Portfolio Re-allocation with Hidden Shocks

Background and Assumptions

Given N assets $S_1 \ldots S_N$ such that

$$dS_1 = S_1\mu_1 dt + S_1\sigma_{11}dW_1 + \cdots + S_1\sigma_{1M}dW_M$$

$$\ldots \tag{1}$$

$$dS_N = S_N\mu_N dt + S_N\sigma_{N1}dW_1 + \cdots + S_N\sigma_{NM}dW_M$$

Where $dW_1 \ldots dW_M$ are M Brownian motions (can be either dependent or independent)

Assuming we want to form a portfolio with those N assets + a risk-free asset, to tell the story in matrix notation, we have:

$$\begin{pmatrix} \mu_1 - r & \mu_2 - r & \cdots & \mu_N - r \\ \sigma_{11} & \sigma_{21} & \ddots & \vdots \\ \ldots & \ldots & \square & \square \\ \sigma_{1M} & \sigma_{2M} & \cdots & \sigma_{NM} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \ldots \\ u_N \end{pmatrix} = \begin{pmatrix} \beta \\ 0 \\ \ldots \\ 0 \end{pmatrix}$$

Where $u_1 + \cdots + u_N = 1 - u_B$; $\beta > 0$, and r is the risk-free rate; u_B is the risk-free asset allocation.

By no-arbitrage theory, the left-hand-side M+1 by N matrix needs to be singular to prevent having a solution, therefore, each asset's risk-premium can be written as a linear combination of its diffusion terms; namely:

$$\mu_i - r = \sum_{j=1}^{M} \sigma_{ij}\lambda_j \, ; \lambda_j > 0 \tag{2}$$

By following the abovementioned relationship should be the best way to define each asset's drift and diffusion terms.

Trading strategy and its Ito's Lemma derivation

We define our claim to be:

$$F = \mathrm{c} \cdot \{\gamma_1 \log(S_1) + \cdots + \gamma_N \log(S_N) + (1 - \gamma_1 - \cdots - \gamma_N)\log(B)\}$$

$$where\ \gamma_B = 1 - \gamma_1 - \cdots - \gamma_N \tag{3}$$

and c is a positive number and $\gamma_i; i = 1, \ldots, N$ are the weights on the assets. To simplify the model, I assume M=2 with two independent diffusion terms for all assets such that $dW_1 \cdot dW_2 = \rho \cdot dt$; hence, using Ito's lemma:

$$
\begin{aligned}
dF &= F(S_1 + dS_1, \dots, S_N + dS_N, B + dB, t + dt) - F(S_1, \dots, S_N, B, t) \\
&= \frac{\partial F}{\partial t}dt + \frac{\partial F}{\partial S_1}dS_1 + \cdots + \frac{\partial F}{\partial S_N}dS_N + \frac{\partial F}{\partial B}dB + \frac{1}{2}\frac{\partial^2 F}{\partial S_1^2}dS_1^2 + \cdots + \frac{1}{2}\frac{\partial^2 F}{\partial S_N^2}dS_N^2 \\
&= c\left\{\frac{\gamma_1}{S_1}(S_1\mu_1 dt + S_1\sigma_{11}dW_1 + S_1\sigma_{12}dW_2) + \cdots\right. \\
&+ \frac{\gamma_N}{S_N}(S_N\mu_N dt + S_N\sigma_{N1}dW_1 + S_1\sigma_{N2}dW_2) + \frac{\gamma_B}{B}rBdt \\
&+ \frac{1}{2}\left(-\frac{\gamma_1}{S_1^2}\right)(S_1^2\sigma_{11}^2 dt + S_1^2\sigma_{12}^2 dt + 2S_1^2\sigma_{11}\sigma_{12}\rho dt) + \cdots \\
&+ \left.\frac{1}{2}\left(-\frac{\gamma_N}{S_N^2}\right)(S_N^2\sigma_{N1}^2 dt + S_N^2\sigma_{N2}^2 dt + 2S_N^2\sigma_{N1}\sigma_{N2}\rho dt)\right\} \\
&= c\left\{\left[\gamma_1\mu_1 + \cdots + \gamma_N\mu_N + \gamma_B r\right.\right. \\
&- \frac{1}{2}(\gamma_1\sigma_{11}^2 + \gamma_1\sigma_{12}^2 + \cdots + \gamma_N\sigma_{N1}^2 + \gamma_N\sigma_{N2}^2 + 2\gamma_1\rho\sigma_{11}\sigma_{12} + \cdots \\
&+ \left.\left. 2\gamma_N\rho\sigma_{N1}\sigma_{N2})\right]dt + (\gamma_1\sigma_{11} + \cdots + \gamma_N\sigma_{N1})dW_1 + (\gamma_1\sigma_{12} + \cdots + \gamma_N\sigma_{N2})dW_2\right\}
\end{aligned}
\tag{4}
$$

Therefore, we have a drift of $c\sum_{i=1}^{N}\gamma_i\mu_i + \gamma_B r - \frac{1}{2}(\gamma_i\sigma_{i1}^2 + \gamma_i\sigma_{i2}^2 + 2\gamma_i\rho\sigma_{i1}\sigma_{i2})$ and two diffusions of $c\sum_{i=1}^{N}\gamma_i\sigma_{i1}$ and $c\sum_{i=1}^{N}\gamma_i\sigma_{i2}$

Under no market shocks, the portfolio is equally weighted with no bond allocation; namely, $\gamma_i = \frac{1}{N}; i = 1, \dots, N$ and $\gamma_B = 0$, however, when there are shocks to the market, or in my case, a idiosyncratic jump diffusion is added to each asset, we need to recalculate the weights on each asset, and potentially, to add bond allocation to the portfolio to match our targeted return and volatility. Nevertheless, market shocks are seen as forthcoming, but directions and magnitudes are unknown information, or in other words, a hidden function of assets' drifts and volatilities under normal market conditions. Therefore, I want to develop a framework that can choose the best portfolio with the best weights when jump diffusions are being considered.

