Generating Synergistic Formulaic Alpha Collections via Reinforcement Learning

Shuo Yu[*] [†]
Hongyan Xue[*] [†]
Xiang Ao[†] [‡]
Institute of Computing Technology,
Chinese Academy of Sciences
University of Chinese Academy of Sciences
Beijing, China
yushuo19b@ict.ac.cn
xuehongyan21b@ict.ac.cn
aoxiang@ict.ac.cn

Feiyang Pan
Jia He
Dandan Tu
Huawei EI Innovation Lab
China
pfy824@gmail.com
hejia0149@gmail.com
tudandan@huawei.com

Qing He[†] [‡]
Institute of Computing Technology,
Chinese Academy of Sciences
University of Chinese Academy of Sciences
Beijing, China
heqing@ict.ac.cn

ABSTRACT

In the field of quantitative trading, it is common practice to transform raw historical stock data into indicative signals for the market trend. Such signals are called alpha factors. Alphas in formula forms are more interpretable and thus favored by practitioners concerned with risk. In practice, a set of formulaic alphas is often used together for better modeling precision, so we need to find synergistic formulaic alpha sets that work well together. However, most traditional alpha generators mine alphas one by one separately, overlooking the fact that the alphas would be combined later. In this paper, we propose a new alpha-mining framework that prioritizes mining a synergistic set of alphas, i.e., it directly uses the performance of the downstream combination model to optimize the alpha generator. Our framework also leverages the strong exploratory capabilities of reinforcement learning (RL) to better explore the vast search space of formulaic alphas. The contribution to the combination models' performance is assigned to be the return used in the RL process. This return drives the alpha generator to find better alphas that improve upon the current set. Experimental evaluations on real-world stock market data demonstrate both the effectiveness and the efficiency of our framework for stock trend forecasting. The investment simulation results show that our framework is able to achieve higher returns compared to previous approaches.

CCS CONCEPTS

• **Computing methodologies** → **Reinforcement learning**; *Search methodologies*; • **Applied computing** → *Economics*.

KEYWORDS

Computational Finance, Stock Trend Forecasting, Reinforcement Learning

ACM Reference Format:
Shuo Yu, Hongyan Xue, Xiang Ao, Feiyang Pan, Jia He, Dandan Tu, and Qing He. 2023. Generating Synergistic Formulaic Alpha Collections via Reinforcement Learning. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3580305.3599831

1 INTRODUCTION

Currently, it is almost a standard paradigm to transform raw historical stock data into indicative signals for the market trend in the field of quantitative trading [14]. These signal patterns are called *alpha factors*, or alphas in short [19]. Discovering alphas with high returns has been a trendy topic among investors and researchers due to the close relatedness between alphas and investment revenues.

The prevailing methods of discovering alphas can be in general divided into two groups, namely machine learning-based and formulaic alphas. Most recent research has focused on the former ones. These more sophisticated alphas are often obtained via deep learning models, e.g., using sequential models like LSTM [5], or more complex ones integrating non-standard data like HIST [23] and REST [24], etc. On the other end of the spectrum, we have the alphas that can be represented in simple formula forms. Such formulaic alphas are traditionally constructed by human experts using their domain knowledge and experience, often expressing clear economic principles. To name some, [7] demonstrates 101 alpha factors tested on the US stock market. Recently, research has also been conducted on frameworks that generate such formulaic alphas automatically [3, 9, 10, 27]. These approaches are able to find loads of new alphas rapidly without human supervision, while still maintaining relatively high interpretability compared to the sophisticated machine learning-alphas.

Despite the existing approaches achieving remarkable success, they still have disadvantages in different aspects. Machine learning-based alpha factors are inherently complex and sometimes require

[*] These authors contributed equally.
[†] Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS). Xiang Ao is also at Institute of Intelligent Computing Technology, Suzhou, China.
[‡] Corresponding authors.

KDD '23, August 6–10, 2023, Long Beach, CA, USA

ACM ISBN 979-8-4007-0103-0/23/08.
https://doi.org/10.1145/3580305.3599831

more advanced data other than the price/volume features. In addition, although they are often more expressive, they often suffer from relatively low interpretability. As a result, when the performance of these "black box" models unexpectedly deteriorates, it is hard for human experts to tune the models accordingly. These algorithms are thus not favored under some circumstances due to concerns about risks. On the other hand, while formulaic alphas are more interpretable, previous research on this matter often focused on finding a single alpha factor that predicts well on its own. Nonetheless, it is often impossible to describe a complex and chaotic system such as the stock market with simple rules that human researchers can comprehend. As a compromise, a set of these alphas are oftentimes used together in practice, instead of using them individually. However, when multiple of these independently mined formula alphas are combined, the final prediction performance may not improve much because not much consideration is put into the synergistic effect between factors (see Section 4.2.2 for detail). In addition, these alphas are often simple in their forms, and their underlying mechanisms are often quite understandable. Once they are released to the public and become well-known among practitioners, their performance may deteriorate rapidly [7].

Therefore, the question we are facing is: *Can we find a way to automatically discover interpretable alpha factors that work well with downstream predictive models, without suffering possible performance deterioration due to them being widely known to the general public?*

To solve the above challenge, we formulate a new research problem in this paper, which is to find *synergistic formulaic alpha factor sets*. Using raw stock price/volume data as the input, we aim to search for a set of formulaic alpha factors instead of individual ones. Recall that finding a single well-performing alpha on given data is already a hard problem to resolve since the search space of valid formulas is vast and hard to navigate. The search space for alpha expressions is often even larger than that of a typical symbolic regression problem [13].

The most intuitive approach to this problem is using genetic programming (GP), performing mutations on expression trees to generate new alphas[3, 9, 10, 27]. The utilization of GP for this task is of course not a serendipitous choice since GP methods generally excel at such problems with large search spaces. However, GP algorithms often scale poorly due to the complexity of maintaining and mutating a huge population [13]. Note that mining a set of synergistic alphas all at once is an even harder problem with a much larger search space, the scale of which makes most existing frameworks infeasible to solve.

Hence, previous works mostly tried to find ways to simplify the problem of alpha set mining, by mining alphas one by one and filtering out a subset of them with respect to some similarity metric. The mutual information coefficient (IC) between the pairs of alpha in the set is often employed as the similarity "metric" [3, 10, 27]. However, as we will demonstrate below, adding a new alpha that is of high mutual IC to the ones in an existing pool of alpha may still bring a non-negligible boost of performance to the combined result, and vice versa. This phenomenon still exists even when the combination model is set to be a simple linear regressor. Therefore, the traditional approach to determining whether a set of alpha could be synergistic does not line up with the expected outcome.

To tackle the challenge that GP methods could be inefficient at exploring the vast search space of formulaic alphas, our framework utilizes reinforcement learning (RL) for achieving better results in exploration. Combined with the strong expressiveness of deep neural networks, RL with its excellent exploratory ability plays a predominant role in numerous areas. To list a few examples, game playing [16], natural language processing [11], symbolic optimization [13], and portfolio management [22]. We implement a sequence generator with constraints to ensure valid formulaic alpha generation and employ a policy gradient-based algorithm to train the generator in the absence of a direct gradient. Since traditional mutual-IC filtering methods do not align well with the target of optimizing the combination model's performance, we propose to use directly the performance as the optimization objective of our alpha generator. Under this new optimization scheme, our generator is able to produce a synergistic set of alpha which fits the mine-and-combine procedure in a more suitable way. To evaluate our alpha-mining framework, we conduct extensive experiments over real-world stock data. Our experiment results demonstrate that the formulaic alpha sets generated by our framework perform better than those generated with previous approaches, shown both on the prediction metrics and investment simulations.

