JUMP CHALLENGE PART II

LESLIE LEI
LL133@DUKE.EDU

1. Arbitrary Input

In first section we establish the existence of winning probability for any allowed betting strategy through absorbing markov chain. In the next section we establish the existence of a betting strategy that is optimal through markov decision process. Finally we introduce a policy iteration algorithm, which output converges to the optimal betting strategy pointwise within finite steps.

1.1. **Markov Chain.** This game is essentially a gambler ruin problem. At each step, the gambler can either increase his money from x to $x + s(x)$ with probability p or lose his bet and decrease his money from x to $x - s(x)$ with probability $1 - p$.

When N is a natural number, consider the set of all possible strategies that satisfies the constraints, Σ. Each strategy can be represented as a vector $s \in \{0, 1, \ldots, N\}^{N+1}$: $s(x)$ means betting $s(x)$ dollars with balance x. Let X_t denote his balanace at time t, and we have $P(X_{t+1} = x + s(x)|X_t = x) = p$ and $P(X_{t+1} = x - s(x)|X_t = x) = 1 - p$. When N is a natural number, the state space S is finite; for $a \in S$, define the first hitting time to be $V_a := \inf\{t \in \mathbb{N}|X_t = a\}$.

We can capture the strategy in a matrix Q_s, a $(N + 1)$-by-$(N + 1)$ transition matrix (each row sums to 1) with entries

$$Q_s(i,j) = \begin{cases} p & \text{if } (i,j) = (x, x + s(x)) \\ q & \text{if } (i,j) = (x, x - s(x)) \\ 0 & \text{otherwise} \end{cases}$$

For a given strategy $s \in \Sigma$, we are interested in the probabilities that the gambler wins, i.e. the house goes bankrupt before he does. When N is finite, this is the exiting probability of hitting N before hitting 0; we can compute it through the following result:

Claim 1. *Consider a Markov chain with finite state space S. Let a and b be two points in S, and let $C = S - \{a, b\}$. Suppose that $h(a) = 1$, $h(b) = 0$ and for $x \in C$ we have*

$$h(x) = \sum_y P_s(x, y)h(y)$$

If $P(V_a \wedge V_b < \infty|X_0 = x) > 0$ for all $x \in C$, then $h(x) = P(V_a < V_b)$.

Proof. See Theorem 1.27 from Reference 3. □

We apply this claim to $S = \{0, 1, 2, \ldots, N\}$ and $(a, b) = (N, 0)$. For $x \in S - \{a, b\}$, let P_s be the transition matrix, which is a submatrix of Q_s. Then define two $(N-1)$ column vectors

$$v^s = \begin{pmatrix} v(1) \\ v(2) \\ \vdots \\ v(N-1) \end{pmatrix}, r_s = \begin{pmatrix} r(1) \\ r(2) \\ \vdots \\ r(N-1) \end{pmatrix}$$

where $v(i) = P(V_N < V_0 | X_0 = i)$ and

$$r(i) = \begin{cases} p & \text{if } i + s(i) = N \\ 0 & \text{otherwise} \end{cases}$$

Then the recurrence can be expressed as:

$$v^s = r_s + P_s v^s$$

Claim 2. *Suppose $v^s = r_s + P_s v^s$. Then it has the following properties:*

(1) *If $s(i) > 0$ for all $i \in \{1, 2, \ldots, N-1\}$, then*

$$v^s = (I_{N-1} - P_s)^{-1} r_s$$

(2) *If $v > r_s + P_s v$, then $v > v_s$.*

(3) *If $v < r_s + P_s v$, then $v < v_s$.*

Proof. First note that $P_s^n(i, j)$ is the probability of ending up with balance j when starting with balance i in n steps. For any strategy with $s(i) > 0$, all states other than 0 and N are transient. To see this, consider the sequence $\{x_i\}$ defined by $x_0 = i, x_{i+1} = x_i + s(x_i)$. This sequence is strictly increasing and we have $x_m = N$ for some $m \in \mathbb{N}$. Thus there is a non-zero probability of reaching N in m step, i.e. $P_s^m(i, N) = p^m > 0$. Hence, by geometric distribution we have $\lim_{n\to\infty} P_s^n(i, j) \leq \lim_{n\to\infty} 1 - P_s^n(i, N) \leq \lim_{k\to\infty} 1 - (1 - p^m)^k = 0$ for all $i, j \in \{1, 2, \ldots, N-1\}$. Thus

$$\lim_{n\to\infty} P_s^n = 0_{(N-1)\times(N-1)}$$

On the other hand,

$$\begin{aligned}
(I - P_s)\sum_{i=0}^{n} P_s^i &= I - P_s^{n+1} \\
v^s &= r_s + P_s v^s \\
&= (I + P_s) r_s + P_s^2 v^s \\
&\vdots \\
&= \sum_{i=0}^{n} P_s^i r_s + P_s^{n+1} v^s
\end{aligned}$$

