AlphaGen

Andrew Huang

August 2024

1 Introduction

This paper is interested in applying directed evolution in the field of quantitative finance with the goal of generating novel trading strategies with superior performance. Using GPT-4o as the source of variation, strategies are evaluated and propagated based on PnL, Sharpe, turnover, and fitness.

Directed Evolution (DE) is a powerful tool commonly used in protein engineering. DE consists of three stages:

1. Introducing variety to the gene pool (usually through artificial means such as applying mutagens).
2. Selecting the most desirable candidates using pre-defined criteria.
3. Amplifying the desirable traits and use it as template for the next round of evolution.

Mimicking natural selection, DE has been successfully used to improve stability of enzymes or augment catalytic rates, often by a factor of a hundred or more.

DE has two main advantages. First, it allows us to narrow down search spaces. One of twenty amino acids chain together to form the primary structure of proteins. The sequence space of merely 100 amino acids has the size of 10E130. Second, DE allows us to optimize proteins without having a full understanding of why certain alterations work.

Similarities can be drawn between protein-engineering and portfolio management. Quantitative strategies are made of numerous statistical operators and parameters that can be nested and adjusted. Together, the search space for a high-performing strategies is vast. Furthermore, successful strategies cannot always be understood rationally. This paper aims to address these problems by using GPT-4o to generate novel trading ideas and selectively propagate those with superior Sharpe and fitness.

2 Platform

To calculate key performance metrics such as PnL, Sharpe, fitness etc, a back testing platform is constructed. EOD data of S&P500 stocks is obtained from Kaggle. As this project was inspired by the WorldQuant Trading Competition, the backtesting window is set according to the rules of the competition: from March 10th, 2016 to February 11th, 2021. The available data fields are open, high, low, close and volume. Ticker OGN, GEHC, CEG are excluded because they are listed on the index after the backtesting end date. GEHC is excluded due to incomplete EOD data.

The book size is kept constant on all trading days. If the portfolio value exceeds the initial book size, the gain is realized the immediately. Similarly, if the portfolio value falls below the book size, new cash is injected to keep the invested amount constant. Though not realistic for a real trader, this treatment prevents overemphasizing gains or loss on any specific day of during backtesting period.

A series of statistical operators is created.

1. rank() takes a vector of numbers, rank them from highest to lowest and output the rank of the elements in the vector

2. delta() calculates the difference between the value of a variable (say open price) today with the variable a specified number of days ago

3. ts_min() and ts_max() returns the smallest and largest value of a variable within the past specified number of days. The same applies to ts_rank(), ts_stdev(), ts_sum()

4. correlation() and covariance() calculates the correlation and covariance between two random variables over a specified period and return the result

5. absolute() returns the absolute value of the input

Alpha signals is any combination of these operators. Example alphas include:
(correlation(rank(delta(log(volume), 2)), rank(((close - open) / open)), 6))
(sign(delta(volume, 1)) * (-1 * delta(close, 1)))

Each alpha expression returns a vector of numbers (termed signal) corresponding to each stock. The signal is normalized each day and assets are purchased in proportion to the normalized signal. If the signal value is invalid (for instance dividing by zero), the signal from the last valid date is forward filled. Though transaction cost is not factored into the simulation, strategies with high turnover is penalized in the evolution process.

3 GPT Mutagenesis

Using GPT-4o, a variety of trading strategies is formulated. The system prompt of the conversation is as follows:

> You are a finance professional with background in economics and quantitative finance. Please direct your response in the form of fast expression language, as presented below

GPT-4o is then supplied with some examples alpha ideas written in fast expression language.

> Here are some example "alphas", written in fast expression language. Alphas are algorithmic strategies that will provide return higher than the market. Learn the fast expression language. Using available data fields, generate 10 new alphas
>
> Alpha#1: (-1 * correlation(rank(delta(log(volume), 2)), rank(((close - open) / open)), 6))
>
> Alpha#2: (-1 * correlation(rank(open), rank(volume), 10)) ... (in total 14 examples are provided)
>
> MAKE SURE THE RESPONSE YOU GIVE IS IN THE FOLLOWING FORMAT: It should begin with "Alpha#xx" Followed by a colon And then a specific type of command or formula related to financial analysis.

Therefore, within each generation of evolution, ten alpha ideas are proposed and evaluated. The evaluation process is described in the next section.

4 Selection

Each trading strategy is back-tested and key metrics are computed (according WorldQuant convention). The Sharpe ratio is defined to be the realized return divided by the standard deviation of the daily returns.

$$S_a = \frac{R_a}{\sigma_a}$$

where R_a is the asset return, and σ_a is the standard deviation of the daily returns.Higher Sharpe is desired because it indicates higher return per unit of volatility.

