Analysis of Continuous Datasets Using Tokenized Correlation Windows in an Apache Spark Map-Reduce Framework

Jason Orender, CS 724 High Performance Computing, Old Dominion University, Norfolk, VA USA, joren001@odu.edu

***Abstract*—Technical analysis of securities has long been used as a method to predict future price movements based on past patterns of behavior. There are several classic patterns such as rising and falling channels, and related concepts like "resistance" or "support" pricing levels. While the merits of charting these specific patterns are open to debate, there is considerable belief that past behavior can, in some specific circumstances, predict a future price movement absent a fundamental shift in the nature of pricing the asset. This is in direct conflict with theories of efficient markets and a large body of modern portfolio theory, which generally holds that active management based on technical indicators cannot outperform the market over the long-term[1]. This project is an attempt to explore the use the parallel computational abilities of Apache Spark to discover technical indicators that display predictive properties for a specific commodity. To that end, cryptocurrency datasets were chosen for the analysis on the assumption that much of the movement in cryptocurrency prices is due to technical factors alone; that is to say, most of the buying and selling of these commodities is assumed to be based on perceptions of past price movements driving a general supply and demand dynamic[3]. While the entertainment value of predicting cryptocurrency price movements is substantial and provides a useful backdrop for the analysis, the primary objective of the project was to use a map-reduce framework to create a generalized correlation engine that will efficiently distill useful traits from a big-data set of arbitrary size.**

I. Introduction

Analyzing securities using technical analysis tools has been tried for hundreds of years[2], with mixed success and much disagreement regarding its effectiveness[1]. This project is intended to be a case study in automated technical analysis using map-reduce techniques, specifically as applied in an Apache Spark implementation. A very limited data set was selected to keep the analysis of the results simple and straightforward while keeping the majority of the focus on the techniques used in the analysis. These techniques are intended to be open-ended and applicable to a wide variety of data sets.

The data is partitioned between a "training" and a "testing" set to facilitate objective evaluation of the effectiveness of the algorithm. The training set is used to generate a correlation pair that is in turn used to make a future price prediction; the testing data set is sequestered and not used until the final testing phase. An explanation of correlation pairs will be covered in part III (Algorithm Explanation).

The results will take the form of a set of confusion matrices and a list of correlation pairs. The confusion matrices will provide a clear indication of both the applicability of the correlation pair to the training data as well as the predictive value of the correlation pair to the testing data.

II. Design

A. The Data Set

The input data is a comma-separated value (csv) file organized sequentially by date with seven additional fields: volume, transactions, market cap, closing price, exchange volume, number of coins generated, and the amount of fees charged. The changes in these fields are considered purely technical indicators since they are not directly related to the value proposition of the currency that they represent; the value of a currency is generally regarded as its ability to be a store of value and a medium of exchange[3]. While there are certain long-term behaviors of the fields in this data set that arguably might affect both of these value propositions, the assumption operative in this analysis is that over the short term (i.e. less than 90 days), none of these attributes will directly affect the underlying value of the cryptocurrency being analyzed.

Two cryptocurrencies were selected to perform the analysis on: Bitcoin (BTC) and Monero (XMR). These were selected because they cover a very similar time span of data and show somewhat different behaviors when run through the analysis tools developed; their similarities and differences make them ideal candidates to provide interesting platforms for discussion.

The data used in this analysis was originally downloaded from "coinmarketcap.com" and covers the time period from May 1st, 2013 through March 5th, 2018.

B. The Code

The code is written in Python and used in an Apache Spark Hadoop Distributed File System (HDFS) environment. Since Python is currently a widely accepted language for machine-learning applications and contains several native language devices and structures that interact synergistically with the Apache Spark framework (e.g. dictionaries, lists, tuples), it seemed to be a straightforward choice for the development language.

There are five general sections of the code: 1) Read in and partition the data, 2) create a list of traits and map the data to those traits, 3) tabulate the frequency with which pairs of traits

are coincident with successful/unsuccessful outcomes and screen out those that failed to meet a minimum threshold for retention, 4) create a list of trait pairs that both meet a minimum standard of frequency and correlate with either successful or unsuccessful outcomes, and 5) create a confusion matrix that expresses the predictive abilities of the correlation pairs selected for the final list.

III. Algorithm Explanation

The general algorithm used was inspired by the market basket analysis typically performed in the retail sector to collocate items that are frequently bought together[4] with the intention of maximizing sales. This algorithm turns that analysis on its head by supposing that if a pairing of traits can be correlated to a specific outcome, a recurrence of those traits in the future might have some predictive value for accurately forecasting an as-yet undetermined outcome.

