An actuarial artificial intelligence for the game rock-paper-scissors

AUTHOR:
Michael B. Jordan[1]

AFFILIATION:
[1]Student Member of the Actuarial Society of South Africa, Cape Town, South Africa

CORRESPONDENCE TO:
Michael Jordan

EMAIL:
bayman1991@gmail.com

POSTAL ADDRESS:
PO Box 50362, Waterfront 8001, South Africa

KEYWORDS:
game theory; AI; machine learning; actuarial science

HOW TO CITE:
Jordan MB. An actuarial artificial intelligence for the game rock-paper-scissors. S Afr J Sci. 2016;112(5/6), Art. #a0154, 3 pages. http://dx.doi.org/10.17159/sajs.2016/a0154

Rock-paper-scissors is a simple game played by two players who use hand gestures that resemble a 'rock' (a fist), a piece of 'paper' (a flat hand) or a pair of 'scissors' (index and middle fingers in the shape of a V). Each player plays a gesture at the same time, with the scoring as follows: 'rock' beats 'scissors'; 'paper' beats 'rock'; and 'scissors' beats 'paper'.

If the game is played only once, no dominating strategy exists and the result is down to chance. If the game is played multiple times the chance element reduces as a player's next gesture is influenced by two factors: the previous result (win, loss or draw) and the previous gesture ('rock', 'paper' or 'scissors').

Previous attempts by RoShamBot to create an artificial intelligence (or AI) for rock-paper-scissors have relied on simple frequency analysis (i.e. the user has played 'rock' the most, therefore the AI must play 'paper' next) or history matching (i.e. the AI matching the last four rounds to a large database of previous games and determining what gesture the user will play next and then countering that gesture).

In an actuarial spirit, I take a stochastic approach and instead of determining the next gesture with certainty, I flex the probabilities for 'rock', 'paper' and 'scissors' and then let my AI randomly choose one. The probabilities are flexed based on previous results and previous gestures.

An outline of various strategies

The question arises as to how the previous results and previous gestures influence the players' next gestures. I propose that there are 10 potential strategies that a player can adopt in a multi-round game of rock-paper-scissors. An example of one strategy is: a player repeats a gesture when they win, alternates gestures when they lose and chooses randomly when they draw. Thus for each of the three results, there are three potential responses (repeat, alternate, random). There are therefore nine (three x three) potential strategies. Another example of this kind is when a player alternates their gestures when they win, chooses randomly when they lose and alternates their gestures when they draw. The tenth strategy is to identify which of the nine strategies the other player has adopted and then to counter that strategy. For example, if an opponent tends to repeat their gestures when they win and if they had won the previous round with the gesture of 'rock', then a player would counter their strategy by using 'paper' in the next round.

It is difficult for a human to accurately determine which strategy their opponent is playing. In order for them to do so they would need to keep track of eight variables:

- The previous gesture
- The previous result
- A count of the number of times their opponent alternates after winning
- A count of the number of times their opponent repeats after winning
- A count of the number of times their opponent alternates after losing
- A count of the number of times their opponent repeats after losing
- A count of the number of times their opponent alternates after drawing
- A count of the number of times their opponent repeats after drawing

The player would also need to do a few calculations. For example, is the count for alternating after winning larger than the count for repeating after winning? If the answer is yes, then what does that say about their next gesture and how do you counter it? The time between game rounds might not be sufficient for an untrained human to adopt the tenth strategy, but it would be very simple for a computer.

