Encoding Stock Charts Patterns for Technical Analysis

Thitaree Tanprasert
Harvey Mudd College
Claremont, CA 91711
ttanprasert@hmc.edu

Julianne Lin
Harvey Mudd College
Claremont, CA 91711
jullin@hmc.edu

May 7, 2019

Abstract

Technical analysis is a process in which stock traders analyze time-series stock charts data, visually, and detecting pre-defined patterns that would help them forecast future stock price. In this project, we aim to create a tool that detects these patterns, automatically, to improve the effectiveness of technical analysis. We propose an algorithm that differs from any previous research on this topic. Instead of looking at the raw data as is, we develop an algorithm which describes each dip and peak in the data in terms of rotation angle, scaling factors, and shearing factors with respect to a normalized vector. Then, we hard-code the characteristics of specific patterns parameterized by four parameters. We tested our system on SP500 data and encoded the cup-and-handle and double-bottom patterns. We found that our tool is much more effective than the baseline system, which utilizes self-organizing maps to create pattern templates from raw data. Expert users can adjust the tool by setting the parameters based on domain-specific knowledge. The tool can also be extended, easily and intuitively, to include other patterns or to work with other time-series data without having to annotate training data.

1 Introduction

Predicting stock trends is a widely explored problem in the financial sector. Through visual analysis of stock data trends, experts can predict future stock prices. This requires domain-specific intuition for detecting financially-significant

visual patterns in the time-series data. A tool that could aid users in accurately identifying these human-recognizable patterns would enable traders to make analyses and predictions quickly to keep up with stock trends that vary daily.

In this project, we performed an analysis on time-series stock data to extract human-recognizable patterns by transforming the time-series data into series of vectors and performing geometric analysis. We developed a scheme to model data patterns using vector transformation. More specifically, we developed an algorithm to describe each dip and peak in the data in terms of rotation, scaling, and shearing of a normalized V-shaped vector - a shape formed by alternation of local maxima and minima that occurs repeatedly in any time-series data. By extracting and analyzing these V-shaped structures in the data, we gained an insight at a higher level than merely numerical patterns, which allowed us to perform pattern detection based on these extracted features. Through this method, we hope to demonstrate the utility of transforming time series data to vectors and the adaptability of this scheme to any type of pattern recognition problem for time series data.

In particular, we developed a tool which optimizes the detection of human-recognizable patterns. With only a few parameters to specify the user can target and easily modify pattern detection. Different from other existing rule-matching algorithms, our time-series data analysis is based on defining the data as vectors. This enables a high degree of manipulation of the pattern matching to any degree of specificity desired. Founded on universal parameters to define the time-series data, this type of data-processing can be generalized to analyze time-series data in any field not limited to only stock data.

This paper consists of 7 sections, including this introduction. In the second section, we summarize previous research that has explored methods of stock data analysis, in particular, stock charts pattern detection research that utilize machine learning techniques. In the third section, we describe the methodology of pattern detection. In the fourth section, we describe our experimentation process, consisting of the dataset we used and the baseline system that we implemented to compare with our proposed system. In the fifth section, we discuss the results of our proposed system in comparison with the results of the baseline system. In the sixth section, we make recommendations for future work. Finally, we conclude our achievements and suggest a promising direction to expand the project in the seventh section.

2 Literature Review

Similar work done previously in stock pattern recognition reply largely on neural networks. Research in computerized stock pattern recognition has turned towards neural networks in an effort to minimize the need for high levels of participation from domain experts. Some explored models have included recurrent neural networks, multi-layered perceptrons, and self-organizing maps. [1] [2] [3] Kamijo and Tanigawa (2012) trained a recurrent neural net model to recognize a triangle pattern in candlestick charts stock data. Recurrent neural network is the most

intuitive model for stock data analysis because it is designed to capture contextual, temporal transition. The model obtained a 98.3% accuracy on test data, which is 6.25% of the whole dataset that they used.