Our contributions can be summarized as follows.

- We propose a new optimization scheme that produces a set of alpha that suits downstream tasks better, regardless of what actual form the combination model takes.
- We introduce a new framework for searching formulaic alpha factors based on policy gradient algorithms, to utilize the strong exploratory power of reinforcement learning.
- We present a series of experimental results demonstrating the effectiveness of our proposed framework. Additional experiments and case studies are also conducted to illustrate why mutual IC-based filtering techniques that are previously commonly used may not work as expected when considering the combined performance of an alpha set.

2 PROBLEM FORMULATION

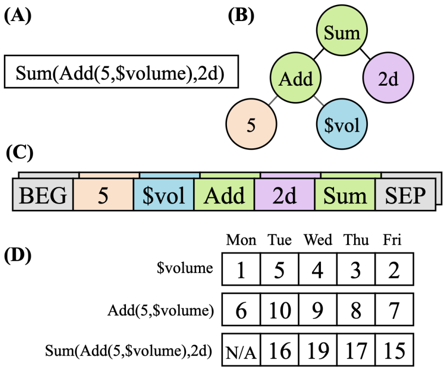
2.1 Alpha Factor

We consider a stock market with n stocks in a period of T trading days in total. On each trading day $t \in \{1, 2, \cdots, T\}$, each stock i corresponds to a feature vector $x_{ti} \in \mathbb{R}^{m\tau}$, comprised of m raw features such as opening/closing price in the recent τ days[1]. Finally, we define an *alpha factor* f as a function mapping feature vectors of all stocks on a trading day $X \in \mathbb{R}^{n \times m\tau}$ into *alpha values* $z = f(X) \in \mathbb{R}^n$. We will use the term "alpha" for both an alpha factor and its corresponding values in the following sections, where it is not ambiguous to do so.

2.2 Alpha Factor Mining

To measure the effectiveness of an alpha, we calculate the information coefficient (IC) between the true stock trend it predicts $y_t \in \mathbb{R}^n$ and the factor values $f(X_t)$. We denote the daily IC function as

[1] This "unrolling" of historical data introduces redundancy. Namely, feature vectors of a stock on consecutive trading days have overlapping sections. This notation is chosen for the convenience of demonstration.



	Mon	Tue	Wed	Thu	Fri
$volume	1	5	4	3	2
Add(5,$volume)	6	10	9	8	7
Sum(Add(5,$volume),2d)	N/A	16	19	17	15

Figure 1: (A) An example of a formulaic alpha. (B) Its equivalent expression tree. (C) Its reverse Polish notation (RPN). Note that *BEG* and *SEP* are sequence indicators later mentioned in our framework. (D). Step-by-step computation of this alpha on an example time series.

$\sigma : \mathbb{R}^n \times \mathbb{R}^n \to [-1, 1]$, which is defined as the Pearson's correlation coefficient:

$$\sigma(u_t, v_t) = \frac{\sum_{i=1}^{n}(u_{ti} - \bar{u}_t)(v_{ti} - \bar{v}_t)}{\sqrt{\sum_{i=1}^{n}(u_{ti} - \bar{u_t})^2 \sum_{i=1}^{n}(v_{ti} - \bar{v_t})^2}}. \tag{1}$$

Such value can be calculated on every trading day between an alpha and the prediction target. For convenience, we denote the IC values between two sets of vectors averaged over all trading days as $\bar{\sigma}(u, v) = \mathbb{E}_t\left[\sigma(u_t, v_t)\right]$.

We use the average IC between an alpha and the return to measure the effectiveness of an alpha factor on a stock trend series $y = \{y_1, y_2, \cdots, y_T\}$:

$$\bar{\sigma}_y(f) = \bar{\sigma}(f(X), y). \tag{2}$$

The output of a combination model can be seen as a "mega-alpha", mapping raw inputs into alpha values. Therefore, we denote the combination model as $c(X; \mathcal{F}, \theta)$, where $\mathcal{F} = \{f_1, f_2, \cdots, f_k\}$ is a set of alphas to combine, and θ denotes the parameters of the combination model. We would like the combination model to be optimal w.r.t. a given alpha set $\mathcal{F}$ on the training dataset, that is:

$$\begin{aligned} c^*(X; \mathcal{F}) &= c(X; \mathcal{F}, \theta^*), \quad \text{where} \\ \theta^* &= \underset{\theta}{\operatorname{argmax}}\, \bar{\sigma}_y(c(\cdot; \mathcal{F}, \theta)). \end{aligned} \tag{3}$$

Conclusively, the task of mining a set of alphas can be defined as the optimization problem $\operatorname{argmax}_{\mathcal{F}} c^*(\cdot; \mathcal{F})$.

2.3 Formulaic Alpha

Formulaic alphas are expressed as mathematical expressions, consisting of various operators and the raw input features mentioned before. Some examples of the operators are the elementary functions (like "+" and "log") operating on one-day data, called cross-section operators, and operators that require data from a series of days, called time-series operators (e.g. "Min(close, 5)" gives the lowest closing price of a stock in the recent 5 days). A list of all the operators used in our framework is given in Appendix A.

Such formulas can be naturally represented by an expression tree, with each non-leaf node representing an operator, and the children of a node representing the operands. To generate such an expression, our model represents the expression tree by its post-order traversal. The order of children is also defined by the traversal order. In other words, the model represents a formula as its reverse Polish notation (RPN). It is easy to see that such notation is unambiguous since the arities of the operators are all known constants. See Figure 1 for an example of a formulaic alpha expression together with its corresponding tree and RPN representations.

3 METHODOLOGY

As illustrated in Figure 2, our alpha-mining framework consists of two main components: 1) the *Alpha Combination Model*, which combines multiple formulaic alphas to achieve optimal performance in prediction, and 2) the *RL-based Alpha Generator*, which generates formulaic alphas in the form of a token sequence. The performance of the Alpha Combination Model is used as the reward signal to train the RL policy in the Alpha Generator using policy gradient-based algorithms, such as PPO [17]. Repeating this process, the generator is continuously trained to generate alphas that boost the combination model, thereby enhancing the overall predictive power.

3.1 Alpha Combination Model

Considering the interpretability of the combined "mega-alpha", the combination model itself should also be interpretable. In this paper, we use a linear model to combine the alphas.

The values evaluated from different alphas have drastically different scales, which might cause problems in the following optimization steps. To remedy this, we centralize and normalize the alpha values with their average and standard deviation. Since Pearson's correlation coefficient is invariant up to linear transformation, this transformation does not affect the performance of the alphas when they are considered separately. Formally, we introduce a normalization operator $\mathcal{N}$, which transforms a vector such that its elements have a mean of 0, and the vector has a length of 1:

$$[\mathcal{N}(u)]_i = \frac{u_i - \bar{u}}{\sqrt{\sum_{j=1}^{n}\left(u_j - \bar{u}\right)^2}}. \tag{4}$$

We will omit explicitly writing the $\mathcal{N}$ operator for simplicity. For the rest of this paper, we will assume that all the $f(X)$ evaluations and the targets y are normalized to have a mean of 0 and a length of 1 before subsequent computations. In other words, treat f as $\mathcal{N} \circ f$ and y as $\mathcal{N}(y)$.

