# AutoAlpha: an Efficient Hierarchical Evolutionary Algorithm for Mining Alpha Factors in Quantitative Investment

**Tianping Zhang , Yuanqi Li , Yifei Jin , Jian Li**
Institute for Interdisciplinary Information Sciences (IIIS), Tsinghua University, China
ztp18@mails.tsinghua.edu.cn, {timezerolyq, yfjin1990}@gmail.com,



## Abstract

The multi-factor model is a widely used model in quantitative investment. The success of a multi-factor model is largely determined by the effectiveness of the alpha factors used in the model. This paper proposes a new evolutionary algorithm called *AutoAlpha* to automatically generate effective formulaic alphas from massive stock datasets. Specifically, first we discover an inherent pattern of the formulaic alphas and propose a hierarchical structure to quickly locate the promising part of space for search. Then we propose a new Quality Diversity search based on the Principal Component Analysis (*PCA-QD*) to guide the search away from the well-explored space for more desirable results. Next, we utilize the warm start method and the replacement method to prevent the premature convergence problem. Based on the formulaic alphas we discover, we propose an ensemble learning-to-rank model for generating the portfolio. The backtests in the Chinese stock market and the comparisons with several baselines further demonstrate the effectiveness of *AutoAlpha* in mining formulaic alphas for quantitative trading.

## 1 Introduction

Predicting the future returns of stocks is one of the most challenging tasks in quantitative trading. Stock prices are affected by many factors such as company performances, investors' sentiment, and new government policies, etc. To explain the fluctuation of stock markets, economists have established several theoretical models. Among the most prominent ones, the Capital Asset Pricing Model (CAPM) [Sharpe, 1964] dictates that the expected return of a financial asset is essentially determined by one factor, that is the market excess return, while the Arbitrage Pricing Theory (APT) [Ross, 2013] models the return by a linear combination of different risk factors. Since then, several multi-factor models have been proposed and numerous such factors (also called abnormal returns) have been found in the economics and finance literature. For example, the celebrated Fama-French Three Factor Model [Fama and French, 1993] discovered three important factors that can explain almost 90% of the stock returns[1]. In quantitative trading practice, designing novel factors that can explain and predict future asset returns are of vital importance to the profitability of a strategy. Such factors are usually called *alpha factors*, or *alphas* in short.

In 2016, the quantitative investment management firm, WorldQuant, made public 101 formulaic *alpha factors* in [Kakushadze, 2016]. Since then, many quantitative trading methods have used these formulaic alphas for stock trend prediction [Chen *et al.*, 2019]. A formulaic alpha, as the name suggests, is a kind of alpha that can be presented as a formula or a mathematical expression.

$$\text{Alpha\#101} = (\text{close} - \text{open})/(\text{high} - \text{low})$$

For example, the above formulaic alpha is one of the alpha factors from [Kakushadze, 2016] and is calculated using the open price, the close price, the highest price and the lowest price of stocks on each trading day. This alpha formula reflects the momentum effect that has been observed in different market (see e.g., [Jegadeesh and Titman, 1993]). For each day, the alpha gives different values for different stocks. The higher the value, it is more likely that the stock will have relatively larger returns in the following days.

There are much more complicated formulaic alphas than the one shown above. In Figure 2, we show two examples, Alpha#71 and Alpha#72 in [Kakushadze, 2016]. The most common way of producing new formulaic alphas is to have economists or financial engineers to come up with new economical ideas, transform these ideas into formulas and then validate its effectiveness on the historical stock datasets. It is known that WorldQuant has been employing a large number of financial engineers and data miners (even part-time online users[2]) to design new alphas. This way of finding good alphas requires tremendous human labor and expertise, which is not realistic for small firms or individual investors. Therefore, there is an urgent need to develop tools for mining new effective alphas from massive stock datasets automatically.

Our goal is to find as many diverse formulaic alphas with desirable performance as possible within limited computational resources. Unlike many optimization and search problems which aim at finding one single desirable solution, we

[1] https://en.wikipedia.org/wiki/Fama%E2%80%93French_three-factor_model

[2] https://www.weareworldquant.com/en/home

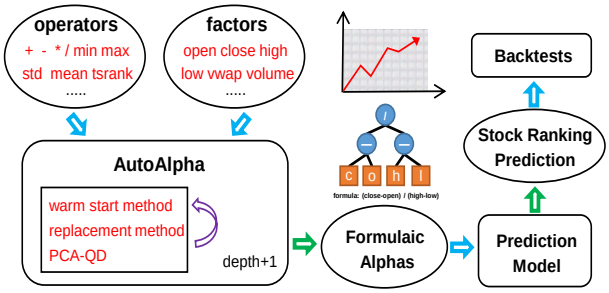



Figure 1: The framework of our approach.

