2009 IEEE International Conference on Data Mining Workshops

Mining for Core Patterns in Stock Market Data

Jianfei Wu and Anne Denton and Omar Elariss
Department of Computer Science and Operations Research
North Dakota State University
Fargo, ND
{jianfei.wu, anne.denton, omar.elariss}@ndsu.edu

Dianxiang Xu
National Center for the Protection of the Financial Infrastructure
Dakota State University
Madison, SD
dianxiang.xu@dsu.edu

***Abstract*—We introduce an algorithm that uses stock sector information directly in conjunction with time series subsequences for mining core patterns within the sectors of stock market data. The core patterns within a sector are representative groups of stocks for the sector when it shows coherent behavior. Multiple core patterns may exist in a sector at the same time. In comparison with clustering algorithms, the core patterns are shown to be more stable as the stock price evolves. The proposed algorithm has only one free parameter, for which we provide an empirical choice. We demonstrate the effectiveness of the algorithm through a comparison with the DBScan clustering algorithm using data from the Standard and Poor 500 Index.**

***Index Terms*—core pattern; time series; desity histogram; quasi-clique;**

I. Introduction

In this paper, we introduce an algorithm for identifying core patterns within sectors of stock market data. We use the sector data directly in conjunction with the stock time series for finding the core patterns. In contrast to standard machine learning approaches, which use Boolean in form of a class label (supervised) or not at all (unsupervised), our algorithm inherently tests the relationship between the Boolean data and the time series, and returns patterns only when the relationship is significant. For the purpose of this paper, a core pattern is a representative group of stocks that show coherent behavior specific to their sector. Multiple core patterns may exist in one sector concurrently. Whether a sector is showing coherent behavior is determined through a modified version of the density histogram technique introduced in [1]. The distribution of stocks within a sector is compared with that of the overall data set, and the statistical significance is calculated. Core patterns of a significant, and hence coherent, sector can be extracted. A detailed discussion is given in Sect. III.

Fig. 1 illustrates the concepts for an example. The top two panels show the preprocessed stock open values in a time window. In the top left panel, the stocks in black highlight a sector of stocks. A randomly selected sample of the same size as the sector is highlighted using charcoal grey color in the top right panel. Two histograms (one in black, the other in charcoal grey) in the bottom panel summarize the neighboring relationships between stocks in the sector and in the random sample respectively. It can be observed that the stocks in the sector have more neighbors on average than those in the random sample. Furthermore, the stocks that are represented

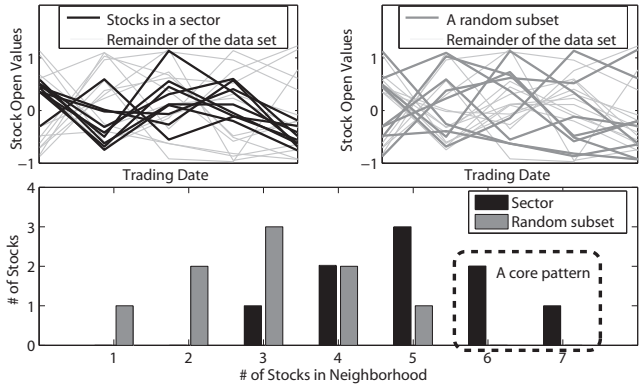

Fig. 1. The top two panels show the preprocessed stock open values in a time window. The top left panel highlights a sector of stocks. The top right panel shows a randomly selected sample which has the same size as the sector. Two histograms (one in black, the other in charcoal grey) in the bottom panel summarize the neighboring relationships between stocks in the sector and in the sample respectively. A core pattern is identified by comparing these two histograms.

by the circled bins of the black histogram have more neighbors than any of the stocks in the random sample. They form a core pattern for this sector.

In Fig. 1, only one random sample is shown. However, in the algorithm, which will be presented shortly, we use multiple random samples and construct a histogram that represents averages over those samples. This histogram acts as a reference for the sector and allows determining if the sector differs significantly from what would be expected by random chance.

The histograms quantify the number of neighbors for a subset of stocks, which can be viewed as a measure of local density of the subset of stocks. We will call them density histograms. We refer to the histogram for a sector as observed density histogram, and the one that was averaged over multiple random samples as expected density histogram.

