Technical Whitepaper: Socean's delegation strategy

Socean Team

2021-07-30

Abstract

A stake pool is an entity that receives stake from depositors and delegates stake to *validator nodes*, which give it a return on funds delegated.

How a stake pool should divide its stake is an open and important question, as it has implications for pool performance and network security. We combine Bayesian mean-variance optimisation and utility theory to derive an optimal delegation strategy.

Contents

Background

Proof-of-stake blockchains require *validator nodes* to vote and come to consensus on the state of the chain. A validator node (or *validator*) receives candidate blocks from different nodes and must vote for the "correct" block.

A validator can have tokens *staked*, or delegated, to it. Validators are rewarded for voting correctly, and stakers are rewarded for delegating to that validator. They can also be punished if they vote incorrectly, where tokens staked with the validator can be "slashed".

A validator's staking rewards are affected by various factors like their own hardware, network latency, power outages, and so on. Performance may also be correlated between validators: if a single data center goes down, for example, validator nodes belonging to the same data center will all perform poorly.

A *stake pool* is an entity that receives (pools together) stake from depositors and delegates stake to validators, which earn staking rewards according to their performance. A stake pool therefore needs to pick the right validators in order to provide a good return for its depositors.

At the same time, however, stake pools have a duty to secure the network and improve its health by increasing the minimum security group (the set of nodes that would have to be compromised in order to stall consensus). The Solana Foundation's stake pool announcement writes:

> ... systematically spreading the staked SOL across a larger number of validators... makes the network harder to attack.

Even if stake pools were completely selfish, they would still want to improve decentralisation to avoid an attack on the network which would decrease the value of their holdings.

How should we delegate stake to validators in a way that maximises staking rewards and censorship resistance/network health? We develop a delegation strategy which trades off returns with network health in an optimal manner.

Delegation strategy

Let x_t^i be the performance validator i obtains at time t. Suppose we have the historical performance matrix $\mathbf{x}$ for all k validators for some time period $(n, ..t-1)$:

$$\mathbf{x} = \begin{vmatrix} x_n^1 & x_{n+1}^1 & .. & x_{t-1}^1 \\ .. & .. & .. & .. \\ x_n^k & x_{n+1}^k & .. & x_{t-1}^k \end{vmatrix}$$

There is remarkable persistence in the performance of validators, and so past performance is in fact predictive of the future. Our job is to decide on an

delegation vector $\mathbf{w}$ for time t, given that the sum of the weights must equal unity ($\sum_{i=1}^{k} w_{it} = 1$), to maximise the total returns $\mathbf{w}^\top \mathbf{x}_t = \sum_{i=1}^{k} x_t^i w_{it}$.

Denote the total returns in period t as C. If we are risk-averse, then the benchmark exponential utility function is written as

$$u(C) = -e^{-\beta C}$$

where β is a constant which expresses the stake pool's risk aversion. When the Socean governance token is issued, depositors will be able to vote on this parameter β.

Given that $C = \mathbf{w}^\top \mathbf{x}_t$, we can rewrite this function as a function of $\mathbf{w}$ alone:

$$u(\mathbf{w}) = -e^{-\beta \mathbf{w}^\top \mathbf{x}_t}$$

We augment the utility function with a *network health function* to get

$$u(\mathbf{w}) = -e^{-\beta \mathbf{w}^\top \mathbf{x}_t - \alpha \eta(\mathbf{w})}$$

Here $\eta(\mathbf{w})$ is the "network health" score that would be obtained by some stake delegation $\mathbf{w}$. The parameter α measures how much weight we put on network health[1].

The network health score $\eta(\mathbf{w})$ is the weighted sum of minimum security group and data center decentralisation:

$$\eta(\mathbf{w}) = MSG_{Stake}(\mathbf{w}) + \gamma MSG_{DataCenter}(\mathbf{w}),$$

where

$$MSG_{Stake}(\mathbf{w}) = 1 - \frac{1}{MinNodes - 1}$$

$$MSG_{DataCenter}(\mathbf{w}) = 1 - \frac{1}{MinDataCenters - 1}$$

$MinNodes$ refers to the minimum number of nodes that would have to go down in order to stall the network and cause a liveness violation. This is equivalent to the largest N nodes that together possess more than 1/3rd of total staked SOL. Similarly, $MinDataCenters$ refers to the minimum number of data centers that would have to go down in order to stall the network.

[1] It can also be thought of (non-rigorously) as one's prior probability that an attack on the network will occur.

The function approaches negative infinity as $MinNodes$ approaches 1. This is desired, as such a situation would be incredibly dangerous for Solana. As $MinNodes$ increases, $MSG(\mathbf{w})$ approaches 1.

Mean-variance equivalence

We have written down the utility function: we now consider how to maximise it under uncertainty. The vector of rewards $\mathbf{X}_t$ is in fact a random variable; $\mathbf{x}_t$ is the *actual performance* we observe at time t. The von Neumann-Morgenstern axioms gives us that the optimal way to maximise our utility function under uncertainty is to maximise *expected utility*. Taking expectations gives us

$$\mathbb{E}[u(\mathbf{w})] = e^{-\alpha\eta(\mathbf{w})}\mathbb{E}[-e^{-\beta\mathbf{w}^\top\mathbf{X}_t}],$$

since $\alpha\eta(\mathbf{w})$ is a deterministic function.

