Mathematical Questions

1. Condition probability and Bernoulli trials[343]	3	*graph+ open tree + closed tree*
	4	*transtivity + recursion of P + recursion of E + Bayesian*
	3	*geometric pdf + memoryless def × 3 + proof*
2. Boy or girl paradox	3	*conditional probability*
3. Monty Hall problem	2	*conditional probability*
4. Faulty coin problem	2	*conditional probability*
5. Collision	5	*Bernoulli trials*
6. Gambler ruin	4	
7. Kelly criterion	3	
8. Tossing a biased coin	3	

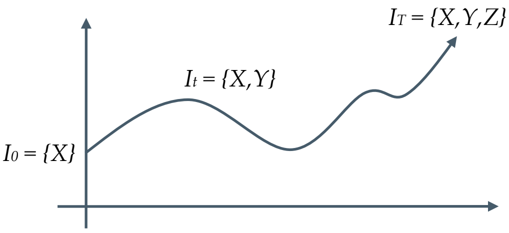
Revision

Difference between random variable and event :

- X is a random variable, x is a realized value
- $X = x_0$ is an event, called event A
- $X = \{ x_0, x_1, x_2 \ldots \}$ is another event, called event B

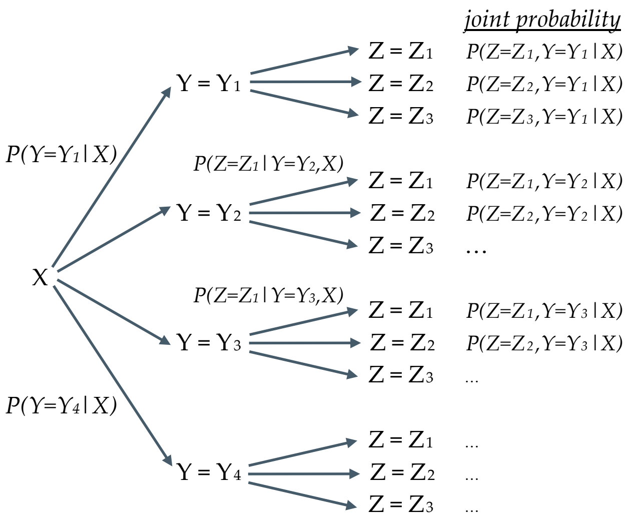
1A. Conditional probability

Given current time 0, intermediate time t and maturity time T, at which random variables $X/Y/Z$ are realized respectively. We have $I_0 = \{X\}$, $I_t = \{X,Y\}$ and $I_T = \{X,Y,Z\}$, such that $I_0 \subset I_t \subset I_T$.

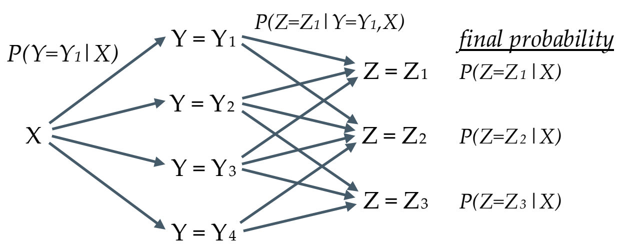


Conditional probabilities can be represented as open tree or closed tree (my terminology) :

- leaves of open tree denote joint probability $\Pr(\overbrace{Z = z, Y = y}^{path} \mid \overbrace{X = z}^{initial})$
- leaves of closed tree denote final probability $\Pr(\overbrace{Z = z}^{final} \mid \overbrace{X = z}^{initial})$



This closed tree offers visualization of recursion (2a).



1.	*Transitivity*	$\Pr(\overbrace{Z=z,Y=y}^{path} \mid \overbrace{X=z}^{initial})$	$= \Pr(Z=z \mid Y=y, X=x)\Pr(Y=y \mid X=x)$	*consider a path in open tree*
2a.	*Recursion of probability*	$\Pr(\overbrace{Z=z}^{final} \mid \overbrace{X=x}^{initial})$	$= \sum_y \Pr(Z=z \mid Y=y, X=x)\Pr(Y=y \mid X=x)$	*consider a leaf in closed tree*
2b.		$\Pr(Z=z)$	$= \sum_y \Pr(Z=z \mid Y=y)\Pr(Y=y)$	
3a.	*Recursion of expectation*	$E[Z \mid X=x]$	$= E[E[Z \mid Y=y, X=x] \mid X=x]$	*known as Tower property*
3b.		$E[Z]$	$= E[E[Z \mid Y=y]]$	
4.	*Bayesian*	$\Pr(\theta \mid X=x)\Pr(X=x)$	$= \Pr(X=x \mid \theta)\Pr(\theta)$	

Proof

1. *Transitivity*

$$\begin{aligned}
\Pr(Z=z,Y=y \mid X=x) &= \frac{\Pr(Z=z,Y=y,X=x)}{\Pr(X=x)} \\
&= \frac{\Pr(Z=z,Y=y,X=x)}{\Pr(Y=y,X=x)}\frac{\Pr(Y=y,X=x)}{\Pr(X=x)} \\
&= \Pr(Z=z \mid Y=y, X=x)\Pr(Y=y \mid X=x)
\end{aligned}$$

2a. *Recursion of probability*

$$\begin{aligned}
\Pr(Z=z \mid X=x) &= \sum_y \Pr(Z=z,Y=y \mid X=x) \\
&= \sum_y \Pr(Z=z \mid Y=y, X=x)\Pr(Y=y \mid X=x) && \textit{using transitivity}
\end{aligned}$$

2b.

$$\begin{aligned}
\Pr(Z=z) &= \Pr(Z=z \mid X=x)\,|_{x=universal} \\
&= \sum_y \Pr(Z=z \mid Y=y, X=x)\Pr(Y=y \mid X=x)\,|_{x=universal} \\
&= \sum_y \Pr(Z=z \mid Y=y)\Pr(Y=y)
\end{aligned}$$

