2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology

Stock Price Prediction in Undirected Graphs Using a Structural Support Vector Machine

Richard Kyle MacKinnon
Department of Computer Science
University of Manitoba
Winnipeg, MB, Canada
Email: ummack57@cs.umanitoba.ca

Carson K. Leung
Department of Computer Science
University of Manitoba
Winnipeg, MB, Canada
Email: kleung@cs.umanitoba.ca

***Abstract*— Business analytics techniques help mine and analyze business/financial data. For instance, a *structural support vector machine* (SSVM) can be used to perform classification on complex inputs such as the nodes of a graph structure. We connect collaborating companies in the information technology sector in an undirected graph and use an SSVM to predict positive or negative movement in their stock prices. By using a minimum graph-cutting algorithm to drive the cutting plane optimization problem of the SSVM, an exact solution is achieved in polynomial time. The learned model exploits the associative relationship between the prices of the collaborating companies to outperform the accuracy of a regular SVM. Experiments were conducted using the companies in the Standard and Poor's 500-45 Information Technology Sector index. Trades based on the learned model achieved superior returns in the range of 10% to 17% while tracking the index alone over the same time periods yielded returns in the range of -17% to 9%.**

***Keywords*-Web intelligence; structural support vector machine (SSVM); minimum graph-cuts; graph labeling; finance; stock price prediction**

I. INTRODUCTION

Data mining algorithms and methods [5], [6], [9], [19] applied to real-world datasets can form the basis for effective web intelligence applications and business analytics reasoning. This allows users to continuously or iteratively explore, investigate and understand past business performance so as to gain new insight and drive business planning.

Stock price prediction is an extremely difficult web intelligence and business analytics problem regarding its basis in optimization and forecasting future trends. A recent statistic showed that from 2008 to 2012, only 9.84% of Canadian equity fund managers achieved better returns than passively managed funds based on the S&P/TSX Composite Index [14]. That means that more than 90% of the time, funds in which stocks were actively selected by fund managers performed worse than the market as a whole. Clearly, there is room for improvement in this area.

Algorithms that can accurately forecast stock price movements have a huge financial incentive for whoever has access to them. Aside from the potential for creating multi-millionaires, such an algorithm would have secondary benefits as well. One possibility would be early identification of investments that are destined to fail, reducing the chance of major disruptions and market crashes when they do. Another possibility is that a successful algorithm could be adapted to other domains with similar problem requirements (e.g., social network mining [21], [31], Big data mining and analytics [18]). We previously conducted a feasibility study [20] to explore the possibility of using machine learning approaches for stock price prediction.

In this paper, we apply data mining and machine learning techniques on business intelligence data consisting of fundamental financial information to develop a model for the business analytics problem of predicting positive or negative movement in stock prices. To perform classification on complex inputs such as financial data that are modelled in terms of nodes in a graph structure, we use a *structural support vector machine* (SSVM) together with a minimum graph-cut approach.

Key contributions in this paper are: (i) a new application of SSVM and minimum graph-cuts to stock price prediction using fundamental analysis, and (ii) the ability to consider the relationships between collaborating companies in the prediction model.

The remainder of this paper is organized as follows. The next section discusses related work covering different approaches to stock price prediction and recent advances in the field. The requisite machine learning background on SVMs and SSVMs follows. The following section outlines the integration of the SSVM and minimum graph-cut algorithms. Next, the graph structure, feature vectors, and training labels are described. Finally, experimental results show the accuracy of the prediction model and the benefit of capturing the collaborating information over a regular SVM. A real-world market evaluation is also conducted showing a favourable comparison of the return on investment when using the model as opposed to just tracking the S&P 500-45 index.

II. RELATED WORK

There are several schools of thought when it comes to stock price prediction. Technical analysis [26] contends that early identification of trend reversals allows the investor to profit. Factors—such as past price, trading volume, moving averages and candlestick patterns—all fall under the envelope of this approach. More recently, attempts to augment these traditionally statistical methods with artificial

978-1-4673-9618-9/15 $31.00 © 2015 IEEE
DOI 10.1109/WI-IAT.2015.189

intelligence algorithms such as neural networks [29], genetic algorithms [2], [13], or a combination of the two [14] have been considered.

