Lab 4

CS 3793/5233 – Fall 2013
Tom Bylander, Instructor

In Lab 4, you will improve a program that plays the game of Yahtzee. Your grade on the lab will depend on the performance of your program on 1000 games of Yahtzee. One improvement is to add learning to make better decisions. The initial program is lab4.zip, which you can download from the course web site.

Yahtzee

You can play Yahtzee by running the main method in Yahtzee.java.

Yahtzee consists of 13 rounds. In each round, you roll the dice up to three times and then choose one of 13 categories for scoring the final roll. You must score once in each category, which means that towards the end of the game you may have to settle for scoring zero in some categories. The score is determined by a different rule for each category; see the section on Scoring below.

The object of the game is to maximize your total score. The game ends once all 13 categories have been scored.

You have five dice which you can roll. To start with, you roll all five dice (well, the program rolls the dice for you). After you roll all dice, you can either score the current roll, or reroll any or all of the five dice.

You can roll the dice a total of three times: the initial roll (in which you must roll all five dice), plus two rerolls of any or all dice. After rolling three times, you must score the roll.

Once you’ve scored the roll, you have completed one round and repeat the process. You continue until all 13 categories have been filled, at which time the game is over.

Scoring

Once you have the dice combination you want to score (or have to score), you score the roll in one of the 13 categories. Once a category has been scored, it is closed out for the rest of the game; you cannot change a category’s score once it’s been set. Each category defines its own scoring rules, as described below. The categories are partitioned into upper scores and lower scores.

Upper Scores

The upper scores consist of six categories corresponding to the six numbers of the dice. In an upper scores category, you total only the specified number. So if you roll:

3 3 4 3 6

and score in the Threes category, your total for that entry would be 9. This same roll would yield 0 points if you scored it in the Ones, Twos, or Fives category, 4 points if you scored it in the Fours category, or 6 points if you scored it in the Sixes category.
When the game is over, if you score 63 or more points as upper scores (an average of 3 dice per category), you will get an upper bonus of 35 points. You don’t need to score exactly 3 dice in each upper category to get the bonus, as long as the upper total is at least 63.

**Lower Scores**

The lower scores consist of seven categories. You score either a set amount (defined by the category), or zero if you don’t satisfy the category requirements.

**3 of a Kind and 4 of a Kind**

For 3 of a Kind, you must have at least 3 of the same number. If so, you total all the dice and score that total. Similarly for 4 of a Kind, except that you must have at least 4 of the 5 numbers the same. So for example, if you rolled:

```
5 5 3 1 5
```

you would receive 19 points for 3 of a Kind, but zero points for 4 of a Kind.

**Small Straight and Large Straight**

A straight is a sequence of consecutive numbers; a Small Straight is 4 consecutive numbers, and a Large Straight is 5 consecutive numbers. Small Straights score 30 points and Large Straights score 40 points. Thus, if you rolled:

```
5 4 3 2 6
```

you could score either a Small Straight or a Large Straight, since this roll satisfies both.

**Full House**

A Full House is a roll where you have both a 3 of a Kind, and a pair. Full houses score 25 points.

**Yahtzee**

A Yahtzee is a 5 of a Kind (i.e. all the numbers are the same), and it scores 50 points. Yahtzee is the only category that you may score more than once. If you roll another Yahtzee in a single game, and if you have already scored one or more 50’s the Yahtzee category, then you may score another Yahtzee for 50 more points, and you also get an additional round of play. If you choose to score a 0 in the Yahtzee category, then you cannot score any subsequent Yahtzees during the current game. You cannot score a 0 in the Yahtzee category after you have already scored a 50.

**Chance**

Chance is the catch-all roll. You can roll anything and you simply total all the dice.
Playing Yahtzee Yourself

Again, you can play Yahtzee by running the main method in Yahtzee.java.

Protocol

Yahtzee.java will behave in the following way. Initially, Yahtzee.java will output the first dice roll as 5 numbers on one line. It expects an input of either the dice to reroll, or a category for scoring. The dice are numbered from 0 to 4, so that an input of:

```
0 3 4
```

would indicate that dice 0, dice 3, and dice 4 should be rerolled. The yahtzee program will then reroll the specified dice and output all 5 numbers on one line. Any attempt to do input more than 2 lines of rerolls will result in a runtime error.

