# Kelly's Criterion

Eldad Kronfeld

2021-09-01

Thank you Hagai for the continuing patience and support along the way.

## Contents

## 1 Abstract

This is a paper about the mathematical model proposed by John Kelly at the early 1950s, however instead of looking at the model from the lenses of games and gambles, each of our gamble is an option we can invest in until the next round of the model, at the beginning I will explain the basic concept behind the model and show simple example for one investments with multiple outcomes at each round, the results will be shown by using Monte Carlo simulation, later on I will expend the module to include multiple investments each with multiple outcomes.

## 2 The Base model

### 2.1 Defining The Model

Kelly's Criterion is an infinite model in discrete time frames, which I will call ticks\rounds. at each round we have sum of money that we would like to invest and the model represent what happens each round with the split that we decided on.
However, the premise of the model is that the investments or gambles we take each round have expectations that are greater then 1,

$$\mathbf{E}_{gain} \geq 1$$

meaning that in the long run we can expect that we will yield revenue from the venture and not loss from it.
**Note:** through all of the explanations and simulations I will normalize the odds to be such as that we start with 1\$ and each gain will be noted as $\gamma \cdot F_{N-1}$, where $F_{N-1}$ denotes the sum of money from round $N-1$ and $\gamma$ denotes the gain and $\gamma \in [0, \infty)$, so when $\gamma = 1$ it means that there is no gain and no loss at all, however when $\gamma > 1$ we gain money and when $0 \leq \gamma < 1$ we can expect to loss money.

if we try to define a simple model, where we have 1 investment with $k$ outcomes, where each outcome have $p_i$ probability of happening and we invest $f$ amount of money and keep $b = 1 - f$

$$F_N = \begin{cases} (b + f \cdot \gamma_i) F_{N-1} & N > 1, \text{ for } i = 1, .., k \\ 1 & for \;\; N = 0 \end{cases} \tag{1}$$

this equation represent that if outcome $i$ happens then the part that we didn't invest in $(1 - f)$ doesn't change, however the part that we did invest $f$ changes by $\gamma_i$.
ultimately the way I looked at it, which helped me later to define more complicated model would be:
*let* $X$ be random variable that represent the gain of each outcome, then if the

outcome is $\gamma_i$ then $P(X = \gamma_i) = p_i$, this will help us look at the exponential growth and develop a way to find the optimal fraction $f$ in each round to achieve the optimal returns.
now let's look for the fraction $f$ that produces the best gain, because equation 1 we can deduce that the growth rate from an investment is exponential , then:

$$F_N = F_0 e^{G_N N} \tag{2}$$

so $G_N = \frac{1}{N} log(\frac{F_N}{F_0}) = \sum_{i=1}^{k} \frac{W_i}{N} log(b + \gamma_i f)$, however when we take N to $\infty$ in equation 2 we get the following equation:

$$G = \lim_{N\to\infty} \sum_{i=1}^{k} \frac{W_i}{N} log(b + \gamma_i f) = \sum_{i=1}^{k} p_i log(b + \gamma_i f) \tag{3}$$

then we would like to look for the maximum.

**Theorem 2.1.** *$f(x) = -log(b(x) + \gamma x)$ is convex when $x \in (0, \infty)$.*

*Proof.* $\frac{\partial f}{\partial x} = -\frac{\gamma-1}{b+\gamma x}$ and the second derivative is $\frac{\partial f}{\partial x^2} = \frac{(\gamma-1)^2}{(b+\gamma x)^2}$
second derivative is always positive, then $f(x)$ is convex.

□

**Lemma 2.2.** *$G(f) = \sum_{i=1}^{k} p_i log(b + \gamma_i f)$ is concave.*

*Proof.* when we look at equation 3 as a function of $f$,
$G(f)$ is concave $\iff$ $-G(f)$ is convex.

$$-G(f) = \sum_{i=1}^{k} p_i(-log(b + \gamma_i f))$$

however a positive weighted sum of convex functions, is a convex function, each component $s_i = p_i log(b + \gamma_i f)$ is convex because $\gamma_i$ is non-negative , and $0 \le p_i \le 1$ and Theorem 2.1 ,then $-G(f)$ is convex, making $G(f)$ concave.