In this study, the stock data come from the Standard and Poor 500 Index from 07/30/07 to 07/29/08. We follow the suggestion of [2] and use the first derivatives instead of raw values, i.e., we take the differences between successive trading days of each stock. The evaluation shows that our algorithm is more effective than a comparison algorithm at determining groups of sequences that show coherent behavior in a successive window. The remainder of the paper is organized

978-0-7695-3902-7/09 $26.00 © 2009 IEEE
DOI 10.1109/ICDMW.2009.115

as follows: In Sect. II we introduce the related work. In Sect. III we present our algorithm in detail. In Sect. IV we systematically evaluate our algorithm, and compare it with the density-based clustering algorithm DBScan. In Sect. V we conclude the paper.

II. Related Work

Many previous studies discuss cluster stability analysis for static data sets [3], [4]. All of them separate the original data set into subsets and then evaluate consistency between clusters found in different subsets. In [5], Ben *et al* present a method for quantitatively assessing the presence of structure in a clustered data set. The method counts the number of similarities between the labels of the objects common to both subsamples. In [3], Tilman *et al* present an approach that evaluates the partitions of a data set of some clustering algorithm, by checking whether similar structures are identified under repeated applications of the clustering algorithm. Volkovicha *et al* present an approach to evaluate the goodness of a cluster by the similarity among the entire cluster and its core [4]. None of these discuss stability in a dynamic data set or time series data set. How stable clusters are expected to be over consecutive time windows depends on factors that influence the time dependence.

For stock data, previous studies [6] have shown that stocks that belong to the same sector tend to move more coherently than stocks from different sectors. From a business point of view this can be expected, since stocks in the same sectors are likely to be influenced by the same external factors, and may also influence each other. That means that coherence that is specific to a sector may be an indication that the stocks are influenced by the same factors and may continue to do so over longer periods of time. We test this coherence using concepts developed in [1]. If the stocks show a statistically significant pattern with respect to their membership in a sector, we expect the core or cores of that pattern to be stable over time.

The term core pattern has previously been used to describe a subset of a frequent itemset with a certain core ratio [7].

III. Algorithm

The proposed algorithm has two main steps. In each time window, the algorithm first identifies significant sectors by comparing the two density histograms, i.e., for each sector the observed density histogram is compared with the expected density histogram that is constructed from random samples. In a second step, the algorithm extracts core patterns from those significant sectors.

In this study, we use the stream sliding window concepts. Two adjacent sliding windows are used, namely training window and evaluation window. The algorithm detects significant sectors in the training window, and builds core patterns for the significant ones. Evaluation windows are used for testing how stable the core patterns are. In the remainder of the paper, the term "time window" is used to represent either a training window or an evaluation window.

A. Outline of the algorithm

The algorithm iteratively executes the following steps until all training and evaluation windows are processed:

(a) Detecting Significant Sectors in a training window

Normalization: Stock data are first normalized within a training window, using a row-wise z-score normalization followed by a column-wise z-score normalization. More details will be given in Sect. III-B.

Significance test: For each sector, an observed density histogram, together with an expected density histogram, is constructed. Then the statistical significance of the sector is calculated by applying a χ^2 goodness-of-fit test on the two density histograms. The sectors which are significant at 0.1% level are used to form core patterns.

(b) Forming core patterns

Extract high-density stocks: By comparing the two density histograms, stocks that have more neighbors than expected are identified.

Form core patterns: Core patterns are extracted from high-density stocks of a sector that is considered to be significant. Details will be discussed in Sect. III-D.

(c) Extract core patterns in the successive evaluation window

Core patterns in the successive evaluation window are extracted with the same procedures as in the training window.

