Momentum Families through Clustering of Sequential Financial Returns

Eesh Naik, Som Naik

Columbia University

Author Note

Email: esn2125@columbia.edu, sn3139@columbia.edu | Date: 5/3/25 | Grants: None

Abstract

This paper partitions a time series of asset returns into vectors of sequential returns. It clusters these vectors into different "momentum families" using K-means clustering. Then, it is probabilistically evaluated whether a member of a certain momentum family is more likely to appear immediately after witnessing members from other momentum families in the time series. Our results show that the distribution of frequencies with which these momentum families follow each other is not uniform, and that some families are more likely to appear in the future given the current family present in the time series. We incorporate this idea into a trading model, outperforming "buy and hold" on assets with minimal external information, specifically Bitcoin ETF and commodity ETFs. It also extracts more information from price movements than a vanilla autoregressive model.

Keywords: Clustering, financial time series, autoregressive

Introduction

Momentum is often of great interest to traders and asset managers. Those who are not implementing arbitrage strategies can look to be long or short in an asset that is showing sustained momentum in a certain direction, instead of looking for orthogonal alpha in the opposite direction. The central philosophy behind momentum is that the past values in the time series will indicate what happens in the future. This is the basis behind processes such as ARMA-GARCH as well. In our opinion, the limitation of AR-like models is that only static snapshots of the past are combined with single coefficients to predict the future. For example, in a financial time series of returns, you might assign a coefficient to yesterday's return and another coefficient to day-before-yesterday's return. This incorporates information from the past but does not consider how yesterday's return might relate to the day before that. It gives a static, sliced view of the past that we believe is a hinderance when trying to capture momentum. In this paper, we believe we improve that framework by creating a more informative data point: a vector of sequential returns. This vector gives us a relative direction in which the sequential returns point, as well as their magnitude. These vectors can be clustered such that similar ones are grouped together.

Sequential Return Vector

The sequential return vector (SRV) is the fundamental basis of our methodology. In our paper, we create it by first obtaining a financial time series with daily resolution, such as the daily price of a stock. After transforming the price into market adjusted log returns, we evenly partition that time series into SRVs of a desired length and put them into a matrix. If our desired length of each vector is three and our total time series is nine data points, the matrix starting at day $\boldsymbol{t}$ is as follows:

$$Let\ S\ be\ a\ time\ series\ of\ market\ adjusted\ \log returns$$

$$\mathbf{M_t} = \begin{vmatrix} S_t & S_{t+3} & S_{t+6} \\ S_{t+1} & S_{t+4} & S_{t+7} \\ S_{t+2} & S_{t+5} & S_{t+8} \end{vmatrix}$$

One thing to note is that if the length of our time series is not a multiple of the length of our SRVs, we simply drop a few datapoints from the end of our time series so that it matches. Other considerations about the SRV matrix are mentioned in the appendix.

Clustering Columns in SRV Matrix

After taking in all the considerations and building our SRV matrix, we must cluster the columns such that similar columns are grouped. Ultimately, when we finish clustering, we expect there to be distinct clusters of SRVs that represent different families of momentum. This clustering approach is effective because it maintains the information we get about the momentum from each SRV. For example, we may have a taxonomy of SRV clusters as seen below:

$$Cluster\ Center\ 1\ (upward\ momentum)\text{: } \begin{vmatrix} 0.1 \\ 0.2 \\ 0.3 \end{vmatrix}$$

$$Cluster\ Center\ 2\ (V-shaped\ momentum)\text{: } \begin{vmatrix} 0.2 \\ -0.1 \\ 0.3 \end{vmatrix}$$

$$Cluster\ Center\ 3\ (noise)\text{: } \begin{vmatrix} 0.001 \\ 0.002 \\ 0.001 \end{vmatrix}$$

Through this sample taxonomy, we can see that every single return in a given SRV simultaneously determines which cluster it belongs to. For example, if the third return in the noise cluster center was dramatically increased, that vector would point somewhere completely different. However, if we dramatically increase the third return in the upward momentum center, it would likely point in a similar direction. This is an improvement to a traditional autoregressive process because if we apply the same "third return changing" test, our final answer will largely depend on the coefficient of that third return in our model. Its value relative to the other S_{t-n} returns would not carry any weightage in the final prediction.

Now that we have established the rationale behind using clustering for grouping the columns in our SRV matrix, we proceed to use K-means clustering[1]. This is a good choice because different groupings of SRVs can be generated while also specifying how many clusters to make, a feature not available in other algorithms like DB Scan.

Results on Bitcoin ETF

We now apply our procedure on the Grayscale Bitcoin Trust ETF (GBTC). This ETF is passively invested in Bitcoin and gives us a way to follow the price movements of Bitcoin while including demand from people without crypto wallets, etc. Unlike traditional securities issued by companies, GBTC does not have earnings, press releases, proxy wars, and other corporate actions. Price movements are primarily affected by investors and their expectations. We obtain our data from the Yahoo Finance API "yfinance" package in Python and pull daily closes starting from 2021/1/01 (Monday) till 2024/12/13 (Friday) and exclude weekends.

