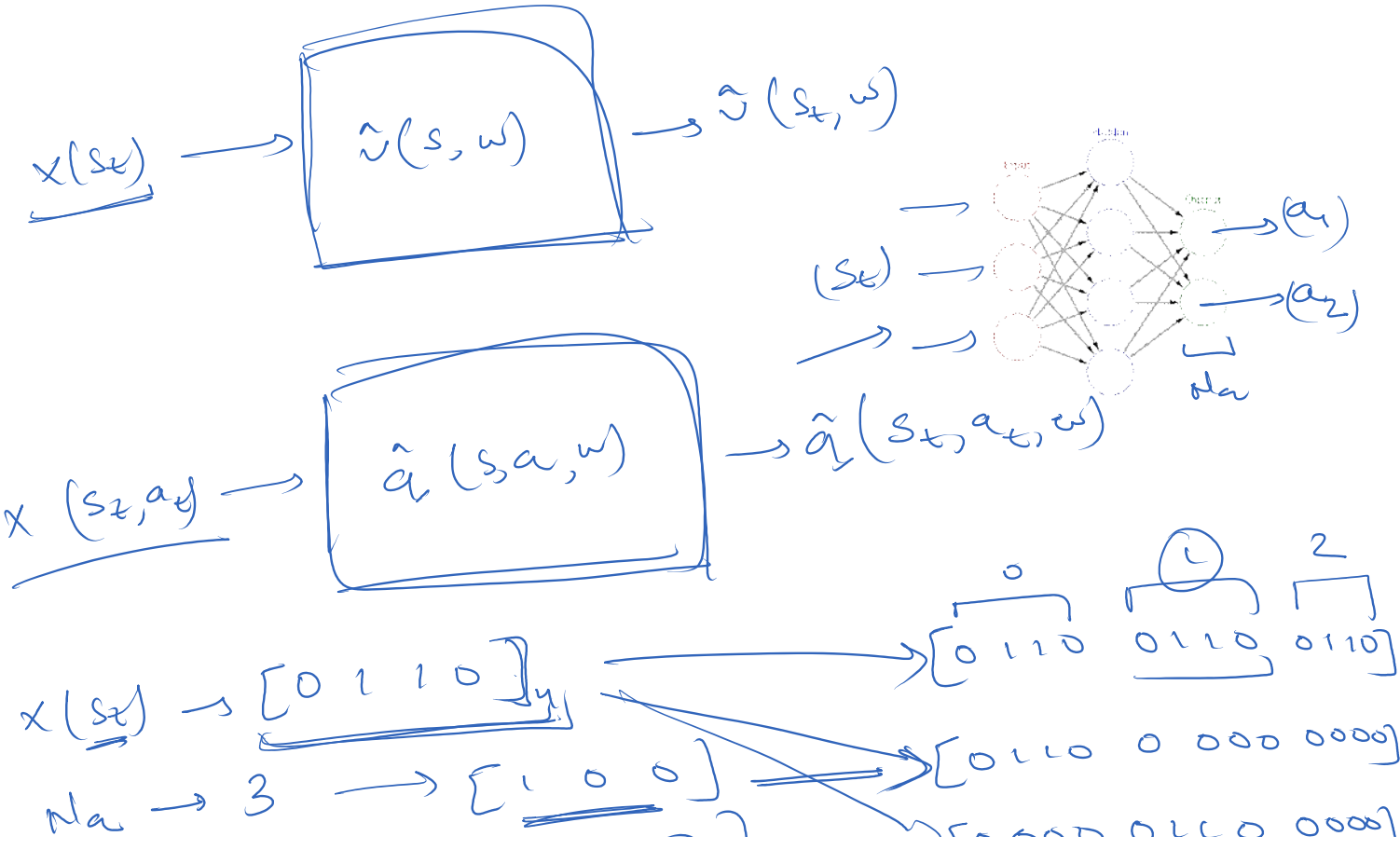
Intro to control using approximation

Sunday, May 24, 2020 12:13 AM

Reinforcement Learning

Introduction to control using approximation

Control using approximation:



General weight update rule for action value function:

$$w_{t+1} = w_t + \alpha[U_t - \hat{q}(S_t, A_t, w_t)]\nabla\hat{q}(S_t, A_t, w_t)$$