B. Normalization

In each time window, we first perform a row-wise z-score normalization on each vector, then apply a column-wise z-score normalization on all dimension values of stock vectors along each dimension, as (1) and (2) show respectively.

$$\begin{aligned} \mathbf{V}_{i,j}^{k,L} &= \frac{\mathbf{V}_{i,j}^{k,L} - mean(\mathbf{V}_i^{k,L})}{std(\mathbf{V}_i^{k,L})} \\ mean(\mathbf{V}_i^{k,L}) &= \frac{1}{L}\sum_{j=1}^{L}\mathbf{V}_{i,j}^{k,L} \\ std(\mathbf{V}_i^{k,L}) &= \sqrt{\frac{1}{L}\sum_{j=1}^{L}(\mathbf{V}_{i,j}^{k,L} - mean(\mathbf{V}_i^{k,L}))^2} \end{aligned} \tag{1}$$

$$\mathbf{V}_{i,j}^{k,L} = \frac{\mathbf{V}_{i,j}^{k,L} - \frac{1}{N}\sum_{i=1}^{N}\mathbf{V}_{i,j}^{k,L}}{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(\mathbf{V}_{i,j}^{k,L} - \frac{1}{N}\sum_{i=1}^{N}\mathbf{V}_{i,j}^{k,L})^2}} \tag{2}$$

N is the number of stock vectors in a time window. It is equal to the number of stocks in the data set. $\mathbf{V}_i^{k,L}$ is the i^{th} ($i \in [1, N]$) stock vector that starts from the k^{th} trading day and ends at the $(k + L - 1)^{th}$ trading day. It has a dimensionality of L. $\mathbf{V}_{i,j}^{k,L}$ is the j^{th} ($j \in [1, L]$) dimension value of the stock vector $\mathbf{V}_i^{k,L}$.

The row-wise z-score normalization was chosen in accordance with the study in [2]. Together with the column-wise z-score normalization, this normalization approach ensures that random data would result in a distribution that closely

resembles the normal distribution that is assumed in the theoretical model.

All stock vectors are further projected to a unit hypersphere, as (3) shows, such that the cosine similarity between two stock vectors can be calculated efficiently.

$$V_{i,j}^{k,L} = \frac{V_{i,j}^{k,L}}{|V_i^{k,L}|}$$
$$|V_i^{k,L}| = \sqrt{\sum_{j=1}^{L} (V_{i,j}^{k,L})^2} \quad (3)$$

C. Significance test

The observed histogram is used to summarize the neighboring relationships between stocks in a sector. If a sector shows coherent behavior, the stocks within this sector have more neighbors than expected. The distribution of expected neighbors is quantified through calculating neighbors of stocks in random samples. In other words, if stocks in a sector have more neighbors than those in random samples, we conclude that the sector is showing coherent behavior. We apply a χ^2 goodness-of-fit test on the observed and expected density histograms for each sector. If a sector is significant at a 0.1% level (P-Value of 0.001), we consider that sector to be significant, and showing coherent behavior.

In this study, cosine similarity is used to determine whether two stock vectors are neighbors of each other. If the cosine similarity of two stock vectors exceeds a threshold τ, they are considered neighbors. The cosine similarity between stock vector **X** and stock vector **Y** is defined as:

$$Sim(\mathbf{X}, \mathbf{Y}) = \sum_{i=1}^{L} (X_i Y_i) \quad (4)$$

X and **Y** are two vectors both having a dimensionality of L. X_i and Y_i are the i^{th} dimension values for **X** and **Y** respectively.

To build an observed density histogram for a sector, the number of neighbors for each stock vector in the sector is calculated, and a contribution of 1 is added to a corresponding bin of the observed density histogram. To build an expected histogram, T random samples are first drawn. Then the contribution of these random samples are averaged to create the expected density histogram for the sector. In this study, T is set to 30.

Fig. 2 shows an example of the observed density histogram and the expected density histogram, together with the theoretical model, which will be presented shortly, for the energy sector from 07/31/07 to 08/06/07. Notice the difference between the two density histograms in Fig. 2. The mean of the observed density histogram is greater than that of the expected density histogram, which implies that the stock vectors in energy sector have a higher density than the overall data set.

After the two density histograms are constructed, statistical significance of the sector is attained by using a χ^2 goodness-of-fit test. Those sectors that are significant at 0.1% level are further used to extract core patterns.

