Modeling Stock Price Trends

Shan Zhong, Yizhou Cai, University of South Carolina

2022

Project: Motivation

- Besides stock price predicting, other aspects such as uncertainty estimating and trading decision making are also important analytical aspects of stock trading.
- To check how sensitive the position is, correlations between different stocks need to be considered as well.
- Professional investors usually manage their position in a portfolio of stocks that can reduce the variance of potential return and thus hedge against potential losses.

Project: Data

Data: scraped list of over 500 current and past S&P 500 companies for the period from January 1, 2000 to December 31, 2021. Original data contains the daily price information of Open, Close, High, Low, Volume.

Table 1: A summary for the technical indicators we used in the model. These indicators are all calculated based on the daily price information.

Indicator Name	Quick Introduction	Window period
CCI	The Commodity Channel Index can help to identify price reversals, price extremes and trend strength.	20 days
MACDH	The Moving Average Convergence Divergence Histogram is the difference between MACD and the MACD signals.	12, 26, and 9 days
RSI	The Relative Strength Index, which measures the average of recent upward movement versus the upward and downward movements combined, in a percentage form.	14 days
KDJ	The Stochastic Oscillator measures where the close is relative to the low and high.	14, 3, and 3 days
WR	The Williams %R index is another index for buy signals by measuring the current price in relation to the past N periods.	14 days
ATR	The Average True Range provides an indicator for the volatility of price. We converted ATR into percentages.	14 days
CMF	The Chaikin Money Flow provides an indicator related to the trading volume.	20 days

Project: S&P 500 Stock Example

We also chose the difference in the weekly average log adjusted closing price for our study, with the relation between the weekly average log adjusted closing price $log(p_t)$ and the log scale return R_t defined as:

$$R_t = log(p_t) - log(p_{t-1})$$

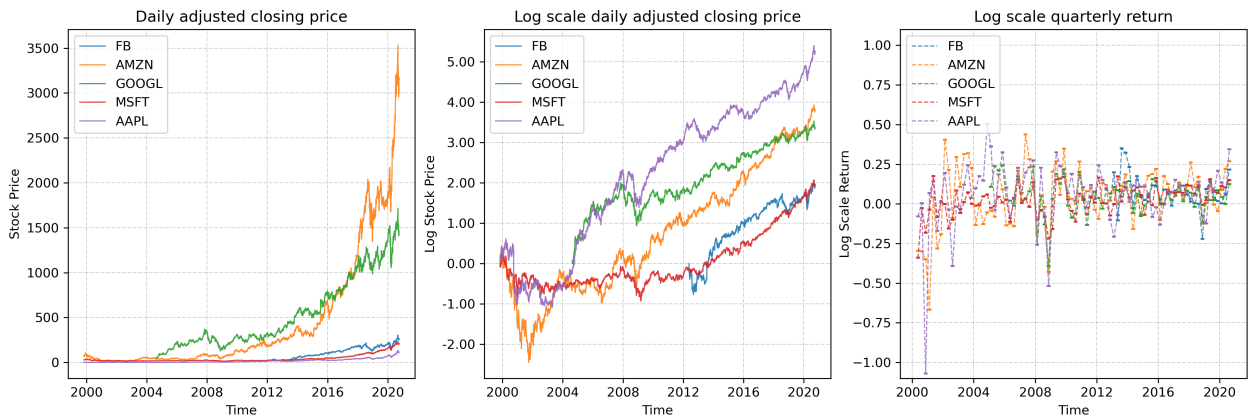


Figure 1: Cleaned daily closing price, log scale price with starting point shifted to 0, and the log scale quarterly return which is defined as the difference of quarterly average log adjusted closing price, for five selected stocks.