3a. *Recursion of expectation*

$$\begin{aligned}
E[Z \mid X=x] &= \sum_z z\Pr(Z=z \mid X=x) \\
&= \sum_z [z\sum_y \Pr(Z=z \mid Y=y, X=x)\Pr(Y=y \mid X=x)] && \textit{using 2a} \\
&= \sum_y [\sum_z z\Pr(Z=z \mid Y=y, X=x)]\Pr(Y=y \mid X=x) && \textit{reverse two summations} \\
&= \sum_y E[Z \mid Y=y, X=x]\Pr(Y=y \mid X=x) \\
&= E[E[Z \mid Y=y, X=x] \mid X=x]
\end{aligned}$$

3b.

$$\begin{aligned}
E[Z] &= \sum_z z\Pr(Z=z) \\
&= \sum_z [z\sum_y \Pr(Z=z \mid Y=y)\Pr(Y=y] && \textit{using 2b} \\
&= \sum_y [\sum_z z\Pr(Z=z \mid Y=y)]\Pr(Y=y) && \textit{reverse two summations} \\
&= \sum_y E[Z \mid Y=y]\Pr(Y=y) \\
&= E[E[Z \mid Y=y]]
\end{aligned}$$

Probability is a function of realised value z, while expectation is not.

$\Pr(Z=z)$	*contains random variable Z and realized value z*	*(it is a function of z)*
$E[Z]$	*contains random variable Z*	*(it is a function of distribution parameters, like Gaussian mean and stdd)*

1B. Bernoulli trials

Collision is considered as a successful event in a sequence of Bernoulli trial with probability p.

- The number of collision n within N trials is a binomial distribution : $\Pr(\#collision = n) = C_n^N p^n q^{N-n}$
- The number of trials N until collision happens is a geometric distribution : $\Pr(\#trial = N) = pq^{N-1}$
  $\Pr(\#trial > N) = q^N$

Three definitions of memoryless :

(1)	$\Pr(\#trial = N \mid \#trial > N_0)$	$=$	$\Pr(\#trial = N - N_0)$	
(2)	$\Pr(\#trial = N)$	$=$	$\Pr(\#trial = N - N_0) \times \Pr(\#trial > N_0)$	*deduction of N_0 for probability*
(3)	$E[\#trial \mid \#trial > N_0]$	$=$	$E[\#trial] + N_0$	*addition of N_0 for expectation*

Prove 1&2 are consistent :

$$\Pr(\#trial = N \mid \#trial > N_0) = \Pr(\#trial = N - N_0)$$

$$\rightarrow \quad \frac{\Pr(\#trial = N \cap \#trial > N_0)}{\Pr(\#trial > N_0)} = \Pr(\#trial = N - N_0)$$

$$\rightarrow \quad \frac{\Pr(\#trial = N)}{\Pr(\#trial > N_0)} = \Pr(\#trial = N - N_0)$$

$$\rightarrow \quad \Pr(\#trial = N) = \Pr(\#trial = N - N_0) \times \Pr(\#trial > N_0) \qquad \textit{QED}$$

Prove 1&3 are consistent :

$$
\begin{aligned}
E[\#trial \mid \#trial > N_0] &= \sum_{N=1}^{\infty} N \Pr(\#trial = N \mid \#trial > N_0) \\
&= \sum_{N=N_0+1}^{\infty} N \Pr(\#trial = N \mid \#trial > N_0) && \Pr(\#trial = N \mid \#trial > N_0) = 0, \forall N \le N_0 \\
&= \sum_{N=N_0+1}^{\infty} N \Pr(\#trial = N - N_0) && \textit{memoryless property (1)} \\
&= \sum_{M=1}^{\infty} (M + N_0) \Pr(\#trial = M) && \textit{put } M = N - N_0 \\
&= \sum_{M=1}^{\infty} M \Pr(\#trial = M) + N_0 \\
&= E[\#trial] + N_0
\end{aligned}
$$

Now, let's show that the geometric distribution is memoryless.

$$
\begin{aligned}
\Pr(\#trial = N \mid \#trial > N_0) &= \frac{\Pr(\#trial = N \cap \#trial > N_0)}{\Pr(\#trial > N_0)} \\
&= \frac{\Pr(\#trial = N)}{\Pr(\#trial > N_0)} \\
&= \frac{pq^{N-1}}{q^{N_0}} \\
&= pq^{(N-N_0)-1} \\
&= \Pr(\#trial = N - N_0) && \textit{satisfying memoryless definition}
\end{aligned}
$$

2. Boy or Girl Paradox

Look at the following, they seem to be counter-intuitive at the first glance, called boy or girl paradox :

- Mr Chan has two kids, what is the probabiliy that both are boys? answer = 1/4
- Mr Chan has two kids, the older one is a boy, what is the probabiliy that both are boys? answer = 1/2
- Mr Chan has two kids, at least one is a boy, what is the probabiliy that both are boys? answer = 1/3

Suppose p is the probability of having a boy, then :

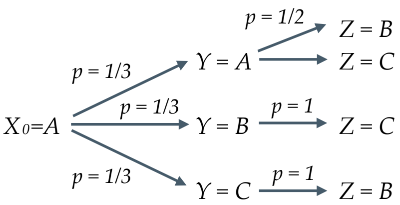
$$
\begin{aligned}
\Pr(X_0 = boy, X_1 = boy) &= p^2 && \textit{joint probability is probability of union of events} \\
\Pr(X_0 = boy, X_1 = boy \mid X_0 = boy) &= \frac{\Pr(X_0 = boy)P(X_1 = boy)}{\Pr(X_0 = boy)} && \textit{since two events are independent} \\
&= P(X_1 = boy) \\
&= p \\
\Pr(X_0 = boy, X_1 = boy \mid X_0 = boy \cup X_1 = boy) &= \frac{\Pr((X_0 = boy \cap X_1 = boy) \cap (X_0 = boy \cup X_1 = boy))}{\Pr(X_0 = boy \cup X_1 = boy)} \\
&= \frac{\Pr(X_0 = boy \cap X_1 = boy)}{\Pr(X_0 = boy \cup X_1 = boy)} \quad = \quad \frac{p^2}{p^2 + 2pq}
\end{aligned}
$$