Given a set of k alpha factors $\mathcal{F} = \{f_1, f_2, \cdots, f_k\}$ and their weights $w = (w_1, w_2, \cdots, w_k) \in \mathbb{R}^k$, the combination model c is defined as follows:

$$c(X; \mathcal{F}, w) = \sum_{j=1}^{k} w_j f_j(X) = z. \tag{5}$$

We define the loss of the combination model as the mean squared error (MSE) between model outputs and true stock trend values:

$$\mathcal{L}(w) = \frac{1}{nT}\sum_{t=1}^{T}\|z_t - y_t\|^2. \tag{6}$$

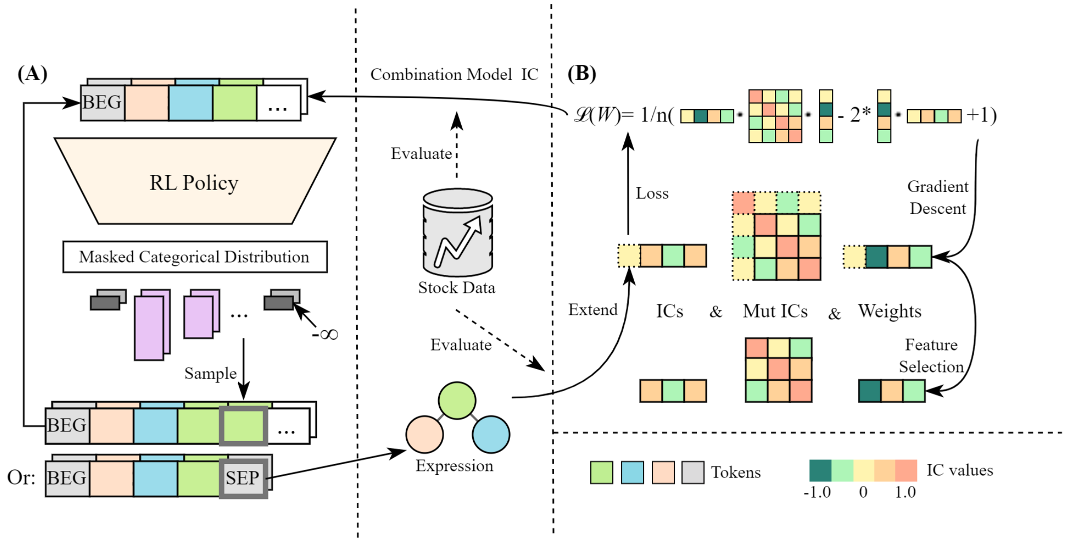


Figure 2: An overview of our alpha-mining framework. (A) An alpha generator that generates expressions, optimized via a policy gradient algorithm. (B) A combination model that maintains a weighted combination of principal factors and, in the meantime, provides evaluative signals to guide the generator.

To simplify the calculation of alpha combination, we have:

THEOREM 3.1. *Let $\mathcal{F}$ be a set of k alphas and w be their respective weights, the MSE loss $\mathcal{L}(w)$ can be represented as:*

$$\mathcal{L}(w) = \frac{1}{n}\left(1 - 2\sum_{i=1}^{k} w_i \bar{\sigma}_y(f_i) + \sum_{i=1}^{k}\sum_{j=1}^{k} w_i w_j \bar{\sigma}(f_i(X), f_j(X))\right). \tag{7}$$

The proof of this theorem is provided in Appendix B. Notice that there is no z_t term on the RHS of Equation 7. Once we have obtained $\bar{\sigma}_y(f)$ for each alpha f and their pairwise mutual correlations $\bar{\sigma}(f_i(X), f_j(X))$, we can then calculate the loss $\mathcal{L}(w)$ solely using these terms, saving time on calculating the relatively large z_t in each gradient descent step.

Considering time and space complexity, it is impractical to combine all generated alphas together, because we need $O(k^2)$ evaluations of mutual IC to calculate mutual correlation for each pair of factors. The quadratic growth of complexity makes it expensive to apply the current procedure to a large number of alphas. However, a few dozen of alphas will suffice for practical uses. To a certain point, more alphas would not bring much more increment in performance, following the law of diminishing returns. We will demonstrate this effect in Section 4.2.2.

After the alpha generator outputs a new alpha, the alpha is first added to the candidate alpha set and assigned a random initial weight. Gradient descent is then performed to optimize the weights with respect to the extended alpha set. We also set a threshold to limit the size of the alpha set, leaving only the principal alphas with the largest absolute weight. If the amount of alphas in the extended set exceeds the threshold, the least principal alpha is removed from the set together with its corresponding weight. The pseudocode of the training procedure is shown in Algorithm 1.

Algorithm 1: Incremental Combination Model Optimization

```
Input: Alpha set $\mathcal{F} = \{f_1, \cdots, f_k\}$, weights
       $w = \{w_1, \cdots, w_k\}$ and a new alpha $f_{\text{new}}$
Output: Optimal alpha subset $\mathcal{F}^* = \left\{f'_1, \cdots, f'_k\right\}$, optimal
        weights $w^* = \left(w'_1, \cdots, w'_k\right)$
$\mathcal{F} \leftarrow \mathcal{F} \cup \{f_{\text{new}}\}$, $w \leftarrow w \| \text{rand}()$;
foreach $f \in \mathcal{F}$ do
    Calculate alpha IC $\bar{\sigma}_y(f)$;
    foreach $f' \in \mathcal{F}$ do
        Calculate mutual IC $\bar{\sigma}(f(X), f'(X))$;
for $i \leftarrow 1$ to $num\_gradient\_steps$ do
    Calculate $\mathcal{L}(w)$ according to Equation 7;
    $w \leftarrow \text{GradientDescent}(\mathcal{L}(w))$;
$p \leftarrow \text{argmin}_i |w_i|$;
$\mathcal{F} \leftarrow \mathcal{F} \setminus \{f_p\}$, $w \leftarrow (w_1, \cdots, w_{p-1}, w_{p+1}, \cdots, w_k)$;
return $\mathcal{F}, w$;
```

3.2 Alpha Generator

The alpha generator models a distribution of mathematical expressions. Traditional auto-regressive generators can only deal with sequences, as a result, we need to represent the expressions in a linear form. Since each expression can be represented as a symbolic expression tree, we use the reverse Polish notation (RPN) to represent it as a linear sequence. To control and evaluate the generation process of valid expressions, we model the generation process as a non-stationary Markov Decision Process (MDP). We will describe the various components of the MDP in the following paragraphs. An overview of the alpha generator is shown in Figure 3.

Table 1: Tokens used in our framework.

Category	Examples
Operators	*CS-Log*, *CS-Add*, *TS-Mean*, , . . .
Features	*$open*, *$volume*, . . .
Constants	−30, −10, −5, −2, −1, −0.5, −0.01, 0.01, 0.5, 1, 2, 5, 10, 30
Time Deltas	10*d*, 20*d*, 30*d*, 40*d*, 50*d*
Sequence Indicator	*BEG*(begin), *SEP*(end of expression)

3.2.1 Tokens. The token is an important abstraction in our framework. A token can be any of the operators, the features, or constant values. Table 1 shows some examples of such tokens. For the full list of operators, please refer to Appendix A; for the full list of features we have chosen, please refer to Section 4.1.1.