A category input must be one of the following:

- Ones
- Twos
- Threes
- Fours
- Fives
- Sixes
- 3 of a Kind
- 4 of a Kind
- Full House
- Small Straight
- Large Straight
- Yahtzee
- Chance

After a category has been input, the next round begins. After all categories have been scored, the current game ends and the yahtzee program outputs the total score for the game. After the game ends, Yahtzee.java starts another game (unless it has reached the upper limits of number of games to play).

If there is an attempt to score a category twice (with the exception of multiple Yahtzees), or if an input is a not a reroll (a list of numbers) or one of the categories above, then the yahtzee program result in a runtime error.

Improvement One

The current player YahtzeePlayer.java simply rerolls all the dice twice and then scores the first category that has not been scored yet. This results in an average score under 50.

A simple improvement is to choose the unscored category that results in the highest score. The scoreCategory method in YahtzeePlayer.java should be useful for coding this. This improvement should result in an average score of about 110.
Improvement Two

Improvement One does not take advantage of the rerolls to increase the score. Your program should find the reroll with the maximum expected value. Much of what is described here is implemented in the `expectedValue` method in `YahtzeePlayer.java`.

Choosing Dice to Reroll

In a reroll, you can choose any subset of the dice to reroll. Ideally, you would like to find the best subset that will maximize the final score. However, the implementation I discuss here will find the best subset for a single reroll to maximize an individual category score. It does not take into account two rerolls, it does not take into account the upper scores bonus, and it does not take into account that some categories (e.g., Chance) are much easier to score than others (e.g., 4 of a Kind). Other improvements below describe how to modify your lab to learn how to make better decisions.

Suppose that the current values of the dice are:

```
2 5 1 2 2
```

and that you are considering rerolling dice 1 and 2, leaving dice 0, 3, and 4 alone. Note that there are 36 possible values for the two dice. Assuming the possible outcomes are equally likely, and assuming that no category has been scored yet, here is the best way to score each possibility, preferring 4 of a Kind over 3 of a Kind over Chance in case of ties.
<table>
<thead>
<tr>
<th>Dice</th>
<th>Category</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 1 1 2 2</td>
<td>Full House</td>
<td>25</td>
</tr>
<tr>
<td>2 1 2 2 2</td>
<td>4 of a Kind</td>
<td>9</td>
</tr>
<tr>
<td>2 1 3 2 2</td>
<td>3 of a Kind</td>
<td>10</td>
</tr>
<tr>
<td>2 1 4 2 2</td>
<td>3 of a Kind</td>
<td>11</td>
</tr>
<tr>
<td>2 1 5 2 2</td>
<td>3 of a Kind</td>
<td>12</td>
</tr>
<tr>
<td>2 1 6 2 2</td>
<td>3 of a Kind</td>
<td>13</td>
</tr>
<tr>
<td>2 2 1 2 2</td>
<td>4 of a Kind</td>
<td>9</td>
</tr>
<tr>
<td>2 2 2 2 2</td>
<td>Yahtzee</td>
<td>50</td>
</tr>
<tr>
<td>2 2 3 2 2</td>
<td>4 of a Kind</td>
<td>11</td>
</tr>
<tr>
<td>2 2 4 2 2</td>
<td>4 of a Kind</td>
<td>12</td>
</tr>
<tr>
<td>2 2 5 2 2</td>
<td>4 of a Kind</td>
<td>13</td>
</tr>
<tr>
<td>2 2 6 2 2</td>
<td>4 of a Kind</td>
<td>14</td>
</tr>
<tr>
<td>2 3 1 2 2</td>
<td>3 of a Kind</td>
<td>10</td>
</tr>
<tr>
<td>2 3 2 2 2</td>
<td>4 of a Kind</td>
<td>11</td>
</tr>
<tr>
<td>2 3 3 2 2</td>
<td>Full House</td>
<td>25</td>
</tr>
<tr>
<td>2 3 4 2 2</td>
<td>3 of a Kind</td>
<td>13</td>
</tr>
<tr>
<td>2 3 5 2 2</td>
<td>3 of a Kind</td>
<td>14</td>
</tr>
<tr>
<td>2 3 6 2 2</td>
<td>3 of a Kind</td>
<td>15</td>
</tr>
<tr>
<td>2 4 1 2 2</td>
<td>3 of a Kind</td>
<td>11</td>
</tr>
<tr>
<td>2 4 2 2 2</td>
<td>4 of a Kind</td>
<td>12</td>
</tr>
<tr>
<td>2 4 3 2 2</td>
<td>3 of a Kind</td>
<td>13</td>
</tr>
<tr>
<td>2 4 4 2 2</td>
<td>Full House</td>
<td>25</td>
</tr>
<tr>
<td>2 4 5 2 2</td>
<td>3 of a Kind</td>
<td>15</td>
</tr>
<tr>
<td>2 4 6 2 2</td>
<td>3 of a Kind</td>
<td>16</td>
</tr>
<tr>
<td>2 5 1 2 2</td>
<td>3 of a Kind</td>
<td>12</td>
</tr>
<tr>
<td>2 5 2 2 2</td>
<td>4 of a Kind</td>
<td>13</td>
</tr>
<tr>
<td>2 5 3 2 2</td>
<td>3 of a Kind</td>
<td>14</td>
</tr>
<tr>
<td>2 5 4 2 2</td>
<td>3 of a Kind</td>
<td>15</td>
</tr>
<tr>
<td>2 5 5 2 2</td>
<td>Full House</td>
<td>25</td>
</tr>
<tr>
<td>2 5 6 2 2</td>
<td>3 of a Kind</td>
<td>17</td>
</tr>
<tr>
<td>2 6 1 2 2</td>
<td>3 of a Kind</td>
<td>13</td>
</tr>
<tr>
<td>2 6 2 2 2</td>
<td>4 of a Kind</td>
<td>14</td>
</tr>
<tr>
<td>2 6 3 2 2</td>
<td>3 of a Kind</td>
<td>15</td>
</tr>
<tr>
<td>2 6 4 2 2</td>
<td>3 of a Kind</td>
<td>16</td>
</tr>
<tr>
<td>2 6 5 2 2</td>
<td>3 of a Kind</td>
<td>17</td>
</tr>
<tr>
<td>2 6 6 2 2</td>
<td>Full House</td>
<td>25</td>
</tr>
</tbody>
</table>