First, we calculate market adjusted log returns[2] on GBTC, using the S&P 500 as the market. Next, we decide on the length of our SRV and the number of clusters to group the SRVs. For the length, we picked 5 days as we hypothesized that most trends in a publicly traded ETF would last approximately 5 business days or one trading week. Anything shorter than a week would include too much noise and anything longer would start to create broad regimes rather than identifying patterns. For the number of clusters, we picked 4. Intuitively, we expect to see the following four clusters: noise, upward trending, downward trending, and V-shaped[3]. After running K-means, we get the following cluster centers for our SRVs:

[1] Details on implementation for K-Means provided in appendix
[2] Details on calculation are provided in appendix
[3] These expectations are intuitive and subject to change based on the asset, time horizon, and more

$$C_1 = \begin{vmatrix} 0.20 \\ 3.97 \\ 9.86 \\ -1.68 \\ 4.26 \end{vmatrix}, C_2 = \begin{vmatrix} 2.42 \\ -5.43 \\ 2.91 \\ 0.04 \\ -4.10 \end{vmatrix}, C_3 = \begin{vmatrix} 2.54 \\ 2.18 \\ -4.40 \\ 2.41 \\ -1.60 \end{vmatrix}, C_4 = \begin{vmatrix} -3.59 \\ -1.25 \\ -1.11 \\ -2.96 \\ 2.09 \end{vmatrix}$$

$$Total\ Return\ of\ Centers: C_1 = 16.61, C_2 = -4.16, C_3 = 1.13, C_4 = -6.82$$

$$Norm\ of\ Centers: C_1 = 11.58, C_2 = 7.78, C_3 = 6.23, C_4 = 5.36$$

$$* Values\ written\ in\ 10^{-2}$$

Consistent with our hypothesis, we see that C_1 can roughly be categorized as upward trending, C_2 as W-shaped, C_3 as zero-sum or noise, and C_4 as downward trending. Now we investigate whether seeing a certain cluster can tell us which cluster will come next. To do this, we use our trained K-Means model to label the SRVs in our study period and count the frequency with which each cluster member follows another. This creates a transition matrix that indicates the likelihood of each cluster member following another. We also implement Chi-Squared Goodness-of-Fit to determine whether these probabilities of appearance are truly different from a uniform distribution. If the p-value is low for a certain cluster, it implies that clusters on $t + 1$ will appear asymmetrically.

GBTC Cluster Probabilities		Likelihood of Observing on $\boldsymbol{t+1}$				Freq.	$\boldsymbol{\chi^2}$ GOF
		C_1 *(Upward)*	C_2 *(W-shaped)*	C_3 *(Noise)*	C_4 *(Downward)*	N	p-value
Cluster Observed on $\boldsymbol{t}$	C_1	16.67%	16.67%	16.67%	50.00%	6	0.57
	C_2	30.00%	0.00%	40.00%	30.00%	10	0.31
	C_3	0.00%	28.57%	28.75%	42.86%	21	0.04
	C_4	9.09%	13.64%	40.91%	36.36%	23	0.08

We can see – with statistical significance in row three – that you are very unlikely to see the upward trending C_1 after C_3 and very likely to see the downward trending C_4 after C_3. Looking at row

four, we see that you are most likely to see noise or further downward trends following the downward trending C_4.

Trading Model Now we incorporate this insight into a trading methodology. In our approach, we iterate through the SRVs in the time series and classify them into a certain cluster using our K-Means model. The K-Means model is trained on the first 70% of the data and used to classify the remaining 30%. Next, we use our predicted label at t along with the transition matrix computed earlier to determine the most likely cluster to appear at $t + 1$. If the most likely cluster is noise, we pick the second most likely cluster as there is no action worth taking if noise is expected. If our hypothesized future cluster has a negative total return, we are short for the duration of the SRV. In all other cases, we are long. The results on the 30% section of test data are seen below, along with a vanilla buy and hold strategy shown in blue. The green dots signify taking long positions and the red dots signify taking short positions.

Fig 1.1 GBTC Trading Performance Comparison

As seen in Fig 1.1, the strategy of SRV clustering outperforms buy and hold in the 30% test period in our study period. It performs very well in bearish periods by making key short positions and follows buy and hold in bullish periods. This is consistent with our design of the trading model.

AR Comparison Now we compare our strategy with a traditional AR-model, proving that it is more effective at extracting information from the past. We use the exact same study period from before with the same 70-30% split of train and test. After plotting ACF and PACF charts, we determine that AR(1) is the most appropriate choice. We use the following formula:

$$\widehat{S_t} = \beta_1 S_{t-1} + \beta_0$$

After fitting the AR(1) model on the first 70% of the data, we make predictions on the remaining 30% of the data. Then, these predictions are incorporated into a trading methodology similar to before. If our AR model predicts a negative return for the next day, we are short for that next day. In all other cases, we are long. The results for the 30% test section are plotted below along with buy and hold, and our momentum clustering strategy:

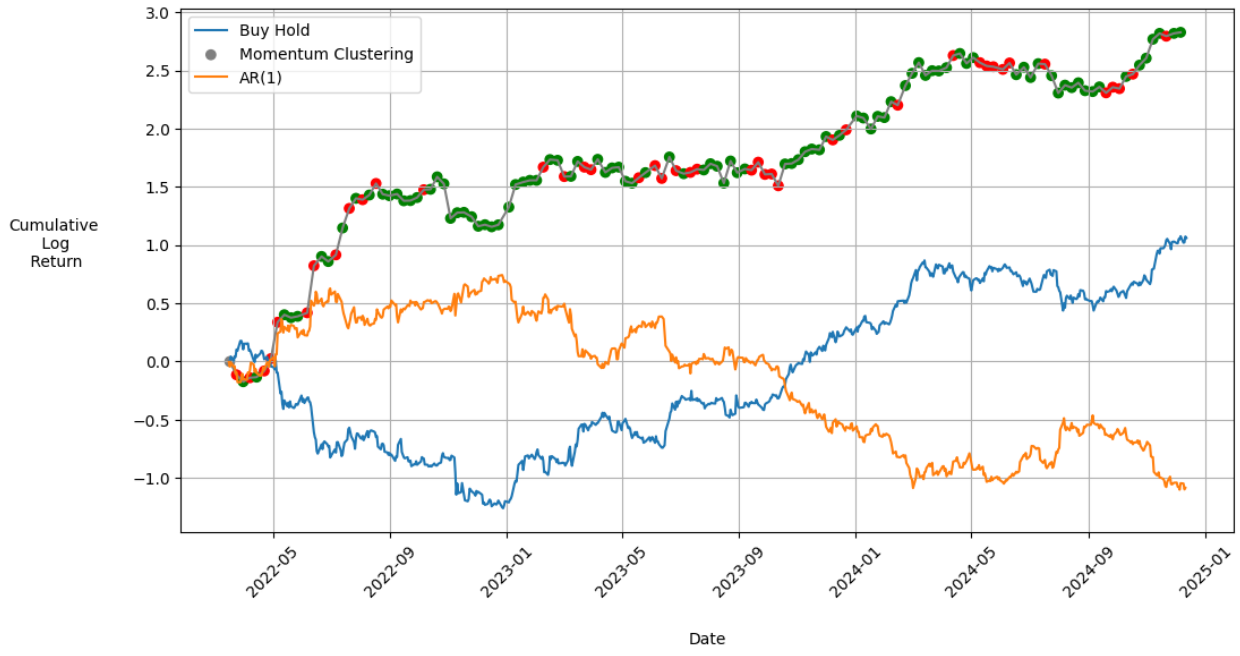

Fig 1.2 GBTC Trading Performance Comparison with AR(1)

From Fig 1.2 we can see that AR(1) primarily expects a negative return in the future, every day. Thus, following our strategy, we end up being short about 98% of the time and our cumulative returns look opposite to buy and hold. The original momentum clustering strategy still outperforms the rest. This is an encouraging result because it indicates that we can extract more information from the past than a traditional AR model and still beat buy and hold.

SRV Directionality Now we analyze the different directions an SRV can point and what that says about the directionality of the SRV right after it. We begin by first using our entire dataset and removing our train/test split. Then the SRVs are converted into unit vectors. Although it is likely that magnitude plays a role in analysis, this section analyzes purely the directionality. The tool used to analyze directionality in SRVs is the dot product. It can give a rough but practical number that shows where one SRV points relative to another. Note that the dot products will always be between -1 and 1, since we are dealing with unit vectors. We begin by calculating the dot product of every adjacent pair of SRVs at t and $t+1$. Next, we come up with a quantitative measure that tells us how closely an SRV represents perfectly positive or perfectly negative momentum. This is done by first creating examples of SRVs that represent these scenarios. For example, if we look at SRVs of length five, we will have the following "perfect" SRVs:

$$Perfect\ Positive = \begin{vmatrix} 1/\sqrt{5} \\ 1/\sqrt{5} \\ 1/\sqrt{5} \\ 1/\sqrt{5} \\ 1/\sqrt{5} \end{vmatrix}, \qquad Perfect\ Negative = \begin{vmatrix} -1/\sqrt{5} \\ -1/\sqrt{5} \\ -1/\sqrt{5} \\ -1/\sqrt{5} \\ -1/\sqrt{5} \end{vmatrix}$$

These examples are determined by maximizing or minimizing each value in the unit vectors. In a unit circle, the "most positive" direction you can go is $(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$. This same logic is applied for higher dimensions. Next, we calculate the dot product of each SRV with each one of these perfect examples. If

an SRV in our data has a dot product of 0.95 with our perfectly positive example, we understand it to have largely positive momentum over those days. The same logic applies to the negative example. Now that we have some measures, we look at some plots. We start off by looking at the SRVs with the largest similarities to our perfect negative example. These would correspond to 5-day periods where GBTC largely trended downward:

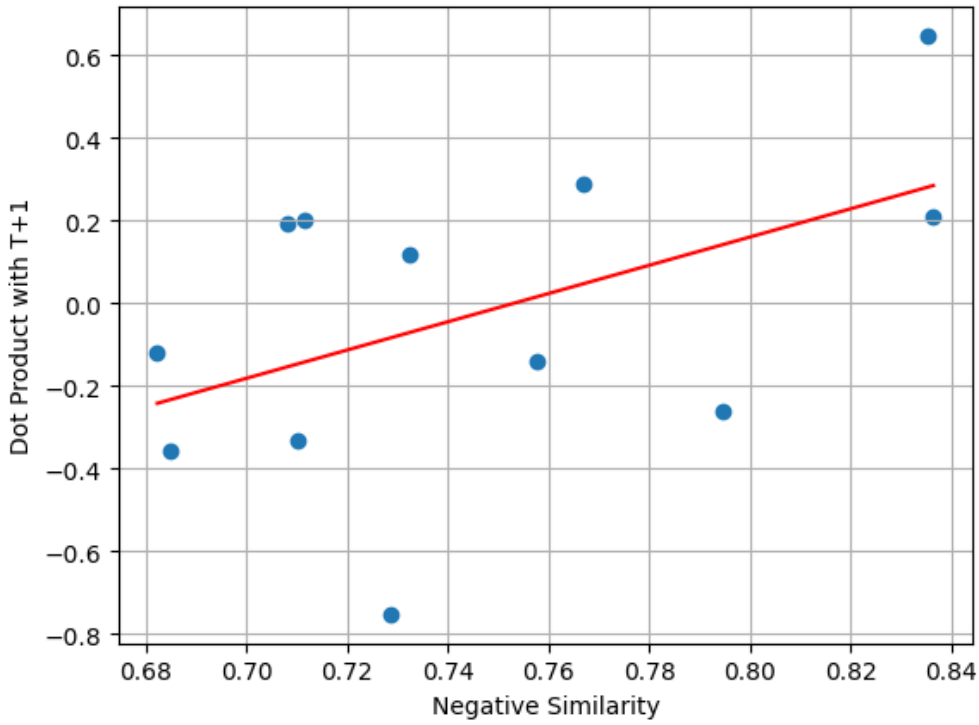

Fig 1.3 GBTC Largest Negative-Similar SRVs

This chart indicates a moderately strong positive association between perfect negative similarity of an SRV and the dot product of the SRV appearing right after. In other words, the more closely an SRV represents perfect negative momentum, the more likely that this sentiment will continue with the SRV at $t + 1$. This is a useful result as it shows that the more negative momentum we observe in GBTC, the more likely it will continue for another five days.

SRV Volatility Prediction In the implementation of our trading model, we simply considered the sign of the net return of our predicted SRV when informing us of our trading decisions. Now we

investigate whether the most likely cluster predicted for t has a similar volatility to the actual SRV witnessed at t. In other words, will the predicted cluster representing the next 5 days of returns have the same spread as the actual returns observed in the next 5 days. This is of great interest to those who construct positions based on volatility rather than directionality. To do this, we simply compare the standard deviation of the elements in our predicted SRV and observed SRV at t. We perform this calculation for our AR(1) predictions as well.

$$Let\ G_t\ represent\ the\ observed\ SRV\ of\ GBTC\ at\ time\ t$$

$$Let\ C_t\ represent\ the\ cluster\ center\ predicted\ for\ time\ t$$

$$Let\ A_t\ represent\ the\ predictions\ of\ AR(1), compiled\ into\ an\ SRV\ at\ time\ t$$

$$G_t = \begin{vmatrix} S_t \\ S_{t+1} \\ S_{t+2} \\ S_{t+3} \\ S_{t+4} \end{vmatrix}, C_t = \begin{vmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{vmatrix}, A_t = \begin{vmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{vmatrix}$$

$$Let\ V_t(C) represent\ the\ volatility\ error\ for\ momentum\ clustering\ at\ time\ t$$

$$Let\ V_t(A)\ represent\ the\ volatility\ error\ for\ AR(1)\ at\ time\ t$$

$$V_t(C) = |SD(G_t) - SD(C_t)|$$

$$V_t(A) = |SD(G_t) - SD(A_t)|$$

We can now plot $V_t(C)$ and $V_t(A)$ over all SRVs in our test period, observing the accuracy of out-of-sample volatility estimation for both methods.[4]

[4] When comparing volatility estimations, one can use ARMA-GARCH instead of AR. However, we obtained similar results and hence decided to continue with AR(1) for consistency