The only disadvantage of this model is that it needs annotated and labeled data for training the model. Moreover, the training process is time-consuming when we want to expand the system to detect new patterns. Similar problems persist with the multi-layered perceptron and self-organizing map approaches. On top of needing labeled data, the self-organizing map method also requires initializing pattern templates. For both requirements, the initiation of these models require human pattern recognition as one of the first steps, which is the problem our system tries to mitigate. However, all approaches employ similar data preprocessing techniques, including data smoothing with moving average and price normalization, which could be beneficial to unsupervised learning approach as well.

3 Pattern Detection Methodology

The pattern detection methodology section will consist of four main subsections: data smoothing, vector extraction, transformation parameters computation, starting vector candidates identification, and pattern encoding.

3.1 Data smoothing

In stock charts data, there are a lot of small oscillations, which are usually neglected by humans as we look for visual patterns at a larger scale of data. Our system would be more robust if we could also remove these oscillations before we begin the data transformation and analysis process. We decide to smooth the data with moving average method, since it is easy to implement, and it works, effectively, in most cases.

The only important we need to specify for moving average method is the size of the moving window. We first experimented with using the average length between consecutive local maxima as the window size. We found that it effectively removes most of the oscillation. Moreover, we found that by applying this method twice - the second time on already-smoothened data - we get data points which are surprisingly accurate to how we view the overall structure of the data, as shown in Figure 1. So, we decided to perform moving average twice before we start extracting the V-shaped vectors.

It should be noted that there are many other data smoothing techniques which are usually used with stock charts data, such as exponential average and random walk. However, we did not explore them after we found that simple moving average works well enough for our purpose.

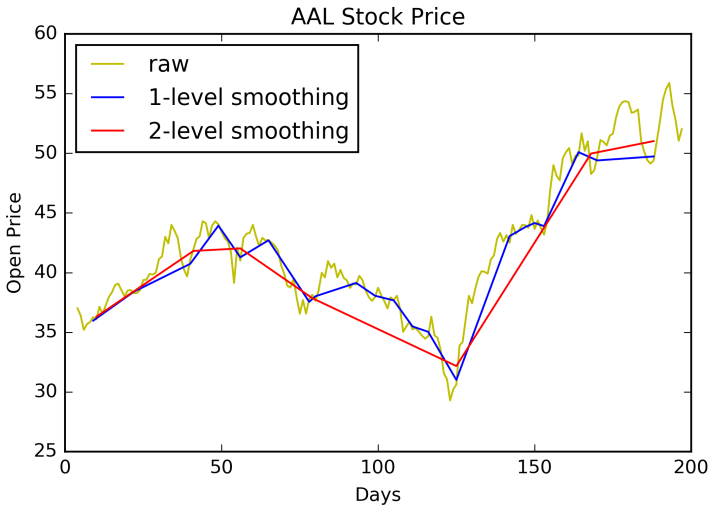

Figure 1: A comparison between raw data, once-smoothened data, and twice-smoothened data.

3.2 Vector extraction

The input to the system is discrete time-series data, where each timestamp corresponds to a value and the timestamps are equally spaced. In case of stock data, the timestamps are the dates in which the stock price is recorded. We would like to transform this input data into a sequence of V-shape vectors, starting from the first local maximum in the data.

In order to do this, we retrieve the x, y-coordinates of all local minima and maxima. The data between two consecutive local maxima constitute a single V vector. Each V vector will be represented by the two local maxima at the end and one local minimum that is the dip of the V-shaped vector.

$$V = \begin{bmatrix} x_m - x_l & x_r - x_m \\ y_l - y_m & y_r - y_m \end{bmatrix},$$

where (x_l, y_l) = timestamp and corresponding value of the left end of the vector