To put this in terms of the market basket analysis, assuming that beer is known to often be purchased with milk and diapers, a relevant question might be "will the customer buy beer?" If both milk and diapers are already in the shopper's basket, then the chances that the customer will buy beer as well might dramatically increase. To approach the analysis in a different way, one might ask "is the customer male?" Supposing that a beer and diaper purchase pair correlates primarily to male customers, a purchase of that pair might also dramatically increase the chances of accurately identifying this outcome.

In terms of financial analysis, an outcome that is desired might be "a rise in the price of a commodity by at least 20% over the next month." Assembling a list of pairs of traits that correlate with an affirmative result of this type might dramatically improve the chances of accurately predicting future price movements.

A. *Defining a trait*

Determining what constitutes a "trait" is somewhat problematic when describing a commodity since many of the descriptive attributes are continuous variables like price, volume, and market cap. This was dealt with by normalizing the variables and then assigning bins to attributes within a certain range. The best results with the data set used seemed to occur when the attribute was transformed into a fractional gain from 30 days prior and 60 days prior, then divided into 30 equal sized bins distributed between the minimum and the maximum recorded levels for the values. As an example, if attribute "B" is currently at 1300, and the level 30 days prior was 1000, the fractional gain would be 0.3; if the maximum fractional gain was 0.5 and the minimum fractional gain was negative 0.2 (that is, a 20% loss), the bin size would be : (0.5 – (-0.2))/30 = 0.023, and the bin that the current level of 1300 for attribute B would occupy is: int((0.3 – (-.2))/0.023) = 21. Additionally, different base levels were assigned to each attribute so that the traits could be differentiated. If a base level of 1000 was assigned to attribute A, 2000 to attribute B, 3000 to attribute C, and so on, the trait assigned to attribute B in the example above would be: 2000 + 21 = 2021. In this way, the traits can be manipulated without losing track of which trait belongs to which attribute. This is the "tokenized correlation window" referred to in the title of the paper (fig. 1 explains why this is labeled a "window").

At the end of stage 2 of the code, this transformation will have been performed on every line in the input data file.

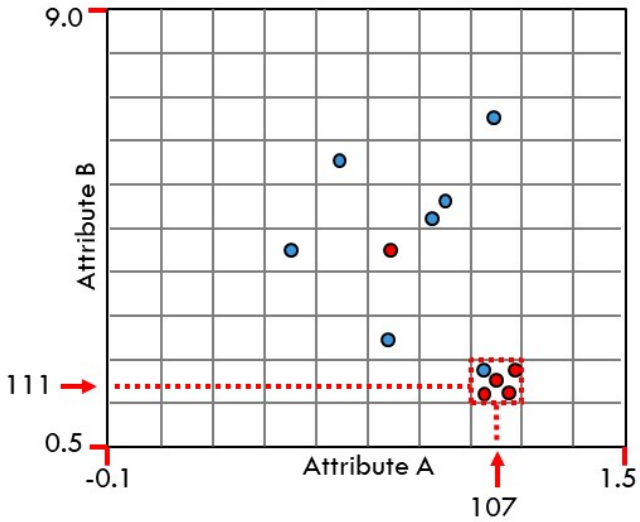

Fig. 1. Visualizing the trait pair. If "successful" trait pairs are coded as red and "unsuccessful" trait pairs coded as blue, the successful data points can be thought of as clustering in the window defined by the trait pair (107, 111).

B. *Pair tabulation*

At the end of stage 2, the data consisted of a date and a list of attributes. The next stage transforms the list of attributes into a list of pairs and then tabulates how often each distinct pair occurs in a line of data that is deemed either successful or unsuccessful.

The number of pairs produced is essentially an arithmetic expansion in (n-1) since the order of the values in the pair is irrelevant and an attribute is not allowed to pair with itself. The list of attributes consisting of [1, 2, 3] would therefore produce a complete list of pairs equivalent to [(1,2), (1,3), (2,3)]. This is given by the following expression (where p is the number of pairs and n is the number of attributes):

$$p = n*(n-1)/2 \quad (1)$$

As a result, the number of pairs grows asymptotically to $O(n^2)$ and can greatly increase the run time of the analysis when a large number of attributes is used. With seven attributes, the run time on the cluster that was used was still a very reasonable 23 seconds.

Tabulating the coincidence of trait pairs with either a successful or unsuccessful result was simply a matter of mapping each trait pair that appeared on a successful day to its own <key,value> pair where the key is the trait pair and the value is one (for example: < (2021, 1011), 1 >), and then performing a reduce to sum up the values and yield the number of occurrences for each trait pair. This procedure was repeated for those pairs meeting the criteria for unsuccessful outcomes also.

