heuristic_analysis

June 29, 2017

1 Artificial Intelligence Nanodegree

1.1 Build a Game-Playing Agent Heuristic Analists

Guillermo Aure Jun 26, 2017

1.1.1 Introduction

This report contains information about the heuristic functions used during the build of a Game-Playing Agent. The information is about an exercise (tournament) conducted using three heuristic functions, and the resultant performance of each one of them. We used a simulated tournament that consisted of multiple variances of AI agents playing a game called Isolation against each other, using the different heuristics to evaluate the game at each state. The report has two sections; the first one contains a description of three heuristics evaluation functions used to determine a particular agent best action at each state. The second section contains the result of the tournament and analysis about each heuristic evaluation performance.

1.1.2 Heuristic Functions

To determine the best action of the AI agents we coded the following three heuristic functions: - **Number of AI agent available moves**: this function calculates on each play the potentially available moves of the AI agent at that particular state of the game, it takes into consideration the state, the actions(possible moves) and the AI player. - **Number of moves resulting from subtracting the AI agent possible moves minus its opponent possible moves**: this function goes beyond the previous one by including also the opponent moves as part of the calculation, it is a function that embodies more information than the previous one. - **Number of total potential moves at each state**: of the three implemented functions, this one does not include information about the players but only about the state and actions, it is the less complete function.

1.1.3 Heuristic Functions Performance

The below tables shows the result of executing the simulate tournament. The tournament script, evaluates the performance of the heuristic evaluation function against a baseline AI agent using alpha-beta search and iterative deepening ID called AB-Improved. The three AB-Custom agents use ID and alpha-beta search with the heuristic evaluation functions defined by the student as part of the Build a Game-Playing Agent project.

In the above table, we can observe that the evaluation functions where information about the players is included the performance of the AI agent are much better than those evaluation functions that only contains information about the game itself.

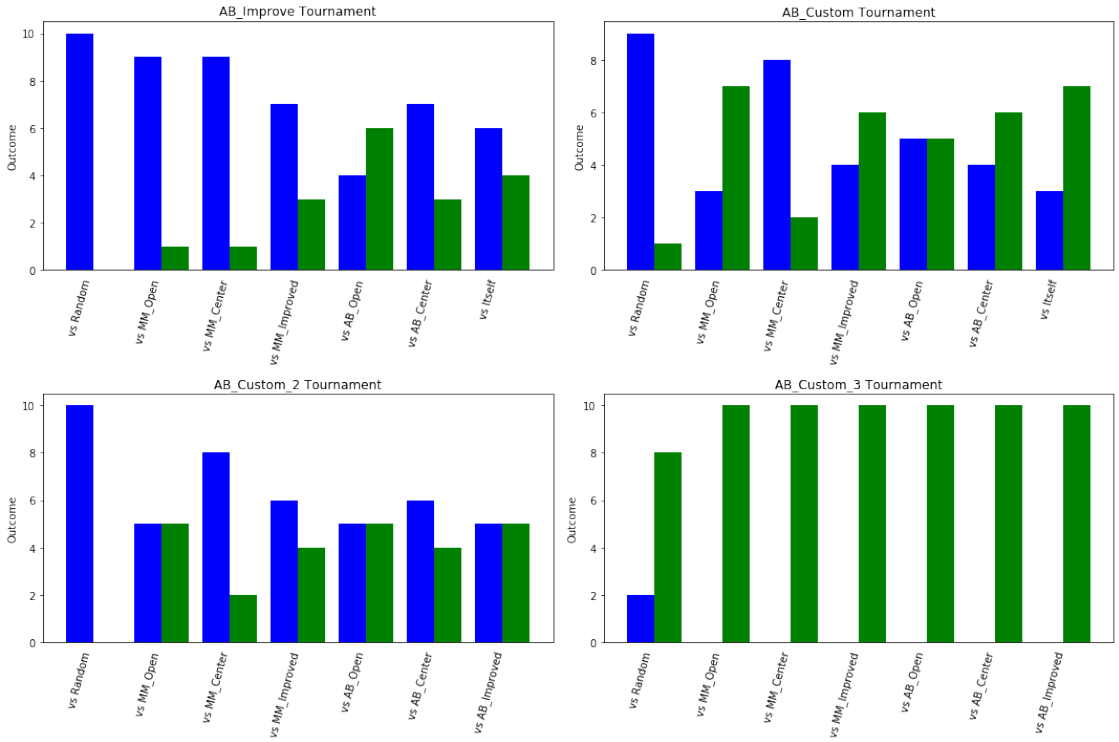
A good evaluation function must include a combination of the game state features and also the calcualtions involve in it have to be executed fast enough to allow the deepest search. The two previous characteristics of an evalution function are proporsionally inverse, the more state features the evaluation function uses the better its approximation is to the real terminal utility value; at the same time the more computational effort is required for its calculation allowing lest time to explore the tree more deep, so its result will be shallow.