□

because $G$ is concave, we will be able to find it's maximum by calculating the root of the Derivative, which is

$$\frac{\partial G}{\partial f} = \sum_{i=0}^{k} p_i \frac{\gamma_i - 1}{b + \gamma_i f} = 0$$

and we know that such a root exist inside $[0, 1]$ because of last lemma, stating that $G(f)$ is concave making it's second derivative always non-positive, then the maximum point should be either at the end of the range at 1 or somewhere inside it, because $G'(0) = \sum_{i=0}^{k} p_i \frac{\gamma_i-1}{b} = \frac{1}{b}(E_{gain} - \sum_{i=1}^{k} p_i) = \frac{1}{b}(E_{gain} - 1) > 0$, this statement is true because the sum of all the possibilities is 1 and $E_{gain} \geq 1$. the actual root is very tough to find, however at my own python implementation I used Brent's method which is a hybrid method combining bisection method, the secant method and inverse quadratic interpolation, which was highly recommended by Scipy's optimization documentation.

## 2.2 First Model's Monte Carlo Simulations

After looking at the model analytically and arguing that an optimal fraction does exist, we can simulate scenarios that will help us validate our statements. In order to accomplish such a task, I wrote code in point that represent a simulation, where in each round of the simulation we will pick one of the outcomes, however the probability to pick a certain outcome will be defined, so less common occurrences will be less likely to happen then the more common ones.
After creating classes in order to define the probabilities and the general flow of the simulation I saved for each instance of the simulation I ran, the amount of fortune at each round in order to plot graphs like Fig 1 in order to show that our calculation were right and the optimal value does exist and it works in real life situations.

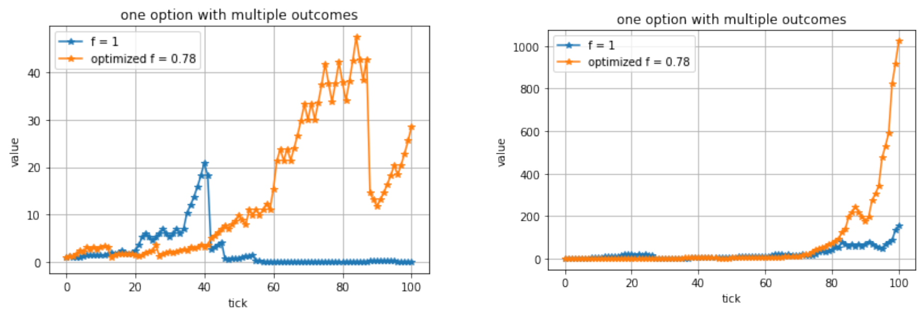

Figure 1: Two random simulation with and without optimal value

# 3 The Expended Model

## 3.1 Defining The Model

The new model will be very similar in some aspects to the Base model, but at the same time different and more robust, In order to include more option with more outcomes so we will be able to calculate something more realistic where each options has it's own distribution and returns and our choice will not be own dimensional, but multidimensional, meaning we will need to know how to split our cash each round between not-investing and all of the options together. in order to actually start talking about the return each round, we will need to define our fortune at each round.
at first we will define a group of vectors to be:

$$F_N = \begin{cases} (b + \overrightarrow{\alpha} \cdot \overrightarrow{f})F_{N-1} & N > 1, \text{ for each posible outcome } \overrightarrow{\alpha} \\ 1 & for\ \ N = 1 \end{cases} \tag{4}$$

unlike in the previous model where each outcome was simply a real number, here it is a vector of real numbers representing the return of each option (returns can be 0 if the option didn't win at all or even negative if the loss was high enough) in the vector the $i$-th element represent the return of option number $i$.
However if an option did win and we didn't invest in it, then the value in the vector $\overrightarrow{f}$ in the appropriate element would be 0 meaning it will not be taken into account in the following round.
sadly finding such the joint distribution is not a simple task so from now on we will assume that the options are independent from one another in order to simplify the calculation of the joint probability, we will denote the random vector as $X$ with $l$ coordinates and each $X_i$ would denote the $i$-th random variable inside X:

$$P(X = \overrightarrow{\alpha}) = \prod_{i=1}^{l} P(X_i = \alpha_i) \tag{5}$$

After defining both the probability of each outcome (eq. 5) and the value at each round (eq. 4) we can create a working Monte-Carlo simulation, However that's not all.
As before I will ask the big question: is there an optimal money distribution that would provide the best gain in the long run?
Turns out, there is but in order to understand why it would take a bit more:
in order to find such a distribution we will define a non-linear optimization problem that we would need to solve in order to find it.
Looking at equation (2) I would like to expend that and find the gain at $\infty$ for our current model.