Those trait pairs that exceeded a minimum threshold (10 in this case) were retained and passed on to the next stage.

C. Creating the model

The model is simply a list of traits with a demonstrated correlation to an outcome that is expected to have some predictive properties regarding that specific outcome.

To assemble the model from a list of pairs and the frequencies with which they are coincident with specific outcomes, the frequencies are first compared with the overall frequency of that pair within the data set. A trait pair that appears on every day, for example, will necessarily appear on every day with the specific outcome that was screened, but will correspondingly have no predictive value regarding whether that outcome will occur. To eliminate these, the following probability was calculated (where f_o is the frequency correlated to a specific outcome, and f_T is the total frequency in the data set):

$$P_o = f_o/f_T \tag{2}$$

All traits with a P_o value of less than 0.9 were eliminated, substantially reducing the number of pairs to examine.

Additionally, the probability of each attribute in the pair was calculated as:

$$P_1 = f_1/f_{T1} \tag{3}$$

and

$$P_2 = f_2/f_{T2} \tag{4}$$

The elements of these equations correspond directly with those in equation (2), but only for each element of the pair. If each element of the pair was completely independent of the other, the following composite probability (P_c) would be expected to hold:

$$P_o = P_c = 1 - (1 - P_1)*(1 - P_2) \tag{5}$$

The objective of this analysis is to find pairs in which $P_o > P_c$, which would indicate that the probability of the outcome coincident with a particular trait pair is greater than what could be accounted for by either trait alone. As such, any pairs that failed to exceed the composite probability of their elements were eliminated.

At the end of this stage, the model is complete.

D. Measuring predictive value

A standard confusion matrix of the following form was used:

TABLE I. CONFUSION MATRIX

Actual	Prediction		
	NO	***Not Predicted***	***YES***
NO	True Negative	No decision	False Positive
YES	False Negative	No decision	True Positive
Rand	Random Negative		Random Positive

Fig. 2. Confusion matrix form

The "Not Predicted" column collects all lines in the data set that do not have either a pair associated with a positive outcome or a pair associated with a negative outcome. No decision is made concerning the projected outcome of these days.

The "YES" result corresponds to a rise in value over the next month, while the "NO" result corresponds to a decline in value over the next month. These could be construed as buy and sell indicators, respectively.

The random row indicates what the result would have been if the prediction had been decided by a simple coin flip. If, for instance there were a total of 10 "YES" results and 90 "NO" results in the data set, flipping a coin would have resulted "YES" being correct nominally 10% of the time. This should be compared with the "True Negative" value in the "NO" column and the "True Positive" value in the "YES" column.

IV. RESULTS

A. Monero (XMR)

The first confusion matrix discussed is for the Monero (XMR) cryptocurrency. This currency is marketed as a "privacy" currency due to its unique encrypted "ring signature" that masks both the sender and the receiver. It was almost an ideal study subject because there were very few extraordinary events that might have affected its adoption or use during the time period analyzed.

TABLE II. TRAINING CONFUSION MATRIX (XMR)

Actual	Prediction		
	NO	***Not Predicted***	***YES***
NO	143 (98%)	346	2 (3%)
YES	3 (2%)	504	77 (97%)
Rand	45.7%		54.3%

Fig. 3. Monero training confusion matrix

TABLE III. TESTING CONFUSION MATRIX (XMR)

Actual	Prediction		
	NO	***Not Predicted***	***YES***
NO	20 (59%)	85	8 (26%)
YES	14 (41%)	123	23 (74%)
Rand	41.3%		58.6%

Fig. 4. Monero testing confusion matrix

What tables II and III show is that the pairs selected correlated extremely well with the results in the training confusion matrix, and while the predictions did not perfectly foretell the results in the testing matrix, they were much more accurate than a random guess would have been. This is likely a greater reflection of the predictive power of the particular types of data fields selected, rather than a flaw in the algorithm.

B. Bitcoin (BTC)

The next confusion matrix is for the Bitcoin (BTC) cryptocurrency. This is the iconic first mover in the cryptocurrency space, and the one that most people will be familiar with. There were several events that occurred over the span of the data, and two unique events in the history of Bitcoin occurred in the testing portion of the data that had no precedent in the training portion of the data set: 1) the Bitcoin Cash (BCH) fork, and 2) the massive uptick in adoption by the general public in December of 2017. By segregating the training portion of the data set in the last 20% of the data collected (as with the Monero analysis), the following results were obtained.