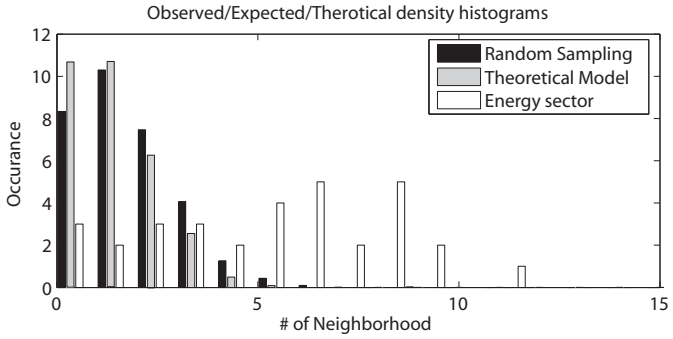

Fig. 2. Density histogram for energy sector using random sampling and the theoretical model for the time window from 07/31/07 to 08/06/07.

D. Forming core patterns

Definition 1: High-density stock vectors: The stock vectors that can be extracted from the tail of the observed density histogram until the aggregated density of the expected density histogram is greater than or equal to 1.

We extract those high-density stock vectors for each significant sector. Once those high-density stock vectors are extracted, core patterns can be formed among them. Quasi-clique detection [8], [9] is a state-of-art technique to mine dense subgraphs in a large graph. It has been used in many applications, including functional prediction of uncharacterized genes [10], mining highly correlated stocks [11]. We use quasi-clique mining technique to extract core patterns from the high-density stock vectors: Let $G = (V, E)$ be a graph where V is a set of vertices representing the high-density stock vectors, and E is a set of edges representing neighboring relationships among the high-density stock vectors. Unfortunately, the problem of mining $\gamma - cliques$ in a graph is a $NP - hard$ problem.

Multiple definitions for quasi-cliques exist [8], [9]. In this study, we largely follow the definition of *quasi-cliques* in [9]. However, considering the graph G is relatively small (remember that we mine quasi-cliques only among high-density stock vectors), we adopt an absolute cutoff for $deg^G(v)$, that is, we conside a threshold (*Th*, which is an integer) for the number of edges that all vertices in a quasi-clique must possess. In other words, we try to find quasi-cliques, in which all vertices have at least *Th* edges (In our study *Th* equals to 3). Such a modification greatly simplifies the quasi-clique mining procedure. Algorithm 1 depicts an algorithm for mining this type of quasi-cliques, and it has a time complexity of $O(n^2)$ in the worst case.

Algorithm 1 iteratively deletes the vertices that do not belong to a quasi-clique (i.e., delete the vertices that have less than *Th* edges), until only those ones that belong to a quasi-clique are left. Fig. 3 illustrates a simple example (*Th* equals to 2 in this example). The original graph has 6 vertices. Before algorithm 1 proceeds to the while loop, vertices 5 and 6 are deleted after executing *Updatematrix* command. After one iteration of the while loop, vertex 4 is deleted, and a quasi-clique which includes vertices 1, 2 and 3 is found.

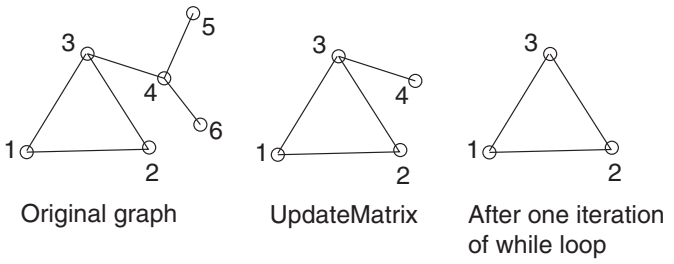

Fig. 3. Illustration of the procedure for extracting quasi-cliques.

Algorithm 1: Mining Quasi-Clique

Data: $graph\_Matrix$; /* graph matrix in which entry value equals to 1 indicates an edge, 0 other wise. */
Data: Th; /* threshold for the minimum number of edges. */
Result: $quasi\_cliques$
1 $candidate\_vertices$ = findVertices($graph\_Matrix$,$\geq$,Th); /* find vertices which have at least Th edges. */
2 $graph\_Matrix$ = updateMarix($graph\_Matrix$,$candidate\_vertices$); /* update graph matrix by eliminating entries having value 1 that are not connecting vertices in $candidate\_vertices$ */
3 **while** *findVertices*($graph\_Matrix$,$<$,Th) $\neq$ *null* **do**
4 $candidate\_vertices$ = findVertices($graph\_Matrix$,$\geq$,Th);
5 $graph\_Matrix$ = updateMarix($graph\_Matrix$,$candidate\_vertices$)
6 $quasi\_cliques$ = $candidate\_vertices$;
7 **return** $quasi\_cliques$;