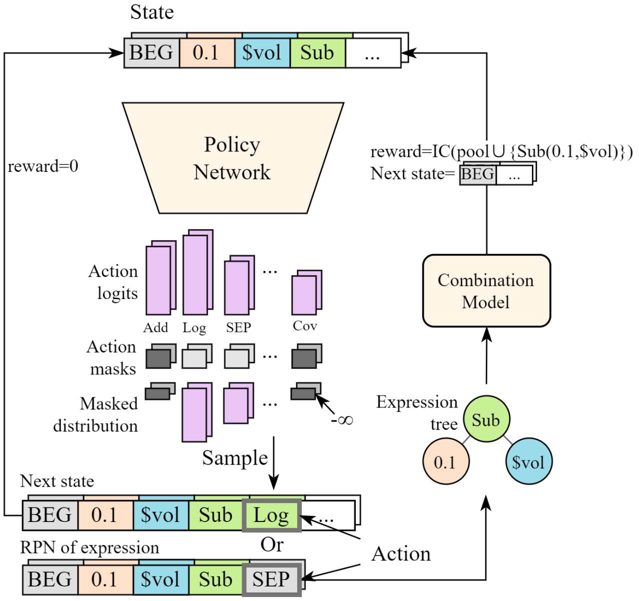


Figure 3: An illustration of our alpha generation framework.

3.2.2 State Space. Each state in the MDP corresponds to a sequence of tokens denoting the currently generated part of the expression. The initial state is always *BEG*, so a valid state always starts with *BEG* and is followed by previously chosen tokens. Since we aim for interpretability of the alphas, and too long of a formula will instead be less interpretable, we cap the length threshold of the formulas at 20 tokens.

3.2.3 Action Space. An action is a token that follows the current state (partially generated sequence). It is obvious that an arbitrarily generated sequence is not guaranteed to be the RPN of a valid expression, so we only allow a subset of actions to be taken at a specific state to guarantee the well-formedness of the RPN sequence. Please refer to Appendix C for more details.

3.2.4 Dynamics. Given a state and an action, we can obtain the next state deterministically. The next state is generated by taking the current state's corresponding sequence and appending the action token to the end.

3.2.5 Rewards and Returns. The MDP does not give immediate rewards for partially formed sequences. At the end of each episode, if the final state is valid, the state will be parsed into a formulaic function and evaluated in the combination model shown in Algorithm 1. To encourage our generator to generate novel alphas, we will then evaluate the new combination model with the new alpha added, and use the model's performance as the return of this episode. Since the reward varies together with the components of the alpha pool, the MDP in our framework is non-stationary.

In practice, longer alphas that perform well are harder to find than shorter ones, due to the exponential explosion of the search space. Therefore, contrary to common RL task settings, we do not necessarily want to penalize longer episodes (longer expressions) for alpha expression generation. We set the discount factor as $\gamma = 1$ (no discount) thusly.

Algorithm 2: Alpha Mining Pipeline

```
Input: Stock dataset, including features and prediction
       targets $X = \{X_t\}, Y = \{y_t\}$
Output: Optimal alpha subset $F^* = \left\{f'_1, \cdots, f'_k\right\}$, optimal
        weights $w^* = \left\{w'_1, \cdots, w'_k\right\}$
Initialize $\mathcal{F}$ and $w$;
Initialize RL policy $\pi_\theta$ with parameters $\theta$ and replay buffer
  $\mathcal{D}$;
for each iteration do
    for each environment step do
        $a_t \sim \pi_\theta(a_t|s_t)$;
        $s_{t+1} \leftarrow [s_t, a_t]$;
        if $a_t$ = SEP or $len(s_{t+1}) \geq threshold$ then
            $f \leftarrow parse(s_{t+1})$;
            Update $\mathcal{F}$, $w$ using $f$ and Algorithm 1;
            $IC_{new} \leftarrow \bar{\sigma}_y(\sum_{i=1}^{k} w_i f_i)$;
            $r_t \leftarrow IC_{new}$;
        else
            $r_t \leftarrow 0$;
        $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r_t, s_{t+1})\}$;
    for each gradient step do
        Use batch $\mathcal{B} \subset \mathcal{D}$ to do gradient descent on PPO
          objective $\mathcal{L}^{CLIP}(\theta)$ to update $\theta$;
return $\mathcal{F}$, $w$;
```

3.2.6 Reinforcement Algorithm. Based on the MDP defined above, we use Proximal Policy Optimization (PPO) [17] to optimize a policy $\pi_\theta(a_t|s_t)$ that takes a state as input and outputs a distribution of action. An actual action will be sampled from the output distribution.

PPO is an on-policy RL algorithm based on the trust region method. It proposed a clipped objective $\mathcal{L}^{CLIP}$ as follows:

$$\mathcal{L}^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min \left\{ r_t(\theta)\hat{A}_t, \operatorname{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t \right\} \right], \quad (8)$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{old}(a_t|s_t)}$, and $\hat{A}_t$ is an estimator of the advantage function at timestep t. Using the importance sampling mechanism, PPO can effectively take the biggest possible improvement while keeping the policy in a trust region that avoids accidental performance collapse.

Since our MDP has complicated rules for the legality of actions, an action sampled from the full discrete action distribution predicted by the learned policy is likely to be invalid as mentioned in Section 3.2.3. We adopt the Invalid Action Masking mechanism [6] to mask out invalid actions and just sample from the set of valid actions.

3.3 Network Architecture

The PPO algorithm requires the agent to have a value network and a policy network. Under our experiment settings, the two networks share a base LSTM feature extractor that converts token sequences into dense vector representations. Separate value and policy "heads" are attached after the LSTM. The hyperparameters of the model are given in Appendix D.

3.4 Training with policy gradient-based methods

For the task of alpha mining, the result we care about is a single synergistic set of alphas that the agent discovers during the whole training process, across all training episodes. In other words, we do not require the agent to achieve relatively high average returns in each episode. For this reason, we maintain a pool of alphas without resetting between episodes. We run the alpha generation procedure mentioned in Section 3.2 and optimize the alpha combination model according to Section 3.1 repeatedly. In this way, we train the policy to continuously generate novel alpha factors that bring improvement to the overall prediction performance.

The proposed alpha mining process is shown in Algorithm 2. Our implementation is publicly available[2].

4 EXPERIMENTS

Our experiments are designed to investigate the following questions:

- **Q1**: How does our proposed framework compare to previous alpha mining methods?
- **Q2**: How well does our model scale as the alpha set size increases?
- **Q3**: Compared to the more commonly used mutual correlation, why is combination model IC a better metric?
- **Q4**: How does our framework perform under more realistic trading settings?

4.1 Experiment Settings

4.1.1 Data. Our experiments are conducted on raw data from the Chinese A-shares market[3]. We select 6 raw features as the inputs to our alphas: {open, close, high, low, volume, vwap (volume-weighted average price)}. The target is set to be the 20-day return of the stocks, selling/buying at the closing price (Ref(close, −20)/close − 1). The dataset is split chronologically into a training set (2009/01/01 to 2018/12/31), a validation set (2019/01/01 to 2019/12/31), and a test set (2020/01/01 to 2021/12/31). In the following experiments, we will use the constituent stocks of the China A-share CSI300 and CSI500 indices as the stock sets.

4.1.2 Compared Methods. To evaluate how well our framework performs against traditional formulaic alpha generation approaches, we implemented two methods that are designed to generate alphas one at a time. **GP** is a genetic programming model using the alpha's IC as the fitness measure to generate expression trees. This model is implemented upon the gplearn framework[4]. **PPO** is a reinforcement learning method, based on the same PPO [17] algorithm and expression generator. It is different from our full framework in that it uses only the single alpha's IC instead of the combined mega-alpha's IC as the episode return.