| |
|---|
| Alpha#71: max(Ts_Rank(decay_linear(correlation(Ts_Rank(close, 3.43976), Ts_Rank(adv180, 12.0647), 18.0175), 4.20501), 15.6948), Ts_Rank(decay_linear((rank(((low + open) - (vwap + vwap)))^2), 16.4662), 4.4388)) |
| Alpha#72: (rank(decay_linear(correlation(((high + low) / 2), adv40, 8.93345), 10.1519)) / rank(decay_linear(correlation(Ts_Rank(vwap, 3.72469), Ts_Rank(volume, 18.5188), 6.86671), 2.95011))) |

Figure 2: Formulas Alpha#71 and Alpha#72.

prefer to look for multiple diverse solutions with high performance and low correlation.

As shown in Figure 3, a formulaic alpha can be expressed as a tree where the leaves correspond to raw data and the inner nodes correspond to various operators. As the discrete search space is very large, it is natural to use genetic algorithms to search for effective alphas in the form of trees. However, as we argue below, this is not straightforward and there are several challenges we need to address.

**Challenge 1: Quickly locate the promising search space.** The vanilla genetic algorithm is generally inefficient in mining effective formulaic alphas, due to the fact that effective alphas are sparse in the huge search space. Therefore, how to quickly locate the promising space for search becomes a critical issue.

**Challenge 2: Guide the search away from the explored search space.** In order to find many diverse and effective formulaic alphas, we need to run the genetic algorithm several times for more results. However, the vanilla genetic algorithm usually converges to the same local minima.

**Challenge 3: Prevent the premature convergence problem.** The premature convergence problem [Gupta and Ghafir, 2012] arises in genetic algorithms when some type of effective genes dominate the whole population and destroy the diversity in the population. When premature convergence happens, the population stucks at a suboptimal state and we can no longer produce offspring with higher performance.

In this paper, we propose a new model called *AutoAlpha* to address the above challenges in a unified framework. The technical contributions of this paper can be summarized as follows:

- We discover an inherent pattern of the formulaic alphas. Based on this, we design a hierarchical structure to quickly locate the promising space for search, which address Challenge 1.
- For Challenge 2, we propose a new Quality Diversity method based on the Principal Component Analysis (*PCA-QD*) to guide the search away from the explored space for more diverse and effective formulaic alphas.
- We introduce the warm start method at the initialization step and the replacement method during reproduction to prevent the premature convergence problem. This addresses Challenge 3.

Based on the formulaic alphas we discover, we propose an ensemble learning-to-rank model to predict the stocks rankings and develop effective stock trading strategies. We perform backtests in the Chinese stock market for different holding periods. The backtesting results show that our method consistently outperforms several baselines and the market index.

## 2 Problem Statement

Mining formulaic alphas can be regarded as a feature extraction problem. We start from an initial set of basic factors (e.g. open, close, volume, etc.) and operators (e.g. +-*/, min, std, etc.), and then build formulaic alphas that satisfy certain performance measurement criterion, in order to reveal some inherent patterns of the stock market. The basic factors and the operators we use can be found in [Kakushadze, 2016]. The data is public in Chinese stock markets and can be accessed through multiple resources[3]. In this section, we formalize the problem of mining formulaic alphas.

### 2.1 Stock Returns

The return of a stock is generally determined by the close price of the stock and the holding period. For a given stock $s$, a given date $t$ and a given holding period $h$, the return of the stock can be calculated as:

$$r_{t,s}^{(h)} = \frac{close_{t+h,s} - close_{t,s}}{close_{t,s}}$$

where $close_{t,s}$ is the close price of stock $s$ at date $t$. Assuming that there are $n$ different stocks in the stock pool, the return vector at date $t$ of holding period $h$ is denoted as $\boldsymbol{r}_t^{(h)} = (r_{t,1}^{(h)}, ..., r_{t,n}^{(h)})$.

### 2.2 Evaluation Metrics

For a given formulaic alpha $i$, its value for the stock $s$ at date $t$ is defined as $a_{t,s}^{(i)}$. We use the *IC* (information coefficient) [Grinold and Kahn, 2000] to evaluate the effectiveness of an alpha in the mining process. For a given formulaic alpha $i$ and a given holding period $h$, the IC can be calculated as the mean of the *IC array*:

$$IC_i = \frac{1}{T}\sum_{t=1}^{T} corr(\boldsymbol{a}_t^{(i)}, \boldsymbol{r}_t^{(h)}) \quad (1)$$

where $\boldsymbol{a}_t^{(i)} = (a_{t,1}^{(i)}, ..., a_{t,n}^{(i)})$ is the value vector of formulaic alpha $i$ at date $t$, $corr$ is the sample Pearson Correlation, $\{corr(\boldsymbol{a}_t^{(i)}, \boldsymbol{r}_t^{(h)})\}_{t=1}^{T}$ is the IC array and $T$ is the number of trading days in the training period.