Despite positive results in many papers, there still exists some question on the profitability of technical analysis [25]. Moreover, proponents of the efficient market hypothesis [22] and the random walk theory [8] refute the very premise of technical analysis. The efficient market hypothesis states that stock prices adjust immediately to changes in information. Since the timing and content of new information is unpredictable by nature, stock prices will follow a 'random walk' and thus current prices must be independent of their previous values (or their dependence is negligible).

Fundamental analysis is an approach to stock price prediction that does not rely on previous prices. Instead, a fundamental analysis [8], [27] seeks to identify variables, or fundamentals, that describe the intrinsic value of a stock. By comparing the actual price of the stock to the intrinsic value, investors can profit by selecting undervalued stocks. Recent work in this area has focused on identifying macroeconomic fundamentals [33], in-depth studies exploring what conditions are most conducive for particular fundamental variables [24], as well as fuzzy expert system development for portfolio recommendation [10].

III. Background

In this section, we provide some background information on SVMs, such as mathematical definitions and the differences between SSVMs and regular SVMs.

A. Support Vector Machines

SVMs in their most basic form act as binary classifiers for linearly separated data. It is essentially an optimization problem that finds a hyperplane that has the greatest distance, or maximum margin, between all the training data. Any point lying on the hyperplane will satisfy the equation: $w^T x + b = 0$ for feature vector x. The margin is then:

$$\frac{2c}{\|w\|}$$

with the optimization problem being:

$$\max_{w,b} \frac{2c}{\|w\|}$$

$$\forall i: y_i(w^T x_i + b) \geq c$$

where y_i is the label and can be either ± 1. Without changing the solution, this can be reformulated and rescaled to make $\|w\|$ easier to calculate, giving the general form of the SVM:

$$\min_{w,b} \frac{1}{2}\|w\|^2$$

$$\forall i: y_i(w^T x_i + b) \geq 1$$

The term b is called the bias and describes how far away from the origin the hyperplane is. In practice, this value can be ignored if the training data are normalized.

B. Slack Variables

In the case that there is no solution to the previous optimization problem, we must have data that are non-linearly separated. To deal with this case slack variables are introduced for each training example that falls within the margin. This changes our SVM formulation to:

$$\min_{w,b,\xi \geq 0} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i$$

$$\forall i: y_i(w^T x_i + b) \geq 1 - \xi_i$$

The purpose of the additional term in the objective function is to increase its value when training data fall within the margin and are misclassified. C is a parameter set by the user that describes the magnitude of the penalty for a misclassification. Properly choosing a value for C is a trade-off between the margin size and the number of misclassified training examples [1]. If C is set too high, the SVM will over-fit the training data. On the other hand, if C is set too low, the SVM will have a high error rate due to the large margin.

C. Multiclass Support Vector Machines

The previous formulae all described binary classifiers, but a more generalized approach is needed when an arbitrary number of labels is possible. Crammer and Singer [4] show that this can be done by representing w as a matrix with one row for each label. The classifier then computes a similarity score between the feature vector x and each row of the matrix. If $\Psi(x_i, y_i)$ is a function relating feature vectors and labels, then the classifier chooses the best label y^* using the following equation:

$$y^* = \underset{y \in \mathcal{Y}}{\operatorname{argmax}}\left(w^T \Psi(x_i, y)\right)$$

The full multiclass SVM then takes the form:

$$\min_{w,\xi \geq 0} \frac{1}{2}\|w\|^2 + \frac{C}{n}\sum_{i=1}^{n} \xi_i$$

$$\forall i, \forall y \in \mathcal{Y} \setminus y_i: w^T \Psi(x_i, y_i) \geq w^T \Psi(x_i, y) + \Delta(y_i, y) - \xi_i$$

The second equation describes constraints imposed by each incorrect labelling y that allow the model parameter w to be learnt: the value of the slack variable ξ_i must be minimized such that the difference in score the SVM calculates between the correct and incorrect label is greater than or equal to the loss function $\Delta(y_i, y)$ (describing the actual difference between labels) minus ξ_i.

