Report for CodeQuest

elagnitram

January 2024

1 Introduction

In this report we will present the analysis of our strategy to supply liquidity in multiple liquidity pools. More precisely, our goal is to determine the optimal distribution across the n pools. This distribution aims to maximise a specific performance metric defined as

$$\min_\theta \ \mathrm{CVaR}_\alpha[r_T]$$

under the condition that the probability $P[r_T > \zeta] > q$. Here, $r_T = \log(x_T) - \log(x_0)$ represents the final performance of the LP, while θ denotes how the initial wealth is allocated among the n pools. The parameters α, ζ, and q are specified in the file `params.py`.

2 Methodology

We are implementing gradient descent on the initial percentages of X-coins assigned to the first five of six pools. The quantity in the sixth pool passively changes to ensure the total sum remains 10. Initially, we set six parameters representing the initial guesses for the six pools. Using Monte Carlo methods, we calculate the conditional Value-at-Risk (CVaR) and the (1-q) quantile ξ.

We define δ_1 and δ_2 as criteria to judge if ξ is approaching the critical value during training. We consider the following three cases:

1) If $\xi < \zeta + \delta_1$, we assume the constraint has been violated, and then set parameters to be the one before updating and try to update the parameters with a smaller step size. If we encounter this case several times, we assume the optimal result has been found and consider the gradient descent to be complete.

2) If $\xi > \zeta + \delta_2$, we update parameters using the gradient of CVaR, assuming the updated parameters would not violate the constraint.

3) If $\zeta + \delta_1 < \xi < \zeta + \delta_2$, we calculate the gradients of both CVaR and ξ. We do not want ξ to be smaller anymore as it is close to ζ. To achieve this, we project the gradient of CVaR to the space where ξ would not descend. More specifically, If the dot product of the gradients of ξ and CVaR is positive, (i.e., $(\nabla\xi)^\top \nabla\mathrm{CVaR} > 0$), then we set our searching direction to be $\nabla\mathrm{CVaR} - \lambda\nabla\xi$, where $\lambda = ((\nabla\xi)^\top \nabla\mathrm{CVaR})/((\nabla\xi)^\top \nabla\xi)$ to make the direction vertical to $\nabla\xi$.

After these, we can determine our search direction, and then we can update the parameters. However, given the parameters are the percentages of X-coins assigned to each pool, they are naturally required to be positive and have a sum being 1. Thus, the following two cases need to be considered.

2.1 Case 1

If some of the updated parameters (assigned weights of each pool) become negative, or the sum of these parameters is greater than 1, then we find the largest admissible step size to make sure that these parameters are all positive with the sum greater than 1. Here, we assume that, if the parameters are on the boundary, they are always on the boundary during the potential subsequent updates.
After finding the largest admissible step size, we update parameters based on this step size. Then, we set all the parameters that are equal to zero after the update to be fixed, meaning these parameters would always be zero during the potential subsequent updates. Because the sum of the parameters should be 1, we set one of the parameters that have not been fixed to change passively, i.e. move according to the movement of other non-fixed parameters to make sure the sum equals 1.

2.2 Case 2

If negative parameter updates do not occur, we proceed with updating the parameters based on the learning rate.

2.3 Further actions

After updating the parameters, we need to evaluate the CVaR and the quantile, repeating the entire procedure until the magnitude of the CVaR change post-update is no greater than a predetermined threshold ϵ.

Once the optimal parameters are determined, new samples should be generated due to the formula's imprecision, and the gradient descent search should be repeated until the CVaR change post-update does not exceed a second threshold ϵ_1 that we set. However, in practice, the Monte Carlo method is computationally expensive, so we limit the gradient descent algorithm to a dozen iterations.

3 Computation

We are interested in implementing Monte Carlo methods to estimate the log return and we will derive an explicit formula for that. Suppose that we have generated M samples of pools. In this case, the return of each scenario is $\{r_j\}_{j=1}^{M}$. Furthermore, we assume that these are sorted from small to large. Then the conditional Value-at-Risk for each return r_j is given by

$$\mathrm{CVaR}(r_j) = \frac{1}{M_\alpha} \sum_{j=1}^{M_\alpha} r_j,$$

where M_α is the index corresponding to the desired quantile. Thus the derivative of the conditional Value-at-Risk can be obtained as the sum of the derivative of the return.