As $n \to \infty$, we have

$$\begin{aligned}
\lim_{n\to\infty} \sum_{i=0}^{n} P_s^i &= (I - P_s)^{-1} \\
v^s &= \lim_{n\to\infty} \sum_{i=0}^{n} P_s^i r_s + P_s^{n+1} v^s \\
&= (I - P_s)^{-1} r_s
\end{aligned}$$

The other two statement follows easily as $v > r_s + P_s v$, then $v > \sum_{i=0}^{n} P_s^i r_s + P_s^{n+1} v$ for all $n \in \mathbb{N}$ and thus $v > (I - P_s)^{-1} r_s$ by the same argument.

$\square$

1.2. **Markov Decision Process.** The above equation reminds of Markov Decision Process in general (See Reference 1). Now define *optimal winning probability*

$$v^*(x) = \max_{s\in\Sigma} v^s(x)$$

and

$$\mathcal{L}v := \max_{s\in\Sigma}\{r_s + P_s v\}$$

where max is taken componentwise. Both are well defined since Σ is finite.

Now we take a digression by introducing a discounting factor, λ. Intuitively, if we think of $v_s(i)$ as expected reward when starting at balance i, we will discount the expected reward of next time period by λ. In the discounted model, there is a nice relationship between $\mathcal{L}$ and v^*:

Claim 3. *Define*

$$\mathcal{L}_\lambda v := \max_{s\in\Sigma}\{r_s + \lambda P_s v\}$$

for $0 < \lambda < 1$*. Then* $\mathcal{L}_\lambda$ *is a contraction mapping on the space* $[0,1]^{N-1}$ *with sup norm. Therefore there exists a unique* $x \in [0,1]^{N-1}$ *s.t.* $\mathcal{L}_\lambda x = x$*.*

Proof. Fix $s \in \Sigma$ and consider $L_{\lambda,s} : v \to r_s + \lambda P_s v$. Then for any two vectors $u, v \in [0,1]^{N-1}$

$$||L_{\lambda,s}(u-v)|| = ||\lambda P_s(u-v)|| \leq \lambda ||P_s|| \cdot ||u-v|| \leq \lambda ||u-v||$$

Here we use the fact that a sub-stochastic matrix has norm less than 1 (See Reference 2). Now take supremum over all $s \in \Sigma$ and $\mathcal{L}_\lambda$ is a contraction mapping as well. The remainder of the claim follows from Banach Fixed-Point Theorem (see Reference 4). □

Claim 4. *Define*

$$v^*_\lambda = \max_{s \in \Sigma} \{v^s_\lambda | v^s_\lambda = r_s + \lambda P_s v^s_\lambda\}$$

Then v^*_λ *has the following properties:*

(1) *If* $v \leq \mathcal{L}_\lambda v$*, then* $v \leq v^*_\lambda$*.*

(2) *If* $v \geq \mathcal{L}_\lambda v$*, then* $v \geq v^*_\lambda$*.*

(3) v^*_λ *satisfies the equation* $\mathcal{L}_\lambda v^*_\lambda = v^*_\lambda$*.*

Proof. If $v \leq \mathcal{L}_\lambda v$, then for each $i \in \{1, 2, \ldots, N_1\}$, there exists $s_i \in \Sigma$ such that $v(i) \leq r_s(i) + \sum_j P_s(i,j)v(j)$. Define a new strategy $s = (s_1(1), s_2(2), \ldots, s_{N-1}(N-1))^T$, we have $v(i) \leq \sum_j (I_{(N-1)\times(N-1)} - \lambda P_s)^{-1}(i,j) r_s(j) = v^s(i)$ for all $i \in \{1, 2, \ldots, N-1\}$. In vector notation, this is

$$v \leq r_s + \lambda P_s v$$

By claim 2, we have $v \leq v^s_\lambda \leq v^*_\lambda$.

If $v \geq \mathcal{L}_\lambda v$, then $v \geq r_s + \lambda P_s v$ for any $s \in \Sigma$. By claim 2, this means $v \geq v^s_\lambda$ for $s \in \Sigma$. Take supremum over all s we have $v \geq v^*_\lambda$.

