Dolphin on GPU

The purpose of this project is to find the best portfolio of size 20 given N different asset.

Portfolio

A portfolio is a combination of assets with respective shares. In order to ease our process, we will represent shares as a percentage of the portfolio. We will represent a portfolio P as follows :

- P is of size n if it has n different asset
- list of assets: $(a_i)_{i\in[[1,n]]}$
- list of weights: $(w_i)_{i\in[[1,n]]}$

To optimize a portfolio, we have to find the best combination of weights and assets. Suppose we have N assets and we want to create the best portfolio of size n. It can be represented by the following function:

$$F : (a_i)_{i\in[[1,N]]} \mapsto \begin{pmatrix} a_i \\ w_i \end{pmatrix}_{i\in[[1,n]]}$$

Then again, this function is supposed to give us the optimal list of assets ande their respective weights. We don't know yet what optimal means and how to compute it. Here comes the metrics.

Metrics

In order to compare portfolios, we need to pass by some metrics. These metrics are defined for one asset and can be generalized to a portfolio. In this section we will define the following metrics:

- Return (Rendement en français)
- Volatility (Volatilité en français)
- Sharpe

Return -- Rendement (Fr)

Return of an asset:

The Return value of an asset is an evaluation of the performance of that asset on a certain period. Let a be an asset, and V_t be the Value of a. The return value R of a on the period $T = [t_{i-1}, t_i]$

Let a be an asset, and V_i be the value of a. The return value R of a on the period $T = [v_{i-1}, v_i]$ is defined as:

$$R(a,T) = \frac{V_i - V_{i-1} + Cash\ flow\ received}{V_{i-1}}$$

Return of a portfolio:
The Return value of a portfolio is an evaluation of the performance of that portfolio on a certain period. It is actually the weighted average of its assets. Let P a portfolio of size n. Let $(a_i)_{i\in[[1,n]]}$ the list of assets in P and their respective weights $(w_i)_{i\in[[1,n]]}$. The return value R of P is defined as:

$$R(P,T) = \sum_{i=1}^{n} w_i R(a_i,T)$$

Volatility -- Volatilité (Fr)

Volatility of an asset:
volatility σ is the degree of variation of a trading price series over time as measured by the standard deviation of logarithmic returns *(Wikipedia)*. In *JUMP*'s slides, it is the standard deviation of the value. We will follow this definition since we are using their API. Let a an asset, σ_a its volatility and V_a its Variance on a period of size m. Let $\overline{a}$ the mean value of a on that period and a_i the value of a at an instant i. The Volatility of a is defined as:

$$\sigma_a = \sqrt{V_a}$$
$$\sigma_a = \sqrt{\frac{\sum_{i=1}^{m}(a_i - \overline{a})^2}{n}}$$

Volatility of a portfolio:
The volatility of a portfolio is standard deviation of the value (or return) of that portfolio. Since any portfolio is a weighted sum of assets and we can compute the variance-covariance matrix of all assets, it is easy to compute the volatility of a portfolio.
Let P a porfolio of size n sush as :

- $(a_i)_{i\in[[1,n]]}$ is the list assets composing P
- $(w_i)_{i\in[[1,n]]}$ is the list of respective weights
- The values of the assets are taken over a period T of size m

- $a_{i,j}$ is the value of asset a_i at the instant j in T
- $\overline{a}_i$ is the mean value of the asset a_i over T
- $Cov(a_i, a_j)$ the covariance between assets a_i and a_j over T

The mean and covariance over T are defined as:

- $\overline{a}_i = \frac{1}{m} \sum_{j=1}^{m} a_{i,j}$
- $Cov(a_i, a_j) = \frac{1}{m} \sum_{k=1}^{m} (a_{i,k} - \overline{a}_i)(a_{j,k} - \overline{a}_j)$

Let p the value of P, we have $p = \sum_{i=1}^{n} w_i a_i$. The volatility σ_P of the portfolio is simply the standard deviation of p, there for $\sigma_P = \sqrt{Var(p)}$ with $Var(p)$ the variance of p over the period T. The volatility of a the portfolio is defined as:

$$\sigma_P = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} w_i w_j Cov(a_i, a_j)}$$

Sharpe ratio

In finance, the Sharpe ratio is a way to examine the performance of an investment by adjusting for its risk. The ratio measures the excess return per unit of deviation in an investment asset, typically referred to as risk.

Let a an asset, R_a the asset return and $Var(a)$ the asset variance. Let b a benchmark asset, R_b the benchmark return and $Var(b)$ the benchmark variance. The sharpe of a is defined as:

$$S_a = \frac{E[R_a - R_b]}{\sqrt{Var(a) - Var(b)}}$$

If we take the asset benchmark as R_b as the *constant* risk-free return, noted R_f, we have $Var(R_f) = 0$ since R_f is a constant. The volatility of a is $\sigma_a = \sqrt{Var(R_a)}$
the sharpe becomes:

$$S_a = \frac{E[R_a - R_f]}{\sigma_a}$$

optimization

In order to optimize a portfolio P of size n (with already specified assets), we need to find the weights $(w_i)_{i \in [[1,n]]}$ that maximizes $R(P)$ and minimizes σ_P. We can represent this optimization problem as a quadratic optimization problem:

$$\begin{aligned} \text{maximize} \quad & w^T R - \gamma w^T \Sigma w \\ \text{subject to} \quad & \begin{cases} w^T 1 & = 1 \\ w & \geq 0 \end{cases} \end{aligned}$$

Where w is the vector of weights $(w_i)_{i\in[[1,n]]}$, R is the vector of asset returns and Σ is the covariance-variance matrix of the assets in P. The risk aversion is represented by the parameter $\gamma > 0$.

Implementation

At first, we may want to directly solve the convex optimization with all assets and try to parallelize this process. However, we have to find an optimal portfolio with a fixed size $n = 20$ using N assets such as $n < N$. In term of convex optimization, such a restriction is hardly achievable.

A simple way to recover from this issue and to make use of the power of a GPU is to find multiple different optimal portfolios of size n and take the best. The number of different portfolios we can create from N assets is $\binom{N}{20}$. It is in our best interest to launch every computation on a new thread.

The data provided by the API is not clean, some information is missing and some is incomplete. We therefore had to fetch and complete it. Also, since we may want to parallelize metrics computation in order to optimize more the process, we chose to compute every ratio ourselves. That is why we started our work by trying to figure out what formulas are used by the API to compute needed metrics (multiple ways exists). This step has been difficult and some formulas are still missing.

Once we figured out what are the steps that leads to finding the optimal portfolio and had fetched and cleaned the data, we started the implementation in c++.

At first comes the parser. it takes two arguments, a start and an end date. Those dates allow us to choose the period of time in which we want to work. If any information during this period of time is missing, we simply ignore the asset. The parser returns a vector of assets containing all data for each valid asset.

Next step

We now have to create different portfolios randomly and call our optimizer on each one of them. The main challenge here is to find a smart way to choose theses portfolios not just take all combinations.