Project: Brownian Motion Model

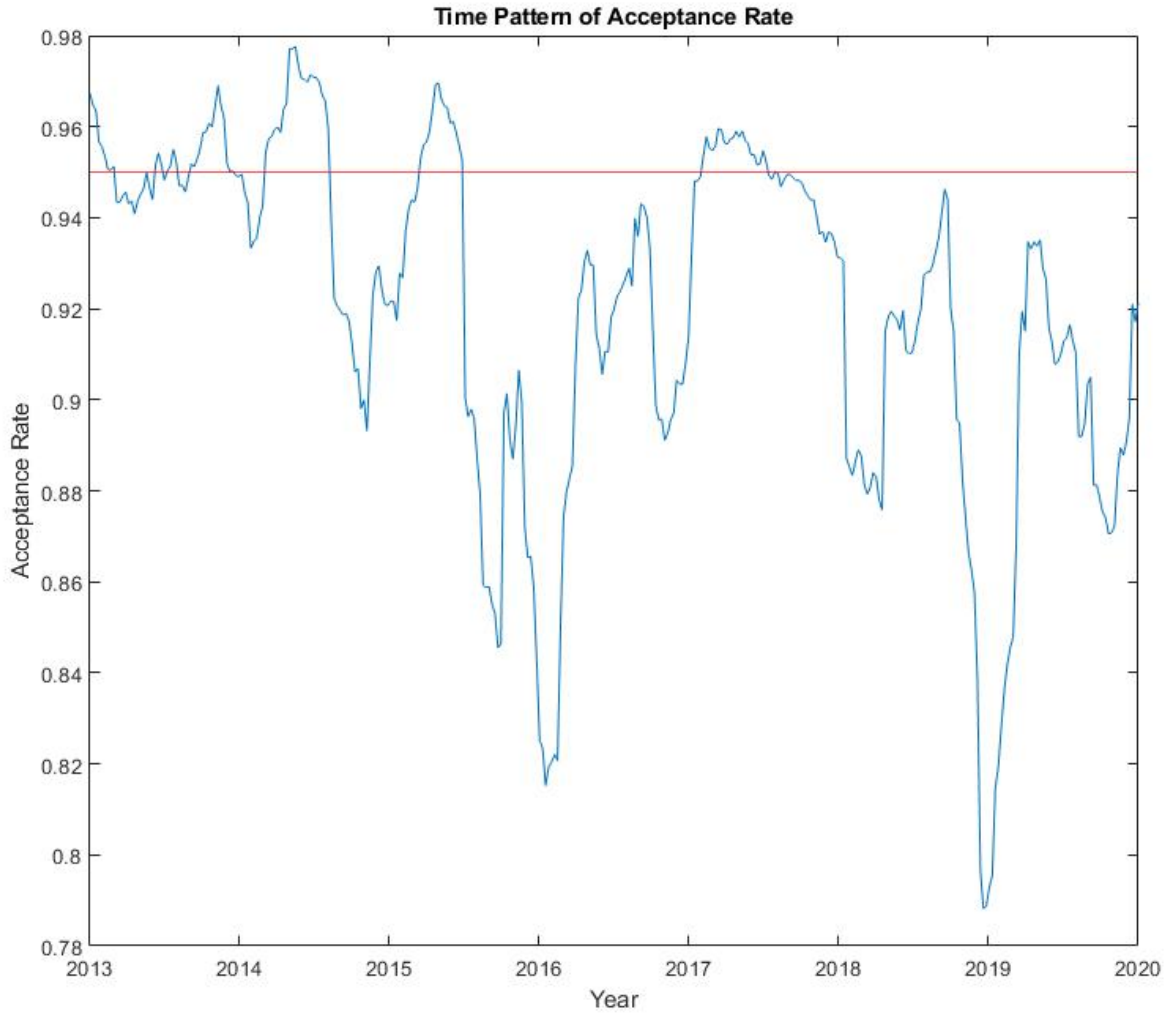


Figure 2: The coverage rate for the 95% prediction interval for the brownian motion methods.

Project: Preprocessing and Evaluation

- The model updates at the end of each year with all previous data.
- All predictor variables were transformed to standard Gaussian distributions via Yeo-Johnson transformations.
- Data cleaning by filling missing values with 0 and capping the absolute values of these preprocessed predictors at 4.5.
- Use Log likihood to measure the performance of models.

Project 1: Model Under RL framework

The state s were defined as the price and indicator information of the stocks we are considering, with state space $\mathcal{S}$ bounded by the predefined range $[-4.5, 4.5] \subset \mathbb{R}$ for the preprocessed technical and price informations of the last three period $t-1, t-2$, and $t-3$:

$$
\begin{aligned}
\boldsymbol{s} = \big\{ & s_{stock_1} : \mathbf{X}^{stock_1}_{t-1,t-2,t-3} = [r^{stock_1}_{t-1,t-2,t-3}, \mathrm{CCI}^{stock_1}_{t-1,t-2,t-3}, \ldots, \mathrm{CMF}^{stock_1}_{t-1,t-2,t-3}] \in [-4.5, 4.5]^p, \\
& s_{stock_2} : \mathbf{X}^{stock_2}_{t-1,t-2,t-3} = [r^{stock_2}_{t-1,t-2,t-3}, \mathrm{CCI}^{stock_2}_{t-1,t-2,t-3}, \ldots, \mathrm{CMF}^{stock_2}_{t-1,t-2,t-3}] \in [-4.5, 4.5]^p, \\
& \ldots \\
& s_{stock_5} : \mathbf{X}^{stock_3}_{t-1,t-2,t-3} = [r^{stock_5}_{t-1,t-2,t-3}, \mathrm{CCI}^{stock_5}_{t-1,t-2,t-3}, \ldots, \mathrm{CMF}^{stock_5}_{t-1,t-2,t-3}] \in [-4.5, 4.5]^p \big\}
\end{aligned}
$$

Suppose the environment knows everything and it determines the trend movement for stocks with it's actions:

$$
\boldsymbol{a} = [r_t^{stock_1}, r_t^{stock_2}, r_t^{stock_3}, r_t^{stock_4}, r_t^{stock_5}] \in \mathcal{A} = \mathbb{R}^5
$$

Project 1: Model Under RL framework

We imitate the behavior of the environmental agent by learning a stochastic policy $\pi_\theta(a|s)$ based on past history of demonstrations.

$$\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}) \sim \mathcal{N}(\boldsymbol{\phi}(\boldsymbol{s})^\intercal \boldsymbol{\theta}, \Sigma)$$

Where $\boldsymbol{\phi}(\boldsymbol{s})$ are basis functions learned via neural network, and Σ modeled by a stationary covariance kernel with a squared exponential (SE) kernel function:

$$K_{SE}(a_{stock_i}, a_{stock_j}) = \sigma_{stock_i}\sigma_{stock_j} exp(-\frac{\|\mathbf{X}_{t-1}^{stock_i} - \mathbf{X}_{t-1}^{stock_j}\|_2^2}{2\ell^2})$$

where $\log\sigma$ are learnt by neural network, and ℓ is the hyperparameter predefined.

For any d-dimensional multivariate normal distribution $X \sim N_d(\mu, \Sigma)$ where $\mu = (\mu_1, \ldots, \mu_d)^T$ and $\Sigma_{jk} = cov(X_j, X_k)\ \ j, k = 1, \ldots, d$. Define $Z = \mathbf{a}^T X = \sum_{j=1}^{d} a_j X_j$, the characteristic function is given by:

$$\varphi_Z(t) = E\left[\exp(itZ)\right] = E\left[\exp(it\mathbf{a}^T X)\right] = \varphi_X(t\mathbf{a})$$

$$= \exp\left(it\sum_{j=1}^{d} a_j\mu_j - \frac{1}{2}t^2\sum_{j=1}^{d}\sum_{k=1}^{d} a_j a_k \Sigma_{jk}\right)$$

Which is normal with:

$$\mu_Z = \sum_{j=1}^{d} a_j\mu_j \ , \ \sigma_Z^2 = \sum_{j=1}^{d}\sum_{k=1}^{d} a_j a_k \Sigma_{jk}$$

Our objective function is log-likelihood:

$$Likihood = \mathbb{P}[r_{t,true}^{stock_1}, \ldots, r_{t,true}^{stock_5} | \mathcal{N}(\boldsymbol{\phi}(\boldsymbol{s})^\intercal \boldsymbol{\theta}, \Sigma)]$$

Our basis function (for simplicity did not apply LSTM structure):

```
# Define model
class my_basis_function(nn.Module):
    def __init__(self):
        super(my_basis_function, self).__init__()
        self.flatten = nn.Flatten()
        self.linear_relu_stack = nn.Sequential(
            nn.Linear(5*8, 128),
            nn.ReLU(),
            nn.Linear(128, 128),
            nn.ReLU(),
            nn.Linear(128, 10)
        )

    def forward(self, x):
        flatten_x = self.flatten(x)
        mean_and_logstd = self.linear_relu_stack(flatten_x)
        return mean_and_logstd
```

