Building predictors for the game mia

Anonymous ACL submission

Abstract

This term paper deals with the topic of predicting different artificial intelligence approaches based on the game mia. It starts with an short introduction to artificial intelligence. Then, the description of the game mia on which the project is based on. Afterwards, the strategies of the different implemented artificial intelligences (AI) are explained. It follows an explanation of our approach predicting the behavior of the different AIs. Finally, the results of the experiments are discussed.

1 Introduction

Artificial intelligence has become an increasingly important field in computer science and other areas such as automotive (self-driving cars) or security (face-detection). In computer games artificial intelligence reaches new stages of success (Google bot AlphaGo for the game go). In this term paper we now want to answer the question if it is possible to predict different artificial intelligence approaches based on the game mia. The main achievement would be the accuracy of correct predictions if a player (AI) will to lie or tells the correct value. If the actual value is lesser or greater seems not as important as prediction liars properly.

2 The Game

Mia is a simple dice game that is played with two dices and a flat bottomed container (or a dice cup). At the beginning each player has a certain amount of lives (e.g. five). The first player rolls the dices but keeps their values hidden from the other players. He then can decide if he wants to tell the truth to the next player and announce a value that was was actually rolled. Alternatively he can lie and announce a greater or lesser value than the rolled one. But each player has to announce a greater value than the previous player. The next player (who still has not seen the actual values) can now believe the passer, call the passer a liar and look on the dice or pass the dice to the next player (still without looking) announcing a higher value. A player looses a life if he called the previous one a liar and looked on the values to find out that they are what the previous player has announced or even higher. Otherwise the previous player looses a life. The higher value of the roll is multiplied by then and then added to the other die (a 4 and a 2 is 42). The **scoring** is from highest to lowest: 21 (Mia), 11, 22, 33, 44, 55, 66, 65, 64, 63, 62, 61, 54, 53, 52, 51, 43, 42, 41, 32, 31. If a player announces mia the next player either believes him, give up (without looking at the dices) and looses one life. Or he may look at the dice. If it was actually mia then he looses two lifes if it was not, the previous player looses a life. (For further information see (mia, 2016).)

3 Implementation of the Game

For the implementation we use C++ and Qt to build the GUI-Application. As IDE we used Qt-Creator. To implement an artificial intelligence according to the support vector machine learner we used the OpenCV library. The ingame AI is used to distinguish between lies and the truth of announced values. For the prediction we used the last announced value, the actual value and an indicator to show the start of a new game. To train the SVM we additionally used an indicator if the actual (value that previous player announced) value was a lie.
The difference between the strategy for liar detection and gameplay follows in section 4.

4 Setup and Strategies

To measure the performance of the predictor different artificial intelligence approaches were used. For data acquisition we used a homogeneous set of players in our game implementation and let it generate game data. Therefore the output of one AI is the input of the following AI in the game. For each turn we reported:

- player number
- previous player number
- if the turn is the first one in the round (new game)
- new announced value
- actual rolled value
- previous announced value
- does the player look onto the dices or not

Since the actual rolled value is mostly hidden from the players. We decided to learn the predictor the players strategies only based on: new game, last announced value and new announced value (this seems most realistic). We then create liar labels for the datasets due to comparison of the new announced value and the actual rolled value for training purposes. Each dataset is split up to training data and test data. So we train the predictor for the different approaches by using cross validation (determination of parameter C) on the training data and measure the performance (generalization) on the test data. The test set is 30 percent of the data set, the others are training data. In the end we will compare the results of the implemented AIs detecting lies (take the *look at* label and the predictions of our predictor).

We examined three different types of artificial intelligence approaches. First the statistical approach where the AI acts very straight forward. Then an AI with a certain degree of randomness in its call and look behavior. Last we implemented a learning AI with different calling behaviors to examine if the predictor is able to learn the strategy of a learning player. Last we wanted to know how many samples it takes until there is convergence of accuracy in the predictor.

4.1 Statistic Approach

Table 1 shows basic statistic probabilities of the game mia. Note: if the value (call/announcement of previous player) is 21 then the current player can either look up the dices and possibly looses two lives or he can save himself by rolling a 21 as well.

Value	31	32	41	42	43	51	52
%	94	89	83	78	72	67	61
Value	53	54	61	62	63	64	65
%	56	50	44	39	33	28	22
Value	11	22	33	44	55	66	21
%	19	17	14	11	8	6	6

Table 1: Shows the probability of getting a higher dice result than the announced value.

The artificial intelligence based on the statistical approach is quite easy. It calls a the previous player a liar if the probability of lying is greater than the probability to say the truth. This AI always says the truth if possible, otherwise it take a random value greater than the last announced value.

4.2 Approach with certain degree of randomness (Primitive Random)

This AI will call with a certain probability a value that is greater than the actual rolled value if the value is lower than the one of the previous player. Also the AI will look on the dices with a probability that arises with decreasing beat probability (see table 1) of a announced value (a 31 will not be looked at with probability near 100 percent, 21 is looked at with 100 percent). This AI always says the truth if it is possible (meaning the rolled value is greater than the last announced value).

