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 $P(s_{t+1}|s_1, \dots, s_t, a_1, \dots, a_t)$, we aim to identify a π^* such that

$$\pi^* = \arg\max_{\pi} E\Big(\sum_{s \in \mathbf{p}} R(s, \pi(s))\Big|\mathbf{p}\Big) \tag{1}$$

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

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

$$\begin{aligned} \mathcal{S} &= \Big[\mathsf{Tree}_{\text{dist}} \quad \mathsf{Tree}_{\text{top}} \quad \mathsf{Tree}_{\text{bot}} \quad \mathsf{Monkey}_{\text{vel}} \quad \mathsf{Monkey}_{\text{top}} \quad \mathsf{Monkey}_{\text{bot}}\Big] \subset \mathbb{R}^6 \\ \mathcal{A} &= \big[\mathsf{NoJump} \quad \mathsf{Jump}\big] \end{aligned}$$

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

The reward function can be partially described as:

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

where $[\texttt{pass\_tree}, \texttt{hit\_trunk}, \texttt{hit\_edge}]$ 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 $P(s_{t+1} \big| \{s_i\}_{i=0}^{t}, \{a_i\}_{i=1}^{t})$ 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. $P(s_{t+1} \big| \{s_i\}_{i=0}^{t}, \{a_i\}_{i=1}^{t}) = P(s_{t+1} \big| s_t, a_t)$, 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 improper 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:

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

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

$$Q(s, a)^{new} \leftarrow Q(s, a)^{old} + \alpha[r + \gamma max_a' Q(s', a')^{old} - Q(s, a)^{old}] \quad (2)$$

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

2.1.1 Tunning Policy Flexibility: State Space Preprocessing

Dimension Reduction
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 $\mathcal{S}$. Specifically, consider the 6 axises of $\mathcal{S}$:

$$\begin{bmatrix} \mathsf{Tree}_{\text{dist}} & \mathsf{Tree}_{\text{top}} & \mathsf{Tree}_{\text{bot}} & \mathsf{Monkey}_{\text{vel}} & \mathsf{Monkey}_{\text{top}} & \mathsf{Monkey}_{\text{bot}} \end{bmatrix}$$

If assuming the length of $\begin{cases} \text{Tree Gap} & = & \mathsf{Tree}_{\text{top}} - \mathsf{Tree}_{\text{bot}} \\ \text{Monkey Height} & = & \mathsf{Monkey}_{\text{top}} - \mathsf{Monkey}_{\text{bot}} \end{cases}$ are fixed, we may discard (WLOG) all the "top" statistic since they are linearly dependent of all the "bottom" statistics. Furthemore, since the primary goal of the game is passing tree gaps, agent should focus primarily on its relative distance from the tree gap. Thus if denote:

$$\begin{aligned} X &= \mathsf{Tree}_{\text{dist}} && \text{Relative distance, horizontal} \\ Y &= \mathsf{Tree}_{\text{bot}} - \mathsf{Monkey}_{\text{bot}} && \text{Relative distance, vertical} \\ V &= \mathsf{Monkey}_{\text{vel}} && \text{Relative distance, speed of change} \end{aligned}$$

we may define $s_p = (X, Y, V)$ which encodes the maximal possible information on monkey's relative distance from tree gap. s_p also intuitively fits the markov assumption since from Physics the "flying" objects' movement in space depends on its current position, speed, and gravity (which is an constant). Note although $\mathsf{Monkey}_{\text{bot}}$ is linearly independent from s_p, we consider it not crucial in terms of providing information on agent's relative distance from tree gap, and hence decide to not include $\mathsf{Monkey}_{\text{bot}}$ in s_p.

<u>Discretization</u>
Discretization of $\mathcal{S}_p$ is necessary due to the fact that Q-learning operates on a finite-dimensional $\mathbf{Q}$ matrix, which impacts the performance of the agent due to the trade-off between computation complexity and policy flexibility: while refined grid allow flexible policy which detects tiny but crucial changes in original state space, it also generates a huge $\mathcal{S}_p$ which takes long time to explore and converge. On the other hand, coarse grid generates a small $\mathcal{S}_p$ that can be well explored in short time, the algorithm may converge to an inflexible and sub-optimal policy.
To assess the impact of partitioning on agent performance, we fixed the grid size of speed space (V) to be 20, and tried two different grid sizes for the spatial state space (X, Y): 50×50 and 25×25 grid. The 50×50 grid is approximately identical to the size of the monkey, which will result in a slightly coarse but reasonable partition of spatial state space. The 25×25 grid (as shown in Figure 1) is selected based on the fact that the estimated speed of the monkey's horizontal movement is 25 pxl/game loop, which represents the finest possible partition of (at least) X space.