(x_m, y_m) = timestamp and corresponding of the local minimum

(x_r, y_r) = timestamp and corresponding of the right end of the vector

3.2.1 Transformation parameters computation

Transformation function computation consists of two steps. The first one is computing transformation matrix. That is, for a given pair of vectors V extracted from the data, we want to find a 2x2-matrix A such that $V = AH$. We can

easily see that

$$A = VH^{-1} = \frac{1}{2}\begin{bmatrix} -V_{11} + V_{12} & V_{11} + V_{12} \\ -V_{21} + V_{22} & V_{21} + V_{22} \end{bmatrix}$$

The next step is transformation matrix decomposition, where we will decompose A into a product of three 2x2 matrices called R, S and SH:

- R corresponds to rotation, defined by an angle . To rotate by angle θ, counterclockwise, we multiply the vector by the matrix

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

- S corresponds to scaling, defined by two scalars s_l and s_r for scaling the magnitude of the left and right vectors, separately. That is we would like to find S such that

$$S * H = \begin{bmatrix} -s_l & s_r \\ s_l & s_r \end{bmatrix}$$

  Therefore, we get

$$S = \frac{1}{2}\begin{bmatrix} -s_l + s_r & s_l + s_r \\ -s_l + s_r & s_l + s_r \end{bmatrix}$$

- SH corresponds to shearing parallel to the x-axis, defined by a scalar m. To shear a plane, we multiply the vector by the matrix

$$SH = \begin{bmatrix} 1 & m \\ 0 & 1 \end{bmatrix}$$

We can decompose A by calculating the product of the matrices listed above

$$\begin{aligned} A &= R * S * SH \\ &= \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} * \begin{bmatrix} -s_l + s_r & s_l + s_r \\ -s_l + s_r & s_l + s_r \end{bmatrix} * \begin{bmatrix} 1 & m \\ 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} s_l \cos\theta & ms_l \cos\theta - s_r \sin\theta \\ s_l \sin\theta & ms_l \sin\theta + s_r \cos\theta \end{bmatrix} \end{aligned}$$

From this, we can calculate the parameters θ, s_l, s_r and m as follow:

$$\begin{aligned} \theta &= \arctan(A_{21}/A_{11}) \\ s_l &= A_{11}/\cos\theta \\ s_r &= A_{22} - A_{12}\sin\theta \\ m &= (A_{12} + s_r \sin\theta)/s_l \cos\theta \end{aligned}$$

Each V-shaped vectors could therefore be represented by a tuple (θ, s_l, s_r, m). This makes it easier for us to categorize changes in the stock data, since we can do it with respect with each of the three parameters, independently.

3.3 Starting vector candidates identification

After we extract the V-shaped vectors, we need to identify the possible starting points for the desired patterns. In this project, we limit the scope of our project to two patterns: cup-and-handle and double-bottom. These are basic patterns for technical analysis of stock charts data, since the occurrence of both patterns indicate that the price will soon increase. The two patterns start with a relatively long decline, so we look for V-shaped vectors that has "big" s_l parameter.

The scheme we use to determine whether the parameter is "big" enough is simply to select ones that are larger than average. We experimented with using mean and median and found that each is suitable for different data. Based on preliminary observation, we found that we should not rule out too many candidates. The candidates which end up being too small will result in a pattern too small and will be ruled out in the pattern verification process in the next step. Therefore, we decided to use the minimum between mean and median as the threshold for identifying starting vector candidates.

It should be note that, occasionally, due to our doubly smoothened data, some window of data only contains big V-shaped vectors. In other word, the data smoothing process is so effective that it already removes all negligible oscillations for us. In that case, both the mean and median will be very big. So, we empirically set a threshold such that if both mean and median exceed the threshold, then we simply use the preset threshold to identify starting vector candidates.

For other patterns not explored in this project, the algorithm for identifying starting vector candidates may need to be modified. For example, for double-top pattern, we would want to start at the middle of a V-shaped vector and look for big s_r parameters instead. It is also possible to extract an upside-down V-shaped vectors in the previous step and calculate the transformation matrix with respect to a normalized upside-down V-shaped vector. The same formula could be applied to calculate the four transformation parameters.