E. Theoretical model

A theoretical model is built to approximate the expected density histogram to improve the efficiency. As we mentioned in Sect. III-B, performing a row-wise z-score normalization then a column-wise z-score normalization ensures that all dimension values of the stock vectors approximately follow a standard normal distribution along each dimension. We further assume that all dimensions of stock vectors are independent. This may not be strictly true but, since we take the first derivative of each stock, the dependency between dimensions is mitigated. Thus, the probability density function (PDF) for a stock vector **X** can be approximated as:

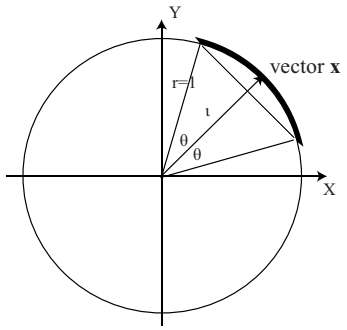

Fig. 4. The relationship between τ and the hyper-surface area of the cap.

$$\begin{aligned}\varphi(\mathbf{X}) = P(x_1, x_2 ... x_L) &= \prod_{i=1}^{L} P(x_i) \\ &= \prod_{i=1}^{L} \frac{1}{\sqrt{2\pi}} e^{-\frac{x_i^2}{2}} = (\frac{1}{\sqrt{2\pi}})^L e^{-\frac{\sum_{i=1}^{L} x_i^2}{2}} \\ &= (\frac{1}{\sqrt{2\pi}})^L e^{-\frac{|\mathbf{X}|^2}{2}}\end{aligned} \tag{5}$$

where x_i is the i^{th} dimension value of the stock vector **X**, and $|\mathbf{X}|$ is the vector length of **X**.

Under such an assumption, the distribution of stock vectors is independent of the orientation of the stock vectors. Therefore, under this approximation the number of neighbors of a stock vector for a random data set can be estimated by computing the hyper-surface area of the cap, which is segmented by the hype-plane that is defined by threshold τ.

Fig. 4 illustrates the concept in two dimensions. In Fig. 4, suppose **X** is a stock vector in a random sample of size $|S|$. Then the number of neighbors of **X** within the random sample can be approximated by $(|S|-1) * \frac{A_L}{A}$, where A is the hyper-surface area of the unit hypersphere (in this example it is the circumference of the circle) and A_L is the hyper-surface area of the cap (in this example it is the length of the arc) and L is the dimensionality of the stock vector (in the example L=2). .

Therefore, we try to find the relationship between the threshold τ and the hyper-surface area of the cap which is segmented by the hype-plane that is defined by threshold τ. The hyper-surface area of the cap divided by that of the hypersphere (i.e., $\frac{A_L}{A}$) is exactly the probability for another stock vector to be neighborhood of stock vector **X**.

The area of the cap can be calculated as follows:

$$\begin{aligned}A_L = &\int_{-a\cos\tau}^{a\cos\tau} \sin^{L-2}(\varphi_1) d\varphi_1 \int_0^{\pi} \sin^{L-3}(\varphi_2) d\varphi_2 \\ &... \int_0^{\pi} \sin(\varphi_{L-2}) d\varphi_{L-2} \int_0^{2\pi} d\varphi_{L-1} \\ =&\Omega * \int_{-a\cos\tau}^{a\cos\tau} \sin^{L-2}(\varphi_1) d\varphi_1\end{aligned} \tag{6}$$

where $\phi_1, \phi_2 ... \phi_{L-1}$ are angular coordinates, and $\Omega = \int_0^{\pi} \sin^{L-3}(\varphi_2) d\varphi_2 ... \int_0^{\pi} \sin(\varphi_{L-2}) d\varphi_{L-2} \int_0^{2\pi} d\varphi_{L-1}$. So the probability of two vectors **X** and **Y** being neighbors is

$$p = \frac{A_L}{A} = \frac{\Omega * \int_{-a\cos\tau}^{a\cos\tau} \sin^{L-2}(\varphi_1) d\varphi_1}{n * C_L} \tag{7}$$

where $C_L = \begin{cases} \frac{\pi^{(L/2)}}{(L/2)!} & \text{for even L} \\ \frac{2^{(L+1)/2}\pi^{(L-1)/2}}{L!!} & \text{for odd L} \end{cases}$

