Rheinische Friedrich-Wilhelms-Universität Bonn
Institut für Informatik
Abteilung VI
Autonome Intelligente Systeme

Prof. Dr. Sven Behnke
Postanschrift: 53012 Bonn
Sitz: Friedrich-Ebert-Allee 144

Robot Learning

Assignment 5

Due Tuesday, June 13th, before class.

5.1)
Develop an implementation of the game BeCareful, a simplified version of Black-jack:

- The game is played with an infinite deck of cards (i.e. cards are sampled with replacement)
- Each draw from the deck results in a value between 1 and 10 (uniformly distributed) with a color of red (probability 1/3) or black (probability 2/3).
- There are no aces or picture (face) cards in this game
- At the start of the game both the player and the dealer draw one black card (fully observed)
- Each turn the player may either stick or hit
- If the player hits then he or she draws another card from the deck
- If the player sticks he or she receives no further cards
- The values of the player's cards are added (black cards) or subtracted (red cards)
- If the player's sum exceeds 21, or becomes less than 1, then she \goes bust" and loses the game (reward -1)
- If the player sticks then the dealer starts taking turns. The dealer always sticks on any sum of 17 or greater, and hits otherwise.
- If the dealer goes bust, then the player wins; otherwise, the outcome { win (reward +1), lose (reward -1), or draw (reward 0) } is the player with the largest sum.

Specifically, write a function $(s', r) = advance(s, a)$, which takes as input a state s (dealer's first card 1–10 and the player's sum 1–21), and an action a (hit or stick), and returns a sample of the next state s' (which may be terminal if the game is finished) and reward $r \in \{1, 0, -1\}$ for winning, draw, and loosing. All non-terminal rewards are zero. There is no discounting ($\gamma = 1$). You should treat the dealer's moves as part of the environment, i.e. calling $advance(s, stick)$ will play out the dealer's cards and return the final reward and terminal state.

6 points

5.2)
Implement Sarsa(λ) for BeCareful. Initialise the value function $Q(s, a)$ to zero. Use a time-varying scalar step-size of $\alpha_t = 1/N(s_t, a_t)$ and an ε-greedy exploration strategy with $\varepsilon_t = N_0/(N_0 + N(s_t))$, where $N_0 = 50$ is a constant, $N(s)$ is the number of times state s has been visited, and $N(s, a)$ is the number of times action a has been selected from state s.
Run the algorithm with parameter values λ ∈ {0, 0.1, 0.2, ..., 1}. Stop exploration and learning after 1000 episodes and plot the accumulated reward for the next 100 episodes against λ.

7 points

Rheinische Friedrich-Wilhelms-Universität Bonn
Institut für Informatik
Abteilung VI
Autonome Intelligente Systeme

5.3)
Use now a simple coarse coding value function approximator that is based on a binary feature vector $\varphi(s, a)$ with 3 ∗ 6 ∗ 2 = 36 features. Each binary feature has a value of 1 iff (s, a) lies within the cuboid of state-space corresponding to that feature, and the action corresponding to that feature. The cuboids have the following overlapping intervals:
dealer(s) = {[1, 4], [4, 7], [7, 10]}
player(s) = {[1, 6], [4, 9], [7, 12], [10, 15], [13, 18], [16, 21]}
a = {hit, stick}
where
• dealer(s) is the value of the dealer's first card (1–10)
• sum(s) is the sum of the player's cards (1–21)
Repeat the Sarsa(λ) experiment from 5.2), but using linear value function approximation $Q(s, a) = \varphi(s, a)^T\theta$. Use a constant exploration of $\varepsilon=0.05$ and a constant step-size of $\alpha=0.01$.
Again, for λ ∈ {0, 0.1, 0.2, ..., 1} learn from 1000 episodes, stop learning and exploration and plot the accumulated reward for the next 100 episodes against λ.

7 points