$$S_l = \left\{ \overrightarrow{\alpha} | \overrightarrow{\alpha} \in \mathbb{R}^l, \text{ for every vector } \overrightarrow{\alpha} \text{ of possible outcome for } l \text{ options} \right\}$$

$$G(\overrightarrow{f}) = \lim_{N\to\infty} \frac{1}{N} \log(\frac{F_N}{F_0}) = \sum_{\overrightarrow{\alpha} \in S_l} P(X = \overrightarrow{\alpha}) \log(b + \overrightarrow{f} \cdot \overrightarrow{\alpha}) \tag{6}$$

In this model we have $l$ options, option $i$ have $k_i$ outcomes, however that is not the end, we have a constraint that make sure that we don't distribute more money then we have, and each fraction is a positive part of our overall, money. making it a maximization problem on a simplex.

**Maximize:**

$$G(\overrightarrow{f}) = \sum_{\overrightarrow{a} \in S_l} P(X = \overrightarrow{a}) \log(b + \overrightarrow{f} \cdot \overrightarrow{a})$$

**s.t:**

$$b + \sum_{i=1}^{l} f_i = 1$$

$$f_i \geq 0, \text{ for } i = 1..l$$

solving such a system is not an easy task, which would require calculating very big system of Lagrangian multipliers, however before trying to actually solve the system I would like to make a few statements.

**Theorem 3.1.** *$G(f)$ (eq. 6) is a concave function in the non-negative part of $\mathbb{R}^l$.*

*Proof.* In order to prove that $G(f)$ is concave, I will prove that $-G(f)$ is convex. So let's look at each component of of the sum, and prove that it is convex

$$C_a(f) = -\log(b + \overrightarrow{f} \cdot \overrightarrow{a})$$

because $C_a(f)$ is a composition of $y(x) = -log(x)$ which is nondecreasing, univariate, convex function with multivariate hyperplane which is a convex function, we get that the composition is convex (see [1] page 99).
and the sum of all $C_a(f)$ is convex making $-G(f)$ convex.

□

because $G(f)$ is concave our problem is finding the maximum of a concave function, which grant us couple of things:

1. we know that a maximum point exists in the constraints.
2. if we use iterative algorithm that converge to a local maxima, we get that the convergence point is in fact the maxima in the constraints.

because of advantage number (2) I decided to solve the optimization problem by using an algorithm called *SLSQP* (Sequential Least SQuares Programming) which aims to find a minima inside the constraints, so I first of all changed the problem to a minimization problem:

**Minimize:**

$$H(\overrightarrow{f}) = -\sum_{\overrightarrow{a} \in S_l} P(X = \overrightarrow{a}) \log(b + \overrightarrow{f} \cdot \overrightarrow{a})$$

**s.t:**

$$b + \sum_{i=1}^{l} f_i = 1$$

$$f_i \geq 0, \text{ for } i = 1..l$$

making it a convex minimization problem, which insures that a global minima exists, which is important because SLSQP solves converges to a local minima, which in our case, would be the optimal solution.
Sequential Quadratic Programming that is used in SLSQP is a way to solve the KKT equations using constrained quasi-Newton methods that guarantee superlinear convergence by accumulating second-order information regarding the KKT equations, since QP subproblems are solved at each major iterations.

It is important to note that running the SLSQP algorithm to find the optimal fraction was extremely fast, which is very assuring if this work will become the basis for an actual trading bot, which would need fast runtime in order to trade in high frequency which is very ideal according to (add physics article).

