CS 236R PSET 1
Florian Berlinger and Lily Xu
March 20, 2020

Summary

Given a training set of 250 3x3 bi-matrix games, we attempted to infer the behavior of the row player and make predictions on a test set accordingly. Evaluation metrics include the quadratic distance of the frequency distribution (Q) and the accuracy in predicting the most frequently chosen action (A). We attempted behavioral inference and prediction using (i) traditional models such as Nash Equilibrium or Level-k, (ii) Machine Learning strategies including Linear Regression and Gradient Boost, as well as (iii) Hybrid models that combine the former two approaches.

A hybrid approach combining Nash Equilibrium and Linear Regression yielded the highest performance with A = 0.752 and Q = 0.076 (Fig. 1) (see predict.py). We chose this hybrid model to evaluate against the test set (see main_test.py).

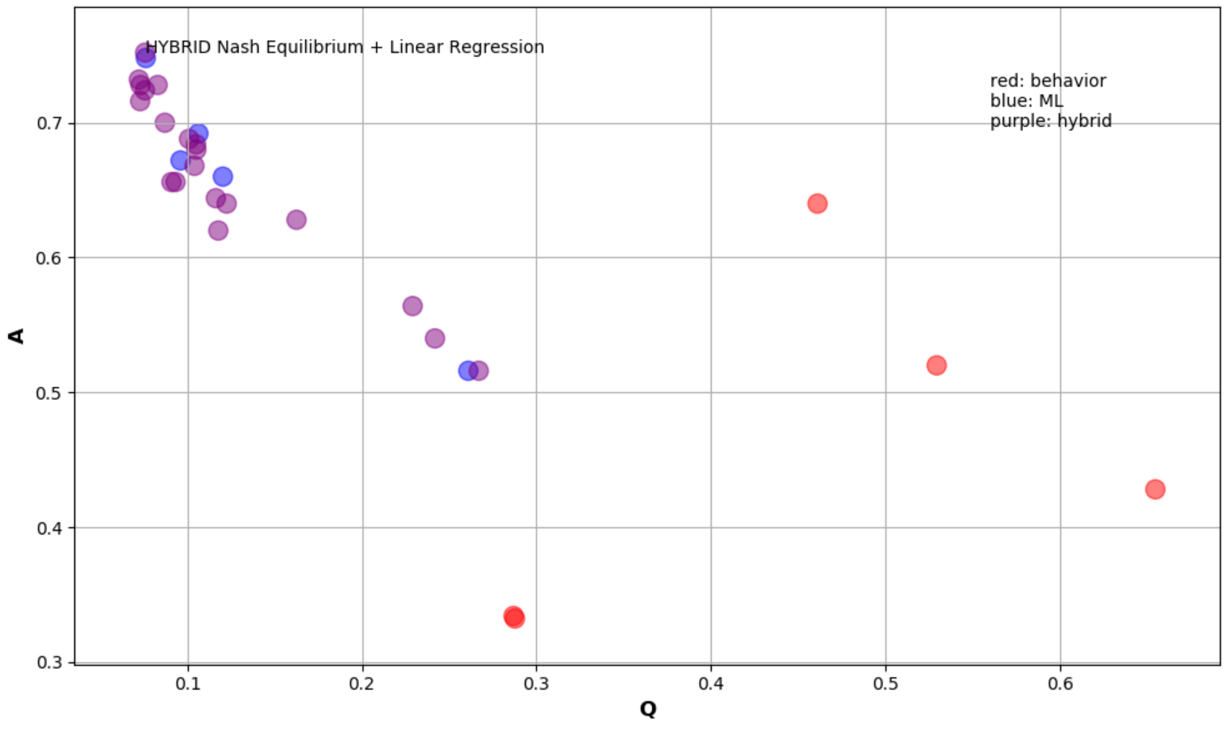

Fig. 1 | Action prediction accuracy (A) and quadratic distance of frequency prediction (Q). High performance manifests in high A scores and low Q scores. 31 experiments were run on the training set including 5 behavioral-based approaches (red), 6 pure ML-based approaches (blue), and 20 combinatory hybrid-based approaches (purple). The winning experiment was a hybrid of Nash Equilibrium and Linear Regression with A = 0.752 and Q = 0.076.

Methodology

The main approach we used was a hybrid combining Nash Equilibrium and Linear Regression with A = 0.752 and Q = 0.070 (Fig. 1). Here we present detailed evaluation of all approaches we tried.

0 Introductory Remarks

Our game predictors were implemented in Python3 using additional packages including numpy, pandas, matplotlib, and sklearn. The final approach is in main_test.py. Running main_test.py from the terminal in a folder with {hb_train_feature, hb_train_truth, hb_test_feature}.csv will train this final approach on the training set and make predictions on the test set. The script main_test.py will output hb_test_pred.csv with header [f1, f2, f3, Action] followed by number of games x 4 corresponding predictions (c.f. format of hb_train_truth.csv). The script predict.py contains the implementations of all approaches and is well documented.