3.4 Pattern encoding

Since we do not have experience or training in technical analysis, we look at several examples in online tutorial to determine the general structure of the cup-and-handle and the double-bottom patterns. Then, we encode these structures as rules for verifying each of the starting vector candidates we identify in the previous step. The rules are as follow:

1. **Cup-and-handle pattern** consists of one, relatively wide V shape followed by a relatively short, shallow fall. The more specific rules to enforce the structure of this shape are:

    - The big V is made of one or more V-shaped vectors.
    - The 2 ends of the V must have similar heights under a preset threshold. (See red horizontal lines in Figure 2)

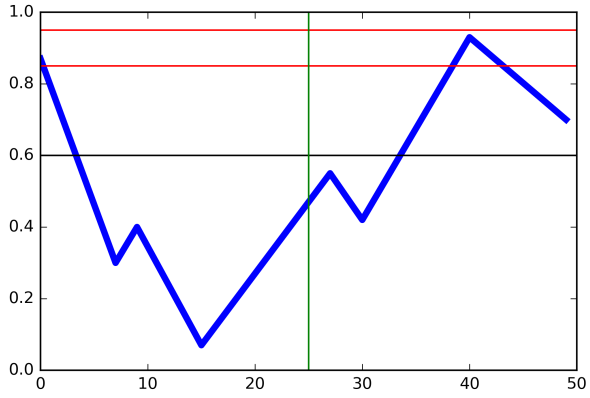

Figure 2: Example of data that fits cup-and-handle pattern encoding. The red horizontal lines indicate the allowable threshold of difference between two ends of the cup. The black horizontal line indicates the maximum height allowable for intermediate peaks between two ends of the cup. The green vertical line indicates the minimum length of data that would be considered a meaningful pattern.

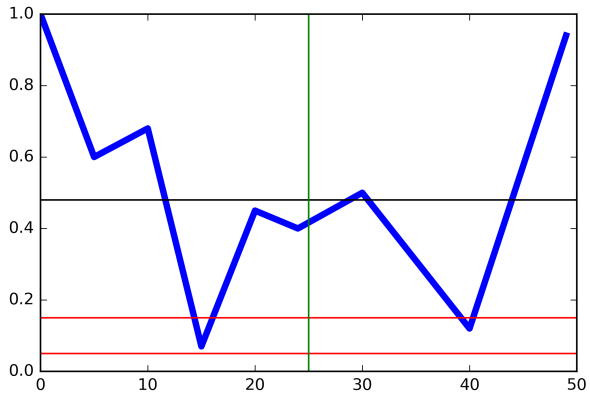

Figure 3: Example of data that fits double-bottom pattern encoding. The red horizontal lines indicate the allowable threshold of difference between two bottoms. The black horizontal line indicates the minimum height for a peak to be considered the middle peak of the pattern. The green vertical line indicates the minimum length of data that would be considered a meaningful pattern.

- The 2 ends of V must be far apart by at least a preset length. (See the green vertical line in Figure 2)
- The big V cannot have a local maximum inside higher than a preset threshold. (See the black horizontal line in Figure 2)

2. **Double-bottom pattern:** consists of two consecutive big V shapes with the "middle" peak between the two shapes being relatively high but not higher than the two ends of the shape. The more specific rules to enforce the structure of this shape are:
   - The big V is made of one or more V-shaped vectors.
   - There must be 2 dips that are a lot lower than others between the 2 ends.
   - The 2 dips must differ by only under a threshold. (See red horizontal lines in Figure 3)
   - The right end has to be higher than the left end.
   - The middle peak must be higher than a preset percentage of the left end. (See black horizontal line in Figure 3) And there must be only one middle peak between the two ends of the pattern.
   - The 2 ends of V must be far apart by at least a preset length. (See the green vertical line in Figure 3)

4 Experiment

The experiment section consists of two subsections: dataset and baseline system. Specifically, we will describe the datasets we use or plan to use to evaluate our system and the baseline system that we will compare our system to.