Turnover measures how frequently assets are traded by the strategy. It is defined as the minimum of the total purchase value and the total sale value divided by the book size.

$$T = \frac{min(V_b, V_s)}{S}$$

where V_b and V_s is the total value of assets bought and sold during the back-testing period and S is the book size. Frequent transaction is associated with higher cost, therefore a smaller turnover is desired.

Finally, fitness is defined as the following

$$F = S_a \cdot \sqrt{\frac{R_a}{max(T, 0.125)}}$$

Fitness takes into account the Sharpe as well as trading frequency. Higher fitness is more desirable as it indicates either higher return or lower turnover.

In each generation, ten strategies are proposed by the model. The Sharpe and fitness is ranked within each generation. The ranks are then added together to produce a final score. Lower score indicates better performing strategies. The top three strategies and the bottom three strategies are presented to the model as inspiration for the next generation.

An intricate balance between learning and variety needs to be maintained. If too much emphasis is placed on variety, the algorithm will mindlessly produce alpha ideas with no attempt to build on the successes from previous generations. If the model tilts too much toward learning, the model will only fine tune existing strategies, potentially overfitting the backtesting data.

To keep this balance, the model is instructed to produce seven strategies based on previous generations and three strategies that are completely novel. The follow-up prompt is as follows:

> {top_alpha}, {second_alpha}, and {third_alpha} performed best, second-best, and third-best in backtesting, respectively. {worst_alpha}, {second_worst_alpha}, {third_worst_alpha} performed worst, second-to-worst, and third-to-worst in the backtesting, respectively. I want you to think about what makes the best-performing alphas good and the worst-performing alphas bad. Using this insight, generate seven more good-performing alpha ideas. In addition, I want three alpha ideas that are COMPLETELY UNRELATED to the previous best performers. Use different ideas, different nesting of function. It needs to be very different. Make sure you use different fields. Increase the variety of alpha ideas produced. Do not simply change the values of the parameters but try to come up with novel ideas. Give the seven good ideas first before the three unrelated ideas. Make sure the response you give is in the following format: It should begin with "Alpha#xx", Followed by a colon, And then a specific alpha idea written in fast expression language.

In an iterative process, 75 rounds of evolution is conducted. Conversations from more than eight generations ago are deleted to save token count and limit context size. In three generations, GPT-4o returned invalid strategies (such as using a function that doesn't exist or improper closing of parentheses). These strategies are discarded without further examinations.

5 Results

5.1 Evidence of Learning

Over the course of evolution, the model proposes a variety of strategies. Some proposed examples are given below:
"(-1 * rank(ts_stdev(delta(field('close'), 1), 5)))"

"(-1 * rank(correlation(rank(field('volume')), rank(field('high')), 10)))"
"(rank(ts_sum(field('high'), 15)) * rank(ts_stdev(field('low'), 15)))"
Overall, the model successfully grasps the syntax of the commands and is capable of nesting different operations to produce novel ideas.

Within each generation, the metrics of the best strategy is recorded and plotted below. I posit that strategy evolution can be divided into two stages: breakthrough and refinement. During refinement, the model tweaks parameters to optimize strategy performance. Some generation witnesses a breakthrough: a new strategy is discovered and outperforms the current winner. The new strategy then becomes the norm and is refined until the next breakthrough occurs. In the evolution plots below, we see distinct phases of breakthrough (when there is a spike in Sharpe or significant drop in turnover), followed by generations where the metrics fluctuate around a fixed value.

In the first generation, the best performing alpha was
"Generation 1: "(rank(ts_stdev(field('volume'), 10)) * rank(ts_sum(field('close'), 10)))""
For a few rounds, the model found different fields (volune, high, low, close) and parameters produced marginally better results. The best performers of the subsequent rounds are given below:
"Generation 2: (rank(ts_sum(field('low'), 20)) * rank(ts_stdev(field('close'), 20)))"
"Generation 3: (rank(ts_sum(field('high'), 20)) * rank(ts_stdev(field('low'), 20)))"
"Generation 4: (rank(ts_sum(field('close'), 20)) * rank(ts_stdev(field('open'), 20)))"

In generation 21, the model discovers that certain pairing between fields and operation, namely ts_min with low and ts_max with high drastically reduces fitness with a slight reduction in Sharpe, and the subsequent proposals followed this rule.

The final strategy at the end of 75 generations is "(rank(ts_min(field('low'), 10)) * rank(ts_max(field('high'), 10)))" is reached. While the model proposes a large variety of strategies, the best performers in each generation takes on similar form. DE holds a laissez-faire attitude towards the product of evolution: it makes little attempt at increasing the variety of best performers. However, modifications can be easily made within the existing framework to promote attempts at different alpha ideas, including:

1. Increasing the ratio of "unrelated alphas" to "refinement alphas" via the follow-up prompt.

2. Introducing a new metric that computes the correlation of returns with previous best performers, selecting those with a lower correlation coefficient.