## 3.2 The Expended Simulations

The new simulation holds in memory $l$ (user defined number) of tables that represent the distribution of each option and general information like: rounds, $F_0$, $\overrightarrow{f}$ and $b$. afterwards it runs the model for the amount of rounds defined at the beginning each time picking an outcome from each of the options and calculating the amount of money at each round. at the end of the simulation the simulation returns an array (Numpy array) of all the results we gathered along the way. which I then take in order to compile graphs and insights from the Monte-Carlo runs. however in order to make the model as general as possible the classes defining the simulation where as general as possible in order to get as many options as we want with as many outcomes as we want for each one of them, which lead to some difficulties when it came time to use SLSQP because I needed to create the target function and the constraints, so what I decided to do at the end is look at the $b$ parameter as the first parameter of the vector $\overrightarrow{f}$ which in a way could be described as an option with 100% chance to return 100% of the value (safe option), looking at this in this way helped me return the target function as the huge sum of all the possibilities and the returns. in all of the work I used Numpy in order to speed the simulation up by saving all of the relevant data in the same memory space which led to less memory fetches overall.
After creating all the necessary underwork and pipelines, I was able to produce a piece of software which is very flexible and able to produce as many different

experiments as I wanted without a need to write a lot more code for each one of the new tests I conducted.

if we look at Fig (2) we can see four random runs of 3 options each with 2 outcomes, while Fig (3) represent a bigger system of 5 options each with 3 outcomes.

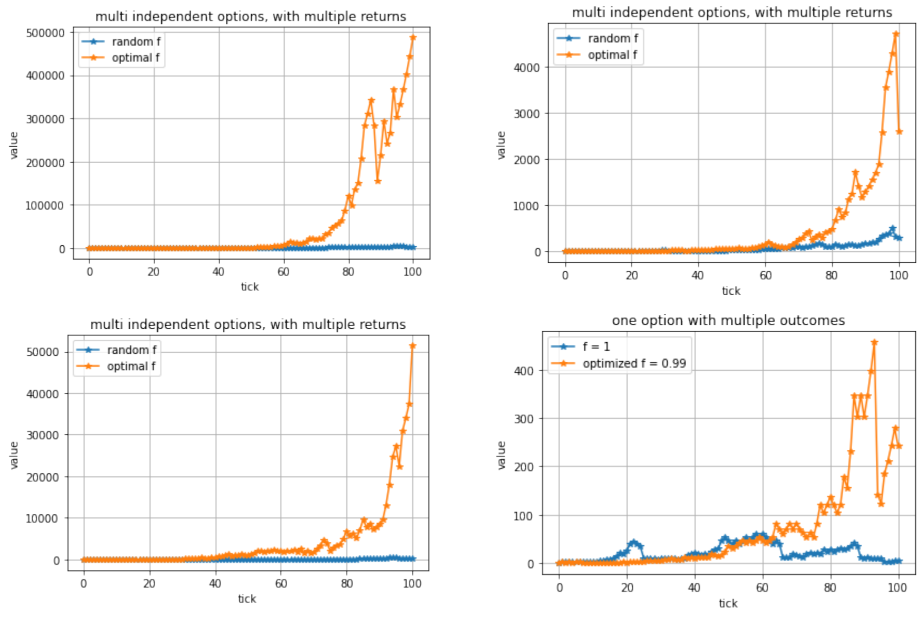

Figure 2: Four random simulation with and without optimal value, 3 options each with 2 outcomes

because of the random nature of the simulation and the fact that we look for the values of $\overrightarrow{f}$ that will produce the best results asymptotically making some of the simulations that I tried to run produce results that are countering our statements where the random money distribution produce better results then the optimal one.