4.1 Datasets

The dataset we use to experiment our method is the S & P500 stock dataset [1] which includes 5 years of historical stock data (February 8, 2013 - February 7, 2018) from 500 S& P companies. The dataset contains 619,040 data points (one for each day.) In each data point, we get the following attributes of the stock data:

- Open - Price of the stock at market open (this is NYSE data so all in USD)
- High - Highest price reached in the day
- Low - Lowest price reached in the day
- Close - Price of the stock at market close

[1] https://www.kaggle.com/camnugent/sandp500

- Volume - Number of shares traded
- Name - the stock's ticker name

Another dataset that we will potentially experiment on later is the Huge Stock Market Dataset [2] which is much larger than the previous dataset. The data provided in this dataset contains the same attributes as in the S & P500 dataset.

4.2 Baseline system

We implement a baseline system that is partially based on self-organizing maps. The system utilizes unsupervised learning to create a template for each pattern of interest based on the unlabeled data. Although supervised methods such as multi-layered perceptrons and recurrent neural net would almost definitely give better performance, they all require us to annotate and label the data from S & P500 dataset. The annotation and labelling processes are very time-consuming and need objective domain-specific knowledge, which is not available to us. Since our system could detect patterns without having to be trained with annotated data, we decide to implement a baseline system that is equally efficient.

The baseline system gets a window of raw data as input. The window is equal to the initialized pattern in length. The general idea of the algorithm is that, if the window of data is "similar enough" to the template, then we update the template by exponential average with the window of data. We determine similarity between data and template with sum-of-squares error and empirically set a threshold to decide whether the error is low enough to update the template with the data. It should be noted that one window of data can only update one template that is most similar to it. In our implementation, we create more than one templates for each pattern to cover several possible lengths of patterns in one training round.

To use this system to detect patterns in test data, we would perform 2-D convolution with the pattern and the data to find segments of data with sum-of-squares error below a reasonable, preset threshold. The result of the baseline system will be discussed in Section 5.1.

5 Result

5.1 Baseline system results

We visualize the templates for cup-and-handle and double-bottom patterns trained with by the baseline system. We tested the system with several different template size.

It is clear from Figure 4 and Figure 5 that the trained templates are not the desirable shapes. For the cup-and-handle pattern, all templates result in curves that resemble a sigmoidal equation. For the double-bottom pattern, the template

[2] https://www.kaggle.com/borismarjanovic/price-volume-data-for-all-us-stocks-etfs

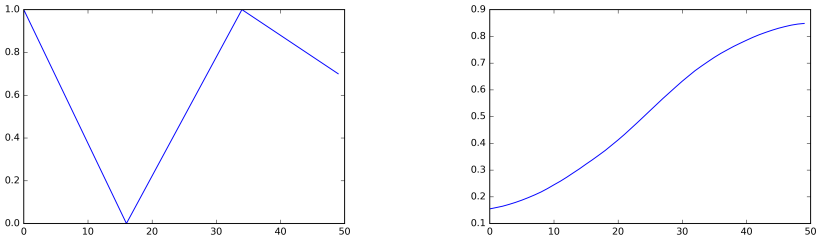

Figure 4: Comparison between (left) initial and (right) trained cup-and-handle pattern templates.

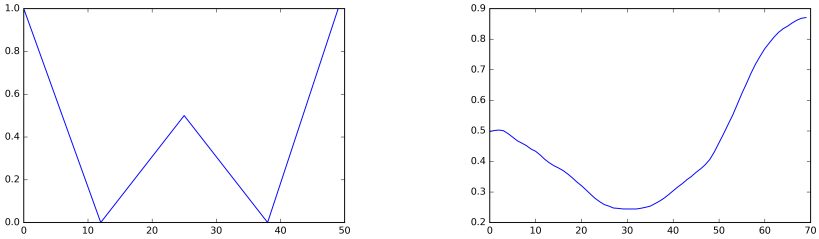

Figure 5: Comparison between (left) initial and (right) trained double-bottom pattern templates.