[3] http://tushare.org/#

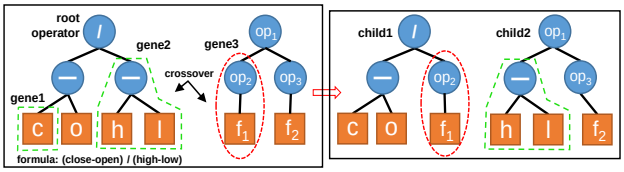



Figure 3: The demonstration of crossover. The leftmost tree shows the tree representation of the formulaic alpha '(close − open)/(high − low)'. The trees on the right are two children after crossover. op: operator. f: factor.

The IC of an alpha indicates the relevance between the alpha and the stock returns, and should be as high as possible[4].

### 2.3 Similarity between Alphas

The similarity between the alpha $i$ and $j$ is calculated as:

$$sim(i,j) = \frac{1}{T}\sum_{t=1}^{T} corr(\boldsymbol{a}_t^{(i)}, \boldsymbol{a}_t^{(j)})$$

A group of alphas is *diverse* if the similarity between any two alphas in the group is lower than 0.7.

In the process of mining formulaic alphas, our goal is to find as many *diverse* formulaic alphas with high IC as possible within limited computational resources.

## 3 AutoAlpha

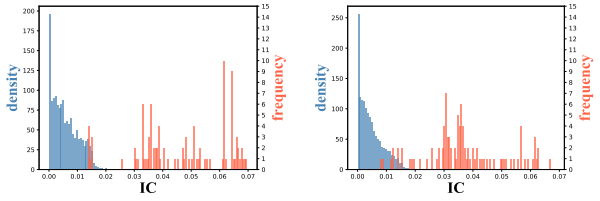
*AutoAlpha* is a framework based on genetic algorithms [Whitley, 1994]. Genetic algorithm is a kind of metaheuristic optimization algorithm which draws inspiration from biological process that produces new offspring and evolve a species. The vanilla genetic algorithm uses mechanisms such as reproduction, crossover, mutation and selection to give birth to new offspring. In each step of regeneration, it uses fitness function to select the best-fit individuals for reproduction. After we give birth to new offspring through crossover and mutation operations, we replace the least-fit individuals in the population with new individuals to realize the mechanism of elimination through competition.

In order to apply the genetic algorithm for mining formulaic alphas, first we need to define the genetic representation of a formulaic alpha. As shown in the leftmost tree of Figure 3, a formulaic alpha can be represented as a formulaic tree. It would be much easier for us to carry out crossover and mutation for trees. Figure 3 shows the crossover between two formulaic alphas of depth 2. We perform the crossover in the same depth level to prevent the depth from increasing. That is, the crossover between *gene1* and *gene3* in Figure 3 is not allowed. The *gene2* and *gene3* are called *root genes* which are directly attached to their root operators while *gene1* is not.

### 3.1 Hierarchical Structure

The search space of trees is huge and the effective alphas are very sparse. In our experiment, we find out that the standard genetic algorithm (e.g., that implemented in python package 'gplearn') is generally inefficient in initializing the population and exploring the search space for mining formulaic alphas (see the results in Section 4.4). For remedy, We propose a novel hierarchical search strategy for the genetic algorithm, that is significantly more efficient in the initialization and exploration of the search space.

[4]W.l.o.g., we assume that $IC \geq 0$ since we can multiply $-1$ to a formulaic alpha which has negative IC.




(a) depth=2 (b) depth=3

Figure 4: The IC of the root genes of the top 100 discovered formulaic alphas and its distribution. For example, in the left figure, the blue histogram is the density plot of IC of the formulaic alphas of depth 2 (estimated by 20000 randomly generated samples). And the red histogram is the frequency plot of IC of the root genes of the top 100 discovered formulaic alphas of depth 3.

**Motivation**

In the early stage of this research, we have been using vanilla genetic algorithms for mining formulaic alphas. An interesting phenomenon occurs during the experiments that the algorithm usually converges to similar formulaic alphas with '*vwap/close*' as a piece of its genes. '*vwap*' is the Volume Weighted Average Price of a stock. The gene *vwap/close* itself is also an effective alpha which relates to the phenomenon of mean reversion. While *vwap/close* itself is an effective formulaic alpha, the formulaic alphas of higher depth which contain *vwap/close* as a piece of its genes usually combine this mean reversion information with some other information and have higher effectiveness.