TABLE IV. TRAINING CONFUSION MATRIX (BTC-1)

Actual	Prediction		
	NO	***Not Predicted***	***YES***
NO	80 (100%)	512	0 (0%)
YES	0 (0%)	755	39 (100%)
Rand	42.7%		57.3%

Fig. 5. Bitcoin training confusion matrix (#1)

TABLE V. TESTING CONFUSION MATRIX (BTC-1)

Actual	Prediction		
	NO	***Not Predicted***	***YES***
NO	8 (25%)	101	10 (71.4%)
YES	24 (75%)	206	4 (28.6%)
Rand	33.7%		66.2%

Fig. 6. Bitcoin testing confusion matrix (#1)

What is immediately noticeable from table V is that using the model significantly *decreased* the chances of successfully predicting the future price movements, even worse than if the model had simply chosen at random. This occurred even though the correlation results, as shown by the training confusion matrix in Table IV, were absolutely perfect.

If, instead, the method for choosing the testing data is altered from selecting the last 20% of the data for testing to sequestering every 5th day from the data set for testing, the results are much more comparable to the results for Monero in tables II and III.

TABLE VI. TRAINING CONFUSION MATRIX (BTC-2)

Actual	Prediction		
	NO	***Not Predicted***	***YES***
NO	82 (100%)	467	0 (0%)
YES	0 (0%)	744	75 (100%)
Rand	40.1%		59.9%

Fig. 7. Bitcoin training confusion matrix (#2)

TABLE VII. TESTING CONFUSION MATRIX (BTC-2)

Actual	Prediction		
	NO	***Not Predicted***	***YES***
NO	19 (61%)	118	2 (9%)
YES	12 (39%)	171	19 (91%)
Rand	40.8%		59.2%

Fig. 8. Bitcoin testing confusion matrix (#2)

In tables VI and VII, it is clear that the correlation was very good; in addition, the predictive value of the analysis was materially more accurate than a random choice would have provided. This method of segregating the testing data, while demonstrating that the algorithm accurately identifies correlations with predictive value, shows an inherent weakness in the practice of technical analysis. It clearly will only work if all of the relationships that were teased out in the training phase continue to exist in the testing phase and beyond.

V. CONCLUSIONS

The algorithm presented in this paper finds correlations among the fields present in a data set very efficiently, though the applicability of those correlations to trend prediction is dependent upon the nature of the fields present in the data set and their usefulness in this application. This is especially noticeable in the analysis of Bitcoin; if the relationships between the fields in the data set change meaningfully over time, the predictive value of any correlation revealed is suspect, even if it is a perfect relationship.

VI. FUTURE WORK

The advantage of doing this sort of analysis as an alternative to "black box" neural net machine learning is twofold: 1) the reasons for each choice are clearly available simply by revealing the pairs present in the model, and 2) the models should be additive, meaning that two models trained separately should be able to be combined into a single model

and perform the functions of both of the original models. Future work should illustrate these advantages and test this analysis model on standard machine learning data sets.

Additionally, a procedure for pruning data that has no predictive value should be developed. This will also allow addition of new prospective fields to be included in the predictive model as the discarded fields make more computing power available. The objective will be to move data points from the "Not Predicted" column into the predict "YES" or predict "NO" columns and also to align the results in the training confusion matrix with those in the testing confusion matrix. Correlation pairs with no predictive value should probably be discarded from the model, but using the results from the testing matrix to alter the model should be approached with caution since this might result in unintentionally biasing the model to produce results that ultimately fare well in the training and testing confusion matrices, but sacrifice generality beyond particular training and testing data sets.

It might be advantageous to adapt a cluster analysis such that if a pair exists that is close to a pair in the model, but not precisely within the window, some predictive value could be extracted from that relationship.

For large sets of data with many parameters, a hierarchical model could be developed to determine predictive value for pairs of pairs (four traits represented), or pairs of pairs of pairs (eight traits represented). Identifying pairs of pairs with common traits, effectively chaining two pairs together with a common element, could have some value as well.

REFERENCES

[1] B. Malkiel, "A random walk down Wall Street: the time-tested strategy for successful investing," New York: W.W. Norton; 2003.

[2] I. Singer, M. Kayserling, "Penso, Joseph" in Jewish Encyclopedia vol. IX. New York: Funk & Wagnalls Company, 1906, p.589.

[3] F. Coppola, "The Illogical Value Proposition Of Bitcoin," in Forbes Magazine [Online], January 1, 2018, Available: https://www.forbes.com/sites/francescoppola/2018/01/01/the-illogical-value-proposition-of-bitcoin/#29d0de346218.

[4] Loraine Charlet Annie M.C. and Ashok Kumar D, "Frequent Item set mining for Market Basket Data using K-Apriori algorithm" , International Journal of Computational Intelligence and Informatics, Volume 1, No. 1, 2011, pp.14-18.