3. Monty Hall Problem

There are three doors, behind one of which there is a gift. You initially pick one door, host of the game opens one of the other doors, revealing an empty space. What is the chance of winning of sticking to initial choice versus switching your choice? The fact is quite counter intuitive, as you can double the winning chance making a switch. Suppose the doors are named A,B,C, the one we initially pick is door A. Define action X, define random variables Y be answer, Z be opened door.

<u>Solution</u>

It is a 2 stages Markov chain, only branches with non zero probability are shown.



stick-strategy	***switch-strategy***
$X_T = A$ *good*	$X_T = C$ *fail*
$X_T = A$ *good*	$X_T = B$ *fail*
$X_T = A$ *fail*	$X_T = B$ *good*
$X_T = A$ *fail*	$X_T = C$ *good*
$P_{stick}(X_T = Y) = 1/3$	$P_{switch}(X_T = Y) = 2/3$

<u>Remark</u>

By extending to N door, it becomes more reasonable :

- given N doors and there exists a gift behind one of which
- we initially pick door A, there is $1/N$ probbility that the gift is behind door A
- if the gift is behind door A, the host will open N-2 other doors, then a switch will lead to a loss
- if the gift is not behind door A, the host will open N-2 other doors leaving the gifted door closed, then a switch will win
- those winning probability of switching is $(N-1)/N$, as opposed to $1/N$ if we stick to initial guess

4. Faulty coin problem

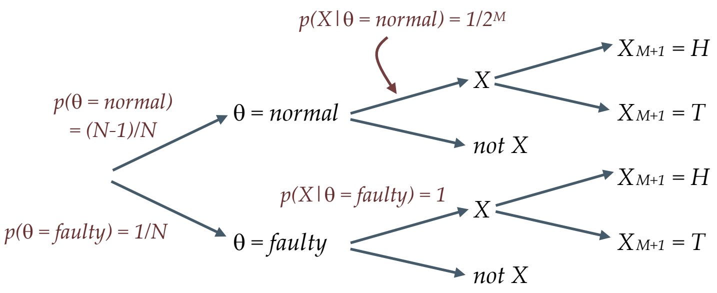
Given N fair coins, one of them has both sides head. A coin is randomly picked which is tossed for M times, outcomes are all head.

(a) What is the probability of the coin being faulty?

(b) What is the probability of one more head if the same coin is tossed?

<u>Solution</u>

Let X be the event of M consecutive head $\{X_1 = H, X_2 = H, ..., X_M = H\}$ while random boolean θ = normal or faulty.



The direction of decision tree can be reversed using Bayesian rule …

(a)

$$
\begin{aligned}
P(\theta = normal \mid X) &= \frac{P(X, \theta = normal)}{P(X)} \\
&= \frac{P(X, \theta = normal)}{P(X, \theta = normal) + P(X, \theta = faulty)} \\
&= \frac{P(X \mid \theta = normal)P(\theta = normal)}{P(X \mid \theta = normal)P(\theta = normal) + P(X \mid \theta = faulty)P(\theta = faulty)} \\
&= \frac{\frac{1}{2^M} \times \frac{N-1}{N}}{(\frac{1}{2^M} \times \frac{N-1}{N}) + (1 \times \frac{1}{N})}
\end{aligned}
$$

or apply recursion 2b directly

$$
P(\theta = faulty \mid X) = \frac{1 \times \frac{1}{N}}{(\frac{1}{2^M} \times \frac{N-1}{N}) + (1 \times \frac{1}{N})}
$$

following same logic

(b)

$$
\begin{aligned}
P(X_{M+1} = H \mid X) &= P(X_{M+1} = H, \theta = normal \mid X) + P(X_{M+1} = H, \theta = faulty \mid X) \\
&= P(X_{M+1} = H \mid \theta = normal, X)P(\theta = normal \mid X) + P(X_{M+1} = H \mid \theta = faulty, X)P(\theta = faulty \mid X) \\
&= \frac{1}{2} \times P(\theta = normal \mid X) + 1 \times P(\theta = faulty \mid X) \\
&= \frac{1}{2} \times \frac{\frac{1}{2^M} \times \frac{N-1}{N}}{(\frac{1}{2^M} \times \frac{N-1}{N}) + \frac{1}{N}} + 1 \times \frac{1 \times \frac{1}{N}}{(\frac{1}{2^M} \times \frac{N-1}{N}) + (1 \times \frac{1}{N})}
\end{aligned}
$$

or apply recursion 2a directly

5. Collision in birthday attack

Given a sequence of Bernoulli trials $x_1, x_2, \ldots$ with successful event A *(prob p)* and otherwise failed event B *(prob q)*, find :

(a) $E[N]$ *such that* $x_N = A$ *and* $x_n = B \ \forall n < N$

(b) $E[N]$ *such that* $x_N = x_{N-1} = A$

(c) $E[N]$ *such that* $x_N = x_{N-1}$

(d) $E[N]$ *such that* $\exists N_0 \in [1, N-1] \ x_N = x_{N_0}$

(e) *by extending to M events with equal chance, then (d) becomes birthday attack*