Equation (7) can be simplified:

$$p = \frac{A_L}{A} = \Omega^{'} * \int_{-a\cos\tau}^{a\cos\tau} \sin^{L-2}(\varphi_1) d\varphi_1 \quad (8)$$

where $\Omega^{'} = \frac{\Omega}{L * C_L}$. To improve performance, p can be saved to a table TB for different values of L and different thresholds τ. Because τ is of type float, we may need to find the closest value in the table. From (8), we notice *p* is a function of cosine similarity threshold τ and vector dimensionality *L*.

Once *p* for a specific threshold τ is calculated using (8), or simply by searching table *TB*, the theoretical model for approximating the expected density histogram is given through a binomial model as follows:

$$h_i = (|S| - 1) * \left(\begin{array}{c} (|S| - 1) \\ i \end{array} \right) * p^i * (1 - p)^{(|S|-1)-i} \quad (9)$$

h_i is the height of i^{th} bin in the expected density histogram, and $|S|$ is the number of stock vectors in a sector. Note that Fig. 2 shows the result of theoretical model together with random sampling and observed histogram.

There is only one free parameter, i.e., the cosine similarity threshold τ for the proposed algorithm with random sampling as well as with using theoretical model. We found that $p = \frac{1}{|S|-1}$ gives consistently good results for a large number of test cases. This choice of p corresponds to centering the mean of the theoretical distribution on the histogram bin corresponding to a single neighbor. A mathematically rigorous derivation of this heuristic is omitted due to space constraints.

F. Summary of the algorithm

The proposed algorithm is summarized in Algorithm 2. $\mathbf{V}_{:}^{k,L}$ is an array of all stock vectors that start from k^{th} trading day and end at $(k + L - 1)^{th}$ trading day. They all have a dimensionality of L. $\mathbf{V}_{:}^{k,L}$ is first normalized using (1) and (2). Then for each sector, the cosine similarity threshold is first calculated, and the observed histogram is built. *AlgoSwitch* is used to switch between theoretical model and random sampling. When *AlgoSwitch* is equal to 0, the algorithm uses the theoretical model from (9), otherwise it uses random sampling with 30 random samples to build the expected histogram. Then the statistical significance of each sector is calculated by applying a χ^2 goodness-of-fit test on the two density histograms. Core patterns are extracted from those that are significant.

Algorithm 2: algorithm summary

Data: $\mathbf{V}_{:}^{k,L}$; /* Stock vectors in a time window. */
Data: B; /* Stock sector filter matrix. */
Data: $AlgoSwitch$; /* Algorithm switch, 0 for theoretical and 1 for random sampling */
Result: $CorePatterns$

```
CorePatterns = [];
V_:^{k,L} = normalize(V_:^{k,L});
foreach sector i do
    S = σ(B_i);        /* Select the subset of stocks in sector i. */
    τ = 1/(|S|-1);
    observed_hist = BuildObservedDensityHist(V_S^{k,L}, τ);
    if AlgoSwitch == 0 then
        expected_hist = BuildTheoreticalModel(|S|);
    else
        T = 30;
        expected_hist = RandomSampling(V_:^{k,L}, τ, |S|, T);
    signif = SignificanceTest(expected_hist, observed_hist);
    if signif ⩽ 0.001 then
        CorePatterns = AddNewCorePattern(CorePatterns,
        ExtractCorePatterns(expected_hist,observed_hist));
return CorePatterns;
```

IV. Evaluation

The algorithm is tested on *S&P500* stock database from 07/30/07 to 07/29/08, which has 252 values after taking the first derivatives. Thus, with training window size of *L* and evaluation window size of *L'*, the algorithm is tested $(252 - L - L' + 1)$ times. For each test, if the algorithm identifies that a sector is coherent (significant) in a training window, core patterns are built separately in the training window and its corresponding evaluation window.

