Learning from Randomness

Jonathan Stokes

July 5, 2024

1 Introduction

I have been able to recreate the results posted previously on training a transformer model to play GOPS but not improve them significantly using bootstrap training. In bootstrap training a data set is generated using uniformly random agents, that data is used to train a transformer agent, the transformer agent playing against a uniformly random agent is then used to generate a second data set, etc.

The difficultly I found in improving the performance of the transformer agent led me to wonder if one can learn from randomness? In this case the uniformly random agent which the transformer agent is playing against. I think the answer is that it is always possible to learn from randomness, but how hard it is to learn from randomness depends not only on the complexity of what is being learned but also on whether or not the learning verification method is deterministic or not. In the case of a simplified version of GOPS I argue that if the learning verification method is not deterministic learning from a random agent is more difficult than it appears from the complexity of what is being learned alone.

1.1 Example of Learning from Randomness

Lets assume there is a door lock controlled by a 2 digit keypad, zero and one. The code that must be entered into the keypad to unlock the door c is unique and length 16. In this case there are

$$2^{16} = 65,536 \tag{1}$$

possible codes. The probability of randomly entering c into the keypad is,

$$p_c = \frac{1}{2^{16}} \tag{2}$$

If one tried to randomly entering 16 digits into the keypad repeatedly, letting k be the number of times c was entered, k would follow a Binomial distribution.

$$p(k,n) = \binom{n}{k} p_c^k (1-p_c)^{n-k} \tag{3}$$

with mean $\mathbb{E}[p(k,n)] = np_c$. Therefore one should expect to have entered c at least once after $n = \frac{1}{p_c} = 2^{16} = 65,536$ trials. In other words, after 2^{16} random trials one should expect to have learned c.

This is a general example of learning from randomness and it could easily be reframed for other scenarios. However I will argue that this only works well when the verification method for the random event, entering random digits, is deterministic. If this is not the case and the keypad is malfunctioning such that it unlocks 50% of the time regardless of the code entered, it is harder to learn c.

To see this notice that if the keypad is malfunctioning and the random sequence of 16 digits entered is r where $r \neq c$, the number of times r will fail to unlock the keypad is Binomially distributed with mean mp_f, where m is the number of times r is entered and p_f is the probability of r failing to unlock the door. If $p_f = \frac{1}{2}$ then $\mathbb{E}[p(k,m)] = \frac{m}{2}$ and for $k = 1$ we would expect it to take at least $m = 2$ trials before r fails to unlock the door and we can assert $r \neq c$. Therefore instead of it taking an expected 2^{16} trials to find c, in the case of the malfunctioning keypad we would expect it to take $2^{17} = 131,072$ trials.

If we want 0.99 confidence that we have found c instead of 0.5 then we need to find n such that,

$$\begin{aligned} \sum_{k=1}^{\infty} p_{f,k} &= \sum_{k=1}^{\infty} \binom{n}{k} p_f^k (1-p_f)^{n-k} && (4) \\ &\geq \alpha && (5) \end{aligned}$$

where $\alpha = 0.99$. This implies that for $k = 0$,

$$\begin{aligned} \binom{n}{0} p_f^0 (1-p_f)^{n-0} &< 1-\alpha && (6) \\ (1-p_f)^n &< 1-\alpha && (7) \\ n \log(1-p_f) &< \log(1-\alpha) && (8) \end{aligned}$$

and since $\log(1-p_f) < 0$,

$$n > \frac{\log(1-\alpha)}{\log(1-p_f)} \tag{9}$$

So if $p_f = \frac{1}{2}$,

$$n > \frac{\log(0.01)}{\log(0.5)} > 6.64 \tag{10}$$

So it would be expected to take $7(2^{16}) = 458,752$ trials to find c. Additionally notice that as $p_f \to 0$ or $\alpha \to 1$, $n \to \infty$ and it becomes computationally intractable to learn c.

1.2 Learning from Randomness in GOPS

How is learning from randomness related to learning good strategies in GOPS by playing against the uniformly random agent? The short answer is I am not sure, learning a GOPS strategy is more complicated than learning a key code and I do not see a reduction from the former to the latter.

Let us instead consider a simpler form of GOPS, *1 Round GOPS*. In *1 Round GOPS* there is 1 round and m ordered bid cards. Who ever bids the highest card wins. In *1 Round GOPS* clearly the optimal strategy is to always bid the highest card, but how easy is it to learn this when playing against a random agent? Lets consider the case where $m = 3$ and the possible bids are $\{1, 2, 3\}$, see Table 1.

Player 1 Bid	Player 2 Bid	Player 1 Result
1	1	Tie
1	2	Lose
1	3	Lose
2	1	Win
2	2	Tie
2	3	Lose
3	1	Win
3	2	Win
3	3	Tie

Table 1: Bid results for Player 1 where $m = 3$.

Looking at Table 1 we see that $\frac{1}{3}$ of the time Player 1 will win with a suboptimal bid, which is analogous to unlocking the door with incorrect code r.