D. Structural Support Vector Machines

SSVMs generalize multiclass SVMs one step further. The optimization equations look the same as multiclass SVMs, but the interpretation of the label space $\mathcal{Y}$ is expanded to include structured labels such as parse trees, sequences; and, as in this paper, undirected graphs, with interdependencies present among the parts of these structures. In most cases, the large label space makes an exact solution intractable [32], which is why we chose binary labels of positive or negative movement for our stock price predictor.

IV. Our Stock Price Predictor

In this section, we describe our stock price predictor, which uses a structural support vector machine (SSVM) to predict stock prices in undirected graphs.

A. Integration of SSVM with Minimum Graph-Cuts

Consider the equations for the multiclass and structural SVMs. The first equation can be solved via a cutting-plane algorithm [12]. The cutting-plane algorithm starts out with only a subset of the constraints and uses a separation oracle to decide which new constraints to add at each step. The new constraint added is always the most violated constraint not yet added using the partial model learnt from the subset. Tsochantaridis *et al.* [32] showed that if the separation oracle can run in polynomial time, then the entire optimization problem could run in polynomial time. The complexity of the separation oracle is equivalent to solving a maximum a posteriori (MAP) inference problem with the addition of the constant loss function term [12]. Considering the following equations reveals this relationship. The first equation rearranges the previous constraint equation to show when a constraint is violated. This allows the most violated constraint $\hat{y}$ to be calculated. The final equation gives the general form of MAP inference for comparison.

$$\exists i, \exists y \in \mathcal{Y} \setminus y_i: \\ (w^T\Psi(x_i, y) - w^T\Psi(x_i, y_i) + \Delta(y_i, y) - \xi_i) > 0,$$

$$\hat{y} = \underset{y \in \mathcal{Y}}{\operatorname{argmax}}\big(w^T\Psi(x_i, y) - w^T\Psi(x_i, y_i) + \Delta(y_i, y)\big)$$

and

$$\underset{y \in \mathcal{Y}}{\operatorname{argmax}}\big(w^T\Psi(x, y)\big)$$

Fortunately, MAP inference is known to have a polynomial run-time for the special case of graphical model in which the potential functions at each node are associative; an exact solution for this case can be obtained by taking minimum cuts in graphs [17].

B. Graph Structure

The stock market is a complicated system with many variables affecting whether or not a particular stock price goes up or down. In order to accurately capture the complex relationships between companies that affect their stock prices, they can be represented as an undirected graph. Each node in the graph corresponds to a single company and each edge represents collaboration between a pair of companies. This work uses the companies appearing in the information technology sector of the S&P 500 index [30]. See Figure 1 for a visualization of our graph.

Because the relationship of collaborating companies is many-to-many, this would normally be an intractable problem if inference were determined using a message-passing algorithm (the complexity is exponential in the tree-width of the graph). Fortunately, the collaboration relationship is an associative one (i.e. if two companies are collaborators, an increase in the stock price of one of them is likely associated with an increase for the other; likewise, a decrease in the stock price of one of them is likely associated with a decrease for the other). As mentioned earlier, we are able to use minimum cuts to solve inference queries in polynomial time.

Edges in the graph are determined by comparing the estimated result count in the Bing search engine for the two queries:

1. "Company A"-"Company B" collaboration;
2. "Company A"-"Company B" competition.

If the first query has more than 1.5 times the results of the second, an edge is added to the graph connecting the nodes for Company A and Company B. This multiplier is necessary to avoid adding edges between companies that are either not strong collaborators or competitors and thus have extremely similar result counts for both queries. This was automated via the Bing Search API from the Windows Azure Marketplace [23]. The values on the edge vectors are simply initialized to one, requiring the SSVM to learn the labelling from the feature vectors at the nodes.