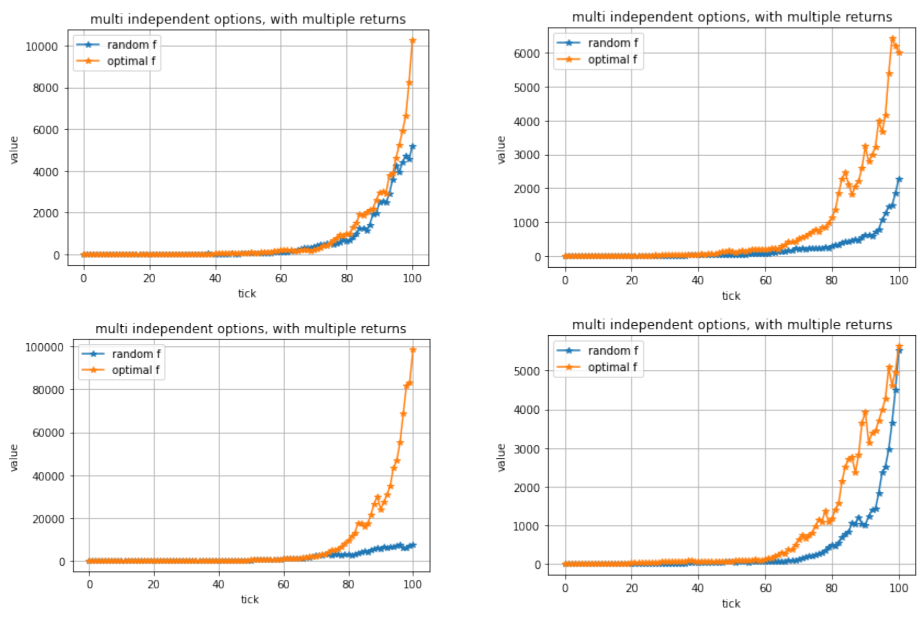

Figure 3: Four random simulation with and without optimal value, 5 options each with 3 outcomes

### 3.2.1 Example

let's look at the following scenario, where we have 5 options, each with the same distribution (each produce $E_{gain} \geq 1$) and see what would be the best money distribution in such a scenario:

| **probability:** | 0.3 | 0.5 | 0.2 |
|---|---|---|---|
| **gain** | 0.5 | 1.15 | 1.5 |

Table 1: The outcome of each option in the example

$$\text{and the optimal value we got is } \overrightarrow{f} = \begin{pmatrix} 0.1666 \\ 0.1666 \\ 0.1666 \\ 0.1666 \\ 0.1666 \\ 0.1666 \end{pmatrix}.$$

which could be surprising considering that all of the options are the same, and could lead us to believe that maybe even if have many options that are similar to each other we should invest only in one or a couple of them, however it turns out that it's the best to stretch our investments over all of them instead of only

a couple of them.
**Note:** this simulation is only a single case, and more work should be put into this question in order to make an absolute statement.

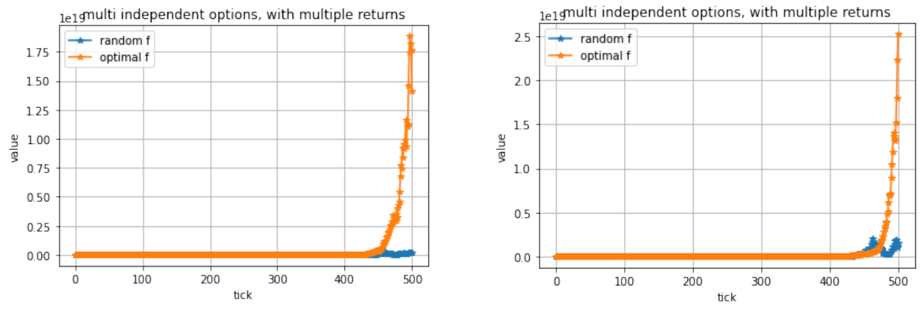

Figure 4: Two simulations for the example with random and optimal values

## 4 Summery and conclusions

Kelly's criterion can be viewed as a simplified game of chance with a very powerful massage, **don't stretch yourself too thin**, the model gives us clues into the world of investments where we could sometimes win and sometimes loss, but the idea is mitigating loss, instead of looking to quick gain we look at the long run by cutting future losses and making sure that when we loss, we won't loss it all, but as a trade off when we gain we will not gain a well as we could have.
When we look at the expended model we learn that splitting our money between similar investments is a good way to go, further enforcing the idea that the poor gets poorer and the rich, richer because they have much [**Feng8388**] more "gas" to stretch themselves instead of just looking for quick gains because of the need to gain. in addition the expended model at least to my own testing runs very fast which is reassuring because if this work will be the basis of a trading bot that would implement a strategy to invest in multiple options, it would be very strong point that the bot will be agile and will be able to trade in high frequency instead of a more fundamental approach in order to maximize the gain it would otherwise have lost because of the money split.

## References

[1] Hanif D. Sherali Mokhtar S. Bazarra C. M. Shetty. *Nonlinear Programming Theory and Algorithms*. Jhon Wiley Sons, 2006. ISBN: 978-0-471-48600-8.