To define our claim $\tilde{F}$ under shock conditions, we write:

$$
\begin{gathered}
\tilde{F} = \mathrm{c}\cdot\{\gamma_1\log(\tilde{S}_1) + \cdots + \gamma_N\log(\tilde{S}_N) + (1 - \gamma_1 - \cdots - \gamma_N)\log(B)\} \\
where\ d\tilde{S}_i = \tilde{S}_i\mu_i dt + \tilde{S}_i\sigma_{i1}dW_1 + \cdots + \tilde{S}_i\sigma_{iM}dW_M + J_i dP \\
st. J_i = f(\mu_i, \sigma_{ij}); j = 1 \dots M \\
and\ \gamma_B = 1 - \gamma_1 - \cdots - \gamma_N; M = 2
\end{gathered}
\tag{4}
$$

Using Ito's lemma, we have:

$$d\tilde{F} = c\left\{\left[\gamma_1\mu_1 + \cdots + \gamma_N\mu_N + \gamma_B r - \frac{1}{2}(\gamma_1\sigma_{11}^2 + \gamma_1\sigma_{12}^2 + \cdots + \gamma_N\sigma_{N1}^2 + \gamma_N\sigma_{N2}^2 + 2\gamma_1\rho\sigma_{11}\sigma_{12} + \cdots + 2\gamma_N\rho\sigma_{N1}\sigma_{N2})\right]dt + (\gamma_1\sigma_{11} + \cdots + \gamma_N\sigma_{N1})dW_1 + (\gamma_1\sigma_{12} + \cdots + \gamma_N\sigma_{N2})dW_2 + (\gamma_1 J_1 + \cdots + \gamma_N J_N)dP\right\} \tag{5}$$

Model Implementation

Assuming we are given H number of assets and to choose N out of it to form a portfolio, we have $\frac{H!}{(H-N)!}$ choices of combinations. Each of those portfolios comprise assets with different theoretical returns μ_i and theoretical volatilities σ_{i1}^2 and σ_{i2}^2, and hidden functions of jump structures with respect to their μ_i, σ_{i1}^2 and σ_{i2}^2. However, at the time of re-allocating the weights to assets and bond, we lack the information of the hidden function of jumps $J_i = f(\mu_i, \sigma_{i1}, \sigma_{i2})$; therefore, we don't have closed-form solutions for asset allocations. As one example of model's X variables, it could be written in the following form when multiplying to its corresponding weights:

$$M = X \cdot \Gamma = \begin{pmatrix} \mu_1 & \mu_2 & \cdots & \mu_N \\ \sigma_{11} & \sigma_{21} & \dots & \sigma_{N1} \\ \sigma_{12} & \sigma_{22} & \cdots & \sigma_{N2} \end{pmatrix} \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \cdots \\ \gamma_N \end{pmatrix} = \begin{pmatrix} \gamma_1\mu_1 & \gamma_2\mu_2 & \cdots & \gamma_N\mu_N \\ \gamma_1\sigma_{11} & \gamma_2\sigma_{21} & \dots & \gamma_N\sigma_{N1} \\ \gamma_1\sigma_{12} & \gamma_2\sigma_{22} & \cdots & \gamma_N\sigma_{N2} \end{pmatrix} \tag{6}$$

Our first step is to randomly generate K sample of Γ by utilizing PyTorch's Convolutional Neural Network (CNN) function *torch.nn.conv2d(in channels, out channels, kernel size)*; in our case, we have *in channels* = 1, *out channels* = K, and *kernel size* = (1, N). As a result, each *out channel* produces a set of randomly generated weights and bias, and after the multiplication of X and all those weights, we obtain K number of weighted sums of μ_i, σ_{i1}^2 and σ_{i2}^2: $\sum_{i=1}^{N}\gamma_{ik}\mu_i$, $\sum_{i=1}^{N}\gamma_{ik}\sigma_{i1}$ and $\sum_{i=1}^{N}\gamma_{ik}\sigma_{i2}$ where k=1,..., K

The second step is to apply logistic regression to K number of weighted sums of μ_i, σ_{i1}^2 and σ_{i2}^2 in the following way:

$$\Pr(Y = 1) = \frac{\exp(A)}{1 + \exp(A)}$$

$$\text{where A} = (\beta_0 + \beta_1\sum_{i=1}^{N}\gamma_{ik}\mu_i + \beta_2\sum_{i=1}^{N}\gamma_{ik}\sigma_{i1} + \beta_3\sum_{i=1}^{N}\gamma_{ik}\sigma_{i2})$$

(7)

The formation of A could be done by using *m=torch.nn.Linear(in features, out features, bias=True)* by setting *in features* = 3, *out features* = 1. By turning bias to be true, we will get one intercept β_0 (*m.bias*) and three coefficients $\beta_1, \beta_2, \beta_3$ (*m.weight*) that correspond to one weighted drift and two weighted diffusions; calling this linear function J times yields J sets of $(\beta_0, \beta_1, \beta_2, \beta_3)$ and each of those parameter sets calculates A, which works as a score of the weighted drift and diffusions. Since we have K number of out channels, so for each X variable, we have K by J times of scores A. Plug A into the probability function $\Pr(Y = 1)$ as showing above is as simple as calling *torch.nn.functional.sigmoid(x)* from the PyTorch library, which results in K by J number of probabilities $\Pr(Y = 1)$.

TBC...