Let T denote the number of Player 1 ties, W the number of wins, L the number of losses, W_{opt} the number of Player 1 optimal wins, and F_{subopt} the fraction of Player 1 wins that are the result of suboptimal moves. Now for arbitrary m the number of ways Player 1 can tie is,

$$T = m \tag{11}$$

since to tie Player 1 must play the same card as Player 2 and there are only m cards Player 2 can play. The number of ways Player 1 can wins is,

$$W = \frac{m^2 - m}{2} \tag{12}$$

since the number of ways to win is 0 for the lowest bid, 1 for the second lowest bid, etc up to the m_{th} bid which has $m-1$ ways to win. In other words, the total ways for Player 1 to win is just the sum $\sum_{i=0}^{m-1} i = \frac{m^2-m}{2}$.

The number of ways to lose is also,

$$L = \frac{m^2 - m}{2} \tag{13}$$

which follows from W by symmetry. The number of ways to win with an optimal bid is,

$$W_{opt} = m - 1 \tag{14}$$

since the only way to not win is if ones opponent also makes an optimal bid.

So what is the fraction of wins resulting from suboptimal moves for arbitrary m? Lets denote this fraction F_{subopt},

$$\begin{aligned} F_{subopt} &= \frac{\frac{m^2-m}{2} - m + 1}{\frac{m^2-m}{2}} && (15) \\ &= \frac{m^2 - 3m + 2}{m^2 - m} && (16) \\ &= 1 - \frac{2}{m} && (17) \\ &\to 1 && (18) \end{aligned}$$

as $m \to \infty$.

Figure 1 plots Equation (17) as a function of m showing how as m increases F_{subopt} it tends towards 1.

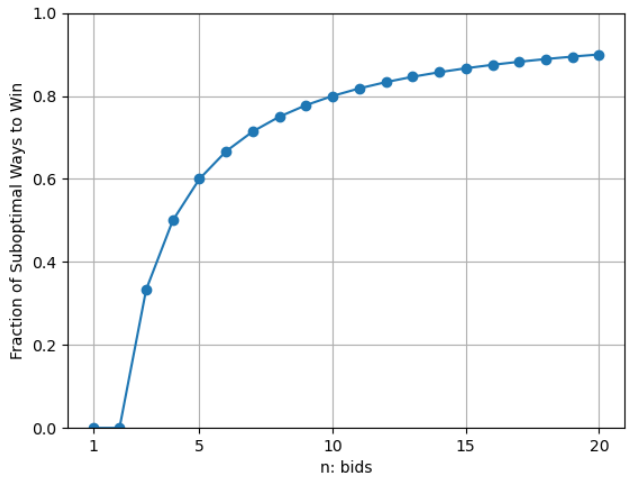

Figure 1: Fraction of suboptimal ways for Player 1 to win.

Now for instance returning to *1 Round GOPS* where $m = 13$. In this case,

$$F_{subopt} = 1 - \frac{2}{13} = 0.846 \tag{19}$$

In other words with $m = 13$ more than 84% of the time a win will result from a suboptimal move in *1 Round GOPS*. Recalling that $(1 - pf)$ in Equation (9) is the probability of unlocking the door with an incorrect keycode r which is analogous here to winning *1 Round GOPS* with a suboptimal bid. Therefore setting $(1 - p_f) = 0.84$ in Equation (9) for learning with randomness we find that for $m = 13$ we would have to see a move result in a win 28 times against a random agent before we have 99% confidence that the move is optimal. Figure 2 shows the wins required to be confident a bid is optimal as a function of the number of bids m and the required confidence α.

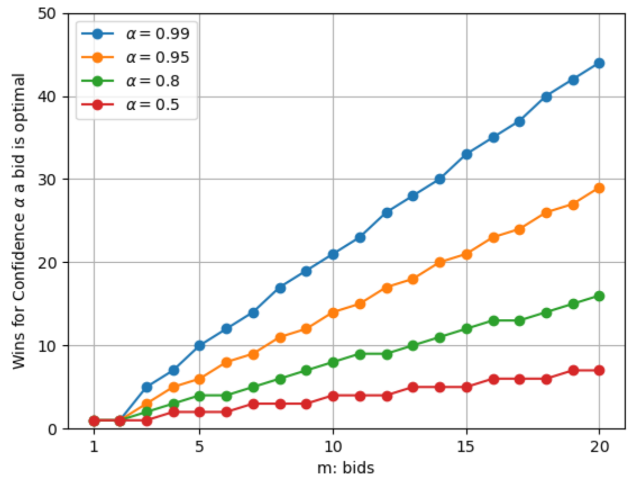

Figure 2: Wins required for confidence α a bid is optimal.

2 Conclusion

The point of this brief write up has been to show that while it is not impossible, it may be more difficult than it appears to learn an optimal strategy in a game like GOPS by playing against a uniformly random agent. This difficultly results from one often winning against a uniformly random agent despite making suboptimal moves, a situation which makes it is hard to determine whether or not a winning strategy is optimal.