{k} \* R\_{t+1+k} given S\_t=s) – Equation 1
4. **Action value function** Q^{pi}:
   1. Q^{pi}(s, a) = Expectation\_{pi} (U\_t given S\_t=s, A\_t=a) = Expectation\_{pi} (sum of gamma^{k} \* R\_{t+1+k} given S\_t=s, A\_t=a)
5. **Recursivity of the value function:**
   1. For any policy pi and state s contained within S,  
        
      V^{pi}(s) = (sum of pi(s,a)) \* (P^{a}{\_{ss’}} \* [R^{a}{\_{ss’}} + gamma\*V^{pi}(s’)]) – Equation 2
6. **Notation:**
   1. We denote random variables by upper case letters and assignments to these variables by corresponding lower case letters
   2. We omit the subscript that typically relates a probability function to random variables when there is no risk of ambiguity
   3. For example, let X and Y be discrete random variables. In the same context, we will let p(x given y) denote P(X = x given Y = y) and p(y given x) denote P(Y = y given X = x)
   4. This notation where the arguments select between different probability functions is standard in machine learning
7. **Recursivity of the value function**:  
     
   Note that Equation 1 can be rewritten as:  
     
   V^{pi}(s) = As T approaches infinity, the (sum of p(r\_{t+1: T+t+1} given S\_t=s)) \* (gamma^{k} \* R\_{t+1+k} given S\_t=s)),  
     
   where the dependency on the policy pi becomes implicit. By marginalization,  
     
   V^{pi}(s) = As T approaches infinity, the sum of[ (double sum of p(r\_{t+1: T+t+1, a\_t, s\_{t+1}} given S\_t=s) \* (sum of (gamma^{k} \* R\_{t+1+k} given S\_t=s) ]  
     
   By the chain rule of probability,  
     
   V^{pi}(s) = As T approaches infinity, the sum of[ (double sum of p(r\_{a\_t given S\_{t}=s} given S\_t=s \* p(r\_{s\_t+1 given S\_t = a, a\_t \* p(r\_{t+1: T+t+1, a\_t, s\_{t+1}} \* (gamma^{k} \* R\_{t+1+k} given S\_t=s )) ]  
     
   By the distributive property and reordering the three outermost summations:  
     
   V^{pi} (s)= (sum of p(a\_t given S\_t=s)) \* (sum of p(s\_t+1 given S\_t=s, a\_t)) \*(sum of p(r\_{t+1: T+t+1} given S\_t= s, a\_t, s\_{t+1}) \*( gamma^{k} \* R\_{t+1+k} given S\_t=s) – Equation 3  
     
   Let E denote the limit in the previous equation, such that  
     
   E = as T approaches infinity, the sum of (p(r\_{t+1: T+t+1} given S\_t= s, a\_t, s\_{t+1}) \*(gamma^{k} \* R\_{t+1+k} given S\_t=s).  
     
   By isolating the first term in the innermost summation,  
     
   E = as T approaches infinity, the sum of (p(r\_{t+1: T+t+1} given S\_t= s, a\_t, s\_{t+1}) \* ((r\_t+1) \* (gamma^{k} \* R\_{t+1+k}))   
     
   By the linearity of expectation, E = E1 + E2, where  
     
   E\_1 = as T approaches infinity the sum of (p(r\_{t+1: T+t+1} given S\_t= s, a\_t, s\_{t+1}) \* (r\_t+1))  
     
   = E\_{pi} \* (R\_{t+1} given S\_{t}=s, a\_t, S\_{t+1}) = R^{a}{\_{ss’}}  
     
   and,  
     
   E\_2 = as T approaches infinity, the sum of (p(r\_{t+1: T+t+1} given S\_t= s, a\_t, s\_{t+1}) \* (gamma^{k} \* R\_{t+1+k} )  
     
   By changing the indices in the innermost summation,  
     
   E\_2 = as T approaches infinity, the sum of (p(r\_{t+1: T+t+1} given S\_t= s, a\_t, s\_{t+1}) \* (gamma^{k+1} \* R\_{t+2+k} )  
     
   By moving a constant factor of gamma outside of the innermost summation,  
     
   E\_2 = (gamma) \* (as T approaches infinity, the sum of (p(r\_{t+1: T+t+1} given S\_t= s, a\_t, s\_{t+1}) \* (gamma^{k} \* R\_{t+2+k} ))  
     
   Because (R\_{t+2: T+t+1 conditionally independent by S\_t, A\_t given S\_t+1}) due to the Markov property,  
     
   E\_2 = gamma \* V^{pi}(s\_t+1)  
     
   Returning to Equation 3,  
     
   By making the dependency on pi explicit and renaming variables,  
     
   V^{pi}(s) = (sum of pi(s, a)) \* (sum of P^{a}{\_{ss’} \* [R^{a}{\_{ss’}} + gamma \* V^{pi}(s’)])  
     
     
     
     
     
   Theorem:  
     
   For any policy pi, state s contained within S, and action a contained within S,  
     
   Q^{pi}(s, a) = sum of(P^{a}{\_{ss’} \* ( R^{a}{\_{ss’}} + gamma \* sum of pi(s’,a’) \* Q^{pi}(s’, a’)))  
     
   Relationship between V^pi and Q^pi:  
     
   V^{pi}(s) = sum of(pi(s, a) \* Q^{pi}(s, a))  
     
   Q^{pi}(s, a) = sum of(P^{a}{\_{ss’} \* ( R^{a}{\_{ss’}} + gamma \* V(s’))
8. **Optimal policies:** 
   1. Let pi >= pi’ if and only if V^{pi}(s) >= V^{pi}(s’) for all s contained within S
   2. A policy pi∗ is optimal if pi\* >= pi for any policy pi
   3. An optimal policy always exists, but is not necessarily unique
   4. Theorem (Bellman optimality equations)
      1. For any action a contained within A and state s contained within S, the optimal state value function V\* and the optimal action value function Q\* are given by:  
           
         V\*(s) = max of the sum of (P^{a}{\_{ss’} \* ( R^{a}{\_{ss’}} + gamma \* V(s’)),  
           
         Q\*(s, a) = sum of (P^{a}{\_{ss’} \* ( R^{a}{\_{ss’}} + gamma \* max of Q\*(s’,a’)),
9. **Reinforcement learning algorithms:** 
   1. Reinforcement learning algorithms aim to find an optimal policy pi\* for a given environment
   2. For any state s contained within S, an optimal policy pi\* can be found given either V\* or Q\*
   3. In the case of Q\* , for any s contained within S, it suffices to choose an a such that Q\*(s, a) is maximal
   4. In the case of V\*, for any s contained within S, it suffices to choose one of the actions a that maximizes the right hand side of the Bellman optimality equation.