Based on such phenomenon, we propose a hypothesis about the inherent pattern of the formulaic alphas, that is, most of the effective alphas have at least one effective root gene. Intuitively, if we want to obtain the formulaic alphas of higher depth, we should search nearby the effective alphas of lower depth. We design an experiment to verifies the hypothesis. First, we use the vanilla genetic algorithm for evolving formulaic alphas. Then we select the top 100 discovered formulaic alphas. For each selected alpha, we further collect its root gene with highest IC. We use the density plot to show that those root genes are effective and are hard to obtain by random generation. The results are shown in the Figure 4.

Based on the analysis, if we maintain a population with diverse and effective root genes in it, the crossover operation attempts to searching nearby effective formulas of lower depth, and thus improve the efficiency in obtaining diverse and effective alphas. In order to establish a diverse and effective gene pool for initializing the population, we use the hierarchical structure as the framework for *AutoAlpha* and generate alphas from lower to higher depth iteratively.

### 3.2 Guide The Search

Now the problem has turned to how to generate formulaic alphas of each depth. Unlike many optimization and search problems which aim at finding one single desirable solution, our goal is to look for as many formulaic alphas with high IC and low correlation as possible. If we have already obtained a group of formulaic alphas, we would like to guide the search into unexplored space for more alphas.

However, the genetic algorithms usually converge into the same local minima. In genetic algorithms community, one method to tackle such problem is the **Quality Diversity (QD)** search [Pugh *et al.*, 2016], which seeks to find a maximally diverse collection of individuals where each individual has desirable performance. If we can calculate the similarity (or distance) between individuals, then we can guide the search by changing the objective landscape through penalty [Lehman and Stanley, 2011]. For example, if the similarity between the new alpha and any of the alphas in the record exceeds a certain threshold, then the fitness of the new alpha is penalized to be 0 (least-fit). However, calculating the similarities is slow, especially when the size of the record grows large. Assuming that the size of the record is $p$, then calculating the similarities between a new alpha and the alphas in the record is $O(npT)$ where $T$ is the number of trading days and $n$ is the number of stocks.

In order to reduce the time complexity, we find a simpler way to approximate the similarity and design our *PCA-QD* search. Firstly, we use the first principal component vector to represent the information of a formulaic alpha. Specifically, the values of the formulaic alpha *i* can be presented as a sample matrix $\boldsymbol{A}^{(i)} = (a^{(i)}_{t,s})_{T\times n}$, where each column (stock) can be treated as a feature and each row (date) can be treated as a sample. Then we can calculate the first principal component of the sample matrix. Next, we use the Pearson Correlation between the first principal components of the two alphas, which we call *PCA-similarity*, to approximate the similarity between the two alphas. The calculation of the first principal component using power method is $O(nT + n^2)$. In this way, we reduce the computational complexity of calculating similarities from $O(npT)$ to $O(pT)$ (in experiments, $n$ is 300 and we have $n < T$ and $n < p$).

We run the experiments to see how the approximation method works by random sampling. The threshold for the *PCA-similarity* is set to be 0.9. When the similarity is over $0.7$, the Mean Absolute Error (MAE) between the *PCA-similarity* and the similarity is $0.092$. And when the *PCA-similarity* is over $0.9$, the MAE is $0.125$. Since we are using the *PCA-similarity* instead of the similarity to guide the search away from the explored area, we hope that this approximation should be accurate when the similarity is high and when the *PCA-similarity* is high as well.

### 3.3 Prevention of Premature Convergence

The premature convergence problem has always been a critical issue in the genetic algorithms [Gupta and Ghafir, 2012]. If we always replace the least-fit individuals with new individuals of greater fitness, it is likely that some type of effective genes may dominate the whole population and destroy the genetic diversity. When premature happens, the whole population gets stuck early in a bad local minima. There are many methods addressing the premature convergence problem [Gupta and Ghafir, 2012], and we select two from them in particular for *AutoAlpha*.

**Warm Start**. In the initialization step, instead of randomly generating individuals to the size of the population, we generate individuals $K$ times the size of the population and then select the individuals which rank at top $\frac{1}{K}$ according to IC into the population as initialization. In this way, we improve the average effectiveness of the initialized individuals, and thus accelerate the evolution. The warm start method in AutoAlpha helps us filter out those genes that are not useful in constructing formulaic alphas of higher depth.