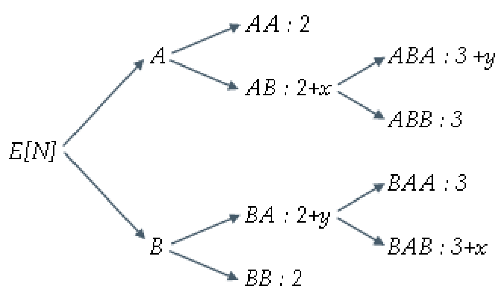
The following calculations mainly make use of ***recursion equation 2b*** and ***memoryless definition 3*** :

$$
\begin{aligned}
(a)\quad E[N] &= p \times E[N \mid x_1 = A] + q \times E[N \mid x_1 = B] \\
&= p + q(E[N]+1) \\
&= 1/p
\end{aligned}
$$

$$\text{start } E[N] \begin{cases} \to A & E[N \mid x_1 = A] = E[N \mid N = 1] = 1 \\ \to B & E[N \mid x_1 = B] = E[N \mid N > 1] = E[N]+1 \end{cases}$$

$$
\begin{aligned}
(b)\quad E[N] &= p^2 \times E[N \mid x_1 = A, x_2 = A] + pq \times E[N \mid x_1 = A, x_2 = B] + q \times E[N \mid x_1 = B] \\
&= p^2 \times 2 + pq \times (E[N]+2) + q \times (E[N]+1) \\
&= \frac{p^2 \times 2 + pq \times 2 + q \times 1}{1 - pq - q} \\
&= \frac{1}{p}\frac{1+p}{p}
\end{aligned}
$$

$$
\begin{aligned}
(c)\quad \begin{bmatrix} 2+x \\ 2+y \end{bmatrix} &= \begin{bmatrix} p(3+y)+q(3) \\ p(3)+q(3+x) \end{bmatrix} \\
\Rightarrow \quad \begin{bmatrix} x \\ y \end{bmatrix} &= \begin{bmatrix} \dfrac{1+p}{1-pq} \\ \dfrac{1+q}{1-pq} \end{bmatrix} \\
E[N] &= p^2 \times 2 + pq(2+x) + pq(2+y) + q^2 \times 2 \\
&= 2\underbrace{(p+q)}_{1}{}^2 + pq(x+y) \\
&= 2 + pq(\frac{1+p+1+q}{1-pq}) \\
&= 2 + \frac{3pq}{1-pq} \\
E[N] &= 2 \qquad \textit{when } p = 1
\end{aligned}
$$



*where x is expected **extra** trials when last realization is A*
*where y is expected **extra** trials when last realization is B*

(d) *We always have* $x_N = x_{N_0}$ *for* $N = 2$ *or* $N = 3$*. Find Pr(N=2) and Pr(N=3), we can then get the answer.*

$$
\begin{aligned}
E[N] &= \Pr(N=2) \times 2 + \Pr(N=3) \times 3 \\
&= (p^2 + q^2) \times 2 + (pqq + pqp + qpq + qpp) \times 3
\end{aligned}
$$

(e) Birthday attack

Given an infinity large population, we keep drawing randomly from the population until the first birthday-collision occurs, assume that *N* people are drawn in total, what is the expected value of *N*? As more people are drawn probability of collision increases, thus memoryless (as part *a-c* in previous part) does not apply, we need to start from basic, i.e. finding *Pr(N= n)* like part *d*.

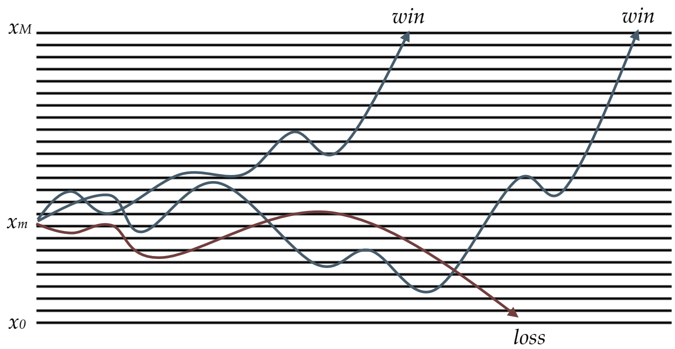
$$
\begin{aligned}
\Pr(\#trial = N) &= \underbrace{\frac{365}{365} \times \frac{364}{365} \times \frac{363}{365} \times \ldots \times \frac{365-(N-2)}{365}}_{N-1} \times \frac{N-1}{365} \\
E[N] &= \sum_{N=2}^{366} \Pr(\#trial = N) N \\
&= \sum_{N=2}^{366} \frac{365}{365} \times \frac{364}{365} \times \frac{363}{365} \times \ldots \times \frac{365-(N-2)}{365} \times \frac{N-1}{365} N
\end{aligned}
$$

6. Gambler's Ruin

Consider a gambler with an initial amount of money x_m participates in a sequence of independent bets, in each bet he either :

- having his wealth incremented to x_{m+1} with a probability of p or
- having his wealth decremented to x_{m-1} with a probability of q
- we don't need an accurate model for x_m as long as $x_{m-1} < x_m < x_{m+1}$ (perhaps $x_{m+1} = s \times x_m$ or $x_{m+1} = x_m + \Delta x$)
- gambler keeps on gambling until his wealth either reaches x_M *(win and exit)* or reaches 0 *(complete ruin)*

\- Find the probability of winning the game. ***This is equivalent to pricing a double barrier option with infinity maturity***.