Defining the variables

The first thing my AI needs is starting probabilities for each gesture. I assign 33.3% each for 'rock', 'paper' and 'scissors' because, for the first round, it is a game of chance. I create six variables that will track how my opponent plays. I create two variables to store the previous result and previous gesture. I also create an AI Confidence and an AI ForgetLimit (these are discussed later). Finally, I create a random variable that can take any value between 0 and 100.

```
//Thinking
private double repeatStyleWin = 0;
private double repeatStyleLose = 0;
private double repeatStyleDraw = 0;
private double alterStyleWin = 0;
private double alterStyleLose = 0;
```

© 2016. The Author(s). Published under a Creative Commons Attribution Licence.

```
private double alterStyleDraw = 0;
//Counters
private String historyChoice = "none";
private String historyResult = "none";
//Memory
private double confidence;
private double forgetLimit;
//Personality
```

The basics

The human opponents to my AI are allowed to input 'rock', 'paper' or 'scissors' as gestures. Inputting a gesture calls the play method, which works as follows: A random number is generated. If it is less than the probability for 'rock', then my AI chooses 'rock'. If it is less than the probability for 'rock' + 'paper', then my AI chooses 'paper'. If it is larger than the probabilities for 'rock' + 'paper', then my AI chooses 'scissors'. The gesture randomly chosen by the AI is then compared to the opponents input gesture, a result is determined and the scores are appropriately updated. Next the AI reviews the previous result and previous gesture and makes the necessary count that will later determine the opponent's strategy.

```
public void rock(View v){
  play("rock");
}
public void paper(View v){
  play("paper");
}
public void scissors(View v){
  play("scissors");
}
```

Making the AI forget

I then call a method known as Forget&Adjust, which is not very important for the first few rounds but is developed so that my AI can continue to counter its opponent if they change their strategy later on. Opponents are likely to change their strategy if they are losing to my AI so it is important for the AI to detect and adapt if they do so. Forget&Adjust is achieved by simply capping the difference between the alternate counts and the repeat counts. For example, if the repeat count for win is 5 and the alternate count for win is 1, the difference between them is 4. If the cap limit is set as 3, the repeat count will be reduced by 1. In this way, the AI still acknowledges that the opponent is repeating for wins but the repeat count does not increase so much that my AI becomes predictable for an extended period of rounds. Capping the counts allows my AI to react sooner to an opponent who changes their strategy.

The cap limit – that is, the AI ForgetLimit – does not have a fixed value. In the example above I set it to 3, but it could be 1 or 5 or 10. The optimal cap limit is unknown, so I let my AI decide for itself. Before each game with a new opponent, the AI ForgetLimit is determined randomly. At the end of the game the AI notes if that cap limit was successful or not. When a new opponent starts the game, a new ForgetLimit is randomly chosen and the success of that game is noted. After this training period, my AI determines which ForgetLimit was empirically the most successful and then adopts that value. This value is semi-permanent. A series of overall game losses would result in the AI wanting to enter training again. The AI will communicate this eventuality to me and I will have the ability to override the decision, as it would not be desirable to enter the AI into a competition when its cap limit is randomly high, because it would be potentially vulnerable to an opponent who changes their strategy. The more games the AI completes in training mode, the more likely an optimal value for the ForgetLimit will be found.

```
public void forgetAndAdjust(){
  if (alterStyleWin > repeatStyleWin + forgetLimit){
    alterStyleWin = alterStyleWin - 1;
  }
  if (repeatStyleWin > alterStyleWin + forgetLimit){
    repeatStyleWin = repeatStyleWin - 1;
  }
  if (alterStyleDraw > repeatStyleDraw + forgetLimit){
    alterStyleDraw = alterStyleDraw - 1;
  }
  if (repeatStyleDraw > alterStyleDraw + forgetLimit){
    repeatStyleDraw = repeatStyleDraw - 1;
  }
  if (alterStyleLose > repeatStyleLose + forgetLimit){
    alterStyleLose = alterStyleLose - 1;
  }
  if (repeatStyleLose > alterStyleLose + forgetLimit){
    repeatStyleLose = repeatStyleLose - 1;
  }
}
```