Our policy, calcuated covariance matrix by *se_kernel_fast* provided by professor Bai in HW3:

```
class my_policy(nn.Module):
    def __init__(self, basis_function):
        nn.Module.__init__(self)
        self.basis_function = basis_function
        self.basis_function.to(device)

    def action_distribution(self, x):
        r_and_logstd = self.basis_function(x)
        r_and_logstd = torch.split(r_and_logstd, 5, dim = 1)
        r = r_and_logstd[0]
        std = torch.exp(r_and_logstd[1])

        cov_matrix = []
        for i in range(x.shape[0]):
            cov_matrix.append( torch.matmul(torch.matmul( torch.diag(std[i]),
                               se_kernel_fast(x[i],x[i]) ),torch.diag(std[i]) )  )

        distribution = ptd.multivariate_normal.MultivariateNormal(loc = r,
                                  covariance_matrix = torch.stack(cov_matrix) )
        return distribution
```

Training (optimizing the log-likelihood, gradient calculated by pytorch):

```
def train(env,policy,optimizer):
    all_log_likihood = []
    policy.train()
    # train with 1 million samples
    for batch in range(1000):
        X, y = sample_from_environment(env)

        # Compute prediction error
        distribution = policy.action_distribution(X.float())

        negative_log_likihood = - torch.mean( distribution.log_prob(y.flatten(start_dim=1)) )

        optimizer.zero_grad()
        negative_log_likihood.backward()
        optimizer.step()
```

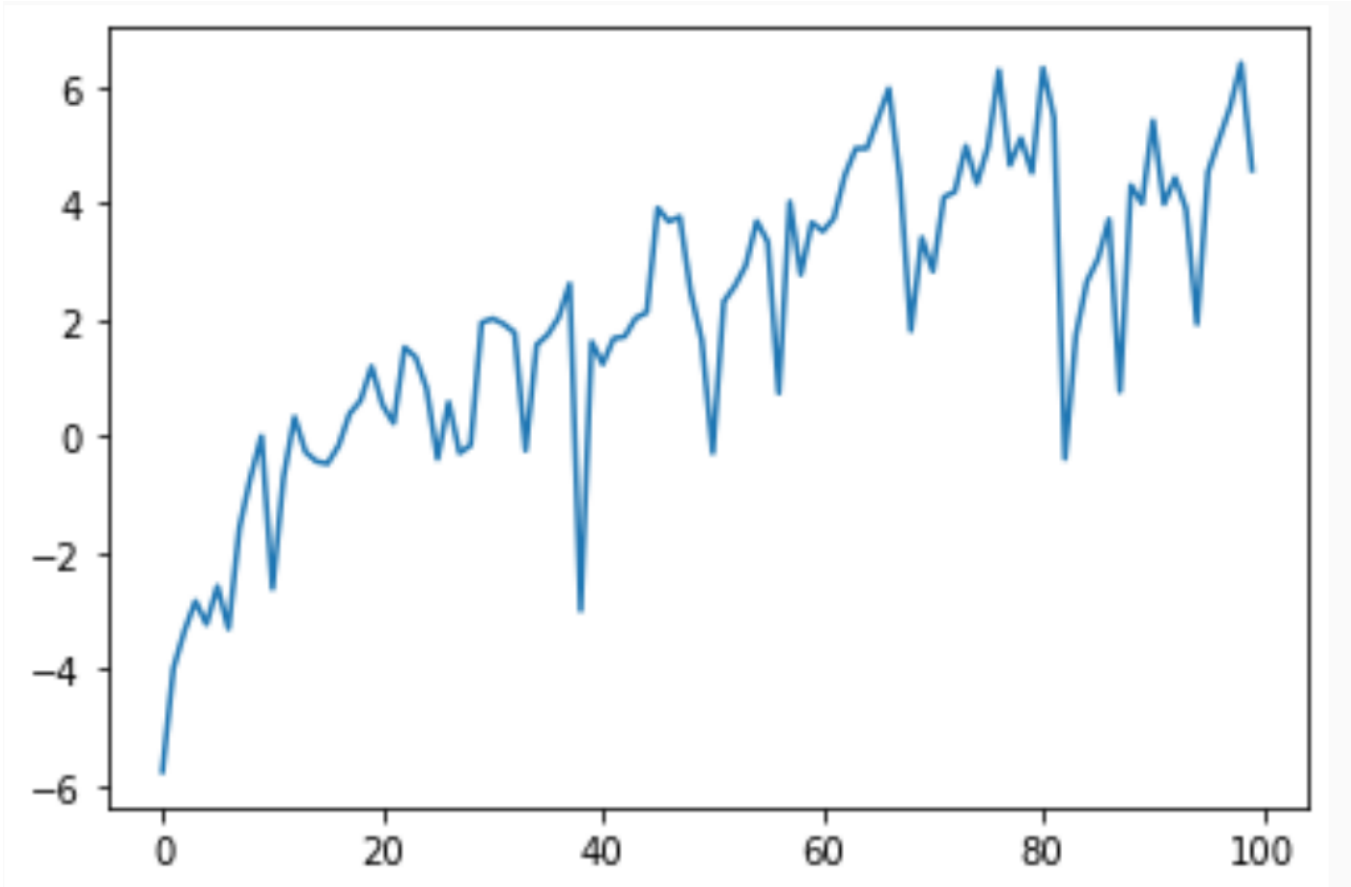
Testing (Coverage rate of 95% prediction interval for portfolio of each 20% with 5 stocks)

```
def test(env,policy):
    policy.eval()
    result = []
    for batch in range(10):
        X, y = sample_from_environment(env,training = False)
        distribution = policy.action_distribution(X.float())
        distribution_sum = ptd.Normal( loc = torch.mean(distribution.mean,dim=1),
         scale = torch.sqrt(torch.mean(distribution.covariance_matrix,dim = [1,2])) )

        # 95% prediction inerval
        lower = distribution_sum.mean - 1.96 * distribution_sum.stddev
        upper = distribution_sum.mean + 1.96 * distribution_sum.stddev
        _within_interval = torch.logical_and(
            ( lower <= torch.mean(y.flatten(start_dim=1),dim=1)),
             ( torch.mean(y.flatten(start_dim=1),dim=1) <= upper ))

        result.append(np.mean(_within_interval.cpu().data.numpy()))
    return result
```