2.1.2 Tunning convergence speed: α, γ and ϵ

Aside from the discretization of $\mathcal{S}$, three native parameters of Q-learning: learning rate (α), discount rate (γ) and the ϵ-greedy factor also impacts algorithm performance in terms of convergence rate. Recall that 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.

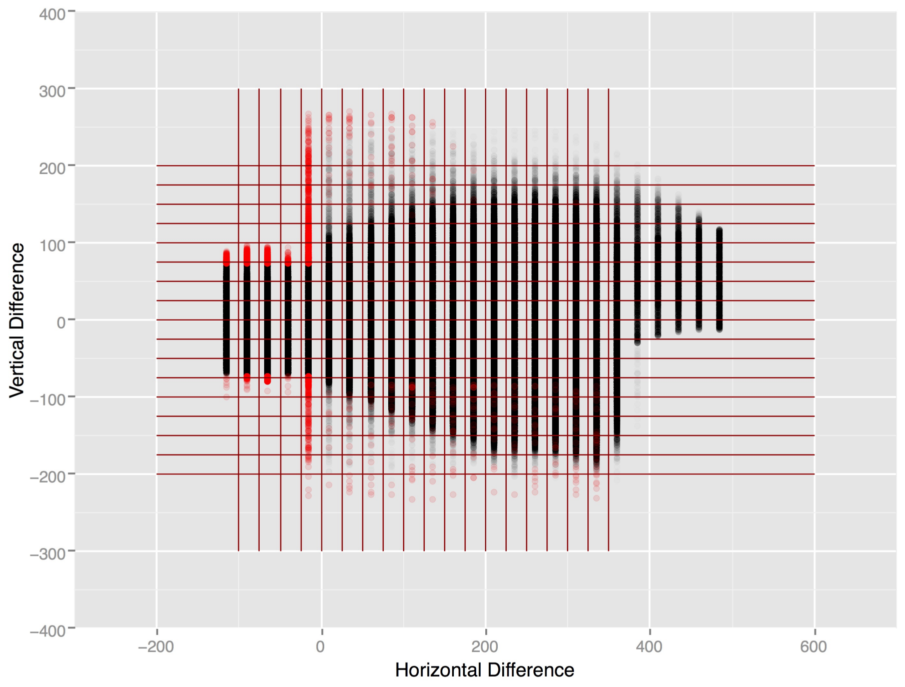


Figure 1: Initial Partition of State Space, 25 × 25 grid

In order to achieve these two requirements, we define the distinct learning rate for each state/action pair and have that rate be $\alpha_k(s,a) = 1/k(s,a)$ where k is the number of times action α has been taken from state s. We also set $\gamma = 0.9$ since the game mechanism is time-stationary and a high discount factor should not "unlearn" previous state-action encounters that quickly. The specification of α and γ are fixed in our implementation since they do not seem to greatly impact agent performance during trial runs.
We also 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. However, in constrast to use $\epsilon = \frac{1}{epoch}$ as introduced in lecture notes, we took $\varepsilon(s,a) = e/k(s,a)$, where e (set at 0.001) is the "base ϵ factor" which affects the initial exploration probability. This exploitation v.s. exploration policy is state-action dependent so that it encourage the agent to explore new actions in new states even if it has already experienced many epoches, we also change e to smaller values due to the fact the agent need to survive long enough in each game run so it can sufficiently explore each game state.

2.2 Performance Evaluation

As shown in (1), our theoretical metric is $E\Big(\sum_{s\in\mathbf{p}} R(s,\pi(s))\Big|\mathbf{p}\Big)$, the expected total reward in a game run. However, as shown in Figure 1 and will be explained Result section, our implemented agent rarely hits screen edge and receives only the tree_hit type penalty. It is thus reasonable to approximate (1) using the aerage total score over repeated game runs. Specifically, as always used in MCMC approach, we evaluate the agents' policy quality and convergence behavior by considering its running average score of the 1000 most recent game runs (denoted as $\overline{\text{Score}}$):

$$\overline{\text{Score}}_i = \frac{1}{1000}\sum_{k=i-1000}^{i}\Big[\sum_{s\in\mathbf{p}_k} R(s,\pi(s))\Big]$$

3 **Result**

3.1 **Initial Run**

The state travelled by agent (black dots) and state when agent fail (red dots) are shown in Figure 1 and 3. As displayed in Figure 1, the agent fail primarily when hitting the top and lower trunk either before or during passing the tree gaps. We also see in Figure 3 (Right) that the agent fail mostly when their vertical speed is too high/low, indicating lack of speed control possibly when close to tree. We also see that our partition of speed space is too coarse with respect to the actual speed distribution.