The expected score after this reroll is $565/36 \approx 15.69$. Note that the expected score would be different if some categories have already been scored. For example, if Full House has already been scored, then the expected score would be $508/36 \approx 14.11$.

The reroll you choose should be the one with the highest expected score. If you have no more rerolls (or if you choose not to reroll any dice), then you should choose the category that has the highest score for the dice.
Choosing Dice to Reroll

The \texttt{expectedValue} method performs the “middle” of this reroll decision making. Here is the sequence:

1. Before starting any games, you should compute the values for the \texttt{diceScore} parameter of \texttt{expectedValue}. Essentially, \texttt{diceScore} caches the values from the \texttt{scoreCategory} method. You need to create a 6-d int array with the first 5 dimensions equal to 7, and the last dimension equal to 13. The first 5 dimensions are for the five dice, and the last dimension for the category to be scored. For example, if \texttt{dice} an int array containing 2, 5, 1, 2, 2, then

\[
\begin{align*}
\text{diceScore}[2][5][1][2][2][0] &= \text{scoreCategory}(0, \text{dice}) = 1 \\
\text{diceScore}[2][5][1][2][2][1] &= \text{scoreCategory}(1, \text{dice}) = 6 \\
\text{diceScore}[2][5][1][2][2][2] &= \text{scoreCategory}(2, \text{dice}) = 0 \\
\text{diceScore}[2][5][1][2][2][3] &= \text{scoreCategory}(3, \text{dice}) = 0 \\
\text{diceScore}[2][5][1][2][2][4] &= \text{scoreCategory}(4, \text{dice}) = 5 \\
\text{diceScore}[2][5][1][2][2][5] &= \text{scoreCategory}(5, \text{dice}) = 0 \\
\text{diceScore}[2][5][1][2][2][6] &= \text{scoreCategory}(6, \text{dice}) = 12 \\
\text{diceScore}[2][5][1][2][2][7] &= \text{scoreCategory}(7, \text{dice}) = 0 \\
\text{diceScore}[2][5][1][2][2][8] &= \text{scoreCategory}(8, \text{dice}) = 0 \\
\text{diceScore}[2][5][1][2][2][9] &= \text{scoreCategory}(9, \text{dice}) = 0 \\
\text{diceScore}[2][5][1][2][2][10] &= \text{scoreCategory}(10, \text{dice}) = 0 \\
\text{diceScore}[2][5][1][2][2][11] &= \text{scoreCategory}(11, \text{dice}) = 0 \\
\text{diceScore}[2][5][1][2][2][12] &= \text{scoreCategory}(12, \text{dice}) = 12
\end{align*}
\]

Making the dimension size 7 is a little wasteful, as the zero index is not used for any of the first 5 dimensions, but hopefully this wastefulness will be made up by avoiding off-by-one errors.