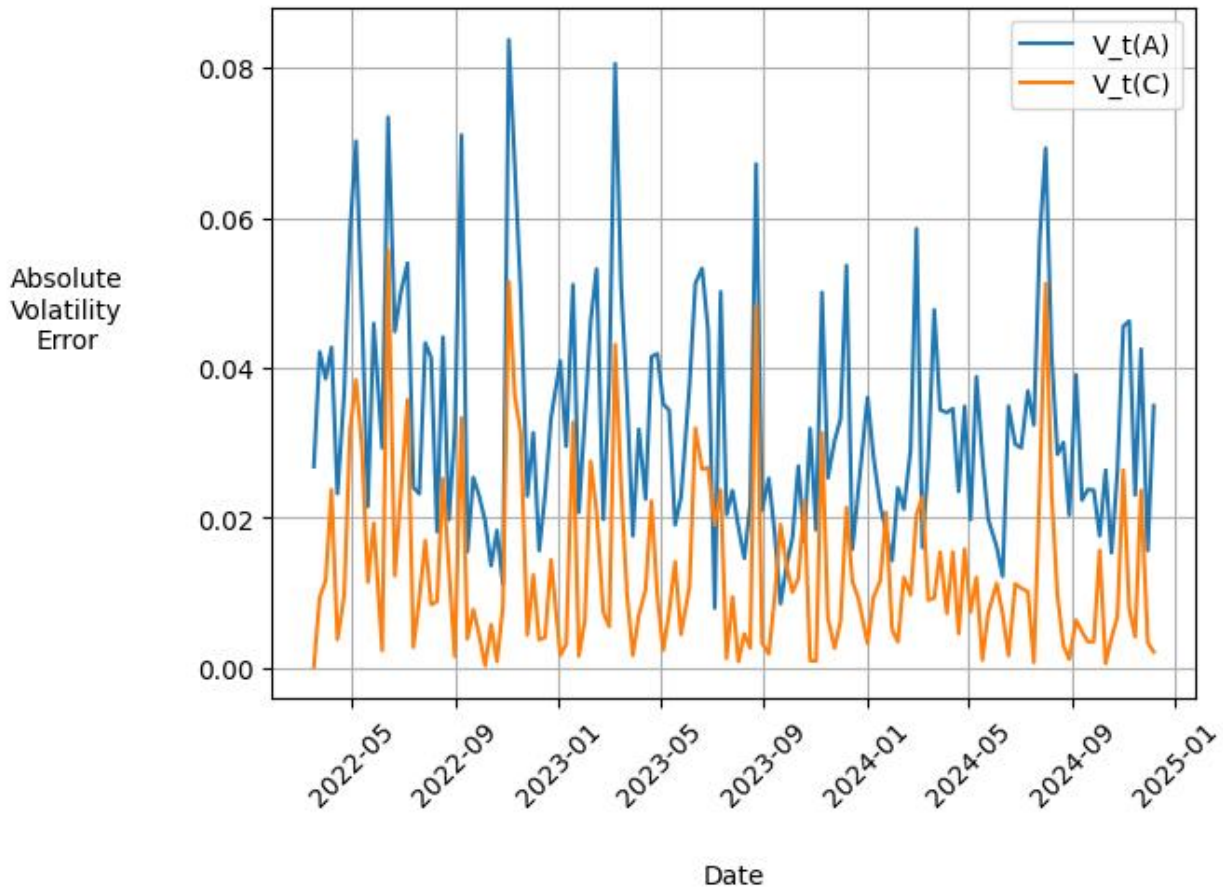

Fig 1.4 GBTC Volatility Estimation

From the chart, we see that the momentum clustering method consistently provides lower volatility error as compared to AR(1) predictions. There seem to be events where both AR(1) and momentum clustering have a sharp increase in error, but as a whole we see better estimation through momentum clustering.

Other Assets All results provided in this section use the study period ranging from 2/1/2022 to 2/1/2025, all at a daily resolution. Parameters of K-Means are tuned as needed, particularly the number of clusters. Results of our trading methodology on USO can be seen below. USO is a publicly traded ETF

that tracks oil.

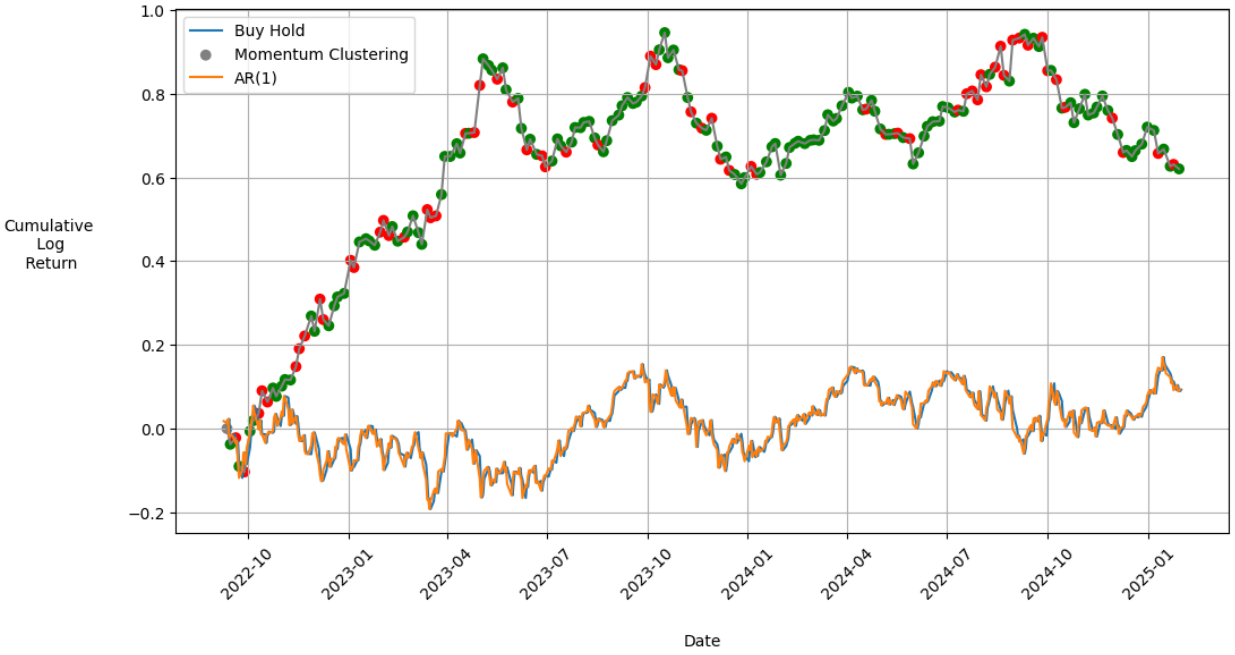

Fig 2.1 USO Trading Model Performance

Results of our trading methodology on WEAT can be seen below. WEAT is a publicly traded ETF that tracks wheat.