<u>*Solution*</u>

Let *f(x)* be a contract on x that pays $1 with gambler wins or $0 on a complete ruin, then its PV is the probability of winning.

$$
\begin{aligned}
f_m &= f(x = x_m) \\
f_0 &= 0 \\
f_M &= 1
\end{aligned}
$$

Please note ***x is underlying*** *while* ***f is derivatives***

Step 1 Derive diff eq Using the memoryless nature, we yield the recursion :

$$
\begin{aligned}
f_m &= p \times f_{m+1} + q \times f_{m-1} \\
p \times f_m + q \times f_m &= p \times f_{m+1} + q \times f_{m-1} \\
f_{m+1} - f_m &= (q/p) \times (f_m - f_{m-1})
\end{aligned}
$$

NOT $x_m \neq p \times x_{m+1} + q \times x_{m-1}$

Step 2 Plug in lower boundary case

$$
\begin{aligned}
f_2 - f_1 &= (q/p) \times (f_1 - f_0) &&= (q/p) \times f_1 \\
f_3 - f_2 &= (q/p) \times (f_2 - f_1) &&= (q/p)^2 \times f_1 \\
f_{m+1} - f_m &= (q/p)^m \times f_1 \\
f_{m+1} - f_1 &= \textstyle\sum_{k=1}^{m} (f_{k+1} - f_k) &&= \textstyle\sum_{k=1}^{m} (q/p)^k \times f_1 \quad \textit{with telescoping sum} \\
f_{m+1} &= f_1 \times \textstyle\sum_{k=0}^{m} (q/p)^k \\
&= f_1 \times \frac{1-(q/p)^{m+1}}{1-(q/p)}
\end{aligned}
$$

since $a + ar + ar^2 + ... + ar^n = a(1-r^{n+1})/(1-r)$ *if* $r \neq 1$

Step 3 Plug in upper boundary case

$$
\begin{aligned}
f_M &= f_1 \times \frac{1-(q/p)^M}{1-(q/p)} \\
f_1 &= \frac{1-(q/p)}{1-(q/p)^M} \qquad \textit{since } f_M = 1 \\
\Rightarrow \quad f_m &= \frac{1-(q/p)}{1-(q/p)^M} \times \frac{1-(q/p)^m}{1-(q/p)} = \frac{1-(q/p)^m}{1-(q/p)^M}
\end{aligned}
$$

Step 4 Consider extreme case Let's extend the game by taking limit M tends to infinty, we have :

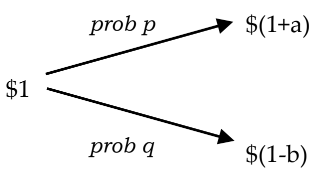
$$
\lim_{M \to \infty} f_m = \lim_{M \to \infty} \frac{1-(q/p)^m}{1-(q/p)^M} = \begin{cases} 1-(q/p)^m & \textit{if} \quad p > q \\ 0 & \textit{if} \quad p < q \end{cases}
$$

When *p>0.5*, there exists a positive probability that the gambler can continue forever, or gambler must ruin in the middle.

7. Kelly Criterion

Given a N independent-stage game, for each stage :

- gambler should decide the amount (which is called **wager** 賭注) to bet that event A will occur (***NO*** short sell is allowed)
- if A does occur, he will earn\$a **net odd** (派彩) per \$1 wager
- if not, he will lose \$b **net loss** per \$1 wager
- suppose p is the probability that event A will occur, **payoff structure** of the bet can be summarised as binomial tree :



Step 1 Derive fair price Assume that it is a fair game, net payoff should be :

$$
\begin{aligned}
& 1 &=&\; p(1+a)+q(1-b) \\
\Rightarrow\; & 0 &=&\; pa-qb \\
\Rightarrow\; & \frac{a}{b} &=&\; \frac{q}{p}
\end{aligned}
$$

like Gambler's ruin, the ratio q/p appears again

Assume that it is not a fair game, then Kelly criterion offers a strategy for making profit and maximising expected wealth.

Step 2 Optimize wrt f

Kelly criterion states that the gambler should not bet all or nothing, but bet a fraction of his wealth depending on the relative values of a, b and p. Assume the total number of games is N, among which, the gambler wins n of them, X_0 is the initial wealth, while X_n is the wealth after the nth game, f is the ***fraction of wealth*** he bets in each game, then the final wealth X_N is :

$$X_N = X_0(1+af)^n(1-bf)^{N-n}$$

where n is a ***binomial random variable***. We try to maximise the expected 'log wealth' :

$$
\begin{aligned}
f_{opt} &= \arg\max_f E[\ln(X_N/X_0)] \\
&= \arg\max_f E[n\ln(1+af)+(N-n)\ln(1-bf)] && \textit{expectation wrt n and maximization wrt f} \\
&= \arg\max_f E[n]\ln(1+af)+E[N-n]\ln(1-bf) \\
&= \arg\max_f Np\ln(1+af)+Nq\ln(1-bf) && \textit{since } E[n]=np \textit{ and } E[N-n]=nq \\
&= \arg\max_f g(f)
\end{aligned}
$$

By taking the first and second derivatives, we have :

$$
\begin{aligned}
g'(f) &= \frac{Npa}{1+af}-\frac{Nqb}{1-bf} \\
g''(f) &= -\frac{Npa^2}{(1+af)^2}-\frac{Nqb^2}{(1-bf)^2} && \textit{hence } g''(f)<0, \forall f\in[0,1], \textit{ it is convex and has a maximum}
\end{aligned}
$$

Setting the first derivative zero, we have the optimal fraction :

$$
\begin{aligned}
\frac{Npa}{1+af_{opt}} &= \frac{Nqb}{1-bf} \\
pa(1-bf_{opt}) &= qb(1+af_{opt}) \\
(p+q)abf_{opt} &= pa-qb \\
f_{opt} &= \frac{pa-qb}{ab} = \frac{p}{b}-\frac{q}{a}
\end{aligned}
$$

The optimal fraction is independent on *N*, it only depends on the relative values of odd and probability.