C. Feature Vectors

Each node in the graph has a 69-element feature vector containing fundamental financial information for the company represented by that node. Raw data for the feature vectors was obtained in csv format from Quandl via their API for stock fundamentals [7].

Stock fundamentals partly consist of both current and historical financial information for a company. Publicly traded companies in the United States are required to disclose their financial information in annual 10-K and quarterly 10-Q filings with the U.S. Securities and Exchange Commission. This information is in a very raw form and is collected and curated before appearing on Quandl. Some examples of individual features are: Number of Shares Outstanding, Net Margin, Ratio of Fixed Assets to Total Assets, and 3-year Standard Deviation of Stock Price.

Some features, such as Book Value of Assets, could be valued in the hundreds of thousands of dollars, while others, such as Ratio of Fixed Assets to Total Assets, always take a value between zero and one. Since these features are not necessarily normalized, finding a hyperplane to separate the data becomes impossible. To address this we automated additional processing of the raw feature values to calculate

their zero-mean unit-deviation normalized form. This transformation is shown below:

$$x_f = \frac{x_i - \mu}{\sigma}$$

The initial and final values of the feature are given by x_i and x_f respectively. The mean is given by μ and the standard deviation is given by σ.

D. Training Labels

The final pieces of data required to train an SSVM are the training labels. In this case, the labels are binary (i.e. consisting of a positive class and a negative class). Nodes whose companies saw an increase in stock price in the months after the release of financial data were labelled with the positive class while nodes whose companies saw a decrease in stock price were labelled with the negative class. Historical stock price data was obtained for all companies from Yahoo! Finance [3].

Since the maximum time span between historical data points obtained from Yahoo Finance is monthly while the Quandl feature vector data was calculated for annual data points, additional processing was done on the Yahoo Finance data to remove the extra data points.

V. Experimental Results and Analysis

Several experiments were conducted on a MacBookPro8,2 with a 2GHz Intel Core i7-2635QM processor and 8GB memory. The SSVM training and testing were performed using the dlib-C++ library with 4 threads [16]. The compiler used was clang-503.0.40 with -O3 optimization level. The dlib-C++ library implements a graph labelling SSVM trainer using minimum cuts for inference and provides functions for cross validation and testing of a trained SSVM. Regular SVM training and testing is also included with the library. All results reported have been averaged over 3 runs.

A. Cross Validation of the SSVM Model

One of the most important parts of training an SVM is choosing the best value for the C parameter. Every dataset will behave differently and the default values suggested by different libraries are not guaranteed to be adequate. In order to determine an appropriate value, cross validation is used.

Cross validation refers to partitioning the training data and training the SSVM using only a subset of those partitions. The SSVM is then tested on the partitions not used in the training and its accuracy on the test data is recorded. The partitions used for training and testing are then switched out until every partition has been tested with. The results on the test partitions are then averaged over the total number of training runs. The results are given in the form of a ⟨percentage of positive class labels correctly assigned, percentage of negative class labels correctly assigned⟩ tuple. Ideally the perfect values would be ⟨1, 1⟩, indicating that all nodes were correctly classified and no node was incorrectly classified. In the case where the two values are different, a bias can be applied during the training process that has the effect of averaging out the results.

Results for the 2-fold and 3-fold cross validation, as well as an average line for our stock collaborators dataset are shown in Figure 2, with raw data available in Tables I and II. As can be seen from the graphs, the two lines representing percentages of correct positive and negative class selections are closely related. If the SSVM finds a hyperplane that favours negative class labels over positive class labels, we would expect to see a higher percentage of negative class labels correctly classified and a lower percentage of positive class labels correctly classified.