For one − step SARSA:

$$w_{t+1} = w_t + \alpha[R_{t+1} + \gamma\hat{q}(S_{t+1}, A_{t+1}, w_t) - \hat{q}(S_t, A_t, w_t)]\nabla\hat{q}(S_t, A_t, w_t)$$

For expected SARSA:

$$w_{t+1} = w_t + \alpha\left[R_{t+1} + \gamma\sum_a \pi(a, S_{t+1})\,\hat{q}(S_{t+1}, a, w_t) - \hat{q}(S_t, A_t, w_t)\right]\nabla\hat{q}(S_t, A_t, w_t)$$

For Q − learning:

$$w_{t+1} = w_t + \alpha\left[R_{t+1} + \gamma\max_a \hat{q}(S_{t+1}, a, w_t) - \hat{q}(S_t, A_t, w_t)\right]\nabla\hat{q}(S_t, A_t, w_t)$$

Episodic Semi-gradient Sarsa for Estimating $\hat{q} \approx q_*$

Input: a differentiable function $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}$

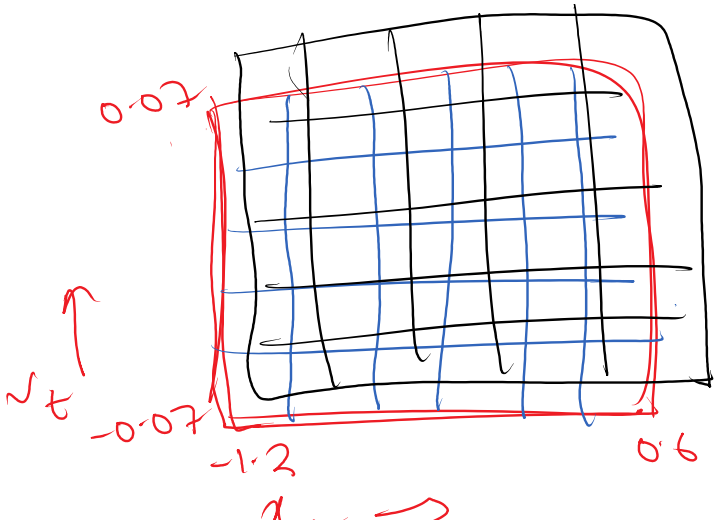
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)
Repeat (for each episode):
  $S, A \leftarrow$ initial state and action of episode (e.g., ε-greedy)
  Repeat (for each step of episode):
    Take action A, observe R, S'
    If S' is terminal:
      $\mathbf{w} \leftarrow \mathbf{w} + \alpha[R - \hat{q}(S, A, \mathbf{w})]\nabla\hat{q}(S, A, \mathbf{w})$
      Go to next episode
    Choose A' as a function of $\hat{q}(S', \cdot, \mathbf{w})$ (e.g., ε-greedy)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha[R + \gamma\hat{q}(S', A', \mathbf{w}) - \hat{q}(S, A, \mathbf{w})]\nabla\hat{q}(S, A, \mathbf{w})$
    $S \leftarrow S'$
    $A \leftarrow A'$