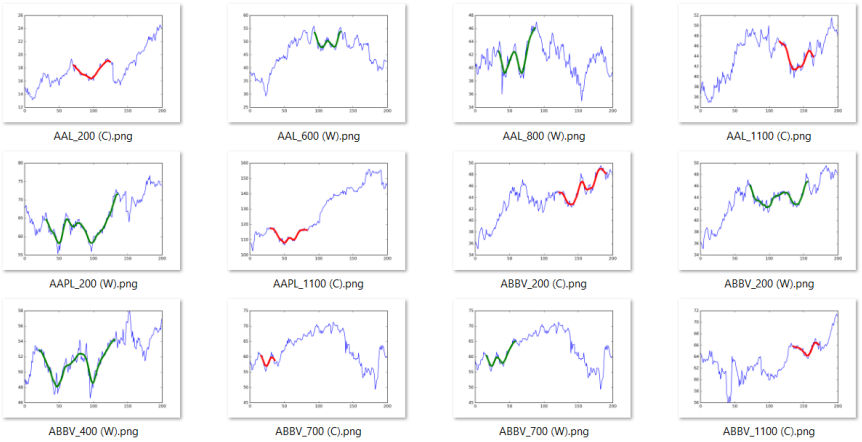

Figure 6: Outputs of the proposed system when run on three stock companies.

of length 70 is the only one that shows some resemblance to initial template with both ends higher than the middle segment. But it does not have the middle peak at all, which means it does not have two clear dips that constitute double-bottom pattern. And we found that running these patterns on test data does not give us the desired detection at all.

One clear limitation we observed with the baseline system are that it requires as many preset parameters as our proposed system, but the parameters setting are not as intuitive. Secondly, it is not flexible to different lengths of patterns. Even though we initialize templates for several window sizes, they are still limited in numbers. The position of each feature of the pattern, for instance, the middle peak of the double-bottom pattern and the right end of the cup in the cup-and-handle pattern, is also not flexible. Therefore, this system could not cover the characteristics of several stock companies at the same time and cannot be modified to trader's personal preference.

5.2 Proposed system results

The system we proposed performs, efficiently, as it does not need to be trained and is not computationally complex. To evaluate our system, we give it a specified date and the window of data prior to the day that we are interested in. The system outputs all occurrences of desirable patterns in the specified window of data.

Figure 6 contains some outputs from the proposed system. In this experiment, we input data from three stock companies: AAL, AAPL, and ABBV. Each window of input has the length of 200 days and the minimum number of data points to be considered a significant pattern is 20 days. The red thick lines indicate detected cup-and-handle patterns. The green lines indicate detected double-bottom patterns. In almost all windows, there are only one pattern