**Replacement Method**. In the reproduction step, instead of comparing the new individuals with the least-fit individuals in the population, we compare the new individuals with their own parents. A pair of parents has two offspring after crossover, and the two offspring can replace their parents when the best of their fitness is greater than that of their parents. In this way, all the genes in the population only have one piece of copy after reproduction, which helps us prevent the premature convergence problem.

### 3.4 Overall Algorithm of AutoAlpha

1. We enumerate the alphas of depth 1 and select the effective ones to set up the gene pool.
2. We use the warm start method and the gene pool to initialize the population of depth 2. Then we use crossover and the replacement method for reproduction. If the new offspring has higher IC than its parents, we will calculate its PCA-similarity with the alphas in the record and determine whether to keep it in the population
3. We repeat step 2 for more alphas of depth 2 and update the gene pool and the record. Then we can start generating alphas of depth 3 and so forth.

We use the formulaic alphas as input features and we use LightGBM [Ke *et al.*, 2017] and XGBoost [Chen and Guestrin, 2016] as learning algorithms to learn-to-rank the stocks. Then we ensemble the results and use it for stock investment in the testing period. Due to space limit, we omit the details of the training and the choice of hyper-parameters.

## 4 Experiments

### 4.1 Experimental Settings

**Tasks.** We conduct the experiments independently for the holding periods of 1 day and 5 days. First we use *AutoAlpha* to generate a group of effective formulaic alphas for the given holding period. Then we use the ensemble model to learn to rank the stocks. Finally we use the predicted rankings to construct the stock portfolio each day and provide backtests to evaluate the effectiveness of the alphas we generate.

**Datasets.** We use the 300 stocks in the CSI 300 Index (hs300)[5] as the stock pool when mining the formulaic al-

[5]CSI 300 is a capitalization-weighted stock market index replicating the performance of top 300 stocks traded in the Shanghai and Shenzhen stock exchanges.

Table 1: Performance of the top 5 formulaic alphas for holding period $h = 1, 5$ in both training and testing period. The values out of/in the brackets are the results in the training/testing period.

| Top-$k$ Alpha | $h = 1$ | | $h = 5$ | |
|---|---|---|---|---|
| | hs300 | zz800 | hs300 | zz800 |
| $Top1$ | 8.36%(7.10%) | 8.41%(7.47%) | 8.47%(6.15%) | 8.82%(5.53%) |
| $Top2$ | 8.30%(6.07%) | 8.17%(3.28%) | 7.87%(6.52%) | 7.80%(5.46%) |
| $Top3$ | 7.84%(5.55%) | 7.24%(5.31%) | 7.69%(5.61%) | 7.90%(4.48%) |
| $Top4$ | 7.84%(6.64%) | 7.64%(6.79%) | 7.67%(7.30%) | 7.88%(6.43%) |
| $Top5$ | 7.74%(6.66%) | 8.02%(5.98%) | 7.50%(5.53%) | 7.40%(4.35%) |

Table 2: Comparison with gplearn and Alpha101. $n$: number of diverse formulaic alphas with IC higher than 0.05. $avgIC$: average IC of the top 50 discovered formulaic alphas.

| Method | $h = 1$ | | $h = 5$ | |
|---|---|---|---|---|
| | $n$ | $avgIC$ | $n$ | $avgIC$ |
| Alpha101 | 0 | 1.02% | 0 | 1.25% |
| gplearn | 35 | 6.10% | 7 | 3.35% |
| AutoAlpha | 434 | 7.50% | 415 | 6.71% |

phas in the training stage. We perform backtests on the stock pool of CSI 800 Index (zz800)[6]. The overall testing period is from 20170901 to 20190731 and the training period is from 20100101 to 20170831.

## 4.2 Evaluation Metrics

We measure the performance of the trading strategies by standard metrics in the literature such as annualized return (AR), annualized volatility and Sharpe ratio (SR).

**Annualized return (AR).** Annualized return calculates the rate of return for a given holding period scaled down to a 12-month period: $\text{AR} = \exp\left\{365/T' \times \log\left(S_T/S_0\right)\right\} - 1$, where $T'$ is the number of days and $T$ is the number of trading days. $S_t$ denotes the total wealth at the end of $t$-th trading day.

**Sharpe ratio (SR).** In finance, the Sharpe ratio is a popular measure of the risk-adjusted performance of an investment strategy: $\text{SR} = (R_p - R_f)/\sigma_p$, where $R_p$ is the annualized return of the portfolio, $R_f$ is the risk-free rate[7], and $\sigma_p$ is the annualized volatility, which is the standard deviation of the strategy's yearly logarithmic returns[8].