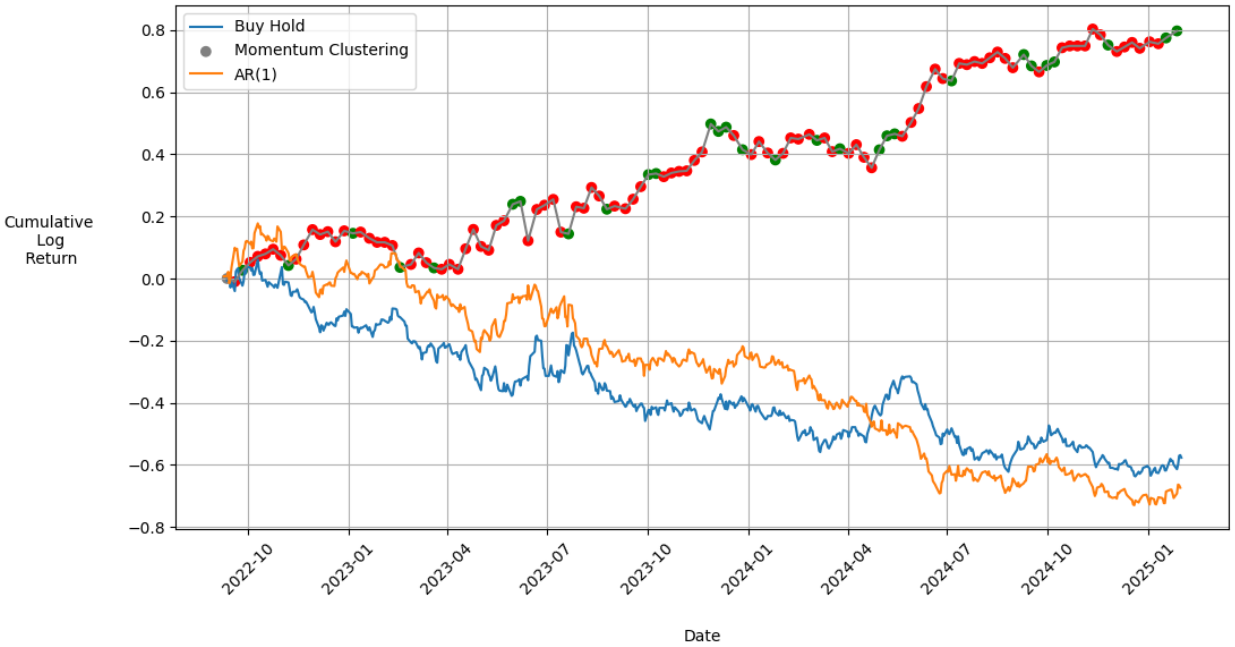

Fig A.3 WEAT Trading Model Performance

Results of our trading methodology on CORN can be seen below. CORN is a publicly traded ETF that tracks corn.