- when $\frac{p}{b} > \frac{q}{a}$, we bet $f = \frac{p}{b} - \frac{q}{a}$
- when $\frac{p}{b} < \frac{q}{a}$, we don't bet as no short sell is allowed

Step 3 Dr Wong argument as linear programming

However Dr Wong argued that we should invest all for unfair game to maximise wealth. Here are his arguments.

- Consider one stage game, we have simple payoff function, hence no log is needed :

$$E(X_1) = X_0(p(1+af)+q(1-bf))$$

- this is a linear programming, no need to take derivative, optimal point must lie on constrains (i.e. $0 \le f \le 1$), so compare :

$$\begin{array}{lllll} E(X_1) & = X_0(p(1+a0)+q(1-b0)) & = X_0 & & \text{when } f=0 \\ E(X_1) & = X_0(p(1+a1)+q(1-b1)) & = X_0 + X_0(pa-qb) & & \text{when } f=1 \end{array}$$

- thus the optimal strategy should be :

when $(pa-qb) > 0$, bet everything $f_{opt} = 1$
when $(pa-qb) \le 0$, bet nothing $f_{opt} = 0$

What makes the differences? The answer lies in the difference in the objective function.

$$\begin{array}{lll} f_{kelly} & = & \arg\max_f E(\ln(X_N / X_0)) \\ f_{Dr} & = & \arg\max_f E(X_N / X_0) \\ & = & \arg\max_f \ln E(X_N / X_0) \\ & \neq & \arg\max_f E(\ln(X_N / X_0)) \end{array}$$

Reference

[1] Betting with the Kelly Criterion. Jane Hung, Mathematics department, University of Washington.

8. Tossing a Biased Coin

Given a biased coin, how can we simulate an unbiased coin toss?

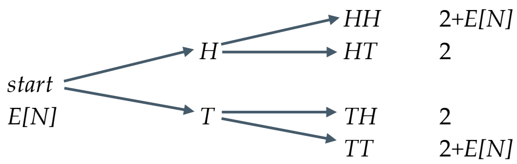
We can evaluate the information of each scheme by calculating :

- number of flips needed to generate 1 bit output *(similar to those E[N] calculations in previous sections)* or
- number of bits generated per each coin flip *(this is called entropy)*
- we will measure *E[N]* for method 1&2 and *entropy* for method 3
- where *N* is the random variable denoting number of coin flips needed to generate the first output bit

<u>*Method 1 : Von Neumann's method*</u>

H/T is called a flip, while 0/1 is called a bit. Given input flip sequence, we will generate output bit sequence.

- biased coin is flipped twice for each round
- for each HT flip, a bit-0 is generated as output
- for each TH flip, a bit-1 is generated as output
- for each HH or TT flip, ignored, hence there is waste of information …



$$
\begin{aligned}
E[N] &= E[N \mid HH]\Pr(HH) + E[N \mid HT]\Pr(HT) + E[N \mid TH]\Pr(TH) + E[N \mid TT]\Pr(TT) \\
&= (2+E[N])p^2 + 2pq + 2qp + (2+E[N])q^2 \\
&= \frac{2(p^2+q^2)+4pq}{1-(p^2+q^2)} \\
&= \frac{2(p+q)^2}{(p+q)^2-(p^2+q^2)} \\
&= \frac{2}{2pq} \\
&= 1/pq
\end{aligned}
$$

Method 2 : Multi level strategy – list of flip sequences

Von Neumann's method is inefficient, as it wastes outcome HH and outcome TT *(as highlighted in red)*. For example :

```
level 0   HHHTTHTTTHTTHTTTHHHHTHHTTTTTHTTTTHTTHTTTTHHHHHHTHHHHTHTTHHHH
output       0 1   1   0       1 0     0   1   0   1     0     1
```

In order to make use of wasted flips, we generate child ***flip-sequence*** (*level* 1) using the original ***flip-sequence*** (*level* 0) :

- for every double flip in level 0 ***flip-sequence***
- if it is HT   add bit 0 to output ***bit-sequence***
- if it is TH   add bit 1 to output ***bit-sequence***
- if it is HH   add flip H to level 1 ***flip-sequence***
- if it is TT   add flip T to level 1 ***flip-sequence***
- whenever the size of level 1 ***flip-sequence*** becomes even, apply Von Neumann on level 1
- do this recursively to create level 2, 3, … and so on, all ***flip-sequences*** share the same output ***bit-sequence***

```
level 0   HHHTTHTTTHTTHTTTHHHHTHHTTTTTHTTTTHTTHTTTTHHHHHHTHHHHTHTTHHHH
level 1    H     T   T   T H H     T T   T   T   T   H H   H H   T H H
level 2                  T   H       T       T             H         H
level 3                                      T                       H
output       0 1 0 1   0     1 1 0     0   1   0   1 1   0     1 0   1
```

(2a) What is the probability of generating zero bit after flipping $N=2^k$ flips?

Multi level strategy cannot generate one single bit in all 2^k flips only if :

- no bit is generated by level 0, implying that only HH or TT exist in level 0
- no bit is generated by level 1, implying that only HH or TT exist in level 1, i.e. only HHHH and TTTT exist in level 0
- no bit is generated by level 2, implying that only HH or TT exist in level 2, i.e. only HHHHHHHH and TTTTTTTT in level 0
- and so on …

$$\Pr(\#\,flip > 2^k) \quad = \quad p^{2^k} + q^{2^k}$$

(2b) What is the probability of generating no bit after flipping K times?