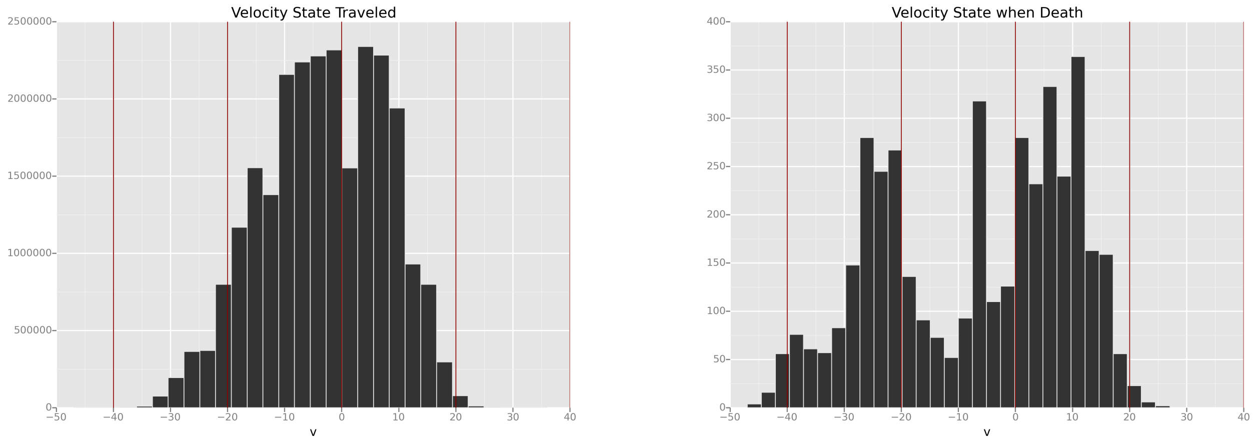


Figure 2: Initial Partition of Speed State Space
Left Distribution of all speed state travelled
Right Distribution of speed state when death

3.2 **Second Run with Improved Grid**

In order to improve our discretization of the state-space, we analysed the distribution of states that agent had travelled during initial runs. Specifically, we calculated the multidimensional quantiles of the (X, Y, V) space. As shown in Figure 3, while the qantiles distributed evenly over X range, quantiles are closer to each other when approaching 0 for both Y and V, indicating higher frequency of these states, and potentially higher importance of decision made within such states. In light of this situation, we refined our grid size along Y and V axis by reducing spatial grid size to 25×10, and reducing speed grid size to 5.

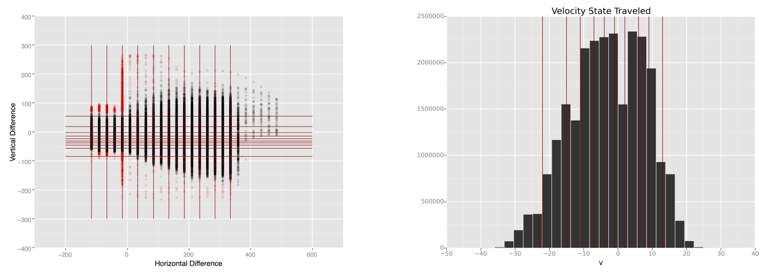
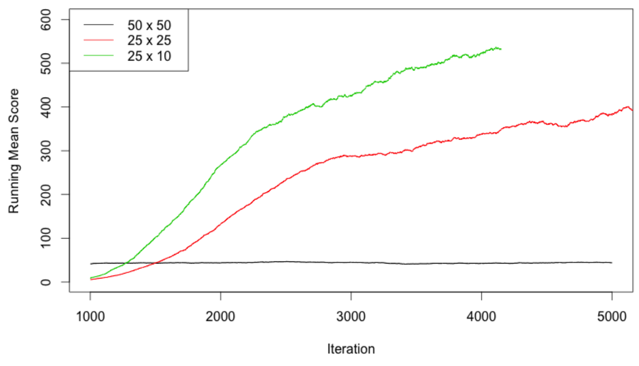