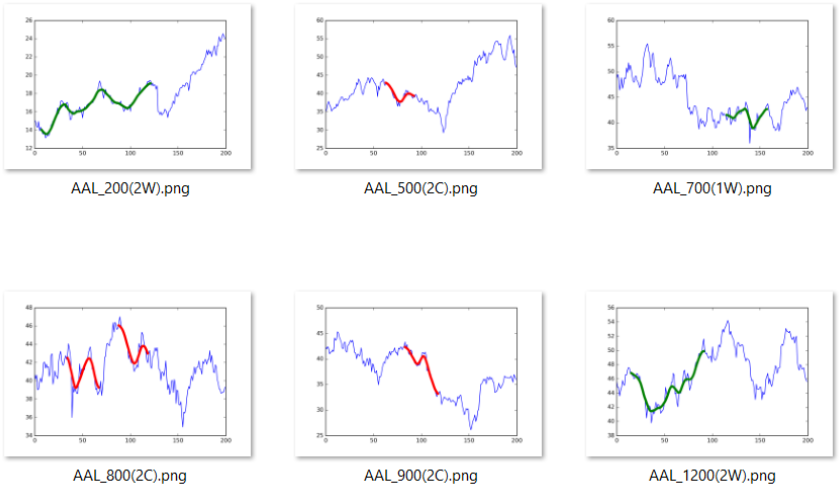

Figure 7: Outputs of the proposed system when run on a stock companies with more relaxed parameters for pattern encoding, resulting in patterns that we do not observe in Figure 6.

detected in the window. But in the window $ABBV_200$ (stock company ABBV, from day 1 to 200) the system manages to detect two consecutive cup-and-handle patterns.

All patterns detected are of the shapes we expect with the rules we encoded in Section 3.4. Moreover, we could see that in these windows, there is no existing patterns the system does not detect. So, our system performs exactly as we expected.

By relaxing the parameters, we can obtain more patterns in the stock charts data and observe a more distorted visual shapes of the desired patterns. For example, when we allowed the two ends of the cup-and-handle patterns to have a larger difference gap and allowed the two dips of the double-bottom pattern to have a larger difference gap, we obtained results in Figure 7 on top of the results we obtained in Figure 6. We can still see how these results resemble cup-and-handle and double-bottom patterns, although they are likely not meaningful patterns to expert traders. This show that the setting of the few human-given parameters in our system relies, greatly, on domain-specific knowledge.

However, these specific set of parameters (end point difference threshold = 2.0 and intermediate threshold = 30%) do not work for all stocks because some stock charts have bigger or smaller fluctuations in price than others. Figure 8 shows some of these cases. This shows that users with domain-specific knowledge should adjust these two parameters in the model according to the nature of each company they may want to run the system on. Moreover, they should adjust the parameters if they want to filter out results or include more results than

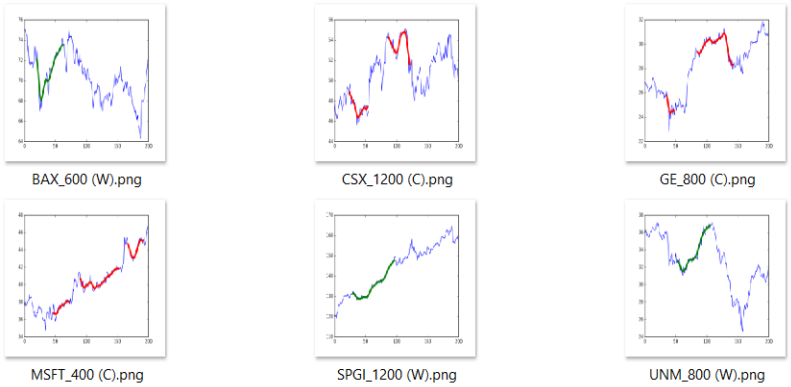

Figure 8: Outputs of the proposed system when run on some randomly selected companies.

ones shown in Figure 6. Again, because the system does not need training and because these parameters are very intuitive, the system could adapt very quickly to user's preference.

6 Future Work

- With more domain-specific knowledge on stock pattern analysis, we can make the pattern encoding more robust and more accurate. Some possible modifications include using angle and shearing information to make patterns more specific (See Figure 9) and refining our criteria for identifying starting vectors to take distribution of parameter values into account.
- Our system has the potential to extend to other patterns, since the encoding of each pattern is intuitive and is based on the same data preprocessing method. The V-shaped vector extraction and transformation parameters calculation could also be modified to detect other shapes such as upside-down V (for patterns that start with and/or end with local minima) and double V shapes. The same formula for parameter calculation in Section 3.2.1 also works for different normalized vector templates.
- This system could be implemented for other time-series data to extract human-recognizable patterns, e.g. rainfall or precipitation data, electricity consumption, sales figures, etc. All time-series could be modeled as a sequence of V-shaped vectors, so if there are useful, visual patterns that experts can identify, then the encoding component of our system could also be modified to detect them.