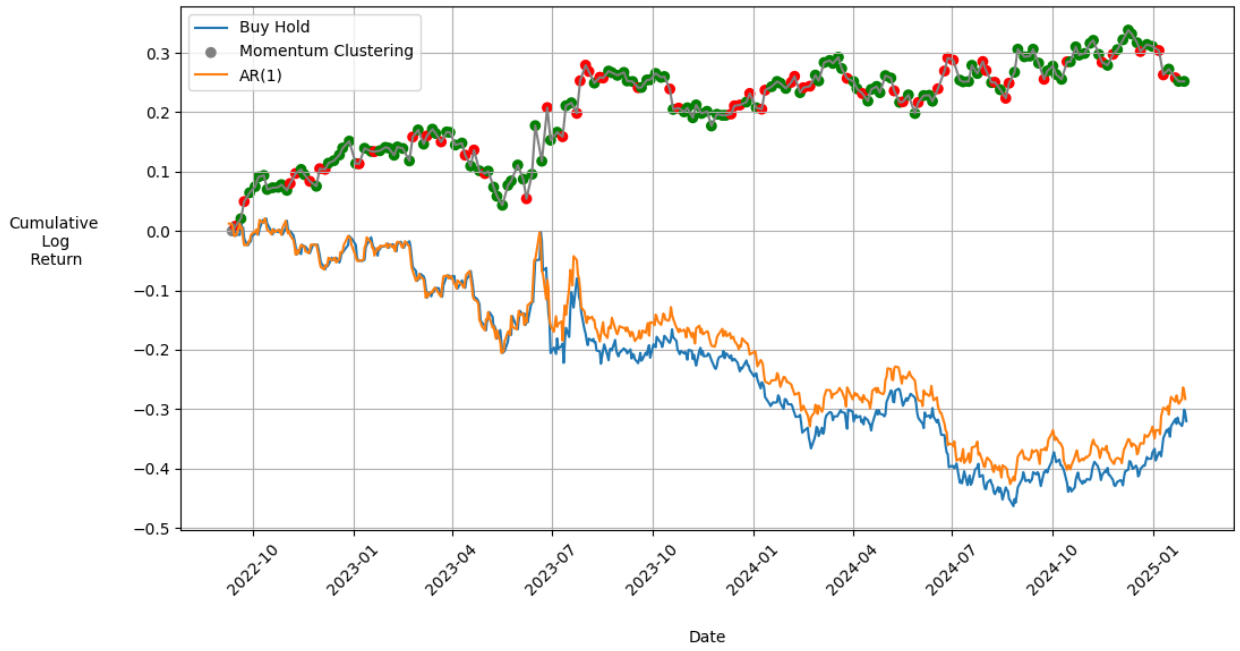

Fig A.4 CORN Trading Model Performance

Results of our trading methodology on GLD can be seen below. GLD is a publicly traded ETF that tracks gold.

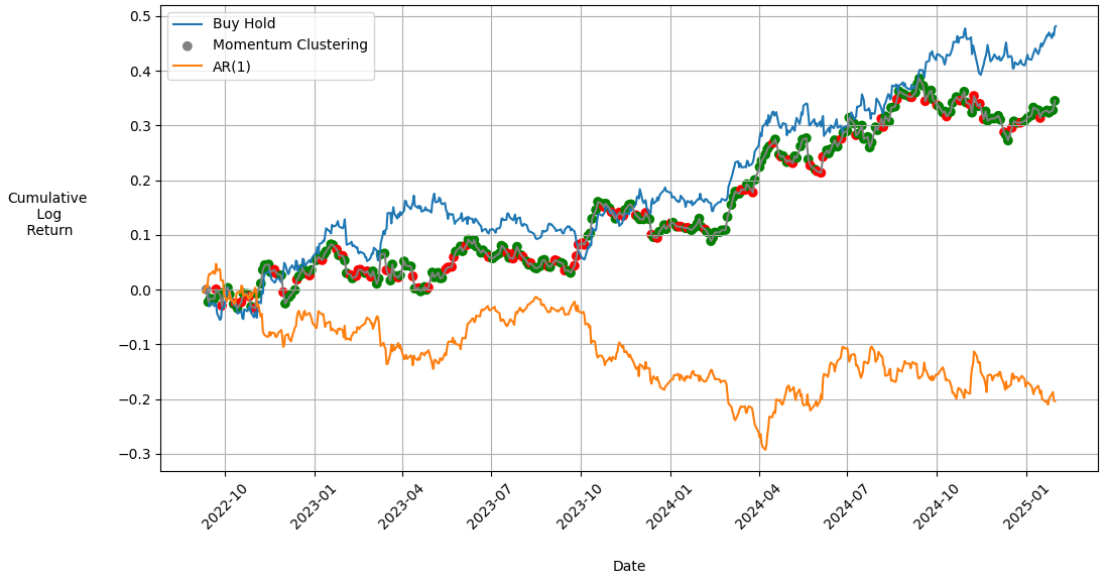

Fig A.5 GLD Trading Model Performance

Results of our trading methodology on SOYB can be seen below. SOYB is a publicly traded ETF that tracks soybeans

