Game Day #1 - Pre-Game Strategies

Kiatten Mittons - Nathan DeMaria, Rees Klintworth, Derek Nordgren

February 12, 2015

Known Rewards vs Better Q-table

In this game, there is a tradeoff between repeatedly hitting known rewards and exploring the entire strategy/action space to attain a better Q-table. During the first round, our team will use a relatively high alpha in order to pursue an accurate-as-possible Q-table. We believe that by attaining a near-optimal Q-table during the first round we can exploit this during the second round to obtain the highest possible score, given the 40/60 weighting of rounds one and two.

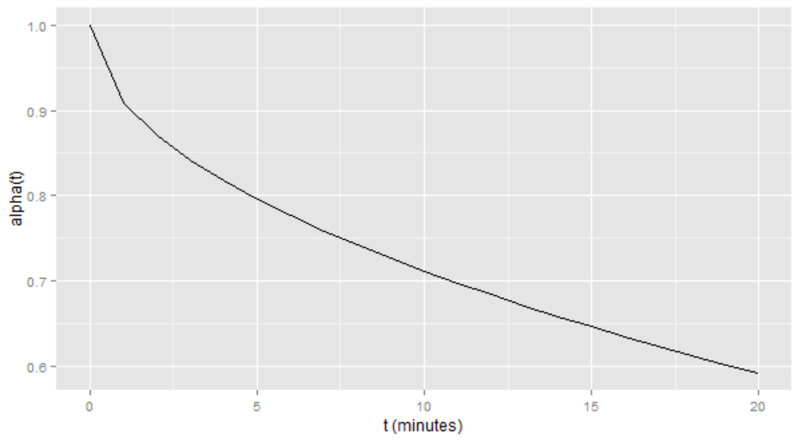
In addition to being able to exploit a near-optimal Q-table, we will have an opportunity to sell our “good” Q-table to other teams.

During the second round, we will decrease our alpha value so that our algorithm relies more heavily upon what it has already learned.

Alpha and Beta 1st Round Selection

Our team will select our alpha to be the function alpha = 1.0 - sqrt(t/6m), where t is the number of minutes that have expired since the round began and m is the total number of minutes in the round. Assuming the round is 20 minutes, alpha will begin at 1.00 at t=0 and converge to roughly 0.60 at t=20. The value of alpha over time can be observed in Figure 1.

Figure 1: Round 1 alpha as a Function of Time



We believe that expressing alpha as a function of time will yield superior results compared to expressing alpha as a constant, as we can begin the round with a very high rate of learning and decrease the rate of learning as the round progresses and as our Q-table becomes closer to optimal. We believe expressing alpha as a function of time will yield a more optimal Q-table, faster.

We will begin the first round with a beta of 0.60. This places some emphasis on projected future rewards, which will be important for making sure we are in good standing with the rest of the class. Our team does not want to focus too much on future rewards, however, so that we can spend the first round exploring the possible state-action pairs and optimizing our Q-table. With a near-optimal Q-table going into Round 2, we can increase beta to focus on chaining possible future state-action pairs together to achieve highly optimal future rewards.

Alpha and beta 2nd Round Selection

During the second round, our team will reduce alpha, as our Q-table will be near-optimal and we will have better results if we rely more heavily on past values. We expect to begin alpha at roughly 0.50 and have it reduce to roughly 0.00 by the end of Round 2, as we expect that our Q-table will be very near to the optimal by the end of the round, and we therefore will be able to rely on the accuracy of past values with relative confidence.

The beta value for our second round will be determined by our perceived performance and our overall confidence in our Q-table. We anticipate it being difficult to analyze our performance relative to the class, but if we have hit multiple large rewards (relative to the $5,000 maximum) or have explored approximately 75% of the Q-table, we will be confident in our overall standing.

In the case that we believe that our standing relative to the rest of the class is very strong, we may lower our beta. A lower beta makes the algorithm myopic, focusing more heavily on the rewards at hand. Even if we have a high degree of confidence in our Q-table, rewards at hand will always be more certain that potential future rewards, so a low-beta approach would be relatively conservative. A low-beta approach could be used to preserve a large lead.

However, if we a) have a high degree of confidence in our algorithm or b) feel that our team is farther back in the class standings, we will raise our beta or leave it near its first round value. A higher beta causes the algorithm to weigh future rewards more heavily than those at hand, resulting in potential high total rewards over the course of the round. If we have a high degree of confidence in our Q-table going into the second round, we would be able to seek high future rewards with relative confidence. If we are lagging the rest of the class a high beta could help our team make up ground during the second round due to the possibility of large future rewards.

Strategies for Buying Q-tables, Handling Other Teams Inquiring About Ours

Based off of both our initial strategy and plans for adjustment, we are confident that we will determine one of the most accurate Q-tables. Therefore, we will not request to receive Q-tables from other teams unless our team believes that we are very far behind after the first round.

No matter how we perceive our performance, we will always tell other teams that our Q-table is well developed. If we are truly doing well, there is no need to lie. If they do request to buy our table, we can simply refuse. If we are not doing as well as we announce, however, the other teams may still be convinced to buy our Q-table. In that case, we could receive extra reward from the sale, without giving anything useful to the other team.

Shiny Application

To speed up our process, we created a Shiny application in R to automate our calculations and store our Q-table. Each time we select an action and receive a reward, we will enter the original state, selection action, resulting state, and resulting reward into the application. The application will then save an updated Q-table, display a heatmap of of the table, suggest the action that will give the highest reward (according to the Q-table), and increment the count of the state/action pair. We will keep a count of each explored state/action pair to keep track of which parts of the table we have and have not already explored.

Initial Size for the Q-table

The Shiny application also contains a script for creating an initial Q-table (stored in .csv format). The number of columns in the table is the number of actions, which will be immediately known from the game page. The number of rows in the table is the number of states. This value is not immediately known; however we will assume that it is a not larger than fifty, since the class is expected to be able to reach all of the states in the limited game time. In reality, the number of states will likely be much smaller than that. However, overestimating the number of states will not have an effect on the Q-learning algorithm, because the rows for states that do not exist will never be used. Therefore, we will have the initial script create the Q table with 50 rows and the correct number of columns. After a few rounds, it will become clear which states are not possible (since no states higher than some n were reached). At that point, we will cut off the extra rows from the Q-table.

Populating the Q-table with Initial Values

When creating the initial Q-table, it is necessary to pick initial values. Since we have no reason to assume that any state action pair is more valuable than another, we will initialize them to all be the same value. We will choose this initial value to be the first positive reward that we find. By doing this, we have a better chance of selecting the right order of magnitude for rewards, instead making an arbitrary guess.