log-likelihood during training process with 100 epochs of 1 million samples:



Some proof:

$$\begin{aligned}
\nabla_\theta \log \pi_\theta(a|s) =& \nabla_\theta \log \left\{ \det(2\pi\Sigma)^{\frac{1}{2}} \exp[-\frac{1}{2}(a-\boldsymbol{\phi}(s)^\intercal\boldsymbol{\theta})^\intercal\Sigma^{-1}(a-\boldsymbol{\phi}(s)^\intercal\boldsymbol{\theta})] \right\} \\
=& \nabla_\theta \left\{ \log(\det(2\pi\Sigma)^{\frac{1}{2}}) + \log(\exp[-\frac{1}{2}(a-\boldsymbol{\phi}(s)^\intercal\boldsymbol{\theta})^\intercal\Sigma^{-1}(a-\boldsymbol{\phi}(s)^\intercal\boldsymbol{\theta})]) \right\} \\
=& \nabla_\theta(-\frac{1}{2}(a-\boldsymbol{\phi}(s)^\intercal\boldsymbol{\theta})^\intercal\Sigma^{-1}(a-\boldsymbol{\phi}(s)^\intercal\boldsymbol{\theta}))
\end{aligned}$$

Let $\boldsymbol{u} = (a-\boldsymbol{\phi}(s)^\intercal\boldsymbol{\theta})$

$$\begin{aligned}
\nabla_\theta \log \pi_\theta(a|s) =& -\frac{1}{2}\nabla_\theta(a-\boldsymbol{\phi}(s)^\intercal\boldsymbol{\theta})\nabla_{\boldsymbol{u}}\boldsymbol{u}^\intercal\Sigma^{-1}\boldsymbol{u} \\
=& -\frac{1}{2}(-\boldsymbol{\phi}(s))[\boldsymbol{u}^\intercal\frac{\partial\Sigma^{-1}\boldsymbol{u}}{\partial\boldsymbol{u}} + (\Sigma^{-1}\boldsymbol{u})^\intercal\frac{\partial\boldsymbol{u}}{\partial\boldsymbol{u}}] \\
=& \frac{1}{2}\boldsymbol{\phi}(s)[\boldsymbol{u}^\intercal(\Sigma^{-1}+(\Sigma^{-1})^\intercal)] \\
=& \frac{1}{2}\boldsymbol{\phi}(s)(a-\boldsymbol{\phi}(s)^\intercal\boldsymbol{\theta})^\intercal[\Sigma^{-1}+(\Sigma^{-1})^\intercal]
\end{aligned}$$