Now by claim 3, we have $\mathcal{L}_\lambda u = u$ has a unique solution, say v. By above arguement, we $v \geq v^*_\lambda$ and $v \leq v^*_\lambda$; hence the solution to $\mathcal{L}_\lambda u = u$ must be v^*_λ. □

The next step is to establish the existence of an *optimal* strategy, i.e. $v^s_\lambda = v^*_\lambda$ for some $s \in \Sigma$. We call a strategy s *conserving* iff $r_s + P_s v^* = v^*$ as it conserves the optimal winning probability if we follow it for an additional period. It may be possible that an optimal strategy does not exist; but if it does, it is conserving:

Claim 5. *s is an optimal strategy, if and only if s is conserving.*

Proof. ($\Rightarrow$) By optimality of s, we have

$$r_s + \lambda P_s v^* = r_s + \lambda P_s v^s = v^s = v^*$$

Thus s is conserving as well.

($\Leftarrow$) Since s is conserving, we have

$$v^* = r_s + \lambda P_s v^*$$

Repeatedly apply it, for any $n \in \mathbb{N}$,

$$v^* = \sum_{i=0}^{n} (\lambda P_s)^i r_s + (\lambda P_s)^{n+1} v^*$$

Thus by argument in claim 2.

$$v^* = (I - \lambda P_s)^{-1} r_s = v_s$$

Note that the argument holds for $\lambda = 1$ as well. $\square$

Putting all these together, we have:

Claim 6. *There exists an optimal strategy $s \in \Sigma$ for the discounted model.*

Proof. Let v be the solution to $\mathcal{L}_\lambda u = u$. We simply choose

$$s(i) \in \underset{x \in \{1,2,\ldots,\min(i,N-i)\}}{\arg\max} \{\lambda p v(i+x) + \lambda(1-p)v(i-x)\}$$

As the number of choice is finite, this is well defined for all $i \in \{1, 2, \ldots, N-1\}$. Hence, $v = r_s + P_s v$. By claim 5, s is conserving and we are done. $\square$

Now define v_λ^s as winning probability for the discounted model by following optimal strategy s. We can construct an optimal strategy for original model, $\lambda = 1$ with the following lemma:

Lemma 1. *Let $\{\lambda_n\}$ be a non-decreasing sequence of discounting factor converging to 1. Then for each $x \in \{1, 2, \ldots, N-1\}$ and $s \in \Sigma$,*

$$v^s(x) = \lim_{n\to\infty} v^s_{\lambda_n}(x)$$

Proof. See Lemma 7.1.8 from Reference 1. $\square$

Claim 7. *There exists an optimal strategy $s \in \Sigma$ for the original model.*

Proof. By previous claim, for every $\lambda, 0 \le \lambda < 1$, there exista an optimal strategy. Since Σ is finite, for any sequence $\{\lambda_n\}$ converging to 1, one of the strategies, say s, will appear infinitely many times as optimal strategy; denote this subsequence as $\{\lambda_{n_k}\}_k$. For any other strategy s', we have

$$v^{s'}(x) = \lim_{k\to\infty} v^{s'}_{\lambda_{n_k}}(x) \le \lim_{k\to\infty} v^s_{\lambda_{n_k}}(x) = v^s(x)$$

Therefore s is the optimal strategy. $\square$

1.3. **Policy Iteration.** The algorithm (See Reference 1) below finds a solution to $v = \mathcal{L}v$ iteratively:

(1) Set $n = 0$ and select $s_0 = (1, 1, \ldots, 1)^T$.

(2) Compute $v^{s_n} = r_{s_n}(I - P_{s_n})^{-1}$.

(3) Choose

$$s_{n+1}(i) \in \underset{0<x<\min(i,N-i)}{\arg\max} \{pv^{s_n}(i+x) + (1-p)v^{s_n}(i-x)\}$$

whenever multiple choices are possible, setting $s_{n+1}(i)$ as small as possible.

(4) If $s_{n+1} = s_n$, stop and output v^{s_n} and s_n. Otherwise increment by 1 and return to step 2.

We will first show that if the algorithm terminates, it gives optimal strategy:

Claim 8. *The policy iteration algorithm terminates with optimal strategy.*

Proof. Upon termination, we have $s' \in \arg\max_{s\in\Sigma}\{r_s + P_s v^{s'}\}$. Since the choise of $s'(i)$ maximaze $\mathcal{L}v^{s'}$ for each component i, we have $\mathcal{L}v^{s'} = r_{s'} + P_{s'}v^{s'} = v^{s'}$. Thus $v^{s'}$ is a solution to $\mathcal{L}v = v$ and s' is optimal by previous claim. $\square$

We now show that if improvement in step 3 is *strict*, then the new strategy is strictly better than the previous one.