Resetting the probabilities

Before I adjust the probabilities to counter my opponent's strategy, I reset the probabilities for 'rock', 'paper' and 'scissors' to 33.3%. I want my AI to make its next decision based only on my opponent's strategy, the last result and the last gesture they played. Their strategy already incorporates all historical results and gestures and if not reset, the probabilities for 'rock', 'paper' and 'scissors' would be distorted. Therefore I flex the probabilities, that is, I adjust them from 33.3% to a new value, and then reset them to 33.3% before I adjust them again.

```
paperChance = 33.3;
rockChance = 33.3;
```

Flexing the probabilities

Once the probabilities have been reset, I flex them to counter my opponent's strategy. (The logic is not straightforward so I will provide an example at the end.) The method starts by looking at the opponent's previous gesture, and comparing the repeating count with the alternating count for the previous result. There are three possible outcomes. Outcome 1: if the repeating count is greater than the alternating count, then it follows that the opponent's result (win/loss/draw) with a particular gesture ('rock', 'paper' or 'scissors') is likely to prompt that same gesture again. Therefore I increase the probability for the gesture that beats that one and slightly increase the probability for the same gesture. Outcome 2: if the alternating count is greater than the repeating count, then it follows that the opponent's result with that gesture is likely to prompt a different gesture in the next round. Therefore I increase the probability for the gesture that loses to that one and slightly increase the probability for the winning gesture. Outcome 3: if the repeating count is the same as the alternating count, I change nothing.

For example: if the opponent previously won with 'rock', and they tend to repeat winning gestures, then I assume that they are more likely to play 'rock' again. To counter this play, the probabilities are flexed to favour 'paper', which would result in a win for the AI. The probabilities are also flexed to slightly increase the chance for 'rock' as the second best result is a draw; the probability for 'scissors' would be decreased.

```
public void adjustStrategy(String choice, double alter, double repeat){
  if (choice.equalsIgnoreCase("rock")){
    if (repeat > alter) {
      rockChance = rockChance + (confidence/2);
    paperChance = paperChance + confidence;
    //She results with rock, likely to do rock again, paper is best strategy next
    //Second best is to play rock and get draw
  }
  else
  if(alter > repeat){
    rockChance = rockChance - (confidence / 2);
    paperChance = paperChance - confidence;
    //She results with rock, likely not to do rock again, scissors is best strategy next
    //Second best is to play rock and go for a 50/50 win/lose
  }
```

Starting confidence

I do not know by how much the probabilities should flex. Therefore I create a variable known as AI Confidence. As for AI ForgetLimit, the AI will learn the optimal value by going through a training session in which it assigns random values for AI Confidence and then selects the value that has the highest end game success rate. I call this type of learning Global Machine Learning, as the learning occurs across various games with various opponents. Unlike AI ForgetLimit, AI Confidence can change during the game. If the AI wins, then its confidence increases; but if the AI loses, then its confidence decreases – I call this Local Machine Learning, as the learning occurs during the game with one opponent.

I am careful when it comes to increasing or decreasing confidence. I only increase confidence when the AI wins with the gesture that has the highest probability, and I only decrease confidence when the AI loses with the gesture that has the highest probability. For example, with the probabilities of 'rock' at 50%, 'paper' at 30% and 'scissors' at 20%, if my opponent plays 'scissors' and my AI plays 'paper' it will lose, but it should not be punished because 'paper' was not the highest probability. If my opponent plays 'scissors' and my AI plays 'rock' it would win and should be rewarded, as 'rock' was the highest probability. The higher the AI Confidence, the more the probabilities will be increased or decreased. Another way of thinking about it is that the AI Confidence represents the clarity of the opponent's strategy. An opponent who always repeats after a win and who always alternates after a loss will cause the AI Confidence to increase. An opponent who is completely random with their gestures will cause the AI Confidence to decrease. Note that it is possible for the AI to lose overall and yet end with a higher confidence and it is possible for the AI to win overall and yet end with a lower confidence. Although this scenario is unlikely as result and confidence are correlated to some extent, it is not improbable because we are rewarding only some wins and punishing only some losses. In the example above, if the opponent plays 'paper' and by chance the AI plays 'scissors' it will win, but should not be rewarded as the opponent has not played what we anticipated them to play; in fact, one could argue that after this type of win the AI should be punished and its confidence reduced. This scenario opens up another possibility – one could base the confidence not on the result and the AI's choice but rather on whether the opponent plays the gesture that they were most anticipated to play. In this way, if the opponent is following a clear strategy, the probabilities will flex more to counter. The worst case scenario is if no clear strategy for the opponent can be identified and thus AI Confidence will reduce and will leave the probabilities at 33.3% for each gesture. In this scenario the final outcome will be based on luck.