Firstly, we expand N as binary representation :

$$N \quad = \quad 2^{k_1} + 2^{k_2} + 2^{k_3} + \ldots 2^{k_M} \qquad \textit{where} \;\; k_1 > k_2 > k_3 > \ldots > k_M$$

$$\begin{aligned}
\Pr(\#\,flip > N) &= \Pr(\#\,flip > 2^{k_1} + 2^{k_2} + 2^{k_3} + \ldots 2^{k_M}) \\
&= \Pr(\#\,flip > 2^{k_2} + 2^{k_3} + \ldots 2^{k_M}) \Pr(\#\,flip > 2^{k_1}) \qquad \textit{memoryless} \;\; \Pr(\#\,flip = N) = \Pr(\#\,flip = N - N_0) \Pr(\#\,flip > N_0) \\
&= \Pr(\#\,flip > 2^{k_3} + \ldots 2^{k_l}) \Pr(\#\,flip > 2^{k_1}) \Pr(\#\,flip > 2^{k_2}) \\
&= \Pr(N > 2^{k_4} + \ldots 2^{k_l}) \Pr(N > 2^{k_1}) \Pr(N > 2^{k_2}) \Pr(N > 2^{k_3}) \\
&= \textstyle\prod_{m=1}^{M} \Pr(\#\,flip > 2^{k_m}) \\
&= \textstyle\prod_{m=1}^{M} (p^{2^{k_m}} + q^{2^{k_m}})
\end{aligned}$$

(2c) What is the expected number of flips needed to generate one bit?

Since N can only be even, we have :

$$\begin{aligned}
\Pr(\#\,flip = N) &= \Pr(\#\,flip > N-2) - \Pr(\#\,flip > N) \\
E[\#\,flip] &= \textstyle\sum_{N=2,4,6\ldots} \Pr(\#\,flip = N) N \\
&= \textstyle\sum_{N=2,4,6\ldots} (\Pr(\#\,flip > N-2) - \Pr(\#\,flip > N)) N \\
&= (\Pr(\#\,flip > 0) - \Pr(\#\,flip > 2))2 + (\Pr(\#\,flip > 2) - \Pr(\#\,flip > 4))4 + (\Pr(\#\,flip > 4) - \Pr(\#\,flip > 6))6 + \ldots \qquad \textit{telescoping sum} \\
&= (\Pr(\#\,flip > 0) + \Pr(\#\,flip > 2) + \Pr(\#\,flip > 4) + \Pr(\#\,flip > 6) + \ldots) \times 2 \\
&= 2(1 + \textstyle\sum_{N=2,4,6\ldots} \Pr(\#\,flip > N))
\end{aligned}$$

Method 3 : Multi level strategy – binary tree of flip sequences

How can we improve the strategy so as to further increase the information generated from each flip?

- method 2 works with a list of ***flip-sequence*** while
- method 3 works with a binary tree of ***flip-sequence*** in which
- each parent ***flip-sequence*** generates two child ***flip-sequences*** and one output ***bit-sequence***, for each double flip in level 0 :

-	if it is HT	add bit 0 to output ***bit-sequence***	and add flip H to level B ***flip-sequence***
-	if it is TH	add bit 1 to output ***bit-sequence***	and add flip H to level B ***flip-sequence***
-	if it is HH	add flip H to level A ***flip-sequence***	and add flip T to level B ***flip-sequence***
-	if it is TT	add flip T to level A ***flip-sequence***	and add flip T to level B ***flip-sequence***

- if a new flip is added to a ***flip-sequence*** and if its size becomes even, then apply Von Neumann on it recursively
- this recursion will effectively create a tree like the following, level naming is different from that in method 2
- each ***flip-sequence*** creates new bit to the same output ***bit-sequence***

```
level 0   ⇒  level A   ⇒  level AA  ⇒  level AAA  ← This row is equivalent to level 0,1,2,3 … in method 2.
                                    ⇒  level AAB  ┐
                       ⇒  level AB  ⇒  level ABA  │
                                    ⇒  level ABB  │
          ⇒  level B   ⇒  level BA  ⇒  level BAA  ├ This part is new to method 2.
                                    ⇒  level BAB  │
                       ⇒  level BB  ⇒  level BBA  │
                                    ⇒  level BBB  ┘
```

- unlike method 2, in which one flip in level 0 can generate at most one output bit, …
- yet in method 3, one flip level 0 may generate multiple output bits

Here is an example. I intentionally separate the output of different levels, in practice, all ***flip-sequences*** share the same ***bit-sequence***.

```
level 0           HHHTTHTTTHTTHTTTHHHHTHHTTTTTHTTTTHTTHTTTTHHHHHTHHHHTHTTHHHH
level A           H     T   T   T H H     T T   T   T   T   H H   H H   T H H
level B           H T T H T H T H H H T T H H T H T H T H T H H T H H T H H H
level AA                        T   H       T       T             H         H
level AB                T       H   H       H       H       T     H     T   H
level BA                            H   T   H                       H       H
level BB            T   T   T   T   H   H   H   T   T   T   T   T   H   T   H

level 0 output      0 1   1   0       1 0     0   1   0   1     0     1
level A output          0                                   1           0
level B output      0   1   1   1               1   1   1   1   0       1
level AA output                     1
level AB output                 1                           0           0
level BA output                         0
level BB output                               0                         0
```

Instead of finding the number of flips needed in level 0 to generate one bit in output we find the expected number of bits generated in output ***bit-sequence*** per flip in level 0, which depends on p. Assume function $f(p)$ to be the efficiency, lets consider level 0 :

flip pair	probability	generate bit	flip to level A	flip to level B
HT	pq	0	-	T
TH	qp	1	-	T
HH	p^2	-	H	H
TT	q^2	-	T	H

We try to find out a recursive formula for $f(p)$ by breaking it down into 3 components :

- average number of bits generated by level 0 alone
- average number of bits generated by level A
- average number of bits generated by level B

$$
\begin{aligned}
2f(p) \quad = \quad & \Pr(generate\_1bit\_to\_output) \times 1 + \\
& \Pr(generate\_1flip\_to\_levelA) \times f(\Pr(head\_in\_levelA)) + \\
& \Pr(generate\_1flip\_to\_levelB) \times f(\Pr(head\_in\_levelB))
\end{aligned}
$$