4.3 A SVM learning approach

The next step was to create a new type of artificial intelligence that changes its behavior during the history/time of the game. Therefore, we used a SVM implementation with different behavior by calling a value (meaning lie or tell the truth).

4.3.1 SVM variant 1

This variant of the AI says the truth or uses the next possible value greater than the last announced one.

4.3.2 SVM variant 2

This AI variant always says the truth if it is possible or calls a random value greater than the last announced one.

4.3.3 SVM variant 3

This AI always calls a possible random value that is independent from the actual rolled dice value.

4.3.4 SVM variant 4

This AI always announces a lower or equal value than the actual rolled value if it is possible. Otherwise it uses a random value greater than the last announced one.

5 The Predictor

To predict the different strategical approaches we used a support vector machine (SVM) predictor with radial basis functions. A linear predictor would not have been sufficient due to the complexity of data, but with the SVM-predictor adequate results could be expected. By using SVM we are interested in separating two (or more) classes by a separating hyperplane with maximal margin. The margin is defined with respect to the training points as the minimal distance between the hyperplane and a training point.(See (Schoellkopf and Smola, 2002, 187–227).)

6 Discussion of Results

An overview of the maintained results can be seen in table 2. In all but one case the learned predictor was able to predict the lier labels better than the AIs. Moreover we can conclude that the predictor was able to learn the strategy of the different approaches quite well.

strategy	sdetect	C	pdetect
statistic	0.473	0.0001	0.863
primitive	0.585	0.5	0.849
SVM1	0.964	2.0	0.971
SVM2	0.810	0.6	0.797
SVM3	0.844	3.0	0.971
SVM4	0.642	2.0	0.989

Table 2: Obtained results in the experiments. Strategy, strategies liar detection (sdetect), parameter C for predictor, predictors liar detection (pdetect) on test data. The predictor is a python SVM implementation.

The artificial intelligence based on the statistic approach (see section 4.1) has the worst results of all our experiments. Our predictor was able to achieve a result of 86% and more or less learn the AIs strategy. The approach with a certain degree of randomness (see section 4.2) was able to detect around 60% (the predictor again beat the ingame AI and was able to learn their strategy). The learning of the svm helps to improve the ingame predictions and learn the strategy of the other players (the ingame results are quite as good as the our predictor implemented in python). Summing up we can say that the ingame AIs results are the better the less randomness exist in the calling behavior of the other players. The predictor was able to learn nearly all strategies good. The following convergences consider the convergence of the python predictor not the ingame AI convergence.

For the statistical approach convergence takes place around 4000 samples if we allow one percent of deviation in both directions (see fig. 7). In fig. 6 the convergence can be seen more easily. It is about 4000 samples. Afterwards there is only little variation of the accuracy. For the SVM approaches can be seen that variant 1 and variant 3 converge to the same value of accuracy (0.97). Convergence for both variants is about 2000 samples. Variant 2 of the SVM approach has a lower lever of accuracy meaning that this one is harder to predict correctly than the others (under the given data). The convergence of variant 2 is at around 3000 samples. Variant 4 has the highest rate of accuracy with 98 percent and converges about 2500 samples (compare fig. 5).

Comparing the convergence of the ingame AI (fig. 8) and the python predictor (fig. 1) we can say that both converge near to 1500 samples. The strategy of a lier has been detected by both ingame and python predictor.

Figures 1 to 4 show the classification of the ingame artificial intelligence based on the SVM learner. The green color represents values that are going to be accepted as the truth by other players. Blue areas represent that the actual value is detected as a lie. On the horizontal axis there is the actual value. On the vertical axis there is the last value [1]. On the left half the running game is represented and on the right side new game predictions are depicted. The origin of the image is on the top left side.

[1]The axis are in numerical order. Therefore, non valid dice values for the mia game are represented.

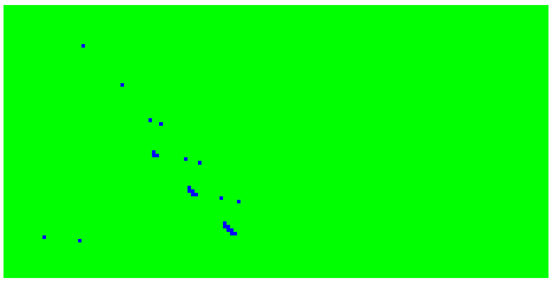
Figure 1: Classification of SVM-AI variant 1.

Figure 1 shows that the SVM-AI (see section 4.3.1) can learn the calling strategy of saying the next greater value. The blue points show that the next greater value is quite often a simple lie.

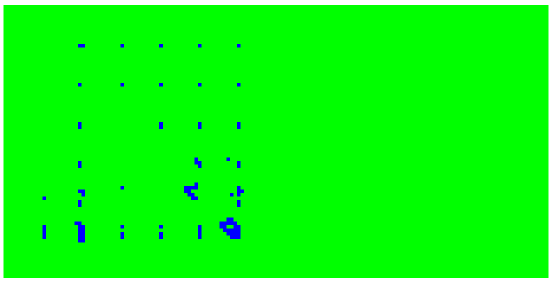
Figure 2: Classification of SVM-AI variant 2

Figure 2 has a green area between the blue dots. That is because the values around 31 and 42 are often true and the AI accepts this calls (see section 4.3.2).