Since only using the topmost alpha to evaluate the frameworks is extremely prone to overfitting on the training data, we also constructed alpha sets with the ones generated by the two single alpha generators. The same combination model is then applied to these alpha sets. Note that the generators still emit alphas in a one-by-one manner, and are agnostic to the combination model's performance. The first method to construct the set (**top**) is to simply select the top-k alphas emitted by the generator with the highest IC on the training set. The second method (**filter**) is to select the top-k performing alphas with a constraint that any pair of alpha from the set must not have a mutual IC higher than 0.7.

To provide a comprehensive evaluation of our approach, we conducted comparisons with several end-to-end machine learning models implemented in the open-source library Qlib [25]. These models were trained to directly predict the 20-day returns using 60 days' worth of raw features as input. It is important to note that, unlike our approach, these models do not generate formulaic alphas. The hyperparameters of these models were set based on the benchmarks provided by Qlib.

- **XGBoost** [2] is an efficient implementation of gradient boosting algorithms, which ensembles decision trees to predict stock trends directly.
- **LightGBM** [8] is another popular implementation of gradient boosting.
- **MLP**: A multilayer perceptron (MLP) is a type of fully-connected feedforward artificial neural network.

In order to account for the effect of stochasticity in the training process, we evaluated each indeterministic experimental combination using 10 different random seeds.

[2] https://github.com/RL-MLDM/alphagen/

[3] The stock price/volume data is retrieved from https://www.baostock.com. Regarding dividend adjustment, the price/volume data are all forward-dividend-adjusted respected to the adjustment factors on 2023/01/15.

[4] https://github.com/trevorstephens/gplearn

Table 2: Main results on CSI 300 and CSI 500. Values outside parentheses are the means, and values inside parentheses are the standard deviations across 10 runs.

Method	CSI 300		CSI 500	
	IC(↑)	Rank IC(↑)	IC(↑)	Rank IC(↑)
MLP	0.0250 (0.0068)	0.0401 (0.0081)	0.0188 (0.0018)	0.0458 (0.0045)
XGBoost	0.0404	0.0576	0.0353	0.0639
LightGBM	0.0259	0.0324	0.0332	0.0609
PPO_top*	−0.0166 (0.0028)	−0.0144 (0.0075)	0.0025 (0.0076)	0.0295 (0.0135)
GP_top*	0.0078 (0.0218)	0.0157 (0.0271)	0.0200 (0.0112)	0.0504 (0.0160)
PPO_filter*	−0.0044 (0.0107)	0.0101 (0.0107)	0.0042 (0.0042)	0.0506 (0.0052)
GP_filter*	0.0183 (0.0190)	0.0298 (0.0227)	0.0117 (0.0083)	0.0562 (0.0105)
Ours*	**0.0725** (0.0105)	**0.0806** (0.0106)	**0.0438** (0.0064)	**0.0727** (0.0112)

*Optimal combination size in $\{10, 20, 50, 100\}$

4.1.3 Evaluation Metrics. We choose two metrics to measure the performance of our models as follows.

- **IC**, the Pearson's correlation coefficient shown in Eq. 1.
- **Rank IC**, the rank information coefficient. The rank IC tells how much the ranks of our alpha values are correlated with the ranks of future returns. It is calculated using Spearman's correlation coefficient, which measures the monotonic relationship between two variables, defined as:

$$\sigma^{\mathrm{rank}}(u, v) = \sigma(r(u), r(v)), \tag{9}$$

where $r(\cdot)$ is the ranking operator. The ranks of repeated values are assigned as the average ranks that they would have been assigned to[5].

Both of the metrics are **the higher the better**.

4.2 Main Results

4.2.1 Comparison across all alpha generators. To answer **Q1**, we first compare our framework against several other alpha-mining methods, including formulaic ones and non-formulaic ones. The target methods for comparison are PPO, GP, MLP, LightGBM, and XGBoost. Experiments are conducted on CSI300 and CSI500 stocks respectively.

The results are shown in Table 2. Our framework is able to achieve the highest IC and rank IC across all the methods we compare to. Note that the framework is only explicitly optimized against the IC metric. The non-formulaic alpha models come in the second tier. The baseline formulaic alpha generators perform poorly on the test set, especially the RL-based ones. The reinforcement learning agent, when optimized only against single-alpha IC, is prone to falling into local optima and thus overfits to the training set, and basically stops searching for new alphas after a certain amount of steps. On the other hand, the GP-based methods avoid the same problem by maintaining and mutating upon a large population of expressions, although they still cannot produce alphas that are synergistic when used together. The results also show that the filtering techniques cannot solve the synergy problem consistently either.

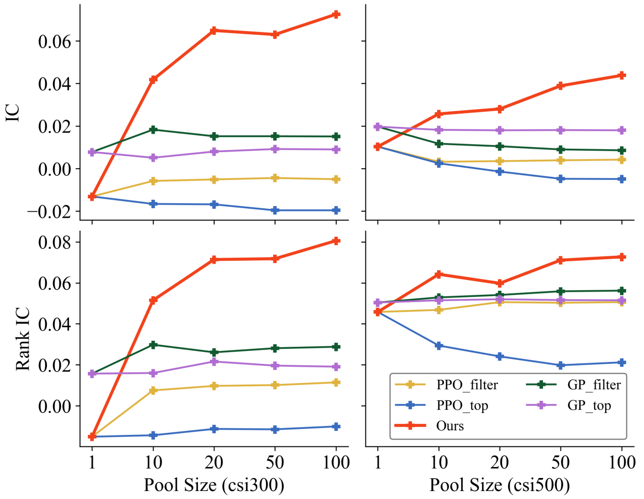


Figure 4: The results of ablation study. A pool size of 1 refers to settings under which only the topmost alpha is evaluated without using a combination model.

4.2.2 Comparison of formulaic generators with varying pool capacity. To answer **Q2**, we study the four baseline formulaic alpha generators more extensively and compare them to our proposed framework. The models are evaluated under pool sizes of $k \in \{1, 10, 20, 50, 100\}$. The results are shown in Figure 4.

Compared to the baseline method PPO_filter, our method directly uses the combination model's performance as the reward to newly generated alphas. This leads to a substantial improvement when the pool size increases, meaning that our method can produce alpha sets with great synergy. Our method shows scalability with pool size: even when the pool size is sufficiently large, it can still continuously find synergistic alphas that boost the performance over the existing pool. Conversely, the combined performance of the alphas generated by other approaches barely improves upon the case with just the top alpha, meaning that these alpha factors have poor synergy. It is noteworthy that the ability to control the reward of individual expressions under a certain alpha pool configuration is granted by the flexibility of the RL scheme. The GP scheme of maintaining a large population at the same time does not work well with fine-grained fitness value control.

Also, we can see that for the CSI500 dataset, GP_filter performs worse than GP_top on the IC metric when the pool size increases. This phenomenon demonstrates that the traditionally used mutual-IC filtering is not always effective, answering the question **Q3**.

4.3 Case Study

Table 3 shows an example combination of 10 alphas generated by our framework, evaluated on the CSI300 constituent stock set. Most of the alpha pairs in this specific set have mutual IC values over 0.7. Previous work [10][27] considered this to be too high for the

[5] For example, $r((3, -2, 6, 4)) = (2, 1, 4, 3)$, while $r((3, -2, 4, 4)) = (2, 1, 3.5, 3.5)$.

Table 3: An example combination of 10 alphas.