We are now in the position to derive the formula of the derivative of the returns. Suppose that initially the X coins and Y coins of each pool are

$$\{(R^i_{(X,0)}, R^i_{(Y,0)})\}_{i=1}^{N},$$

and the weights of investment are $\theta = (\theta_1, \ldots, \theta_N)^\top$, where $\sum_{i=1}^{N} \theta_i = 1$. Because the X coins that we owned at the beginning are small compared to the reserve in each initial pool, we can

approximately assume that the increment of X coin in each pool is

$$\left\{\frac{1}{2}x_0\theta_i\right\}_{i=1}^{N},$$

where x_0 is the total amount of X coins that we owned at the beginning. Thus the LP coins in each pool are about

$$\left\{\frac{\frac{1}{2}x_0\theta_i}{R^i_{(X,0)}}\right\}_{i=1}^{N},$$

given the initial LP coins being 1. Let us denote

$$\left\{\frac{\frac{1}{2}x_0\theta_i}{R^i_{(X,0)}+\frac{1}{2}x_0\theta_i}\right\}_{i=1}^{N},$$

by $\{Q_i\}_{i=1}^N$. Suppose that the reserve of pools at the end time T are

$$\{(R^i_{(X,T)}, R^i_{(Y,T)})\}_{i=1}^N.$$

Then the X coins and Y coins are

$$\{(R^i_{(X,T)}Q_i, R^i_{(Y,T)}Q_i)\}_{i=1}^N,$$

denoted by $\{(X_T^{(i,\theta_i)}, Y_T^{(i,\theta_i)})\}_{i=1}^N$. To swap Y to X, we need to compare the X coins that we can gain from each pool, which are

$$\left(\frac{(1-\phi_i)R^i_{(X,T)}}{R^i_{(Y,T)}+(1-\phi_i)\sum_{j=1}^N(Y_T^{(j,\theta_j)})-Y_T^{(i,\theta_i)}}\right)\cdot\left(\sum_{j=1}^N(Y_T^{(j,\theta_j)})\right).$$

Thus the total X coins that we gain in the end are

$$\left(\sum_{i=1}^N X_T^{(i,\theta_i)}\right)+\left[\max_{1\le i\le N}\left\{\frac{(1-\phi_i)R^i_{(X,T)}}{R^i_{(Y,T)}+(1-\phi_i)\sum_{j=1}^N(Y_T^{(j,\theta_j)})-Y_T^{(i,\theta_i)}}\right\}\cdot\left(\sum_{j=1}^N Y_T^{(j,\theta_j)}\right)\right].$$

This is quite a complicated formula. Only an approximate derivative can be obtained, as it is meaningless to get a precise derivative from an estimation. So, at this point, we assume that the main contribution of the derivative comes from the derivative of $\{Q_i\}_{i=1}^N$. Thus, the derivative with respect to θ_i is given by

$$R^i_{(X,T)}\cdot\frac{\partial}{\partial\theta_i}Q_i+\max_{1\le i\le N}\left\{\frac{(1-\phi_i)R^i_{(X,T)}}{R^i_{(Y,T)}+(1-\phi_i)\sum_{j=1}^N(Y_T^{(j,\theta_j)})-Y_T^{(i,\theta_i)}}\right\}\cdot R^i_{(Y,T)}\cdot\frac{\partial}{\partial\theta_i}Q_i$$

Then using the chain rule, we can get the derivative of log return.

4 Discussion

We begin our initial estimate by assigning equal weights to each pool, i.e., each parameter is equal to 1/6. Based on our simulation, we obtain the optimal investment weight is given by

$$(\theta_1, \ldots, \theta_6) = [0.17468533, 0.16354817, 0.14823748, 0.18512584, 0.22718518, 0.10121879].$$

Moreover, in this case, the conditional Value-at-Risk of current weights is 8.00461, and the probability of larger than ζ is 0.839, suggesting that the performance of our implementation is good enough.

However, there may exist several ways to refine our methods. For instance, we could initiate our initial guesses by exploring a broader range of weight distributions across the pools, not just limiting ourselves to equal weights. This approach might uncover more efficient allocations that could potentially enhance the overall performance metrics. Apart from this, the formula that was used to calculate derivatives can be improved by taking the derivative of the Reserves of each pool into consideration or finding a better approach to deal with the second constrain. The previous one requires a good formula to describe the relationship between X, Y coins and the weights we assigned to each pool which is assumed to be constant in our model.