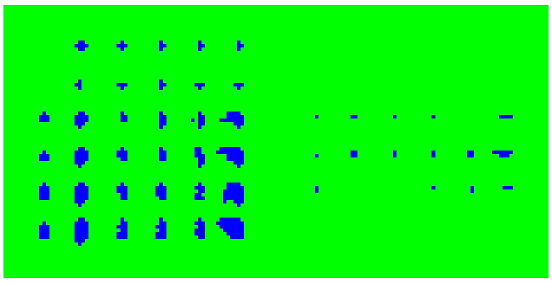
Figure 3: Classification of SVM-AI variant 1

In fig. 3 you can see clearly that the variance of the liars classification increases due to the random calling procedure of the third variant (see section 4.3.3). And since the AI is likely to lie there are blue points on the right side of the image, meaning that start values are considered to be likely a lie.

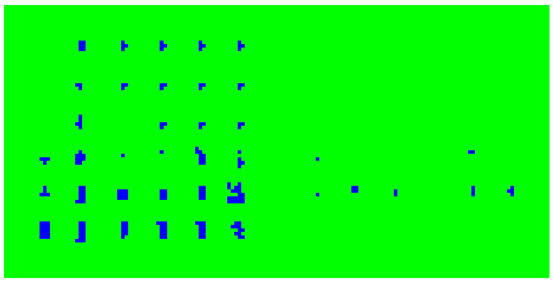
Figure 4: Classification of SVM-AI variant 1

In fig. 4 where the calling behavior is to say a smaller value (see section 4.3.4) if possible the variance of the lier calling is a bit smaller than in section 4.3.3.

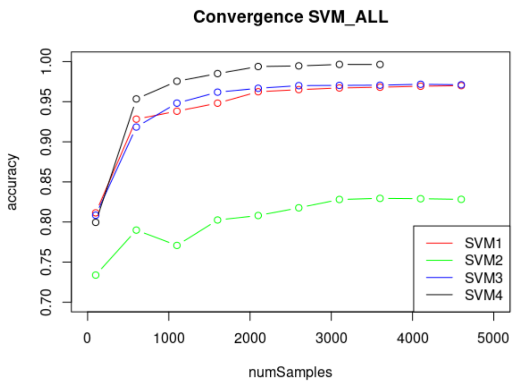

Figure 5: Comparison of predictor based on all four SVM variants data sets.

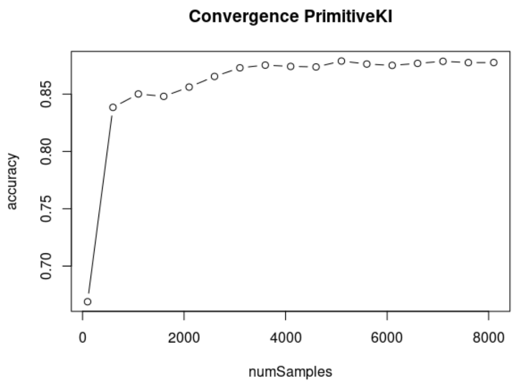

Figure 6: Convergence of the python predictor on data generated by the approach with certain degree of randomness.

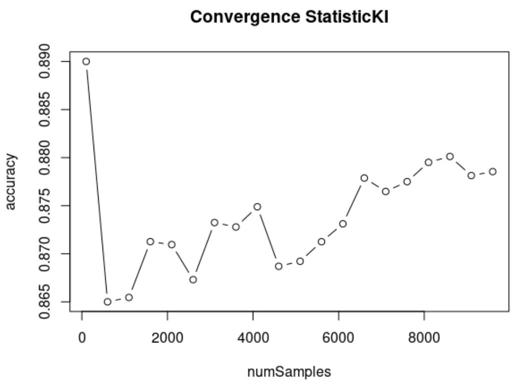

Figure 7: Convergence of python predictor on data generated by the statistical approach.

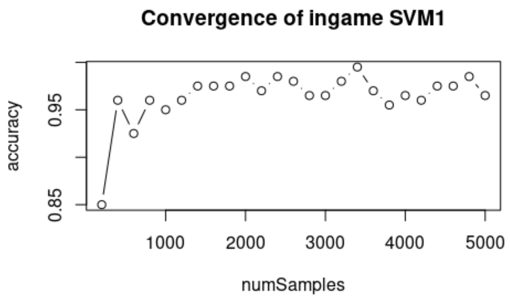

Figure 8: Convergence of the ingame SVM in variant 1.

Acknowledgments

The acknowledgments should go immediately before the references. Do not number the acknowledgments section. Do not include this section when submitting your paper for review.

References

[mia2016] 2016. Mia (game). `https://en.wikipedia.org/wiki/Mia_(game)`.

[Schoellkopf and Smola2002] Bernhard Schoellkopf and Alexander J. Smola. 2002. *Learning with Kernels*. Massachusetts Institute of Technology, Cambridge, Massachusetts, London, England.

A Supplemental Material