#	Alpha	Weight	IC (CSI300)
1	Var(Greater(Greater(Var(low, 50), high), open), 30)	−0.0295	0.0011
2	Max((Min(Max(close, 20) − 30, 20) + 100)/30, 20)	0.0515	0.0262
3	(Mad(high, 50) + 0.5)vwap/close	0.0343	0.0447
4	Ref(low, 50)	0.0260	0.0241
5	Min(high/close, 50) − Greater(−0.05/close, −10)	0.0437	−0.0211
6	Delta(high, 20) + high − 12	−0.0997	0.0165
7	Less(Min(2(vwap − volume), 30) + 30, 30vwap/low) − 5	0.0276	0.0025
8	Corr(Greater(Greater(vwap, volume), Greater(close, Greater(Log(Var(volume, 10)), 10))/close), close, 10)	−0.0279	−0.0338
9	$\lvert \mathrm{low} - 30 \rvert$	0.0319	0.0073
10	$\mathrm{Max}(1 - \mathrm{Max}(\mathrm{Corr}(\mathrm{low}, \mathrm{volume} \cdot \mathrm{Log}(10^{-4}\mathrm{Max}(\mathrm{volume}, 10)), 10), 10), 30)$	0.0312	0.0488
	Weighted Combination		0.0511

individual alphas to be regarded as "diverse", yet these alphas are able to work well in a synergistic manner. For example, the alphas #2 and #6 have a mutual IC of 0.9746, thus traditionally considered too similar to be useful cooperatively. However, the combination $0.09317f_2 - 0.07163f_6$ achieves an IC of 0.0458 on the test set, even higher than the sum of the respective ICs, showing the synergy effect.

Also, although alpha #1 only has an IC of 0.0011, it still plays a vital role in the final combination. Once we remove alpha #1 from the combination and re-train the combination weights on the remaining set, the combination's IC drops to merely 0.0447. The two observations above show that neither the single alpha IC nor the mutual IC between alpha pairs is a good indicator of how well the combined alpha would perform, answering **Q3**.

One possible explanation for these phenomena is that: Although traditionally these alphas are similar due to the high mutual IC, some linear combinations of the alphas could point to a completely different direction from the original ones in the vector space. Consider two unit vectors in a linear space: The more similar these two vectors are, the less similar either of these vectors is to the difference between the two vectors, because as the two unit vectors get closer, the difference vector approaches perpendicular to both of the original vectors.

4.4 Investment Simulation

To demonstrate the effectiveness of our factors in more realistic investing settings, we use a simple investment strategy and conducted backtests in the testing period (2020/01/01 to 2021/12/31) on the CSI300 dataset. We use a simple *top-k/drop-n* strategy to simulate the investment: On each trading day, we first sort the alpha values of the stocks, and then select the top k stocks in that sorted list. We evenly invest across the k stocks if possible, but restrict the strategy to only buy/sell at most n stocks on each day to reduce excessive trading costs. In our experiment, k is set to 50 and n to 5.

We recorded the net worth of the respective strategies in the testing period, of which a line chart is shown in Figure 5. Although our framework does not explicitly optimize towards the absolute returns, the framework still performs well in the backtest. Our framework is able to gain the most profit compared to the other methods.

5 RELATED WORK

Formulaic alphas. The search space of formulaic alphas is enormous, due to the large amount of possible operators and features to choose from. To our best knowledge, all notable former work uses genetic programming to explore this huge search space. [10] augmented the gplearn library with formulaic-alpha-specific time-series operators, upon which an alpha-mining framework is built. [9] further improved the framework to also mine alphas with non-linear relations with the returns by using mutual information as the fitness measure. [27] used mutual IC to filter out alphas that are too similar to existing ones, improving the diversity of resulting alpha sets. PCA is carried out on the alpha values for reducing the algorithmic complexity of computing the mutual ICs, and various other tricks are also applied to aid the evolution process. AlphaEvolve [3] evolves new alphas upon existing ones. It allows combinations of much more complex operations (for example matrix-wise operations), and uses computation graphs instead of trees to represent the alphas. This leads to more sophisticated alphas and better prediction accuracy, although at the risk of lowering the alphas' interpretability. Mutual IC is also used as a measure of alpha synergy in this work.

Machine learning-based alphas. The development of deep learning in recent years has brought about various new ideas on how to accurately model stock trends. Early work on stock trend forecasting treats the movement of each stock as a separate time series, and applies time series models like LSTM [5] or Transformer [21] to the data. Specific network structures catered to stock forecasting like the SFM [26] which uses a DFT-like mechanism have also been developed. Recently, methods for integrating non-standard data with time series have also been investigated. REST [24] fuses multi-granular time series data together with historical event data to model the market as a whole. HIST [23] utilizes concept graphs on top of the regular time series data to model shared commonalities between future trends of various stock groups. One specific type of machine learning-based model is also worth mentioning. Decision tree models, notably XGBoost [2], LightGBM [8],

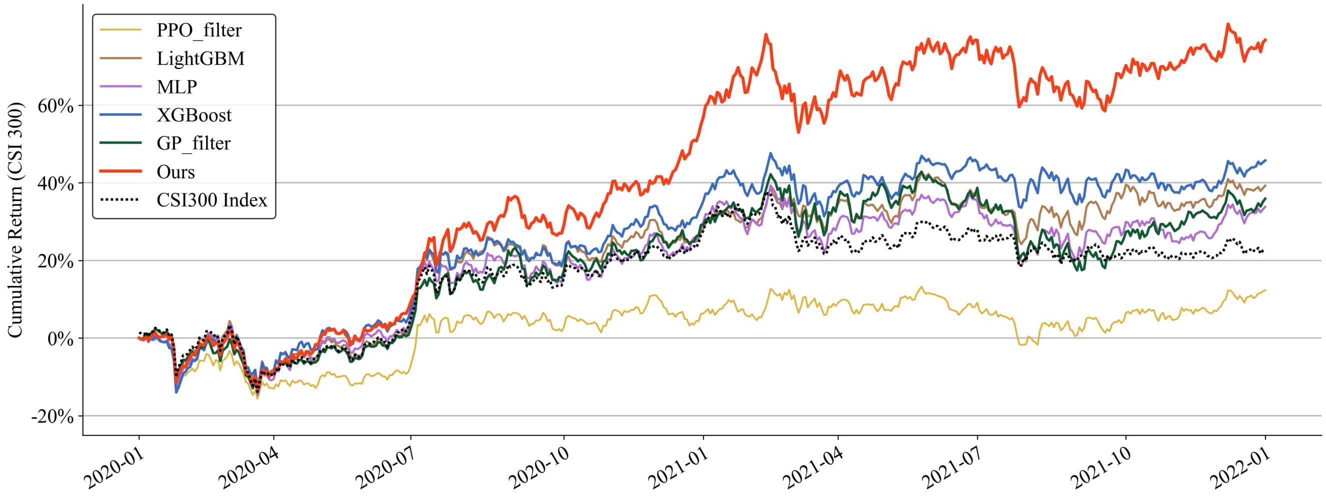


Figure 5: Backtest results on CSI 300. The lines track the net worth of simulated trading agents utilizing the various alpha-mining approaches.

etc., are often considered interpretable, and they could also achieve relatively good performance on stock trend forecasting tasks. However, whether a decision tree with an extremely complex structure is considered "interpretable" is at least questionable. When these tree-based models are applied to raw stock data, the high dimensionality of input only exacerbates the aforementioned problem. Our formulaic alphas use operators that apply to the input data in a more structured manner, making them more easily interpretable by curious investors.