Directly maximising this expected utility function is difficult. In order to do so one must determine the joint probability distribution and estimate the parameters for that joint distribution. We therefore use *mean-variance optimisation* as an approximation to expected utility maximisation. It is known that the exponential utility function is well-approximated by a mean variance representation: see Markowitz 2014, Kroll, Levy and Markowitz 1984.

We can thus rewrite the expected utility function in mean-variance terms, which gives

$$\mathbb{E}[u(\mathbf{w})] = -e^{-\beta[\mathbb{E}[(\mathbf{w}^\top\mathbf{X}_t)]-\frac{\beta}{2}Var(\mathbf{w}^\top\mathbf{X}_t)]-\alpha\eta(\mathbf{w})}.$$

Utility functions are ordinal, so we can take logs and divide through by β to obtain the familiar functional form

$$\mathbb{E}[u(\mathbf{w})] = \mathbb{E}[(\mathbf{w}^\top\mathbf{X_t})] - \frac{\beta}{2}Var(\mathbf{w}^\top\mathbf{X_t}) + \frac{\alpha}{\beta}\eta(\mathbf{w}).$$

Deriving the optimal weights

We have just written the utility function for a stake pool parameterised by some weight allocation $\mathbf{w}$ in the mean-variance form. We now have to estimate the first and second moments of $\mathbf{X_t}^\top\mathbf{w}$ *conditional on the historical performance we observe*: that is to say,

$$\mathbb{E}[u(\mathbf{w})] = \mathbb{E}[(\mathbf{w}^\top\mathbf{X_t}|\mathbf{x_{(t-1)}})] - \frac{\beta}{2}Var(\mathbf{w}^\top\mathbf{X_t}|\mathbf{x_{(t-1)}}) + \frac{\alpha}{\beta}\eta(\mathbf{w}).$$

Writing the utility function in this form allows us to use the literature on mean-variance optimisation. The traditional approach to mean-variance optimisation is

to estimate the optimal portfolio weights by applying the historical observations of the asset returns. However, replacing these optimal parameters by these sample estimators introduces sampling error, which can cause the estimated optimal weights to deviate substantially from the true optimal weights.

Bauder et al. (2020) derive a Bayesian method of constructing the mean-variance optimal portfolio. Crucially, this allows us to optimise directly in terms of the historical data $\mathbf{x}$, rather than having to estimate the variance and covariance matrices. Given a number of observations n and k different validators, they show that the utility function we just derived can be shown to equal

$$U(\mathbf{w}) = \mathbf{w}^\top \bar{\mathbf{x}}_{t-1} - c_{k,n} \frac{\beta}{2} \mathbf{w}^\top \mathbf{S}_{t-1} \mathbf{w} + \alpha \eta(\mathbf{w}),$$

where

$$c_{k,n} = \frac{1}{n-k-1} + \frac{2n-k-1}{n(n-k-1)(n-k-2)},$$

$$\bar{\mathbf{x}}_{t-1} = \frac{1}{n} \sum_{i=t-n}^{t-1} \mathbf{x}_i,$$

and sample variance

$$\mathbf{S}_{t-1} = \sum_{i=t-n}^{t-1} (\mathbf{x}_i - \bar{\mathbf{x}}_{t-1})(\mathbf{x}_i - \bar{\mathbf{x}}_{t-1})^\top$$

We can now obtain the optimal weight vectors $\mathbf{w}$ by a hill-climbing/gradient descent algorithm.

Properties of the solution

This strategy has several attractive properties:

- **Optimal.** Provided you share the same preferences as the stake pool (and you will be able to vote on those preferences), the utility function we derive is optimal (it is the allocation of stake to validators that maximises your expected utility).
- **Parameterisable.** The strategy lets us adjust the degree of risk-aversion and network safety in a rigorous and systematic way. These parameters can be voted on via governance tokens in the future, for delegators themselves to choose their risk appetite.

Bibliography

- Bade, Frahm, and Jaekel (2008). A General Approach to Bayesian Portfolio Optimization.
- Baron (1977). On the Utility Theoretic Foundations of Mean-Variance Analysis.
- Bauder, Bodnar, Parolya, and Schmid (2020). Bayesian mean-variance analysis: optimal portfolio selection under parameter uncertainty
- Bernado and Smith (2000). Bayesian Theory.
- Boadway, Cuff and Marchand (1999). Optimal Income Taxation with Quasi-Linear Preferences Revisited.
- Bowles, Carlin, and Stevens (2007). 5.4.1 Quasi-linear preferences in The CORE Team, The Economy. Retrieved 2021-07-04.
- Gelman (2009). Can you do Bayesian inference without strong assumptions about the functional form of the unknown distribution? Retrieved 2021-07-04.
- Haugh (2016). Mean-Variance Optimization and the CAPM.
- Idzorek (2004). A Step-by-Step Guide to the Black-Litterman Model.
- Johnstone and Lindley (2013). Mean-Variance and the Borch Paradox.
- Kroll, Levy and Markowitz. (1984). Mean-Variance Versus Direct Utility Maximisation.
- Levy and Sarnat (1969). A Note on Indifference Curves and Uncertainty.
- Markowitz (2014). Mean-Variance approximations to expected utility.
- Spiegel. Mean Variance Utility.