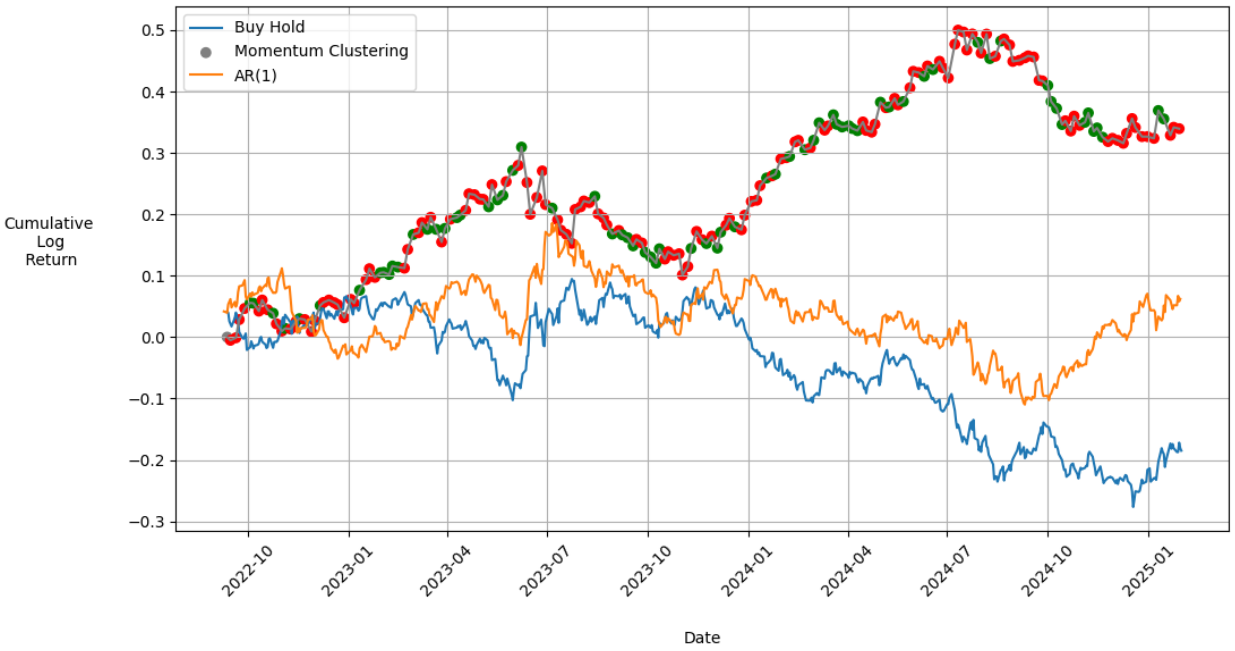

Fig A.6 SOYB Trading Model Performance

As seen in the charts, the trading methodology is quite successful for the provided assets, particularly during drawdowns. In practice, it can be used as a signal to deviate from a buy and hold strategy, especially when the appearance of a negative cluster is very likely.

Conclusion

In this paper, we developed a new methodology to analyze information in a financial time series: momentum vector clustering. We partitioned a time series into vectors and clustered the vectors into families of momentum. These results could be used in a trading model to outperform buy and hold and AR(1) predictions on a Bitcoin ETF and other assets. Our results suggest that some financial time series can be viewed as a linear combination of distinct momentum vectors along with some assumed

noise. Our methodology proved to be successful on assets where there is minimal dissolution of external information, such as cryptocurrency and commodity ETFs.

Appendix

Significance of SRV Length The length of our SRVs carries a lot of significance. It is our hypothesis on the duration of momentum regimes that exist within the time series. For example, if we use three as the length of our SRVs, we are assuming that most different types of momentum in our time series last for three days. Another way to interpret this is the time it takes for momentum to dissolve within participants in our financial time series. For example, if most periods of upward momentum last for three days, and most periods of downward momentum also lasts for three days, three is a safe estimate for the length of our SRVs.

Starting Point of SRV Matrix In our methodology, it is also important to consider where to start our matrix of SRVs. We obviously want to partition the entire time series, so we must start early enough so that we have a sufficient amount of data. If we want 4 years of data, we must start building our matrix four years before the present. However, given our definition of SRVs, it is obvious that shifting our matrix one day forward or one day backward can dramatically change the shape of the vectors in our matrix. For example, let S be a hypothetical time series:

$$Let\ S = \{0.1, 0.9, -0.3, 0.4, 0.5, -0.6, 0.1, -0.4, 0.9 \ldots S_n\}$$

$$Let\ desired\ SRV\ length = 3$$

$$Some\ possible\ M_t$$

$$\mathbf{M_1} = \begin{bmatrix} 0.1 & 0.4 & 0.1 & S_{10} \\ 0.9 & 0.5 & -0.4 & S_{11} \\ -0.3 & -0.6 & 0.9 & \ldots \end{bmatrix}, \mathbf{M_2} = \begin{bmatrix} 0.9 & 0.5 & -0.4 & S_{11} \\ -0.3 & -0.6 & 0.9 & S_{12} \\ 0.4 & 0.1 & S_{10} & \ldots \end{bmatrix}, \boldsymbol{M_3} = \begin{bmatrix} -0.3 & -0.6 & 0.9 & S_{12} \\ 0.4 & 0.1 & S_{10} & S_{13} \\ 0.5 & -0.4 & S_{11} & \ldots \end{bmatrix}$$

We can see that just starting our matrix one or two days ahead can change the shape of each SRV in the matrix. To address this fact, we must remember that we are ultimately clustering the columns in our SRV matrix. If our assumption that three-day momentum families exist within this time series is truly correct,

shifting the matrix by a few days will simply change the location of the cluster center of each family in our results. This is an effect that we are okay with in this paper.

Market Adjusted Log Returns In this paper, we calculate market adjusted returns in the following manner, using the S&P 500 as the market of choice since all of our tested assets are listed in the United States.

$$S_t = Price\ of\ instrument\ at\ day\ t$$

$$R_t = \ln\left(\frac{S_t}{S_{t-1}}\right)$$

$$M_t = \log return\ of\ S\&P\ 500\ at\ day\ t$$

$$B = \frac{Cov(R_t, M_t)}{Var(M_t)}$$

$$A_t = Market\ adjusted\ log\ return\ of\ instrument$$

$$A_t = R_t - (B * M_t)$$

K-Means Implementation In this paper, K-Means was implemented through the sci-kit learn library in Python. The parameters used for GBTC are as follows:

Parameter	Value
N_clusters	4
N_init	20
Algorithm	Elkan
Random_state	20
Max_iter	700

Fig A.1 GBTC K-Means hyperparameters

These parameters can be varied when studying different assets or time periods. Time periods with minimal characteristic families of momentum should utilize a smaller number of clusters. Assets that are hypothesized to show more complex SRVs should use a higher number for Max_iter.