random_resistance_strategy

February 4, 2021

1 Winning probability of a random strategy

On this notebook we compute the probability that the resistance wins the game under a naive random strategy: 1. All players make their decisions independently (of each other and the game history). 2. Spies sabotage with probability 50%. 3. The team leader chooses a team from a uniform distribution on the set of all subsets of players of the 'size' given by the table of the rules.

[1]:
```
from scipy.special import binom
```

[2]:
```
N = 5 # number of players
R = 3 # number of resistance fighters
S = 2 # number of spies in game
M = [2, 3, 2, 3, 3] # mission sizes
m = len(M) # number of missions
```

1.1 P(mission is won)

Let W be the event that the mission is **w**on by the resistance fighers and let *size* be the number of players in the missions team. Then

$$P(W) = \sum_{s=0}^{\min\{S,\text{size}\}} P(s \text{ spies in mission}) \cdot P(W \mid s \text{ spies in mission}) = \sum_{s=0}^{\min\{S,\text{size}\}} \frac{\binom{S}{s} \cdot \binom{R}{\text{size}-s}}{\binom{N}{size}} \frac{1}{2^s}$$

[3]:
```
def mission_success_prob (mission_size, verbose = False):
    P_W = 0
    for s in range (1 + min(S, mission_size)): # s is number of spies in mission
        # number of spies in mission (s) has hypergeometric distribution␣
↪(drawing without replacement)
        p_s = binom(S, s) * binom(R, mission_size - s) / binom(N, mission_size)
        p_success_given_s = .5 ** s # mission succeeds iff all s spies vote for␣
↪success
        if verbose:
            print ("s=", s, "\tp_s=", p_s, "\tp_success_given_s=",␣
↪p_success_given_s)
        P_W += p_s * p_success_given_s
    return P_W
```

[4]:
```python
mission_success_prob(2, True)
```

```
s= 0    p_s= 0.3        p_success_given_s= 1.0
s= 1    p_s= 0.6        p_success_given_s= 0.5
s= 2    p_s= 0.1        p_success_given_s= 0.25
```

[4]:
```
0.625
```

[5]:
```python
mission_success_prob(3, True)
```

```
s= 0    p_s= 0.1        p_success_given_s= 1.0
s= 1    p_s= 0.6        p_success_given_s= 0.5
s= 2    p_s= 0.3        p_success_given_s= 0.25
```

[5]:
```
0.47500000000000003
```

1.2 P(game is won)

We perform m independent Bernoulli trials with different success probabilities. The game is won for the resistance if at least 3 missions are won by them.

[6]:
```python
winprob = 0
p = [None] * 5
a = [None] * 5
for outcome_idx in range(2**m): # loop over all outcome combinations of all␣
↪missions
    for j in range(m):
        a[j] = int(bool(outcome_idx  & 1 << j)) # j-th bit
        p[j] = mission_success_prob(M[j])
    if sum(a) >= 3: # more than half of the 5 missions are successes
        q = 1.0
        for j in range(5):
            q *= p[j] ** a[j] * (1. - p[j]) ** (1. - a[j])
        print ("{:2d}-th winning outcome {} has probability {:.4f}".
↪format(outcome_idx, a, q))
        winprob += q

print ("\nProbability that resistance wins game is {:.5f}".format(winprob))
```

```
 7-th winning outcome [1, 1, 1, 0, 0] has probability 0.0511
11-th winning outcome [1, 1, 0, 1, 0] has probability 0.0278
13-th winning outcome [1, 0, 1, 1, 0] has probability 0.0511
14-th winning outcome [0, 1, 1, 1, 0] has probability 0.0278
15-th winning outcome [1, 1, 1, 1, 0] has probability 0.0463
19-th winning outcome [1, 1, 0, 0, 1] has probability 0.0278
21-th winning outcome [1, 0, 1, 0, 1] has probability 0.0511
22-th winning outcome [0, 1, 1, 0, 1] has probability 0.0278
23-th winning outcome [1, 1, 1, 0, 1] has probability 0.0463
```

```
25-th winning outcome [1, 0, 0, 1, 1] has probability 0.0278
26-th winning outcome [0, 1, 0, 1, 1] has probability 0.0151
27-th winning outcome [1, 1, 0, 1, 1] has probability 0.0251
28-th winning outcome [0, 0, 1, 1, 1] has probability 0.0278
29-th winning outcome [1, 0, 1, 1, 1] has probability 0.0463
30-th winning outcome [0, 1, 1, 1, 1] has probability 0.0251
31-th winning outcome [1, 1, 1, 1, 1] has probability 0.0419

Probability that resistance wins game is 0.56598
```

[ ]:

```

```