Math Modeling, Week 2

Full RL problem
Learning to predict reward
Multiple actions
State dynamics

Multiple actions
Decision-making
Prediction for each action, $P(A)$
Action-selection rules
Max selection: $a = \text{argmax}_A \{P(A)\}$
Exploration/exploitation dilemma
    Use predictions, but give all actions a chance
Luce choice: $\Pr[a=A] \propto P(A)$
Problem: negative values, interval scale
Monotonic transform: $\Pr[a=A] \propto f(P(A))$
Softmax: $\Pr[a=A] \propto e^{P(A)/T}$
    Input always positive
    Only differences matter, $P(A) - P(A')$
    Temperature parameter T

State dynamics
Stimuli predict future stimuli, not just immediate rewards
Conditioned reinforcement
Evaluate outcomes based on what they predict
    S_1 followed by $S_2 \rightarrow V(S_1) = R(S_1) + V(S_2)$
    Circular; well-defined?
Episodic tasks
    Tree example
    V anchored on terminal states
Continuing tasks: (effectively) infinite reward sequence
Return: $\mathbf{R} = \Sigma_t R_t \cdot \gamma^t$
Temporal discounting
    Exponential – strong psychological assumption
    Horizon parameter $\gamma \in [0,1]$
Value function: $V(S) = \mathrm{E}[\mathbf{R} | s_0 = S]$
Recursion: $V(s_t) = R_t + \gamma \cdot V(s_{t+1})$

Markov process
State space, transition matrix
    $T(S,S') = \Pr[s_{t+1} = S' \mid s_t = S]$
Also: reward function $R(S)$
Bellman equation: $V(S) = R(S) + \gamma \cdot \Sigma_{S'} T(S,S') V(S')$
Direct solution
    $V = R + \gamma T V \rightarrow V = (\mathrm{I} - \gamma T)^{-1} R$
    Exists if $\gamma < 1$ (eigenvalue argument)
Gridworld MP batch demo

Learning from prediction error

  Prediction is $V(s_t)$

  Outcome is R_t, $s_{t+1} \rightarrow R_t + \gamma V(s_{t+1})$

  $\Delta V(s_t) = \varepsilon[R_t + \gamma V(s_{t+1}) - V(s_t)]$

Gridworld MP incremental demo

Markov Decision Process

Action selection + State dynamics

  Reward $R(S,A)$

  Transitions $T(S,A,S') = \Pr[s_{t+1} = S' \mid s_t = S, a_t = A]$

Tree example, with actions

  Value of state is value of best action

State-action values

  $Q(S,A) = \mathrm{E}[\mathbf{R}|s_0 = S, a_0 = A]$

Reciprocal recursive equations

  $Q(S,A) = R(S,A) + \Sigma_{S'}\, \gamma\cdot\Sigma_{S'} T(S,A,S')V(S')$

  $V(S) = \max_A Q(S,A)$

  Values assuming optimal action in future

Q-learning

  Prediction $Q(s_t,a_t)$

  Outcome R_t, $s_{t+1} \rightarrow R_t + \gamma\cdot\max_A Q(s_{t+1},A)$

  $\Delta Q(s_t,a_t) = \varepsilon[R_t + \gamma\cdot\max_A Q(s_{t+1},A) - Q(s_t,a_t)]$

  Converges to optimal action values

Gridworld MDP simulation

Exercises

1. Show that probability matching is a special case of Luce choice. That is, consider a task with two actions, A and B, exactly one of which is correct on each trial. Probability matching means making a prediction $P(A)$ for the probability that A will be correct, and choosing actions with probabilities $\Pr[a = A] = P(A)$ and $\Pr[a = B] = 1 - P(A)$. (This is what the simulation from last week did.) Assuming a reward of 1 for being correct and 0 for being incorrect, work out the expected rewards for both actions according to P(A), and then derive the action probabilities given by the Luce choice rule. If you're enjoying this, work out the action probabilities for softmax, and for Luce and softmax under different reward values for right/wrong (instead of 1/0).

2. Special cases of state-value learning ($\Delta V(s_t) = \varepsilon[R_t + \gamma V(s_{t+1}) - V(s_t)]$)
(a) What happens when $\gamma = 0$? How does the model compare to the simpler model from last week?
(b) What happens when there's only one state? Write a simplified version of the learning rule for that case. What does the value converge to, i.e. when is it in equilibrium?
(c) For the one-state case, define a new variable $W = (1-\gamma)V$. How does W behave?

3. Think of some ways to make the Q-learner smarter in the Gridworld task. If you can, implement one and try it out.