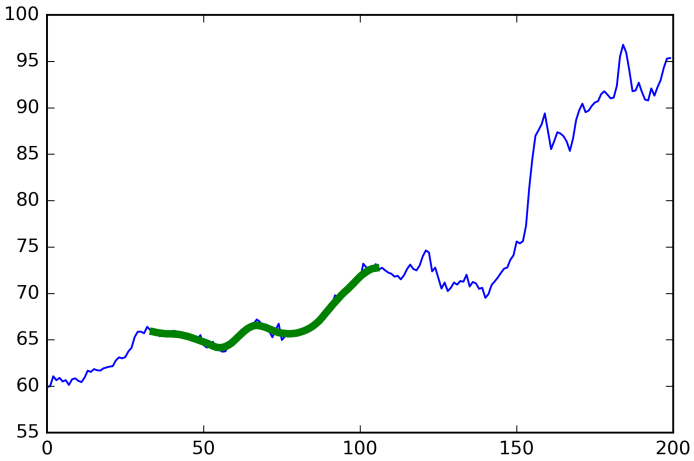

Figure 9: This double-bottom pattern detected fits all rules from section 3.4 but could be too wide to be considered a significant double-bottom pattern in which expert traders are interested in. It could be filtered out by limiting the angle and shearing parameters of the two big V shapes.

7 Conclusion

We developed a tool that, given 4 parameters of user's specification, recognizes basic stock charts patterns. The tool does not need training data or data annotation, only domain-specific language on the general structure of the patterns. The tool works, effectively, for detecting cup-and-handle and double-bottom patterns. It can be easily extended to include other patterns as well. This shows that transforming data into V-shaped vectors make pattern recognition more intuitive and more robust to negligible oscillation.

Acknowledgments

We would to express our appreciation to Professor Weiqing Gu for initiating our idea and advicing us throughout this project.

References

[1] X. Y. Guo, X. Liang, and X. Li. A Stock Pattern Recognition Algorithm Based on Neural Networks. *Third International Conference on Natural Computation (ICNC 2007)*, 2007.

[2] T. C. Fu, F. L. Chung, V. Ng, and R. Luk. Pattern Discovery from Stock

Time Series Using Self-Organizing Maps. *Workshop Notes of KDD2001 Workshop on Temporal Data Mining*, San Francisco, CA, pages 27–37, 2001.

[3] K. Kamijo and T. Tanigawa. Stock price pattern recognition-a recurrent neural network approach.*1990 IJCNN International Joint Conference on Neural Networks*, San Diego, CA, 2012.

[4] Ou, Phichhang and Wang, Hengshan (2009) Prediction of Stock Market Index Movement by Ten Data Mining Techniques. *Modern Applied Science*, Vol. 3 No. 12.

[5] Saad, Emad W., Prokhorov, Danil V, Wunsch, Donald C. (1998) Comparative Study of Stock Trend Prediction Using Time Delay, Recurrent and Probabilistic Neural Networks. *IEEE Transactions on Neural Networks*, Vol. 9 No. 6.

[6] Tsai M-C, Cheng C-H, Tsai M-I, Shiu H-Y (2018) Forecasting leading industry stock prices based on a hybrid time-series forecast model. *PLoS ONE*.

[7] Zhang, Hui, Ho, Tu Bao, Zhang, Yang, Lin, Mao-Song (2006) Unsupervised Feature Extraction for Time Series Clustering Using Orthogonal Wavelet Transform. *Informatica*.

[8] Bonde, Ganesh and Khaled, Rasheed (2012) Extracting the best features for predicting stock prices using machine learning. *Proceedings of the 2012 International Conference on Artificial Intelligence*, ICAI 2012.