Ch. 20 - Reinforcement Learning

Reinforcement Learning

- Passive learning
  - known environment
  - unknown environment
- Active learning
  - unknown environment
- Exploration
- Action-Value function
- Genetic algorithms and evolutionary programming*
Reinforcement Learning

- Agent receives:
  - no examples
  - no environment model
  - no utility function
- Uses feedback (reinforcement)
  - use rewards to learn successful agent function
  - never told correct actions
- Passive vs. active learners:
  - passive - watches to learn utility function
  - active - acts using learned information
  - suggests explorations of unknown portions of environment

Passive Learning in a Known Environment

- Environment generates state transitions
  - agent perceives them
- Provided with model $M_{ij}$ - probability of transition from state $i$ to state $j$
- Uses information about rewards to learn utility function $U(I)$ for each state
- Simplifying assumption - additive utility function
  - utility of sequence = sum of rewards in sequence
Passive Learning Example

Agent trying to learn utilities of terminal states (4,3) and (4,2)

Transitions among adjacent states

Exact utility values

Naïve Updating

- Least mean squares (LMS) approach
- Assumes:
  - observed reward-to-go provides direct evidence of actual expected reward-to-go
- Calculate observed reward-to-go for each state
  - updates estimated utility for each state
- Learning utility function directly from examples
  - reduced reinforcement learning to standard inductive learning (Ch 18)
- Misses important fact:
  - utilities of states are not independent
  - converges very slowly to correct utility values
Adaptive Dynamic Programming

- Solving utility equations with DP algorithm
- After agent has observed rewards for all states
  - compute utilities by solving set of equations:
    \[ U(i) = R(i) + \sum_j M_{ij} U(j) \]
    - \( R(i) \) - reward associated with state \( i \)
    - \( M_{ij} \) - probability that transition will occur from state \( i \) to state \( j \)
- Provides standard to measure reinforcement algorithms
- Intractable in large state spaces

Temporal Difference Learning

- Approximate constraint equations
  - without solving for all possible states
  - adjust values of observed states to agree with constraints
- Updating rule (temporal difference equation)
  - \( U(i) \leftarrow U(i) + \alpha (R(i) + U(j) - U(i)) \)
  - \( \alpha \) - learning rate parameter
- Define conditions that hold locally when utility estimates are correct
- Write update equation that moves estimates towards equilibrium equation (previous slide)
Passive Learning in an Unknown Environment

- LMS and TD will work the same in an unknown environment - don’t use environment model
- ADP will change
  - adds a step to update estimated model of environment
  - used a basis for utility estimates
  - as environment model approaches correct model, utility estimates converge on correct utilities
- Env. Model learned by observation of transitions

Active Learning in an Unknown Environment

- Active agent must consider what actions to take
- Env. Model must consider probabilities of transitions given certain actions:
  \[ M_{ij}^a \]
- Update utility equation:
  \[ U(i) = R(i) + \max_a \sum_j M_{ij}^a U(j) \]
- Needs performance element to choose action at each step
Active ADP Agent

function ACTIVE-ADP-AGENT(e) returns an action
    static: $U$, a table of utility estimates
    $M$, a table of transition probabilities from state to state for each action
    $R$, a table of rewards for states
    $\text{percepts}$, a percept sequence (initially empty)
    last-action, the action just executed

    add $e$ to $\text{percepts}$
    $R[SATE][e] \leftarrow \text{REDUCE}[e]$
    $M \leftarrow \text{UPDATE-ACTIVE-MODEL}(M, \text{percepts}, \text{last-action})$
    $U \leftarrow \text{VALUE-ITERATION}(U, M, R)$
    if TERMINAL[$e$] then
        $\text{percepts} \leftarrow$ the empty sequence
        last-action $\leftarrow \text{PERFORMANCE-ELEMENT}(e)$
    return last-action

Exploration

- What actions should the agent take?
- Each action has two outcomes:
  - gains rewards on current sequence
  - affects percepts received and ability to learn
- Must make trade-off
  - immediate good - current utility estimates
  - long-term well-being
- Don’t want to get stuck in a rut - maximizing rewards on current sequence
- Decide between staying comfortable and exploring to learn more
Example

Moves:
- North, South, East, West
- Prob(action works)=0.8

Rewards:
- -0.04 doesn’t reach terminal state

Two approaches:
- "wacky" - acts randomly
  • eventually explore entire environment
- "greedy" - maximize utility using current estimates

Results:
- "wacky" - learns good utility for all states
  • but never uses the estimates to receive the ultimate reward
- "greedy" - often finds path to terminal state
  • but only sticks to that path
  • never explores for a possibly better path

Exploration Function

\[ U^+(i) \leftarrow R(i) + \max_a f \left( \sum_j M_{ij} U^+(j), N(a,i) \right) \]

- \( U^+(i) \) = optimistic estimate of utility
- \( N(a,i) \) = number of times action a has been tried in state i
- \( f(u,n) \) = exploration function
  - determines how greed is traded off against curiosity
  - ex: \( f(u,n) = \begin{cases} R^+ & \text{if } n < N_e \\ u & \text{otherwise} \end{cases} \)
  - \( R^+ \) - optimistic estimate of best reward
  - \( N_e \) - try each state at least Ne times
Genetic Algorithms and Evolutionary Programming

- Student presentation