Symbolic regression. Symbolic regression (SR) concerns the problem of discovering relations between variables represented in closed-form mathematical formulas. SR problems are different from our problem settings in that there always exists a "groundtruth" formula that precisely describes the data points in an SR problem, while stock market trends are far too complex to be expressed in the space of formulaic alphas. Nevertheless, there remain similarities between the two fields since similar techniques can be applied to the expression generator and the optimization procedure. [15] suggested using a custom neural network whose activation functions are symbolic operators to solve the SR problem. [13] proposed a novel symbolic regression framework based on an autoregressive expression generator. The generator is optimized using an augmented version of the policy gradient algorithm that values the top performance of the agent more than the average. [12] developed a method similar to [13], but also introduced GP into the optimization loop, seeding the GP population with RL outputs. [20] applied the language model pretraining scheme to symbolic regression, training a generative autoregressive "language model" of expressions on a large dataset of synthetic expressions.

Discussions. Although the term "formulaic alpha" is often associated with investing, the concept of simple and interpretable formulaic predictors that could be combined into more expressive models is not limited to quantitative trading scenarios. Our framework can be adapted to solve other time-series forecasting problems, for example, energy consumption prediction [4], anomaly detection [1], biomedical settings [18], etc. In addition, we chose the linear combination model in this paper for its simplicity. Meanwhile, in theory, other types of interpretable combination models, for example, decision trees can also be integrated into our framework. In that sense, providing these combination models with features expressed in relatively straightforward formulas might help provide investigators with more insights into how the models come to the final results.

6 CONCLUSION

In this paper, we presented a new framework for generating interpretable formulaic alphas, designed to aid investors in quantitative trading. Our framework introduces a metric for alpha synergy, using the performance of the alpha combination as a direct evaluation criterion. This allows us to produce sets of alphas that effectively cooperate with a combination model, irrespective of its specific form. Additionally, we formulated the alpha-searching procedure as an MDP and leveraged reinforcement learning techniques to optimize the alpha generator, enabling more efficient exploration of the vast search space of formulaic alphas. Through extensive experiments, we have demonstrated the superior performance of our framework compared to previous formulaic alpha-mining approaches. Our method can also perform well under more realistic trading settings, reinforcing its practical applicability.

ACKNOWLEDGMENTS

The research work was supported by the National Key Research and Development Program of China under Grant No. 2022YFC3303302, and the National Natural Science Foundation of China under Grant No. 61976204. Xiang Ao is also supported by the Project of Youth Innovation Promotion Association CAS, Beijing Nova Program Z201100006820062.

REFERENCES

[1] Aditya Ashok, Manimaran Govindarasu, and Venkataramana Ajjarapu. 2018. Online Detection of Stealthy False Data Injection Attacks in Power System State Estimation. *IEEE Trans. Smart Grid* 9, 3 (2018), 1636–1646.

[2] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi (Eds.). ACM, 785–794.

[3] Can Cui, Wei Wang, Meihui Zhang, Gang Chen, Zhaojing Luo, and Beng Chin Ooi. 2021. AlphaEvolve: A Learning Framework to Discover Novel Alphas in Quantitative Investment. In *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, Guoliang Li, Zhanhuai Li, Stratos Idreos, and Divesh Srivastava (Eds.). ACM, 2208–2216.

[4] Chirag Deb, Fan Zhang, Junjing Yang, Siew Eang Lee, and Kwok Wei Shah. 2017. A review on time series forecasting techniques for building energy consumption. *Renewable and Sustainable Energy Reviews* 74 (2017), 902–924.

[5] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (1997), 1735–1780.

[6] Shengyi Huang and Santiago Ontañón. 2022. A Closer Look at Invalid Action Masking in Policy Gradient Algorithms. In *Proceedings of the Thirty-Fifth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2022, Hutchinson Island, Jensen Beach, Florida, USA, May 15-18, 2022*, Roman Barták, Fazel Keshtkar, and Michael Franklin (Eds.).

[7] Zura Kakushadze. 2016. 101 Formulaic Alphas. arXiv:1601.00991

[8] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 3146–3154.

[9] Xiaoming Lin, Ye Chen, Ziyu Li, and Kang He. 2019. *Revisiting Stock Alpha Mining Based On Genetic Algorithm.* Technical Report. Huatai Securities Research Center. https://crm.htsc.com.cn/doc/2019/10750101/3f178e66-597a-4639-a34d-45f0558e2bce.pdf

[10] Xiaoming Lin, Ye Chen, Ziyu Li, and Kang He. 2019. *Stock Alpha Mining Based On Genetic Algorithm.* Technical Report. Huatai Securities Research Center. https://crm.htsc.com.cn/doc/2019/10750101/f75b4b6a-2bdd-4694-b696-4c62528791ea.pdf

[11] Qian Liu, Yihong Chen, Bei Chen, Jian-Guang Lou, Zixuan Chen, Bin Zhou, and Dongmei Zhang. 2020. You Impress Me: Dialogue Generation via Mutual Persona Perception. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault (Eds.). Association for Computational Linguistics, 1417–1427.

[12] T. Nathan Mundhenk, Mikel Landajuela, Ruben Glatt, Daniel M. Faissol, and Brenden K. Petersen. 2021. Symbolic Regression via Neural-Guided Genetic Programming Population Seeding. (2021). arXiv:2111.00053

[13] Brenden K. Petersen, Mikel Landajuela, T. Nathan Mundhenk, Cláudio Prata Santiago, Sookyung Kim, and Joanne Taery Kim. 2021. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021.*

[14] Edward E Qian. 2007. *Quantitative equity portfolio management: modern techniques and applications.* Chapman and Hall/CRC.

[15] Subham S. Sahoo, Christoph H. Lampert, and Georg Martius. 2018. Learning Equations for Extrapolation and Control. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018 (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer G. Dy and Andreas Krause (Eds.). PMLR, 4439–4447.

[16] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. 2020. Mastering atari, go, chess and shogi by planning with a learned model. *Nature* 588, 7839 (2020), 604–609.

[17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. (2017). arXiv:1707.06347

[18] Suppawong Tuarob, Conrad S. Tucker, Soundar R. T. Kumara, C. Lee Giles, Aaron L. Pincus, David E. Conroy, and Nilam Ram. 2017. How are you feeling?: A personalized methodology for predicting mental states from temporally observable physical and behavioral information. *J. Biomed. Informatics* 68 (2017), 1–19.

[19] I. Tulchinsky. 2015. *Finding Alphas: A Quantitative Approach to Building Trading Strategies.* 1–253 pages.

[20] Mojtaba Valipour, Bowen You, Maysum Panju, and Ali Ghodsi. 2021. SymbolicGPT: A Generative Transformer Model for Symbolic Regression. (2021). arXiv:2106.14131

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 5998–6008.

[22] Zhicheng Wang, Biwei Huang, Shikui Tu, Kun Zhang, and Lei Xu. 2021. DeepTrader: A Deep Reinforcement Learning Approach for Risk-Return Balanced Portfolio Management with Market Conditions Embedding. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021.* AAAI Press, 643–650.

[23] Wentao Xu, Weiqing Liu, Lewen Wang, Yingce Xia, Jiang Bian, Jian Yin, and Tie-Yan Liu. 2021. HIST: A Graph-based Framework for Stock Trend Forecasting via Mining Concept-Oriented Shared Information. (2021). arXiv:2110.13716

[24] Wentao Xu, Weiqing Liu, Chang Xu, Jiang Bian, Jian Yin, and Tie-Yan Liu. 2021. REST: Relational Event-driven Stock Trend Forecasting. In *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, Jure Leskovec, Marko Grobelnik, Marc Najork, Jie Tang, and Leila Zia (Eds.). ACM / IW3C2, 1–10.