The stability of core patterns can be evaluated by examining whether the stock vectors in the core patterns found in a training window can still form core patterns in the consecutive evaluation window. Thus, we apply sensitivity, specificity and accuracy measurements to evaluate the stability of a core pattern. Notice that if there are more than one core patterns, we aggregate them together for the purpose of simplifying the evaluation.

A prominent density-based clustering algorithm, DBScan, is used to compare with the proposed algorithm. DBScan[12] is a well-known density-based clustering algorithm. It has two parameters: minimum points (*minPts*) and epsilon (*eps*). We perform DBScan in the same stock data set. Since stock vectors change over each time window, it is very inefficient to use a k-dist graph [12] to estimate proper values for *eps* and *minPts* in each time window. Therefore, we test DBScan with *minPts* being equal to 3,4,5,...,11 in each time window. A corresponding *eps* is chosen such that the average sizes of clusters found by DBScan algorithm are close to the average size of core patterns.

The comparisons of sensitivity, specificity and accuracy are shown in Fig. 5. It can be seen that the proposed algorithm,

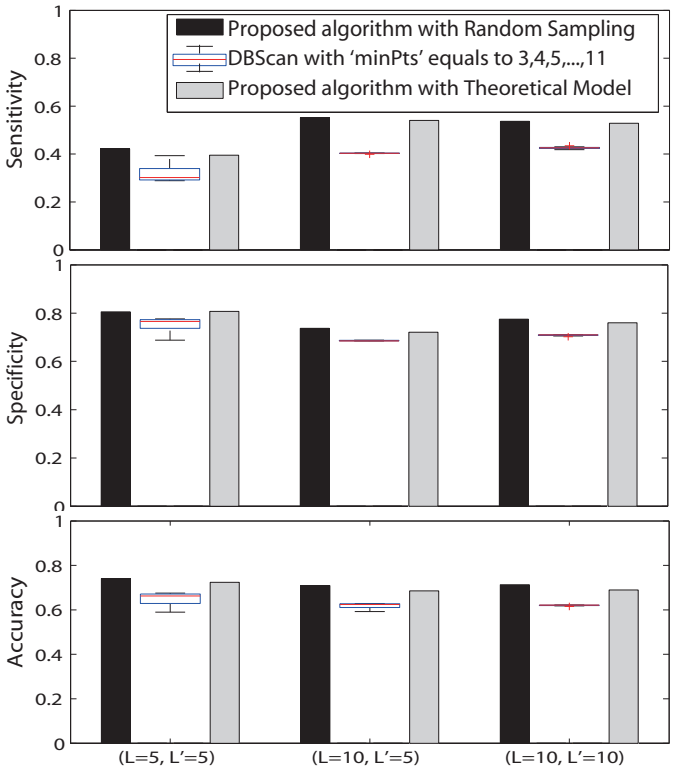

Fig. 5. Comparisons of sensitivity, specificity and accuracy among algorithms.

using both random sampling and the theoretical model, outperforms DBScan algorithm. The proposed algorithm with theoretical model is comparable with the proposed algorithm with random sampling. With increasing window size, the variance of the results of DBScan decreases. The accuracies and specificities of the proposed algorithms are not significantly different among $(L = 5, L^{'} = 5)$, $(L = 10, L^{'} = 5)$ and $(L = 10, L^{'} = 10)$. However, the sensitivities of $(L = 5, L^{'} = 5)$ for the proposed algorithms are relatively worse than those of $(L = 10, L^{'} = 5)$ and $(L = 10, L^{'} = 10)$.

Fig. 6 shows the comparison of the performance among the algorithms (The run times for DBScan with different parameters are averaged). Notice that the performance of the proposed algorithm with Random Sampling depends heavily on the number of samples used to create an expected density histogram. Although we only use 30 samples in this study, its performance is much worse than the other algorithms. However, using the theoretical model in the proposed algorithm achieves comparable performance with DBScan.

Summarizing, the core patterns extracted by the proposed algorithms with both random sampling and with theoretical model are more stable than the clusters found by DBScan algorithm. Furthermore, The proposed algorithm with theoretical model has a comparable performance with DBScan.