My AI can be considered successful if it wins the majority of games, it is possible for it to lose and it is not designed to guarantee a win every time. One of the benefits of taking on a stochastic nature is that my AI cannot be outwitted by its opponent. If my AI was deterministic, then patterns would emerge which could be taken advantage of by its opponents. To further prevent being outwitted, the final values for AI Confidence and AI ForgetLimit should be kept secret from all parties, including myself as the creator.

A colleague pointed out the weakness of these check digit verification (or CDV) checks and recommended using a stochastic variable in the key to improve security. My AI could use this more secure method of encryption for protecting its AI Confidence and AI ForgetLimit.

Fluctuating confidence

One consideration is by how much the confidence should increase or decrease. Again, the optimal value of this variable is unknown. I could make it random and let the AI decide the optimal value through the same training process for AI Confidence and AI ForgetLimit. But that option is not ideal as the more variables that require optimisation, the more extended the AI's training period becomes. That being said, I arbitrarily adjust confidence by 1 point, but I acknowledge that this value may not be optimal. I could train the AI for AI Confidence and AI ForgetLimit and then create AI Adjust as a random variable and train again. But optimising two variables and then another is inferior to simultaneously optimising all three because of potential dependency.

Another consideration is whether there should be limits for AI Confidence – the minimum could be zero but what should the maximum be? Also, should the probabilities be limited? During my testing of earlier versions, I did occasionally get negative probabilities and probabilities exceeding 100%; these unrealistic values distort the AI's performance and can be avoided by either limiting the probabilities or limiting AI Confidence. I also question by what fraction of confidence the second gesture should be adjusted. The higher the fraction, the lower I must limit the AI Confidence if I want to avoid impossible probabilities. The limit for AI Confidence must be less than $(33.3(x/x+1))$, where $1/x$ represents the selected fraction. Therefore, if the second gesture changes by half the confidence then the limit is 22.2, and if the second gesture changes by a third of the confidence then the limit is 25.

Future applications

Humanity is on a journey to make a cognitive computer that can rival our own intellect. We are going to create machines that humans can interact with on a social and emotional level. Playing games together is one way we can socially connect with them.

When we think of an AI we think of it as an individual computer, but it could very well take on a hive model. This AI that plays rock-paper-scissors could be just one of millions of AIs in a hive. Each AI could be assigned certain tasks instead of having one system that does everything. It is my hope that the AI described above gets adopted into future hives and is the one called upon when a computer is challenged to a game of rock-paper-scissors.

This application is just the beginning. This AI can be extended to determine strategies for more complicated games like American football. AI could also provide solutions in education. AI is about using rules, semantics and logic to program a machine to learn – why not reverse engineer these instructions into a study method that helps students to understand information in the most efficient way possible?

I have already started building an AI for American football and have already started developing the most efficient way to study. Actuarial science has provided us with a skill set to approach this subject and I believe actuaries will play a major role in creating the cognitive computer.

Marc Andreesen predicted that there would be two kinds of people in tomorrow's world: those who tell computers what to do and those who are told by computers what to do. I think those in the actuarial profession will agree that we do not want to be the latter.

Download an app of this AI here: https://play.google.com/store/apps/details?id=mjprojects.rockpaperscissors&hl=en

Or watch the YouTube video here: https://youtu.be/K4uH3BqPe3c