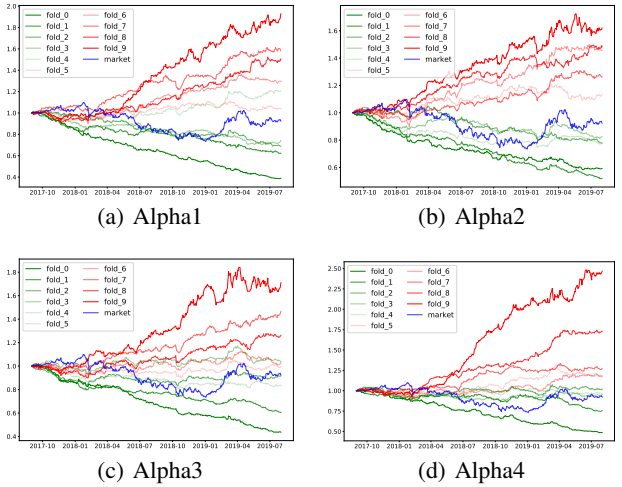
## 4.3 Baselines for Comparison

To further demonstrate the effectiveness of our method, we compare it with the following baselines:

- **Alpha101**: We compare the formulaic alphas we automatically discover with the 101 formulaic alphas in [Kakushadze, 2016]. We use the 101 alphas to train the models and perform backtests under the same settings.
- **gplearn**: 'gplearn' is a popular python package which performs the standard genetic algorithm. It can also be used to generate formulaic alphas.
- **SFM**: [Zhang *et al.*, 2017] proposed SFM (the State Frequency Memory recurrent network) which is an end-to-end deep learning method and applied it to the stock prediction tasks.
- **Market**: The market is represented by the CSI 800 Index. We show that our method is able to outperform the market signficantly.

## 4.4 Results of Generated Formulaic Alphas

Table 1 shows the IC of the top 5 generated formulaic alphas. We show that the alphas can not only generalize in the testing period, but also generalize in the different stock pool of CSI 800 Index. Table 2 shows the comparison between the alphas generated by *AutoAlpha*, the alphas generated by gplearn and the 101 alphas in [Kakushadze, 2016]. We use two metrics to compare the results from both quantitative and qualitative aspects. One is the number of diverse formulaic alphas with IC higher than 0.05. Another is the average IC of the top 50 diverse formulaic alphas. We use the stratified backtests to show the alphas' capability in ranking stocks. The stock pool is divided equally into 10 fold each day according to the ratings the alpha gives and fold 9 has the stocks with highest ratings. Then the $i$-th strategy always buy the stocks in the $i$-th fold each day. The results are shown in Figure 5.




(a) Alpha1 (b) Alpha2

(c) Alpha3 (d) Alpha4

Figure 5: The results of stratified backtests of the top 4 alphas (h=1).

## 4.5 Backtesting Results

For each holding period, we collect the top 150 formulaic alphas generated by *AutoAlpha* which have the highest IC in the training period for model training. Then we use the trained model to predict the stock rankings in the testing period. We ensure the overall procedure does not use future information that is not available at the trading time. We perform backtests using historical data for holding periods $h = 1$ and 5. For a given holding period $h$, the investor invests in the top 10

[6]CSI 800 Index is comprised of all the constituents of CSI 300 Index and CSI 500 Index. It is designed to reflect the overall performance of the large, middle and small capital stocks in Chinese stock markets.

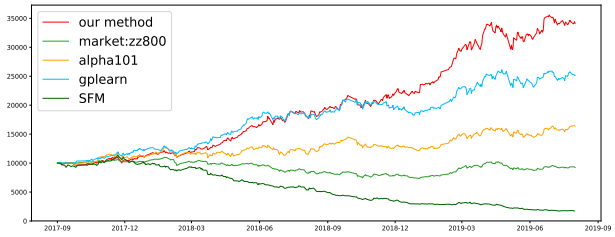
[7]We set the risk-free rate in the experiments to be 0 for simplicity, since there is no consensus on the value of risk-free rate.