V. Conclusion

In this paper, we presented an algorithm for mining core patterns in stock market data. The algorithm identifies whether

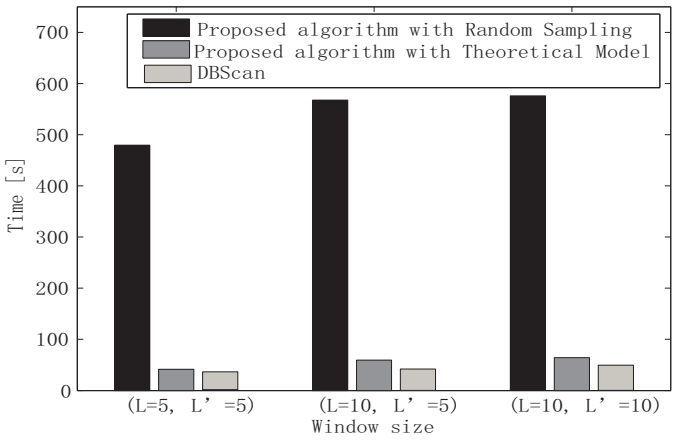

Fig. 6. Comparisons of efficiency among algorithms. The run times for DBScan with different parameters are averaged.

a stock sector currently shows coherent behavior. When coherent behavior of a stock sector is detected, core patterns are extracted. The core patterns are more stable than clusters found by some clustering algorithm DBScan. Such core patterns can be used in many existing stock mining algorithms that would otherwise use clusters. Through comparing with DBScan, we show the effectiveness of the proposed algorithm.

References

[1] J. Wu and A. M. Denton, "Mining vector-item patterns for annotating protein domains," in *Proc. of the Workshop on Mining Multiple Information in conj. with the ACM SIGKDD Int'l Conf. on Data Mining (KDD)*, San Jose, Aug 2007.

[2] M. Gavrilov, D. Anguelov, P. Indyk, and R. Motwani, "Mining the stock market: Which measure is best? (extended abstract)," in *In proceedings of the 6 th ACM Int'l Conference on Knowledge Discovery and Data Mining*, 2000, pp. 487–496.

[3] T. Lange, V. Roth, M. L. Braun, and J. M. Buhmann, "Stability-based validation of clustering solutions," *Neural Comput.*, vol. 16, no. 6, pp. 1299–1323, 2004.

[4] Z. Volkovich, Z. Barzily, and L. Morozensky, "A statistical model of cluster stability," *Pattern Recognition*, vol. 41, no. 7, pp. 2174–2188, 2008.

[5] A. Ben-Hur, A. Elisseeff, and I. Guyon, "A stability based method for discovering structure in clustered data," in *Pacific Symposium on Biocomputing*, 2002, pp. 6–17.

[6] R. Coelho, S. Hutzler, P. Repetowicz, and P. Richmond, "Sector analysis for a ftse portfolio of stocks," *Physica A Statistical Mechanics and its Applications*, vol. 373, pp. 615–626, Jan. 2007.

[7] F. Zhu, X. Yan, J. Han, P. S. Yu, and H. Cheng, "Mining colossal frequent patterns by core pattern fusion," in *ICDE*, 2007, pp. 706–715.

[8] J. Abello, M. G. C. Resende, and S. Sudarsky, "Massive quasi-clique detection," in *In Latin American Theoretical Informatics (LATIN)*, 2002, pp. 598–612.

[9] G. Liu and L. Wong, "Effective pruning techniques for mining quasi-cliques," in *ECML PKDD '08: Proceedings of the European conference on Machine Learning and Knowledge Discovery in Databases - Part II*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 33–49.

[10] H. Hu, X. Yan, Y. Huang, J. Han, and X. J. Zhou, "Mining coherent dense subgraphs across massive biological networks for functional discovery," *Bioinformatics*, vol. 21, no. 1, pp. 213–221, 2005.

[11] V. Boginski, S. Butenko, and P. M. Pardalos, "Mining market data: a network approach," *Comput. Oper. Res.*, vol. 33, no. 11, pp. 3171–3184, 2006.

[12] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *KDD*, 1996, pp. 226–231.