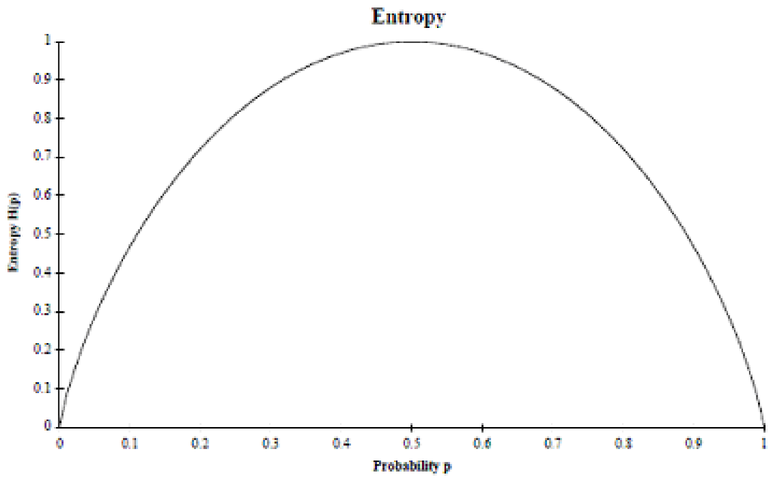
$$
\begin{aligned}
\Pr(generate\_1bit\_to\_output) &= 2pq \\
\Pr(generate\_1flip\_to\_levelA) &= p^2 + q^2 \\
\Pr(generate\_1flip\_to\_levelB) &= 1 \\
\Pr(head\_in\_levelA) &= p^2/(p^2+q^2) \\
\Pr(head\_in\_levelB) &= p^2 + q^2
\end{aligned}
$$

$$
\Rightarrow \qquad 2f(p) = 2pq \times 1 + (p^2+q^2) \times f(p^2/(p^2+q^2)) + 1 \times f(p^2+q^2)
$$

It is difficult to solve for f(p) from the above formula. However entropy gives us the maximum information generated by a given p :

$$
H(p) = -p\log_2 p - q\log_2 q
$$

If the advanced multi level strategy is an optimum strategy, then entropu *H(p)* should fulfill the recursive formula.



Lets verify…

$$
\begin{aligned}
H(p) &= -p\log_2 p - q\log_2 q \\
H(p^2+q^2) &= -(p^2+q^2)\log_2(p^2+q^2) - (2pq)\log_2(2pq) \\
H(p^2/(p^2+q^2)) &= -(p^2/(p^2+q^2))\log_2(p^2/(p^2+q^2)) - (q^2/(p^2+q^2))\log_2(q^2/(p^2+q^2)) \\
&= \begin{array}{l} -(p^2/(p^2+q^2))\log_2(p^2) + (p^2/(p^2+q^2))\log_2(p^2+q^2)) \\ -(q^2/(p^2+q^2))\log_2(q^2) + (q^2/(p^2+q^2))\log_2(p^2+q^2)) \end{array} \\
&= -(p^2/(p^2+q^2))\log_2(p^2) - (q^2/(p^2+q^2))\log_2(q^2) + \log_2(p^2+q^2)) \\
(p^2+q^2)H(p^2/(p^2+q^2)) &= -p^2\log_2(p^2) - q^2\log_2(q^2) + (p^2+q^2)\log_2(p^2+q^2)
\end{aligned}
$$

Lets consider the RHS of recursive function.

$$
\begin{aligned}
RHS &= pq + (p^2+q^2)H(p^2/(p^2+q^2))/2 + H(p^2+q^2)/2 \\
&= pq - p^2\log_2(p^2)/2 - q^2\log_2(q^2)/2 + (p^2+q^2)\log_2(p^2+q^2)/2 - (p^2+q^2)\log_2(p^2+q^2)/2 - pq\log_2(2pq) \\
&= pq - p^2\log_2 p - q^2\log_2 q - pq - pq\log_2(pq) \\
&= -p^2\log_2 p - q^2\log_2 q - pq\log_2 p - pq\log_2 q \\
&= -(p+q)p\log_2 p - (p+q)q\log_2 q \\
&= -p\log_2 p - q\log_2 q \qquad \text{since } p+q=1 \\
&= H(p)
\end{aligned}
$$

Here is the implementation of 3 methods together :

```
class unbiased_coin
{
    enum STEP { NONE, HALF };
    enum FLIP { HEAD, TAIL };

    std::vector<bool> add_flip(const FLIP& flip)
    {
        std::vector<bool> output;
        if (step == NONE)
        {
            step = HALF;
            flip_one = flip;
        }
        else
        {
            step = NONE;

            if (flip_one != flip)
            {
                // *** Method 1 *** //
                result.push_back(flip_one == HEAD? true:false);

                // *** Method 3 *** //
                if (!rhs) rhs = std::make_shared<unbiased_coin>();
                auto tmp = rhs->add_flip(HEAD);
                for(auto& x:tmp) result.push_back(x);
            }
            else
            {
                // *** Method 2 *** //
                if (!lhs) lhs = std::make_shared<unbiased_coin>();
                auto tmp = lhs->add_flip(flip);
                for(auto& x:tmp) result.push_back(x);

                // *** Method 3 *** //
                if (!rhs) rhs = std::make_shared<unbiased_coin>();
                auto tmp = rhs->add_flip(TAIL);
                for(auto& x:tmp) result.push_back(x);
            }
        }
        return output;
    }

    STEP step = NONE;
    FLIP flip_one;
    std::shared_ptr<unbiased_coin> lhs;
    std::shared_ptr<unbiased_coin> rhs;
};
```

Reference

[1] Tossing a Biased Coin, by Michael Mitzenmacher.