Figure 3: Analysis and Improved Partition of State Space
Left Empirical joint quantile of spatial state space
Right Empirical joint quantile of speed state space

3.3 Convergence Behavior and Resulting Policy

The running $\overline{\text{Score}}$ for three grid sizes used are displayed as in Figure 4(Left). As shown in , while coarse grid size (50×50) converged quickly before 1000 iterations, it stucked at a consistently lower range. Comparatively, finer grid size shows slower convergence, with 25×25 grid stabled at around 375.2 after 5000 iterations, while 25×10 grid still shows improvement. The observed score per play of 25×10 grid size is shown on right. As shown, we can observe extremely large score variation between plays, with high score at around 3900 and low score at 0 even when mean score is at around 500.



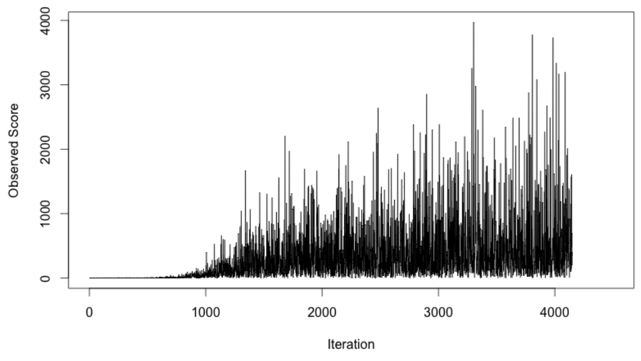


Figure 4: Observed (**Left**) and Running Mean (**Right**) of scores per play
Right Observed Score of Improved Grid
Left Running Mean of Initial and Improved Grid

Finally, the policy generated by the 25×10 grid agent is shown in Figure 5, with hue level (more red) indicating higher chance of pushing "Jump". As shown, the agent did learn a reasonable policy that coincides with human intuition. It tends to jump when lower than tree gap, do not jump when far from tree and at the same horizontal level as tree gaps, and jump occasionally during or right after passing tree gap so it keeps flying. We are not able to explain why the agent still tends to jump when it is close to tree and higher than tree trunk, which is possibly the result of insufficient experience (hence not yet converged Q values) in these states.

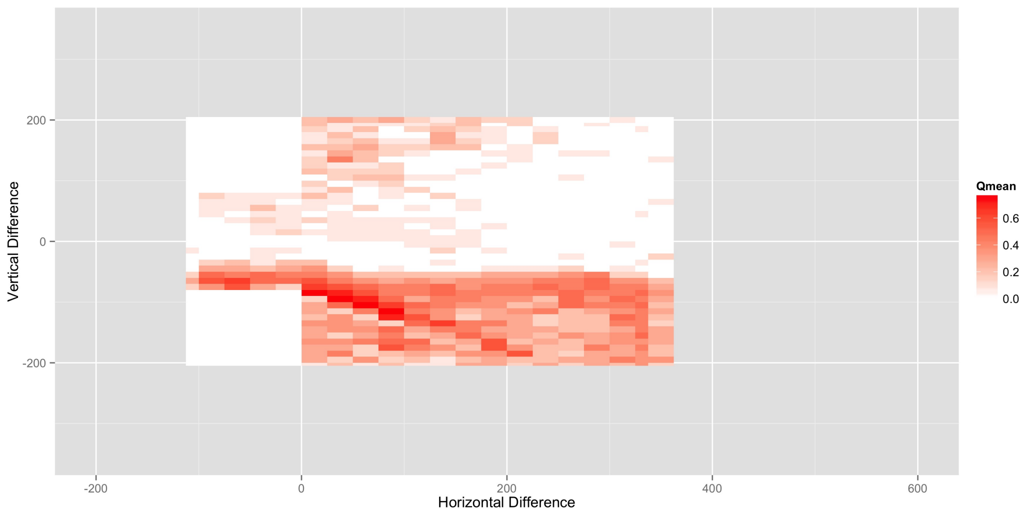


Figure 5: Mean Decision over Spatial States

Reference

1. Moerman W, Bakker B, Wiering M. (2009) **Hierarchical Assignment of Behaviours to Subpolicies**. *Cognitive Artificial Intelligence, Utrecht University*.

2. Aljibury H. (2001) **Improving the Performance of Q-learning with Locally Weighted Regression**. *Electrical Engineering, University of Florida*.

3. Smart W, Kaelbling L.(2003) **Weighted low-rank approximations**. *Proceedings of the Twentieth International Conference* 720727.