[8]https://en.wikipedia.org/wiki/Volatility_(finance)

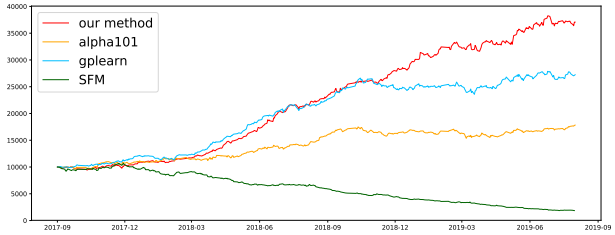
Table 3: Results of backtests and comparisons with baselines. Results relative to the market are in the brackets.

| Method | $h = 1$ | | $h = 5$ | |
|---|---|---|---|---|
| | $AR$ | $SR$ | $AR$ | $SR$ |
| market | -4.1% | -0.20 | -6.4% | -0.31 |
| SFM | -60.0%(-58.5%) | -2.05(-2.89%) | -23.7%(-17.7%) | -1.01(-1.53) |
| gplearn | 61.8%(68.7%) | 2.34(4.26) | 16.9%(22.6%) | 0.72(1.93) |
| Alpha101 | 29.5%(35.3%) | 1.06(2.02%) | 16.3%(23.3%) | 0.70(2.09) |
| our method | **90.0%(98.2%)** | **3.39(6.02)** | **28.0%(34.0%)** | **1.20(3.05)** |

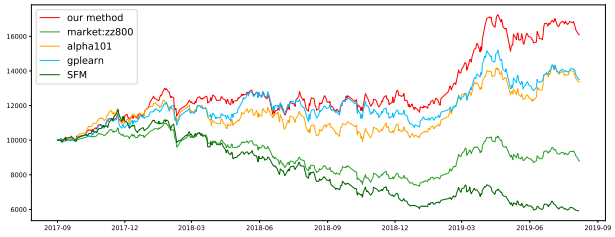
stocks at the close price according to the predicted rankings in each day. Then the stocks are held for $h$ days and sold at the close price at the end of the holding period. The transaction cost is $0.3\%$ as accustomed. The stock pool for trading simulation is the CSI 800 Index (zz800). The backtesting results are shown in the Figure 6, 7 and Table 3.



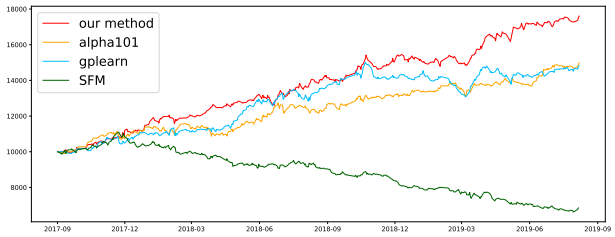

(a) holding period $h = 1$




(b) holding period $h = 1$ (relative to the market)




(c) holding period $h = 5$




(d) holding period $h = 5$ (relative to the market)

Figure 6: The Profit&Loss graph of backtests.

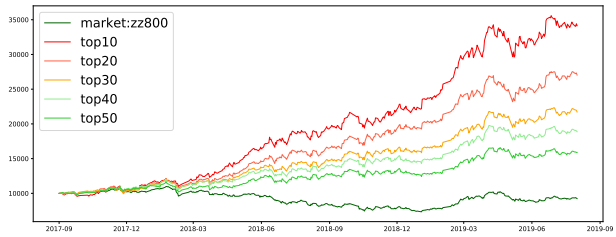



Figure 7: Comparison on backtesting strategies (longing **top 10, top 20, ..., top50**). Holding period $h = 1$.

## 5 Related Work

Data mining and machine learning techniques have been used extensively to address a variety of problems in finance. In particular, our work is closely related to feature extraction in machine learning (see e.g., [Guyon *et al.*, 2008]). Feature extraction has been widely adopted in financial prediction. [Zhang *et al.*, 2018] extracted features from the semantic information in stock comment text for reliability modeling of stock comments. [Huang and Wu, 2008] proposed a GA-based model to extract wavelet features for stock index prediction. Our work develops a substantially different GA-based algorithm and extracts general formulaic alpha factors.

How to maintain diversity has always been a critical issue in genetic algorithms [Gupta and Ghafir, 2012]. The methods to encourage diversity includes Nitching [Sareni and Krahenbuhl, 1998], Crowding [Mahfoud, 1992], Sharing [Goldberg *et al.*, 1987], etc. The replacement method we use derives from the Steady State Genetic Algorithms (SSGAs) [Engelbrecht, 2007]. A number of replacement methods have been developed for SSGAs, including the parent-offspring competition we use [Smith and Vavak, 1999]. The Quality Diversity (QD) is designed to illuminate the diverse individuals with high performance. There are many quality diversity algorithms designated for different kinds of problems [Pugh *et al.*, 2016]. Common examples include Novelty Search with Local Competition (NSLC) [Lehman and Stanley, 2011] and Multi-dimensional Archive of Phenotypic Elites (MAP-Elites) [Mouret and Clune, 2015].