The best numbers are obtained in the case of 3-fold cross validation where the C parameter is set to 10. In this case the tuple returned was $\langle 0.49,\ 0.72 \rangle$, with the averaged value equal to 61%. It should be noted that the number of training examples used was quite small, however 61% is clearly above the realm of random chance. It can be expected from the cross validation that slightly better results be obtained when the SSVM is trained on the entire training set due to the usage of all the training data instead of just a portion of it, as is shown in the next experiment.

B. Test Data Accuracy

After choosing the value of C following the cross validation process, the SSVM was trained with all the training samples. The order of the samples was randomized to reduce any bias due to chronological trends in the data. The accuracy of the learned model was measured against both the training data and new testing data that were not used in any training sample. The results for this test are shown in Table III, with each like-coloured column corresponding to one random arrangement of the data.

Clearly the SSVM has a much higher accuracy on the training samples than it does on the test samples. For the training samples the percentage of labels correctly classified was in the range [65%, 97%] with a mean of 81%. This shows that the SSVM has learnt the problem well enough to correctly classify the majority of nodes in the training samples. However, it has not over-fit the data (otherwise the accuracies would all be 100%). The test samples elicited slightly lower accuracies, appearing in the range [51%, 91%] with a mean of 67%. One of the reasons for these lower numbers could be that the number of training samples was not large enough to truly learn all the details of the model. Ideally, hundreds of training samples would be optimal but we were limited by the availability of publicly accessible data.

C. Comparison to Regular SVM

One question to ask at this point is: does the use of an SSVM give any benefit over a regular SVM in terms of accuracy? The SSVM captures the information about collaborating companies in the graph structure, but a regular SVM considers each company separately, losing this structured information. For this reason we expected that the collaborating information would increase the accuracy of prediction of the SSVM over the regular SVM. To verify this claim we additionally trained a regular SVM on the same

training data and observed the reported accuracy over three runs.

Since we cannot expect the same parameter values for our regular SVM as for our SSVM, it was necessary to cross validate our SVM on the training data as well before testing. Because the SVM considers each company separately, the number of training samples is vastly increased, so 10-fold cross validation was used to give the best chance of generalizing the learned model to handle the test data. Note that we used dlib-C++'s implementation of a ν-SVM [28] with a radial basis kernel, so our corresponding parameters are ν and γ. The cross validation accuracies resulting in the best parameter values are given below in Table IV for 3 different randomized orderings of the training data.

Following the cross validation, the SVM was trained with all the training samples. The results of running the learned model on the test data are shown in Table V. It it important to note that since regular SVMs treat each company separately, a single sample run with our SSVM was comprised of multiple companies while a simple sample run with our SVM is comprised of only a single company. To more easily compare samples together between the two types of SVM, multiple single company samples have been grouped together so that the number and composition of companies in each grouped sample matches that of the corresponding SSVM sample in Table III. However, unlike with the SSVM case, there are no dependencies on the results within each group.

There are a few important observations that can be gathered from this experiment. The first is that the accuracy of the SVM on the training data was in the range [57%, 100%], with a mean of 84%. Compared to the SSVM the difference is negligible. What is interesting to note is the much larger gap in accuracy between the two cases for the test data. The accuracy of the regular SVM for the test data falls in the range [32%, 76%] with a mean of 54%. The SSVM is the obvious winner in this case.

Looking at the actual classification of positive and negative labels, it is clear that the SSVM correctly classifies more nodes than the SVM when the class itself has a larger size. This reflects the associative nature of the underlying data that can be captured in its graph structure. The regular SVM does not have this advantage and loses out.

D. Market Evaluation

The final experiment we conducted was a test of the profitability of our learned model. We made stock picks based on the positive and negative class predictions of the model (whether or not they were correct) and compared the return on investment to what would have been obtained by investing the same amount of money as a lump sum tracking the S&P 500-45 Information Technology Index [11] for the same time periods. If the model predicted a positive class label (the value of the stock was predicted to rise) we took a buy and hold position. If the model predicted a negative class label (the value of the stock was predicted to fall) we exploited this as well by taking a short selling position.