Claim 9. *Let s and t be two viable betting strategies, and*

$$r_t(x) + P_t v^s(x) \geq v^s(x)$$

with $s(x) = t(x)$ for $x \in \{1, 2, \ldots, N-1 | r_t(x) + P_t v^s(x) = v^s(x)\}$. Then $v^t \geq v^s$. Furthermore, if the inequality above is strict at x, then $v^t(x) > v^s(x)$ as well.

Proof. The condition is equivalent to

$$r_t \geq (I - P_t)v^s$$

By claim 2, we have

$$v^t = (I - P_t)^{-1} r_t \geq (I - P_t)^{-1}(I - P_t)v^s = v^s$$

If the first inequality is strict for some component x, so is the second one. $\square$

Claim 10. *The policy iteration algorithm terminates within finite number of steps.*

Proof. Consider the sequence of betting strategies constructed from algorithm $\{s_n\}_n$ and associated winning probabilities $\{v^{s_n}\}_n$. If the algorithm does not terminate at n, we have $v^{s_{n+1}} \geq v^{s_n}$ with strict inequality in at least one component by previous claim. This shows that s_{n+1} is a distinct strategy from $\{s_1, s_2, \ldots, s_n\}$. As Σ is finite, for some finite $M \in \mathbb{N}$ we must have $s_{M+1} = s_M$ and algorithm will terminate. $\square$

2. Arbitrary Real-Valued Input

In first section we made a few conjecture and verify them experimentally through policy iteration algorithm. In the next section we implement the algorithm in C.

2.1. **Observation and Conjectures.** When the game is to the casino's advantage ($0 < p < 0.5$), we conjecture that the player can achieve optimum winning probability by employing bold play: betting what is allows maximum each round, $s(x) = \min(x, 1 - x)$.

Let the winning probablity under bold play be $P(x)$. It satisfies the following relationship:

$$
\begin{aligned}
&P(x) = p \cdot P(2x) && \text{if } x < \frac{1}{2} \\
&P(x) = p + (1-p) \cdot P(2x-1) && \text{if } x \geq \frac{1}{2} \\
&P(0) = 0, \quad P(1) = 1
\end{aligned}
$$

Consider the game in part I with fnite states N. Relabel the starting balance $\{1, 2, \ldots, N-1\}$ as $\{1/N, 2/N, \ldots, (N-1)/N\}$ and limit bet sizes to multiples of $1/N$. As $N \to \infty$, this game converges to arbitrary real input. By running policy iteration for different values of N and comparing it with bold play, we conjecture that

Claim 11. *For any $p \in (0, 0.5)$, bold play gives optimal winning probability.*

From running policy iteration algorithm on $N = 2^n$, we observe that for the best strategy s, $s(x)$ is always highest power of 2 that divides x. That means if rescale x to $\frac{x}{N} = \frac{j}{2^m}$ for some odd $j \in \mathbb{N}$, then $s(x) = \frac{1}{2^m}$. In general this strategy is as good as bold play:

Claim 12. *Let the winning probablity under this strategy s be $Q(x)$. It satisfies the following relationship:*

$$Q(\frac{j}{2^n}) = Q(\frac{j/2}{2^{n-1}}) \qquad \text{if } 2 \mid j \tag{1}$$

$$Q(\frac{j}{2^n}) = p \cdot Q(\frac{j+1}{2^n}) + (1-p) Q(\frac{j-1}{2^n}) \qquad \text{if } 2 \nmid j \tag{2}$$

$$Q(0) = 0, \quad Q(1) = 1 \tag{3}$$

Furthermore, we have

$$Q(x) = P(x) \quad \forall x \text{ of the form } \frac{j}{2^n} \tag{4}$$

Proof. We can verify the recurrence by Markov property. For the second part, suppose the statement holds for all x of the form $\frac{j}{2^n}$ wher $n = m$ and odd $j < 2^m$. Now consider an odd $j < 2^m$ and $n = m + 1$, we have

$$\begin{aligned}
P(\frac{j}{2^{m+1}}) &= p \cdot P(\frac{j}{2^m}) \\
&= p \cdot Q(\frac{j}{2^m}) \\
&= p \cdot (pQ(\frac{j+1}{2^m}) + (1-p)Q(\frac{j-1}{2^m})) \\
&= p^2 P(\frac{j+1}{2^m}) + (1-p)pP(\frac{j-1}{2^m}) \\
&= p \cdot P(\frac{(j+1)/2}{2^m}) + (1-p)P(\frac{(j-1)/2}{2^m}) \\
&= p \cdot Q(\frac{j+1}{2^{m+1}}) + (1-p)Q(\frac{j-1}{2^{m+1}}) \\
&= Q(\frac{j}{2^{m+1}})
\end{aligned}$$