The plot below shows graphically the result of the tournament. Of the three custom functions the one that produced the best result was the "Custom_2" function.

The reason why the "Custom_2" function provided the best results was because it has the more number of state features, with allows it to get a better estimate of the utility value; its computational cost is low enough to allow further level exploration.

```
In [197]: import pandas as pd
          import numpy as np
          import matplotlib.pyplot as plt
          %matplotlib inline

          data = pd.read_csv("tournament.csv")



          ind = np.arange(1,8)  # the x locations for the groups
          width = 0.40       # the width of the bars

          #figure AB_Improve
          plt.figure(figsize=(15,10))

          AB_Improved = data[0:7]

          ax = plt.subplot(221)
          ax.tick_params(axis='x',which='both', bottom='off')
          ax.title.set_text('AB_Improve Tournament')
          ax.set_xticklabels(AB_Improved['players'],rotation=75,fontsize=10 )
          ax.set_xticks(ind)
          ax.bar(ind, AB_Improved['win'],width=0.4,color='b',align='center')
          ax.bar(ind+width, AB_Improved['lost'],width=0.4,color='g',align='center')
          ax.set_ylabel('Outcome')

          #figure AB_Custom
          AB_Custom = data[7:14]

          ax1 = plt.subplot(222)
          ax1.tick_params(axis='x',which='both', bottom='off')
          ax1.title.set_text('AB_Custom Tournament')
          ax1.set_xticklabels(AB_Improved['players'],rotation=75,fontsize=10 )
```

```
ax1.set_xticks(ind)
ax1.bar(ind, AB_Custom ['win'],width=0.4,color='b',align='center')
ax1.bar(ind+width, AB_Custom ['lost'],width=0.4,color='g',align='center')
ax1.set_ylabel('Outcome')

#figure AB_Custom_2
AB_Custom_2 = data[14:21]

ax2 = plt.subplot(223)
ax2.tick_params(axis='x',which='both', bottom='off')
ax2.title.set_text('AB_Custom_2 Tournament')
ax2.set_xticklabels(AB_Custom_2['players'],rotation=75,fontsize=10 )
ax2.set_xticks(ind)
ax2.bar(ind, AB_Custom_2 ['win'],width=0.4,color='b',align='center')
ax2.bar(ind+width, AB_Custom_2 ['lost'],width=0.4,color='g',align='center')
ax2.set_ylabel('Outcome')

#figure AB_custom_3
AB_Custom_3 = data[21:29]

ax3 = plt.subplot(224)
ax3.tick_params(axis='x',which='both', bottom='off')
ax3.title.set_text('AB_Custom_3 Tournament')
ax3.set_xticklabels(AB_Custom_3['players'],rotation=75,fontsize=10 )
ax3.set_xticks(ind)
ax3.bar(ind, AB_Custom_3 ['win'],width=0.4,color='b',align='center')
ax3.bar(ind+width, AB_Custom_3 ['lost'],width=0.4,color='g',align='center')
ax3.set_ylabel('Outcome')

plt.tight_layout()
plt.show()
```


AB_Improve Tournament
Outcome
vs Random
vs MM_Open
vs MM_Center
vs MM_Improved
vs AB_Open
vs AB_Center
vs Itself
AB_Custom Tournament
Outcome
vs Random
vs MM_Open
vs MM_Center
vs MM_Improved
vs AB_Open
vs AB_Center
vs Itself
AB_Custom_2 Tournament
Outcome
vs Random
vs MM_Open
vs MM_Center
vs MM_Improved
vs AB_Open
vs AB_Center
vs AB_Improved
AB_Custom_3 Tournament
Outcome
vs Random
vs MM_Open
vs MM_Center
vs MM_Improved
vs AB_Open
vs AB_Center
vs AB_Improved