CRWN 88
Homework 2

Anders Poirel

October 10, 2018

1 FSA Predictor designs

Note: The motivation behind the designs is explained in Question (1)

Deterministic design As the strategy discussed in (1) is non-deterministic, we cannot implement it perfectly through a deterministic FSA design. However, we can approximate a stochastic design through pseudo-random number generation. We here implement a linear congruential generator using a FSA. Pick a modulus m, and let the set of states be $S = \{1, 2, \dots, m-1\}$ (note that this set is finite, thus satisfying the definition of a FSA). Pick an initials state s_0 from that set. Define the next-state function f as follows:

$$f(s) = (as + c) \mod m$$

For some integers a and c. Finally, for each $s \in S$ define an output functions as follows: "Predictor guesses Heads" if $s < \frac{m}{2}$ and "Predictor guesses Tails" if $s \geq \frac{m}{2}$.
For "good" choices of initial states, a, m and c, the sequence of successive states s passes weak statistics tests and thus provides a reasonable approximation of a stochastic design. In our implementation, we use $m = 2^{28}$, $c = 1013904223$ and $a = 1664525$.

Stochastic design We implement exactly the strategy discussed in (1). Let the set states be $\{H, T\}$, denoting the states "Predictor guesses Heads" and "Predictor guesses Tails" respectively. Pick arbitrarily the initial state. Define the next-state function f as follows: f keeps the current state with probability 0.5 or assigns the other state with probability 0.5.
We can encode this design in the following transition matrix:

$$\begin{bmatrix} P(H|H) & P(T|H) \\ P(H|T) & P(T|T) \end{bmatrix} = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$$

2 Questions

(1) We begin our discussion of the best possible design by determining the Nash equilibrium for this game. First, given the knowledge that one of the participants plays a pure strategy, the other participant is has an incentive to change his strategy: for the Predictor, this is picking always the same as the player, and for the Player, this is picking always the opposite of the Predictor. Hence no pure strategy equilibrium can exist for this game. We therefore determine the mixed strategy equilibrium for this game. Denote H the strategy of picking Heads and T the strategy of picking Tails. The payoff matrix for the penny game is then:

		Player H	*Player* T
Predictor	H	$(1,0)$	$(0,1)$
	T	$(0,1)$	$(1,0)$

Now let x be the probability that the Predictor uses strategy H, and $(1-x)$ the probability that he uses T, y be the probability that the Players uses strategy H, and $(1-y)$ the probability that he uses T. To find the equilibrium, we determine the values of y such that the Predictor's expected payoff is the same for both possible Player moves by solving

$$\begin{aligned} 1 \cdot y &= 1 \cdot (1-y) \\ 2y &= 1 \\ y &= 0.5 \end{aligned}$$

By symmetry, we also have $x = 0.5$. Therefore in equilibrium, both participants choose between Heads and Tails strategies with equal probability.
To see why we should design the Predictor automaton to play a mixed strategy with this probability, recall the following property of a Nash equilibrium: *Supposing the first participant plays the mixed strategy outlined above, the second has no incentive to switch strategies.* That is, if the automaton picks either Heads or Tails each with probability 0.5, the Player cannot improve his expected payoff (note that we can clearly see that the Player can not get odds of winning a given round of more than 50% in this situation.
The deterministic Predictor FSA we described approximates well this strategy, given a good choice of parameters for the linear congruential generator. However, for a deterministic FSA like the one we implemented, the pattern of Heads and Tails is periodic and hence becomes predictable after enough repetitions. Furthermore the pattern will be the same each game, we would need to provide a new random seed to the generator each game to avoid this.

(2) Clearly, if it appears that one of the participants is playing a pure strategy, the best strategy for the other participants will be fully determined by what the first is playing. However, as seen above, assuming one of the participants are

playing a mixed strategy, the best strategy for both participants to is choose randomly between Heads and Tails, each with probability 0.5. A deterministic FSA playing against such a Player would win no more than 50% of the time (as seen in (1)). Furthermore, once the human Player has guessed the pure strategy used by a deterministic FSA Predictor, they can win every round by picking the opposite of what they know the Predictor will pick. Therefore, any Predictor built towards "predictive" and deterministic strategy can easily be exploited by a competent Player. This implies that a good Predictor design will necessarily be stochastic.

3 Code for deterministic design

C99 code used for play-testing the deterministic design:

```
#include <stdlib.h>
#include <stdio.h>

// parameters of the pseudorandom number generator
#define MOD 268435456
#define MULT 1664525
#define INCR 1013904223
#define MID MOD/2

// seed of the pseudorandom number generator
#define SEED 57

// linear congruential pseudo-random number generator
int psrd(int i)
{
    return (i*MULT+INCR) % MOD;
}

int main(void)
{
    char c;
    int curr = SEED;
    printf("Press m to get a new Predictor move, or q to quit
        program:\n");
    while(c != 'q') {
        c = fgetc(stdin);
        if (c == 'm') {
                curr = psrd(curr);
                if (curr < MID)
                    printf("Pick 1.\n" );
                else
                    printf("Pick 0.\n");
            }
    }
}
```

As the stochastic design was overall simpler, it was play-tested by simply using dice.