This array should be created once and filled in once for all 1000 games. Other code will decide which categories to use.

2. The \texttt{expectedValue} method expects the \texttt{diceScore} array, a boolean \texttt{scored} array, and a 5-d double array \texttt{diceEV}. The \texttt{scored} array should have length 13 and should indicate which categories have already been scored in this game.

The \texttt{diceEV} should have each dimension equal to 7. You should only need to create this array once. Each call to \texttt{expectedValue} will overwrite the previous values. Whenever the \texttt{scored} array changes, \texttt{expectedValue} needs to be called.

Part of the reason for providing the \texttt{expectedValue} method to you is so that playing 1000 Yahtzee games will not take up too much time.

3. After calling \texttt{expectedValues}, \texttt{diceEV} contains the expected values of every dice reroll of every dice combination. A dice reroll is indicated by a zero index. For example, at the beginning of a game:

\[
diceEV[2][0][0][2][2] \approx 15.69
\]
This indicates that 15.69 is the expected value of rerolling dice 1 and 2 when the three other dice are equal to 2.

For the current dice, you need to find out which reroll has the highest expected value. There are 32 possibilities, from rerolling none of the dice to rerolling all five dice.

You should only need to fill in \texttt{diceScore} once for all 1000 games. You need to call \texttt{expectedValue} once at the beginning of each round. You need to decide on rerolls once or twice each round.

**Expected Results**

If you only reroll the dice once for each round, the above computation will result in an average score of about 170. If you reroll the dice twice using the above computation, you should get an average result of about 210.

Note: You do not have to reroll any dice. For example, if the first roll is a Yahtzee, you probably don’t want to reroll any dice. Instead, you can score a Yatzee and move on to the next round.

**Improvement Three**

In Improvement Two, the choices of which dice to roll and which category to score are based on maximizing the expected value of one category. In Improvement Three, these choices will be based on maximizing the expected value of the “reward” in one category plus the future rewards in the remaining categories.

Let \( C \) be the set of categories that have not been scored. Let \( e_c \) the expected reward for scoring in category \( c \).

Let \( s_c \) be the score in category \( c \) for given values of the dice. This is what is being stored in the \texttt{diceScore} array for Improvement Two.

In Improvement Two, you chose rerolls and the final category by finding the category that maximizes \( s_c \). In Improvement Three, you will instead maximize:

\[
s_c - e_c + \sum_{i \in C} e_i
\]

which is equivalent to maximizing \( s_c - e_c \).

What this means: The summation is the expected reward of scoring the categories in \( C \). In the current round, the expected reward of one category will be replaced by an actual score. The idea to score categories with better than expected scores.

One simple, but effective, choice for \( e_c \) is the average score in this category; however, it is difficult to estimate these averages directly because your program will play differently for different values. An alternative is to learn these values as the program plays a sequence of games. Thus, after the program has reward \( s_c \) for scoring in category \( c \), the expected reward \( e_c \) will be modified by:

\[
e_c \leftarrow e_c + \eta (s_c - e_c)
\]

where \( \eta \) is a learning rate. Determining an appropriate learning rate is one the tasks you need to perform (try values around \( \eta = 0.01 \)).
This improvement implements LMS (least mean squared) learning of these values. This is equivalent to undiscounted reinforcement learning as described in the book, where a linear function (in this case, the sum of the rewards) is used to calculate future reward.

**Implementation**

Wherever there is a loop for the unscored category with the maximum score, it should be a loop for the unscored category that maximizes expected reward. Note that there is one deep inside the `expectedValue` method. The weight updating should happen whenever a category is scored.

This improvement should increase your average score to above 220.

**Your Grade and Other Possible Improvements**

Your grade on this lab will range from 80 for an average of 100 to 100 for an average of 200. An average over 200 might receive a bonus. Just like the previous lab, there will be bonuses for the best performers. The five best performances in this lab (must be over 200 average and finish within a minute) will receive an additional 50, 40, 30, 20, or 10 points.

The above does not account for the 35 points bonus, more than one Yahtzee, or calculating expected values over two rerolls. You are free to pursue your own improvements.