1 Performance Evaluation

We implemented functions to evaluate the performance of our predictors as per the "Forecast Accuracy" section in the PSET1 guidelines (i.e., for the Q and A metrics). We ensured that these functions are implemented correctly by testing them with random predictions. Averaged over 1000 trials to address variation in the randomness, we found A = 0.333 as expected and Q = 0.288.

2 Mixed-Strategy Nash Equilibrium

As a first predictor, we implemented mixed-strategy Nash equilibrium as per the Lemke-Howson Algorithm [Lemke and Howson, 1964]. We acquired a basic understanding of the algorithm from online sources[1] and based our Python3 implementation on a publicly available MATLAB version[2]. The performance on the training set was Q = 0.529 and A = 0.520.

3 Level-k Reasoning

Implemented for k = 0, 1, 2. Note that Level-0 is just a random strategy, akin to the random baseline. Level-1 assumes the column player is using a random strategy and takes the action that yields the best payoff relative to that strategy. Level-2 assumes the column player is using a level-1 strategy. Although level-2 is a more sophisticated strategy than level-1, its performance is worse, suggesting that the players are limited in their reasoning.

The performance results are shown in Table 1. Note that the level-k behavioral model yields a binary frequency for levels k > 0. Thus, the Q-score for level-1 and level-2 are relatively high, and in fact higher than that of level-0 despite higher A-score performances. To extend this work, we could consider the quantal level-k model which would produce a distribution over the actions rather than produce a fixed response.

Table 1

[1] http://ints.io/daveagp/gta/lecture6.pdf
[2] https://www.mathworks.com/matlabcentral/fileexchange/44279-lemke-howson-algorithm-for-2-player-games

	Q	A
Level-0	0.287	0.334
Level-1	0.461	0.640
Level-2	0.655	0.428

4 Machine-Learning Models

We experimented with five different machine learning models: linear regression, decision tree, random forest, AdaBoost, and gradient boosting. All of these models were implemented with the sklearn package in Python.

To evaluate the performance of these models, we performed 5-fold cross validation, in each iteration holding out a validation set containing 20% of the training points to compute Q and A scores. The results below give performance scores averaged over all 5 iterations. As shown in Table 2, these models all beat 0.5 accuracy, although linear regression (despite being the simplest model) performs best in both Q and A scores.

Table 2

	Q	A
Linear regression	0.071	0.748
Decision tree	0.243	0.548
Random forest	0.120	0.660
Gradient boost	0.096	0.672
AdaBoost	0.106	0.692

5 Hybrid Models

For the hybrid models, we augmented the features going into the machine learning model with the predictions of a behavior model, similar to the papers read in class about predicting initial play. In this section, we experimented with all combinations of the 5 machine-learning models and the 4 behavioral models. The results are shown in Table 3. Linear regression performed best across the board, indicating that using the correct machine learning model is critical to good performance. The performance scores are notably higher than that of any pure behavioral model. However, observe that the performance scores for nearly every ML model is higher when combined with a behavioral model, regardless of which model is selected. This result highlights the importance of using hybrid models to drive performance.

Table 3

	Nash EQ		**Level-0**		**Level-1**		**Level-2**	
	Q	**A**	**Q**	**A**	**Q**	**A**	**Q**	**A**
Linear regression	0.070	0.752	0.071	0.760	0.069	0.732	0.068	0.716
Decision tree	0.234	0.564	0.251	0.504	0.144	0.632	0.236	0.556
Random forest	0.118	0.620	0.123	0.644	0.087	0.700	0.116	0.644
Gradient boost	0.093	0.656	0.100	0.668	0.073	0.728	0.091	0.656
AdaBoost	0.105	0.680	0.105	0.672	0.083	0.728	0.104	0.668

Discussion

We prioritized A-scores over Q-scores as per the grading scheme. Additionally, A-scores are more useful in practice, as the row player will want to take the best-performing action in a single-shot game. All approaches using machine learning yielded lower Q-scores than purely behavior-based approaches. This is expected for at least Level-k approaches, since they only yield the best strategy with frequency 1, rather than provide any distribution over the three actions. In general, low Q-scores were correlated with high A-scores in machine learning and hybrid approaches; i.e., approaches with low Q-scores also showed high A-scores (Fig. 1). The winning hybrid approach combined Nash Equilibrium with a simple Linear Regression (A = 0.752, Q = 0.070). However, it was only marginally better than the best purely ML-based approach, namely Linear Regression without the additional information from Nash Equilibrium (A = 0.748, Q = 0.071).

Bibliography

C. E. Lemke and J. J. T. Howson. Equilibrium points of bimatrix games. SIAM Journal on Applied Mathematics, 12(2):413-423, 1964.