FAI Final Project

Abstract

Using different method to train models and compare result, I finally choose Gradient Boosting Regression to be the model I use for the baseline
the winning rate comparison is at the last part of the report
Winning rate for `agent.py` :

baseline 1	baseline 2	baseline 3
0.61	0.57	0.54

Method Tried

1. No Method
2. Linear Regression
3. Gradient Boosting Regression
4. Random Forest Regression
5. Lasso

No Method

Use simple strategy: *fold* when hole_card is small, *call* if hole_card might win, *raise* when I think hole_card can lead me to win.
baseline 1: 0.6
baseline 2: 0.4
baseline 3: 0
I also added some more condition to make increase win rate. But I did not try the result on it own, I add those conditions in the regression and ensemble models. Ex: always fold if my stack is more than `left_round * 10` , forbid `call` if the amount is to high, etc.

Regression and Ensemble

In each round, I recorded hole card, pot amount, action and amount, and sent the information into regression to fit using sklearn models(Gradient Boosting Regression, Lasso, Linear Regression, Random Forest Regression). When it is my turn, I simulate the loss of three taken action, and choose the action with the smallest loss.

Linear Regression

linear regression is the simplest way to do regression, but the result is not very satisfying

Gradient Boosting Regression

This regression method use ensemble, and it is said to have good performance in Reinforcement Learning
half AI:
while traing, I let my *no method* model and GBR trained together, so the machine can learn from my choice and adjust it. I use this method to support my ai because it often overestimate the reward, and also because I misunderstood the `start_game` code at first. So it is just a try and the result I found while training.
mod 5, baseline 1: 0.65
mod 3, baseline 1: 0.54
mod 7, baseline 1: 0.56

Random Forest Regression

From the course and previous assignment, random forest can have good performance and is worth trying

Lasso

Lasso can prevent overfitting, so I give it a try

Comparison between 4 trained models

Train with multiple players at the same time (It was for fun and just a try, the result is quite bad actually)

Baseline	Gradient Boosting Regression	Lasso	Linear Regression	Random forest Regression
Training	0.36	0.4	0.345	0.335
baseline 0	0.01	0.01	0.09	0.48
baseline 1	0.51	0	0.01	0.48
baseline 2	0.36	0.18	0.29	0.26
baseline 3	0.64	0.48	0.47	0.72

Discover:

1. Multi-player game training strategy might not be suitable for one-vs-one mode, which is quite reasonable since
2. This game depends on **LUCK** a lot!
3. My model fit baseline 3 more than baseline 1 and 2 (but I don't know the reason, due to the action chosen while training, it might because that my model tend to fold in these two baselines). However, I discover a bug in the counting system which skip some of the success counts.

This result is also trained with 3 baselines, but I take turn fighting with these baselines, the result return out quite satisfying. (The actual success rate is higher since I cannot found the bug.)

Baseline	Gradient Boosting Regression	Lasso	Linear Regression	Random forest Regression
Training	0.54	0	0.45	0.48
baseline 0	0.6	0.01	0	0.05
baseline 1	0.6	0.02	0	0.41
baseline 2	0.63	0.6	0.06	0.6
baseline 3	0.6	0.64	0.71	0.45

And I found out that although my ai learn how to calculate the loss, it did not learn that it can get $2000 at most, therefore, I add an additional condition for the robot to ensure that once it has enough money for surviving till round 20, it stop joining the game.

Final result

Baseline	Gradient Boosting Regression (200)	Lasso	Linear Regression	Random forest Regression	Gradient Boosting Regression (400)	Gra Boc Reg witl add con (20
Training	0.545	0.575	0.455	0.51	0.54	0.3
baseline 0	0.45	0.38	0.1	0.38	0.42	0.4
baseline 1	0.53	0.54	0.28	0.52	0.55	0.5
baseline 2	0.59	0.07	0	0.57	0.24	0.6
baseline 3	0.69	0.71	0.45	0.7	0.67	0.3

Reference:

https://github.com/chasembowers/poker-learn

(https://github.com/chasembowers/poker-learn)

I follow the models the author tried

Also tried

A3C but the result is not satisfying so I did not put it here