## 6 Conclusions

In this paper, we propose *AutoAlpha*, an efficient algorithm that automatically discovers effective and diverse formulaic alphas for quantitative investment. We first propose a hierarchical structure to quickly locate the promising space for search. Secondly, we propose a new *PCA-QD* search to guide the search away from the explored areas. Thirdly, we utilize the warm start method and the parent-offspring replacement method to prevent the premature convergence problem. The backtests and comparisons with several baselines show the effectiveness of our method. Finally, we remark that *AutoAlpha* can be also viewed as an approach for automatic feature extraction. As the market becomes more efficient, discovering alpha factors becomes more difficult and automatically extracting effective features is a promising future direction for quantitative investment.

## References


[Chen and Guestrin, 2016] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016.

[Chen *et al.*, 2019] Chi Chen, Li Zhao, Jiang Bian, Chunxiao Xing, and Tie-Yan Liu. Investment behaviors can tell what inside: Exploring stock intrinsic properties for stock trend prediction. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2376–2384. ACM, 2019.

[Engelbrecht, 2007] Andries P Engelbrecht. *Computational intelligence: an introduction*. John Wiley & Sons, 2007.

[Fama and French, 1993] Eugene F Fama and Kenneth R French. Common risk factors in the returns on stocks and bonds. *Journal of financial economics*, 33(1):3–56, 1993.

[Goldberg *et al.*, 1987] David E Goldberg, Jon Richardson, et al. Genetic algorithms with sharing for multimodal function optimization. In *Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49. Hillsdale, NJ: Lawrence Erlbaum, 1987.

[Grinold and Kahn, 2000] Richard C Grinold and Ronald N Kahn. Active portfolio management. 2000.

[Gupta and Ghafir, 2012] Deepti Gupta and Shabina Ghafir. An overview of methods maintaining diversity in genetic algorithms. *International journal of emerging technology and advanced engineering*, 2(5):56–60, 2012.

[Guyon *et al.*, 2008] Isabelle Guyon, Steve Gunn, Masoud Nikravesh, and Lofti A Zadeh. *Feature extraction: foundations and applications*, volume 207. Springer, 2008.

[Huang and Wu, 2008] Shian-Chang Huang and Tung-Kuang Wu. Integrating ga-based time-scale feature extractions with svms for stock index forecasting. *Expert Systems with Applications*, 35(4):2080–2088, 2008.

[Jegadeesh and Titman, 1993] Narasimhan Jegadeesh and Sheridan Titman. Returns to buying winners and selling losers: Implications for stock market efficiency. *The Journal of finance*, 48(1):65–91, 1993.

[Kakushadze, 2016] Zura Kakushadze. 101 formulaic alphas. *Wilmott*, 2016(84):72–81, 2016.

[Ke *et al.*, 2017] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, pages 3146–3154, 2017.

[Lehman and Stanley, 2011] Joel Lehman and Kenneth O Stanley. Evolving a diversity of virtual creatures through novelty search and local competition. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 211–218. ACM, 2011.

[Mahfoud, 1992] Samir W Mahfoud. Crowding and preselection revisited. In *PPSN*, volume 2, pages 27–36, 1992.

[Mouret and Clune, 2015] Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*, 2015.

[Pugh *et al.*, 2016] Justin K Pugh, Lisa B Soros, and Kenneth O Stanley. Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI*, 3:40, 2016.

[Ross, 2013] Stephen A Ross. The arbitrage theory of capital asset pricing. In *HANDBOOK OF THE FUNDAMENTALS OF FINANCIAL DECISION MAKING: Part I*, pages 11–30. World Scientific, 2013.

[Sareni and Krahenbuhl, 1998] Bruno Sareni and Laurent Krahenbuhl. Fitness sharing and niching methods revisited. *IEEE transactions on Evolutionary Computation*, 2(3):97–106, 1998.

[Sharpe, 1964] William F Sharpe. Capital asset prices: A theory of market equilibrium under conditions of risk. *The journal of finance*, 19(3):425–442, 1964.

[Smith and Vavak, 1999] Jim E Smith and Frantisek Vavak. Replacement strategies in steady state genetic algorithms: dynamic environments. *Journal of computing and information technology*, 7(1):49–59, 1999.

[Whitley, 1994] Darrell Whitley. A genetic algorithm tutorial. *Statistics and computing*, 4(2):65–85, 1994.

[Zhang *et al.*, 2017] Liheng Zhang, Charu Aggarwal, and Guo-Jun Qi. Stock price prediction via discovering multi-frequency trading patterns. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 2141–2149. ACM, 2017.

[Zhang *et al.*, 2018] Chen Zhang, Yijun Wang, Can Chen, Changying Du, Hongzhi Yin, and Hao Wang. Stockassistant: A stock ai assistant for reliability modeling of stock comments. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2710–2719. ACM, 2018.