[25] Xiao Yang, Weiqing Liu, Dong Zhou, Jiang Bian, and Tie-Yan Liu. 2020. Qlib: An AI-oriented Quantitative Investment Platform. (2020). arXiv:2009.11189

[26] Liheng Zhang, Charu C. Aggarwal, and Guo-Jun Qi. 2017. Stock Price Prediction via Discovering Multi-Frequency Trading Patterns. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017.* ACM, 2141–2149.

[27] Tianping Zhang, Yuanqi Li, Yifei Jin, and Jian Li. 2020. AutoAlpha: an Efficient Hierarchical Evolutionary Algorithm for Mining Alpha Factors in Quantitative Investment. *arXiv preprint arXiv:2002.08245* (2020).

A LIST OF OPERATORS

There are four types of operators used in our framework. The four types break down into two groups: cross-section operators, and time-series operators. Cross-section operators (indicated by "CS" in the table) only deal with data on the current trading day, while time-series operators (indicated by "TS") take into consideration data from a consecutive period of time. Each of the two groups further divides into unary (indicated by "U") and binary (indicated by "B") operators that apply to one or two series respectively.

B PROOF OF THEOREM 3.1

Proof. We know that the elements of a vector u that is centralized and normalized (using the $\mathcal{N}$ operator mentioned above) have a variance of $1/n$, since:

$$\begin{aligned}\mathrm{Var}\left[u\right] &= \mathbb{E}_i\left[u_i^2\right] - \mathbb{E}_i\left[u_i\right]^2 \\ &= \frac{1}{n}\|u\|^2 - 0 \\ &= \frac{1}{n}.\end{aligned} \tag{10}$$

Using the original definition of Pearson's correlation coefficient, we have:

$$\begin{aligned}\sigma(u,v) &= \frac{\mathrm{Cov}\left[u,v\right]}{\sqrt{\mathrm{Var}\left[u\right]\cdot\mathrm{Var}\left[v\right]}} \\ &= \mathbb{E}_i\left[\frac{(u_i-\bar{u})}{\sqrt{\mathrm{Var}\left[u\right]}}\cdot\frac{(v_i-\bar{v})}{\sqrt{\mathrm{Var}\left[v\right]}}\right] \\ &= \mathbb{E}_i\left[\frac{\left[\mathcal{N}(u)\right]_i}{\sqrt{1/n}}\cdot\frac{\left[\mathcal{N}(v)\right]_i}{\sqrt{1/n}}\right] \\ &= n\mathbb{E}_i\left[\left[\mathcal{N}(u)\right]_i\left[\mathcal{N}(v)\right]_i\right] \\ &= \left\langle\mathcal{N}(u),\mathcal{N}(v)\right\rangle.\end{aligned} \tag{11}$$

Table 4: All the operators used in our framework. CS: cross-section, TS: time-series, U: unary, B: binary.

Operator	Category	Descriptions
Abs(x)	CS–U	The absolute value $\lvert x\rvert$.
Log(x)	CS–U	Natural logarithmic function $\log(x)$.
$x+y$, $x-y$, $x\cdot y$, x/y	CS–B	Arithmetic operators.
Greater(x, y), Less(x, y)	CS–B	The larger/smaller one of the two values.
Ref(x, t)	TS–U	The expression x evaluated at t days before the current day.
Mean(x, t), Med(x, t), Sum(x, t)	TS–U	The mean/median/sum value of the expression x evaluated on the recent t days.
Std(x, t), Var(x, t)	TS–U	The standard deviation/variance of the expression x evaluated on recent t days.
Max(x, t), Min(x, t)	TS–U	The maximum/minimum value of the expression x evaluated on the recent t days.
Mad(x, t)	TS–U	The mean absolute deviation $\mathbb{E}\left[\lvert x - \mathbb{E}\left[x\right]\rvert\right]$ of the expression x evaluated on the recent t days.
Delta(x, t)	TS–U	The relative difference of x compared to t days ago, $x - \text{Ref}(x, t)$.
WMA(x, t), EMA(x, t)	TS–U	Weighted moving average and exponential moving average of the expression x evaluated on the recent t days.
Cov(x, y, t)	TS–B	The covariance between two time series x and y in the recent t days.
Corr(x, y, t)	TS–B	The Pearson's correlation coefficient between two time series x and y in recent t days.

That is to say, the Pearson's correlation coefficient between two vectors equals the inner product of the two vectors centralized and normalized.

Therefore the theorem can be proved as follows. Recall that $f_i(x_t)$ and y_t are normalized.

$$\begin{aligned}
n\mathcal{L}(w) &= \frac{1}{T}\sum_{t=1}^{T}\|z_t - y_t\|_2^2 \\
&= \mathbb{E}_t\left[\|z_t\|^2 - 2\langle z_t, y_t\rangle + \|y_t\|^2\right] \\
&= \mathbb{E}_t\left[\left\|\sum_{i=1}^{k} w_i f_i(X_t)\right\|^2 - 2\left\langle\sum_{i=1}^{k} w_i f_i(X_t), y_t\right\rangle + 1\right] \\
&= \mathbb{E}_t\left[\sum_{i=1}^{k}\sum_{j=1}^{k} w_i w_j \sigma(f_i(X_t), f_j(X_t))\right. \\
&\qquad \left. - 2\sum_{i=1}^{k} w_i\,\langle f_i(X_t), y_t\rangle + 1\right] \\
&= \sum_{i=1}^{k}\sum_{j=1}^{k} w_i w_j \bar{\sigma}(f_i(X), f_j(X)) - 2\sum_{i=1}^{k} w_i \bar{\sigma}_y(f_i) + 1.
\end{aligned} \tag{12}$$

□

C EXPRESSION LEGALITY GUARANTEE

The legality of expressions divides into two parts: *Formal* legality and *semantic* legality.

C.1 Formal Legality

In this section, we focus on formal legality, which ensures that expressions adhere to specific rules during the construction of an RPN. The RPN is built using a stack of expressions, constants, or raw features. The construction procedure follows the following rules, and any actions that violate these rules are masked:

- TS (time-series) operators must take a time-delta (e.g. 10d for a time-difference of 10 days) as its last parameter;
- Excluding the aforementioned time-delta, each operator must take enough expressions as operands, according to the arity of the operator (one for *-Unary, two for *-Binary);
- A multi-token expression should not be equivalent to a constant;
- The special SEP token (end of expression) is only allowed when the generated sequence is already a valid RPN.

For example, when the stack (state) is currently [$open, 0.5], we can choose the "Add" token (a binary operator), building an expression "Add($open, 0.5)". Meanwhile, the operator "Log" is not allowed here because "Log" will take "0.5" and "Log(0.5)" is a constant; similarly, the operator "TS-Mean" is also invalid because "Mean($open, 0.5)" is illegal.

C.2 Semantic Legality

Some expressions that are formally legal may still fail to evaluate due to additional constraints imposed by the operators, resulting in semantic invalidity. An example of such a constraint is the inability to apply the logarithm operator to a non-positive value. Unlike formal legality, which can be detected through the procedure described earlier, semantic invalidity is not directly detected.

To handle these cases, in our experiments, expressions that exhibit semantic invalidity are assigned a reward of -1. This value represents the minimum value of Pearson's correlation coefficient and serves as a discouragement for the agent to generate such expressions.

D HYPERPARAMETERS

The LSTM feature extractor used in the RL agent has a 2-layer structure with a hidden layer dimension of 128. A dropout rate of 0.1 is used in the LSTM network. The separate value and policy heads are MLPs with two hidden layers of 64 dimensions. PPO clipping range ϵ is set to 0.2.