Similiarly for an odd $j > 2^m$ and $n = m + 1$, we have

$$\begin{aligned}
P(\frac{j}{2^{m+1}}) &= p + (1-p) \cdot P(\frac{j}{2^m} - 1) \\
&= p + (1-p) \cdot Q(\frac{j}{2^m} - 1) \\
&= p + (1-p) \left[p \cdot Q(\frac{j+1}{2^m} - 1) + (1-p) \cdot Q(\frac{j-1}{2^m} - 1) \right] \\
&= p \cdot \left[p + (1-p) \cdot Q(\frac{j+1}{2^m} - 1) \right] + (1-p) \left[p + (1-p) \cdot Q(\frac{j-1}{2^m} - 1) \right] \\
&= p \cdot \left[p + (1-p) \cdot P(\frac{j+1}{2^m} - 1) \right] + (1-p) \left[p + (1-p) \cdot P(\frac{j-1}{2^m} - 1) \right] \\
&= p \cdot P(\frac{(j+1)/2}{2^m}) + (1-p) \cdot P(\frac{(j-1)/2}{2^m}) \\
&= p \cdot Q(\frac{j+1}{2^{m+1}}) + (1-p) \cdot Q(\frac{j-1}{2^{m+1}}) \\
&= Q(\frac{j}{2^{m+1}})
\end{aligned}$$

$\square$

2.2. **Implementation.** All real numbers expressable in IEEE 754 double-precision are stored as dydic rational (see Reference 5), and the minimal positive dydic rational representable is $\frac{1}{2^{1277}}$; therefore all bet sizes are multiples of $\frac{1}{2^{1277}}$. Hence the problem is equivalent to arbitrary integer input $N = 2^{1277}$.

An IEEE754 double precision float is expressed as

$$x = \begin{cases} (-1)^{\text{sgn}}(1.b_{51}b_{50}\dots b_0)_2 \times 2^{e-1023} & \text{if } e > 0 \\ (-1)^{\text{sgn}}(0.b_{51}b_{50}\dots b_0)_2 \times 2^{1-1023} & \text{if } e = 0 \end{cases}$$

First we note that for any values of $p \in (0, 0.5)$, the betting strategy is the same. From conjecture and observation in part 1, We are interested to find the highest power of 2 in the denominator of x in reduced fraction. This is equivalent to 1. find the lowest non-zero bit in the fractional part, 2. shift the decimal point to the right of it, 3. adjust exponent accordingly. The function `bet_size()` implements it. We assume the user inputs are valid parameters.

```
#include <ieee754.h>

double bet_size(double P, double x)
{
  double a = x;
  union ieee754_double *pa;
  unsigned int aexp, asgn, asf0, asf1, y;
  unsigned int asf0_mask = 0x000FFFFF;

  pa = (union ieee754_double*) &a;
  aexp = pa->ieee.exponent;
  asgn = pa->ieee.negative;
  asf0 = pa->ieee.mantissa0;
  asf1 = pa->ieee.mantissa1;

  //case 1: x is normal double
  while(aexp>0)
  {
    if(asf0 == 0 && asf1 == 0) break;
    //left shift mantissa til mantissa == 0
    y = asf1 & (1<<31);
    y = y >> 31;
    asf0 = asf0 << 1;
    asf0 = asf0 | y;
    asf0 = asf0 & asf0_mask;
    asf1 = asf1 << 1;
    aexp--;
  }
  //case 2: x is subnormal double
  if(aexp == 0)
  {
    //mantissa = lowest nonzero bit
    if(asf1 == 0) asf0 = asf0 & ~(asf0 - 1);
    else
```

```
    {
      asf0 = 0;
      asf1 = asf1 & ~(asf1 - 1);
    }
  }

  a.ieee.negative = asgn;
  a.ieee.exponent = aexp;
  a.ieee.mantissa0 = asf0;
  a.ieee.mantissa1 = asf1;

  return a.d;
}
```

3. References

(1) Markov Decision Processes, Martin L. Puterman

(2) Mathoverflow Post on norm of substochastics matrix

(3) Essentials of Stochastics Processes, Richard Durrett

(4) Wikipedia Entry on Banach fixed point theorem

(5) Wikipedia Entry on Double-precision floating-point format