CS 181 Machine Learning
Practical 4 Report, Team *la Dernière Dame M*

(Jeremiah) Zhe Liu[1], (Vivian) Wenwan Yang[2], and Jing Wen[1]

[1]Department of Biostatistics, Harvard School of Public Health
[2]Department of Computational Science and Engineering, SEAS

May 6, 2015

1 Problem Description

Set in a *Flappy Bird*-type game *Swingy Monkey*, our current learning task is to estimate the optimal policy function $\pi : \mathcal{S} \to \mathcal{A}$ such that the expectation of reward function $R : \mathcal{S} \times \mathcal{A} \to \mathcal{R}$ is maximized, i.e. if define a stochastic process of game state $\{s_t\}$ with unknown transition probability $P(s_{t+1} | s_1, \ldots, s_t, a_1, \ldots, a_t)$, we aim to identify a π^* such that

$$\pi^* = \arg\max_{\pi} E\Big(\sum_{s \in \mathbf{p}} R(s, \pi(s)) \Big| \mathbf{p} \Big)$$

where $\mathbf{p}$ a sample path of $\{S_t\}$.

In current setting, the state and action spaces are defined as:

$$\mathcal{S} = \begin{bmatrix} \text{Tree}_{\text{dist}} & \text{Tree}_{\text{top}} & \text{Tree}_{\text{bot}} & \text{Monkey}_{\text{vel}} & \text{Monkey}_{\text{top}} & \text{Monkey}_{\text{bot}} \end{bmatrix} \subset \mathbb{R}^6$$
$$\mathcal{A} = \begin{bmatrix} \text{NoJump} & \text{Jump} \end{bmatrix}$$

Note theta $\left[\text{Monkey}_{\text{top}}, \text{Monkey}_{\text{bot}}, \text{Tree}_{\text{top}}, \text{Tree}_{\text{bot}}\right]$ are in fact bounded by screen size (600 pxls).

The reward function can be partially described as:

$$R : \quad \begin{bmatrix} \texttt{pass\_tree} \\ \texttt{hit\_trunk} \\ \texttt{hit\_edge} \\ \texttt{otherwise} \end{bmatrix} \to \begin{bmatrix} 1 \\ -5 \\ -10 \\ 0 \end{bmatrix}$$

where $\left[\texttt{pass\_tree}, \texttt{hit\_trunk}, \texttt{hit\_edge}\right]$ are unknown boolean functions of $s \in \mathcal{S}$.

2 Method

2.1 Rationale on Model Choice

In the previous section we identified below characteristics of the task at hand:

(1) Available Information:

   (a) Known $\mathcal{S}$, $\mathcal{A}$ spaces.
   (b) Unknown transition probability $P(s_{t+1}|\{s_i\}_{i=0}^t, \{a_i\}_{i=1}^t)$ and unknown reward function $R : \mathcal{S} \times \mathcal{A} \to \mathcal{R}$

(2) $|\mathcal{A}| = 2$, while $\mathcal{S} \subset \mathbb{R}^6$ is continous with $|\mathcal{S}| = \infty$.

(3) Outcome metric: $E\Big(\sum_{s\in\mathbf{p}} R(s, \pi(s))\Big|\mathbf{p}\Big)$ the expected number of total reward in each play.

If we are willing to assume Markovian property for the process $\{s_t\}$, i.e. $P(s_{t+1}|\{s_i\}_{i=0}^t, \{a_i\}_{i=1}^t) = P(s_{t+1}|s_t, a_t)$, such that the transition information can be described by a transition matrix $\mathbf{P}$, the availabile information listed in (1) put us into a Reinforcement Learning setting. Furthemore, based on information from (2), we are relunctant to deploy model-based approach due to the lareg $\mathcal{S} \times \mathcal{A}$ space. More specifically, since the model-based approach requires reasonably accurate estimation of $\mathbf{P}$ (a $|\mathcal{S}| \times |\mathcal{S}| \times 2$ matrix), we need to visit each nontrivial position of $\mathbf{P}$ sufficiently often in order to achieve reliable $\hat{P}(s'|s, a)$ estimates. Such computation, even after the discreting of the state space, is intractable in terms of both complexity and storage. Although the estimation of $\mathbf{P}$ and $\mathbf{Q}$ can be simplified by assuming $P(s'|s, a) = f(s', s, a)$ and $Q(s, a) = g(s, a)$ and model $f(.), g(.)$ using flexible, kernel-based methods, we decided not to add this extra layer of complexity in order to avoid skewed $\mathbf{P}$ and $\mathbf{Q}$ estimates due to wrong functional assumption.
Based on above consideration, we focused on Q-learning in our implementation, where the estimation task is greatly simplified by considering only the $\mathbf{Q}$ matrix. Specifically, by expanding the Bellman equations, we have:

$$\begin{aligned} Q(s, a) &= R(s, a) + \gamma \sum P(s'|s, a) max_{a'\in\mathcal{A}} Q(s', a') \\ &= R(s, a) + \gamma E_{s'}[max_{a'\in\mathcal{A}} Q(s', a')] \\ &= E_{s'}[R(s, a) + \gamma max_{a'\in\mathcal{A}} Q(s', a')] \end{aligned}$$

and estimation may proceed through Stochastic Gradient Descent/Temporal Difference Learning, where $\hat{Q}_{\text{target}} = r_{s,a}^{obs} + \gamma max_{a'\in\mathcal{A}} Q^{old}(s', a')$. We thus have below updating algorithm:

$$Q(s, a)^{new} \leftarrow Q(s, a)^{old} + \alpha[r + \gamma max'_a Q(s', a')^{old} - Q(s, a)^{old}]$$

where $\gamma \in (0, 1)$ denotes the discount factor, and $\alpha \in (0, 1)$ denotes the learning rate.

2.1.1 Representing $\mathcal{S}$ in low-dimension space

Due to the continous and high-dimensional nature of original $\mathcal{S}$, it is essential to identify $\mathcal{S}_p$ a minimal-information-loss projection of $\mathcal{S}$ in discrete, lower dimensional space. Specifically, we impose below criteria for our projection $s_p \in \mathcal{S}_p$:

1. $\mathcal{S}_p$ contains maximal possible information from $\mathcal{S}$ for the purpose of optimizing $\mathbf{Q}$, i.e. $\inf|Q(s, a) - \hat{Q}(s_p, a)| < \epsilon$
2. $\mathcal{S}_p$ reasonably satisfies markov assumption, i.e. $P(s_{p,t+1}|s_{p,t}, a_t) = P(s_{p,t+1}|\{s_{p,i}\}_{i=1}^t, \{a_{p,i}\}_{i=1}^t)$

Although above criteria are difficult to verify theoretically, they did provide some intuitive guidance in terms of selecting minimal-information-reduction transformation of $\mathcal{S}$.

2.1.2 Exploration/Exploitation Parameters

The learning rate α determining how much of an effect the new sample has on the current estimate. If α is large, we will adjust quickly but may not converge. What we did is to decrease α gradually as the number of samples of $Q(s, a)$ increases.
Learning rate
ϵ-greedy

2.2 Exploration vs Exploitation

The monkey has to determine what action to take even while it is learning. It receives rewards and punishments even as it is learning. The monkey has to spend time to learn the pipes, but we will expect the monkey to start being productive before the monkey has learned everything there is to know about the pipe. The monkey needs to decide what to do considering both the effect of its action on its immediate rewards and future state, and the need to learn for the future. This issue is known as the exploration-exploitation tradeoff. Q-learning has the following two theoretical properties:

i If every state-action pair(s,a) is visited an unbounded number of times and the learning rate α is "eventually small enough" then the Q-values converge to the limit.

ii If we exploit the Q-values in the limit, then the policy converges to the optimal policy in the limit.

In order to achieve these two requirements, we define the distinct learning rate for each state/action pair and have that rate be $\alpha_k(s, a) = 1/k$ where k is the number of times action α has been taken from state s. We adopt the "$\varepsilon - greedy$" policy that the optimal action is taken with probability $1 - \varepsilon$, but with probability ε, a uniformly random action is taken to induce exploration.We took $\varepsilon = 1/t$, where t is the number of time periods that the monkey has experience.

2.3 Performance Evaluation

3 Result

3.1 State Exploration

3.2 Convergence Behavior

4 Discussion & Possible Directions

Reference

1. Ricci F, Rokach L, Shapira B et al. (2010) **Recommender Systems Handbook**. *Springer*.

2. Koren Y, Bell R, Volinsky C. (2009) **Matrix factorization techniques for recommender systems**. *IEEE Computer* Aug 2009, 42-49.

3. Srebro N, Jaakkola T.(2003) **Weighted low-rank approximations**. *Proceedings of the Twentieth International Conference* 720727.

4. R Salakhutdinov, A Mnih. (2008) **Probabilistic Matrix Factorization**. *Advances in Neural Information Processing Systems* Vol. 20

5. Koren, Y. (2008) **Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model**, *Proc. 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.