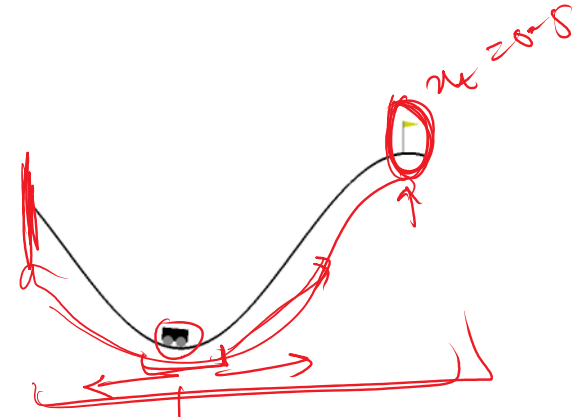
Reinforcement Learning

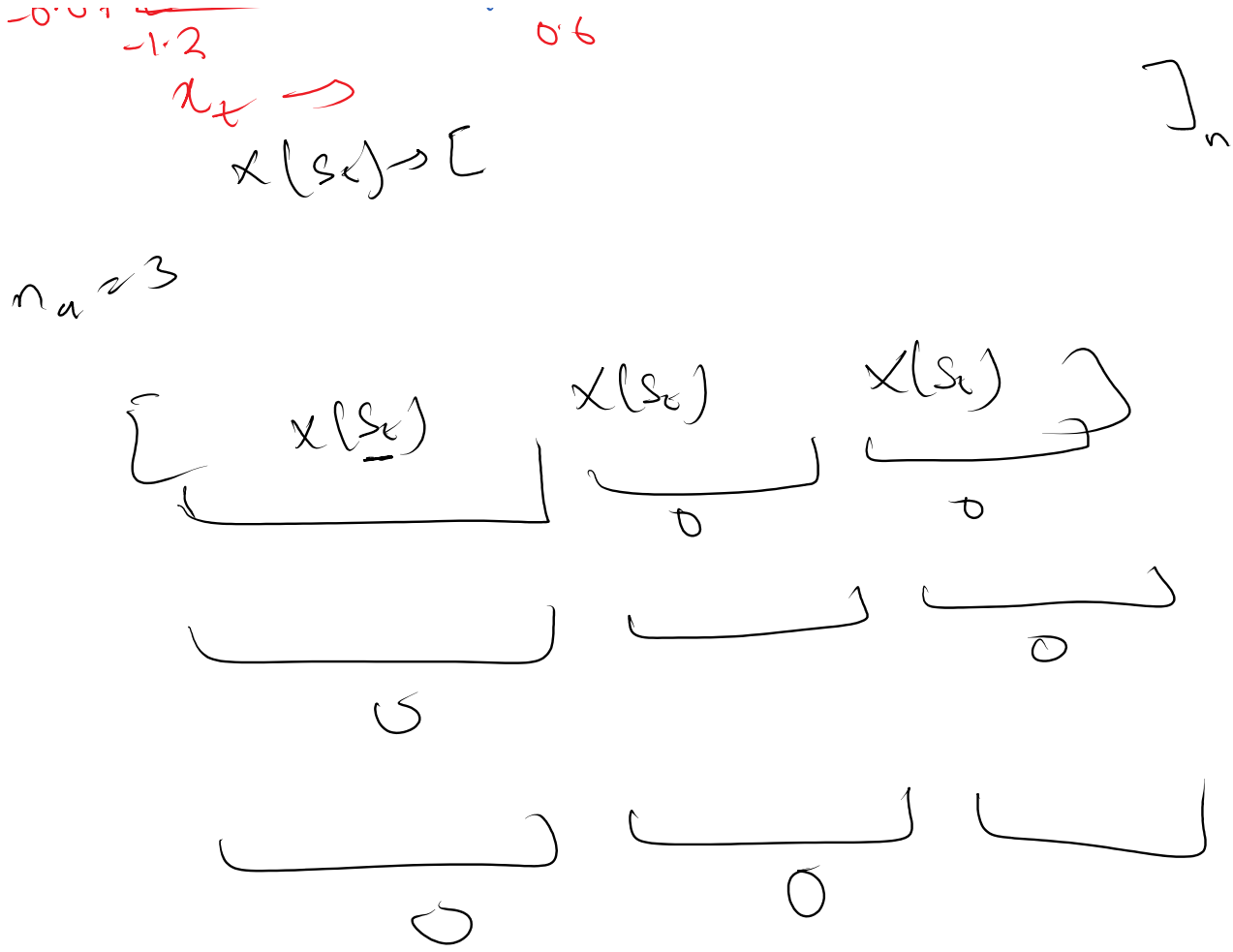
Control using approximation - Mountain Car Example

Mountain Car Example

1. A car is on a one-dimensional track, positioned between two "mountains". The goal is to drive up the mountain on the right; however, the car's engine is not strong enough to scale the mountain in a single pass. Therefore, the only way to succeed is to drive back and forth to build up momentum.
2. State is continuous in two dimensions:
   a. Position from -1.2 to 0.6
   b. Velocity from -0.07 to 0.07
3. Action is discrete:
   a. 0 : Push left
   b. 1 : No push
   c. 2 : Push right
4. Reward is -1 for each time step, until goal position of 0.5 is reached. There is no penalty for climbing the left hill, which upon reached acts as a wall.
5. Episode starts in a random position from -0.6 to -0.4 with no velocity.
6. The episode ends when you reach 0.5 position, or if 200 iterations are reached.






-1.2
0.6
x_t
$x(s_t) \to [$
$]_n$
$n_a = 3$
$x(s_t)$
$x(s_t)$
$x(s_t)$