3. Increasing the number of generations (though associated with more API calls and higher cost)

4. Increase the number of operations and data fields.

The metrics of the best performers in each generation is plotted below:

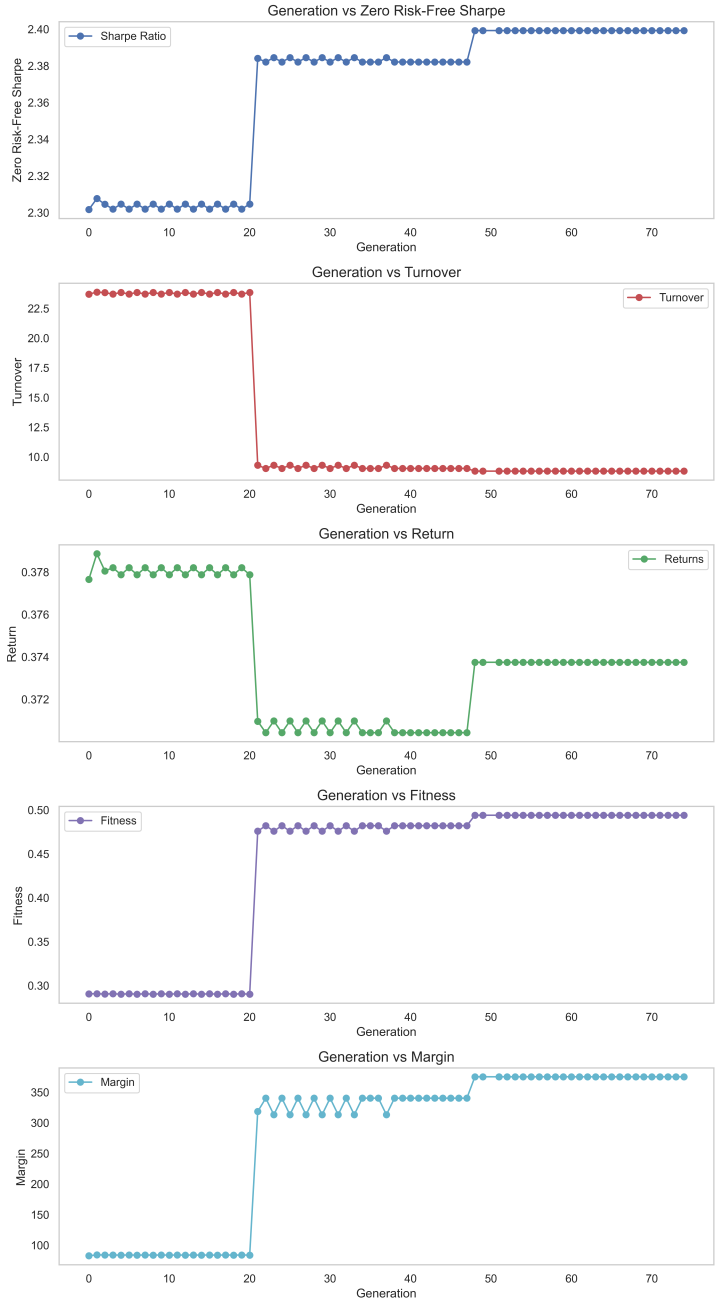

Figure 1: Sharpe, turnover, return, fitness, and margin of the best strategy within each generation.

To quantify the result, the performance metric is regressed on generation. The result of the regression is presented in the table below.

Table 1:

	Performance Metrics			
	Sharpe	Turnover	Returns	Fitness
generation	0.0016^{***}	-0.2422^{***}	$-6.352 \times 10^{-5***}$	0.0033^{***}
	(0.000)	(0.022)	(1.44×10^{-5})	(0.000)
constant	2.3081^{***}	22.0893^{***}	0.3762^{***}	0.3091^{***}
	(0.005)	(0.954)	(0.001)	(0.012)
R-squared	0.732	0.621	0.213	0.659
N	75	75	75	75

Note: $^{*}p < 0.1$; $^{**}p < 0.05$; $^{***}p < 0.01$

As seen in the regression table, there is strong evidence that the proposed strategy commands higher Sharpe, fitness, and lower turnover over time. The result also suggests that evolution can be encouraging lower returns, though the size of the effect is very small (on the order of 10E-5). Since return is not included in the evaluation criteria, it is possible that the model sacrifices some return for a lower turnover or a higher Sharpe, which can be desirable when executing the strategy.

5.2 Validation Result

To ensure that the model is not overfitting to the backtesting data, the strategies are back-tested against validation data. Using the same simulation infrastructure, the strategy is tested on S&P500 data from 2021-02-11 to 2022-09-29 (in accordance with WorldQuant competition data).

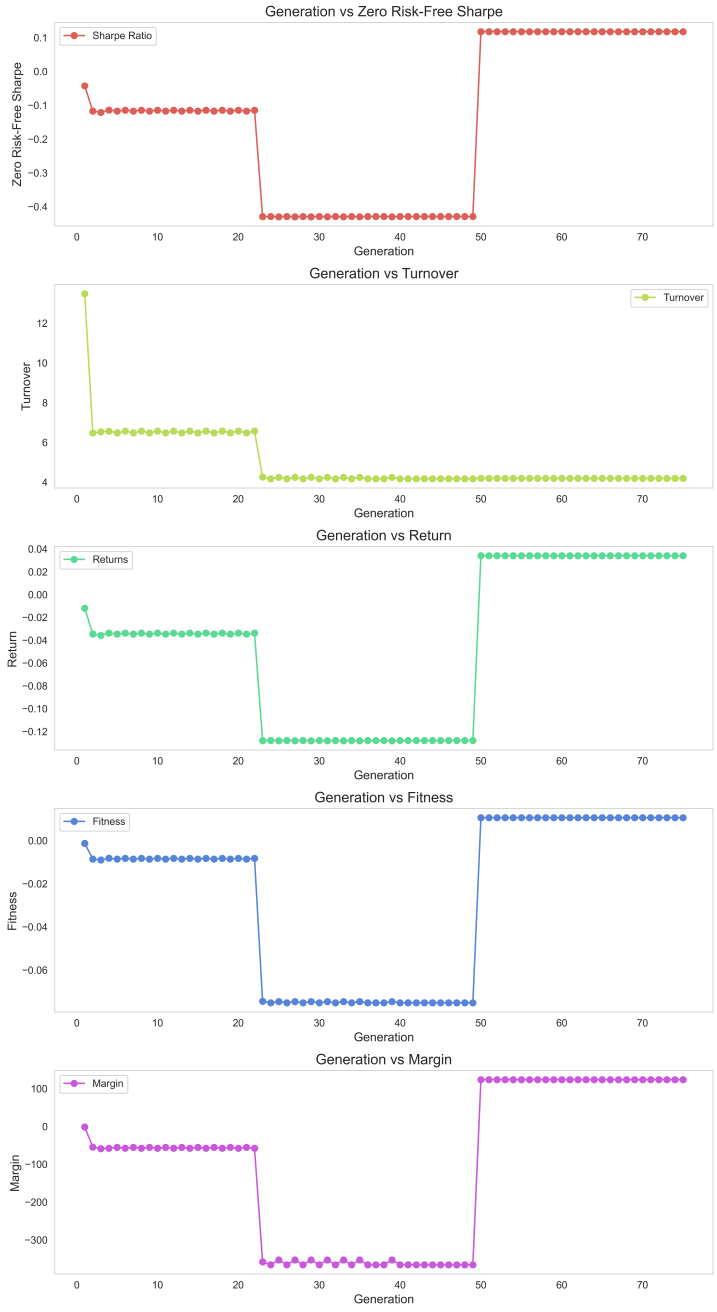

Figure 2: Sharpe, turnover, return, fitness, and margin of the best strategy within each generation in validation set.

Regression of performance metrics on generation is again performed. The result is as follows:

Table 2:

	Performance Metrics			
	Sharpe	Turnover	Returns	Fitness
generation	0.0046^{***}	-0.0461^{***}	0.0014^{***}	0.0004^{***}
	(0.001)	(0.006)	(0.000)	(0.000)
constant	-0.3232^{***}	6.7248^{***}	-0.0957^{***}	-0.0424^{***}
	(0.049)	(0.245)	(0.015)	(0.009)
R-squared	0.189	0.124	0.482	0.185
N	75	75	75	75

Note: $^{*}p < 0.1$; $^{**}p < 0.05$; $^{***}p < 0.01$

Though performance between generation 20-50 is a bit lower than expected, we see overall that metrics in the validation set largely corresponds with the metrics in training set.

6 Conclusion

The application of Directed Evolution (DE) in quantitative finance, specifically in the generation and optimization of trading strategies, demonstrates a promising avenue for innovation. By leveraging GPT-4o, we have shown that it is possible to mimic the natural selection process to iteratively improve trading strategies based on key performance metrics such as Sharpe ratio, turnover, and overall fitness. The evolutionary process has successfully generated novel strategies that outperform their predecessors, confirming the system's ability to learn and refine over successive generations.

However, the experiment also revealed limitations, particularly in the tendency of the model to converge on similar strategies rather than diversifying its approach. Future work could address this by adjusting the balance between exploration and exploitation, perhaps by increasing the proportion of novel strategies generated or by incorporating additional performance metrics to encourage diversity in the strategies developed.

1.https://www.nature.com/articles/s41929-022-00821-3 2.