The results for this experiment are given in Table VI. The initial investment was selected to be $10K per stock when picking stocks based on the model. The total amount of the initial investment was the same in both cases (when tracking the index a single lump sum was used), but the SSVM showed much higher returns. In the three test cases, the return based on the SSVM surpassed the index by 34.59%, 19.03% and 8.4% respectively.

VI. Conclusions

Minimum graph cuts were used as part of a cutting plane algorithm for solving the optimization problem of an SSVM. This allowed the SSVM to learn a prediction model for a complex graph input with multiple edges per node. We applied this approach to the problem of stock price prediction, where the positive and negative class labels corresponded to increasing and decreasing stock prices respectively. Accuracy of the SSVM on the training samples in terms of correctly classified nodes had a mean of 81%, suggesting the model was learnt successfully without over-fitting. Although accuracy on the test samples was a little lower with a mean of 67%, this is beyond the level of random chance and more accurate than a regular SVM that cannot incorporate the collaboration data. When testing the model against the historical levels of the S&P 500-45 Information Technology Index, returns in the range of 10% to 18% were realized. Overall, SSVM classification augmented with minimum graph cuts is a viable approach for predicting stock prices while taking into account collaborating companies. This web intelligence and business analytics approach serves as a useful and practical business intelligence solution that helps optimize business processes in the financial sector.

Acknowledgment

This project is partially supported by NSERC (Canada) and the University of Manitoba in the form of research grants.

References

[1] A. Ben-Hur & J. Weston, "A user's guide to support vector machines," in O. Carugo & F. Eisenhaber (eds.), *Data Mining Techniques for the Life Sciences*, pp. 223-239, Oct. 2009.

[2] C.-H. Cheng, T.-L. Chen, & L.-Y. Wei, "A hybrid model based on rough sets theory and genetic algorithms for stock price forecasting," *Information Sciences*, **180**(9): 1610-1629, May 2010.

[3] Commodity Systems Inc. Historical prices (var.). *Yahoo* https://help.yahoo.com/kb/finance/SLN2311.html.

[4] K. Crammer & Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," *Journal of Machine Learning Research*, **2**: 265-292, Dec. 2001.

[5] A. Cuzzocrea, F. Jiang, C.K. Leung, D. Liu, A. Peddle, & S.K. Tanbeer, "Mining popular patterns: a novel mining problem and its application to static transactional databases and dynamic data streams," *Transactions on Large-Scale Data- and Knowledge-Centered Systems*, **21**: 115-139, 2015.

[6] A. Cuzzocrea, C.K. Leung, & R.K. MacKinnon, "Mining constrained frequent itemsets from distributed uncertain data," *Future Generation Computer Systems*, **37**: 117-126, July 2014.

[7] Damodaran, A. Financial ratios (var.). *Quandl* http://www.quandl.com/stocks.

[8] E.F. Fama, "Random walks in stock market prices," *Financial Analysts Journal*, **51**(1): 75-80, Jan.-Feb. 1995.

[9] A. Fariha, C.F. Ahmed, C.K. Leung, S.M. Abdullah, S. Pervin, & L. Cao, "A new framework for mining frequent interaction patterns from meeting databases," *Engineering Applications of Artificial Intelligence*, **45**: 103-118, Oct. 2015.

[10] M. Fasanghari & G.A. Montazer, "Design and implementation of fuzzy expert system for Tehran Stock Exchange portfolio recommendation," *Expert Systems with Applications* **37**(9): 6138-6147, Sept. 2010.

[11] The Financial Times Ltd. Historical prices (S&P 500). http://markets.ft.com/research/Markets/Tearsheets/Summary?s=SP500-45:IOM.

[12] T. Finley & T. Joachims, "Training structural SVMs when exact inference is intractable," in *Proc. ICML 2008*, pp. 304-311.

[13] A. Gorgulho, R. Neves, & N. Horta, "Applying a GA kernel on optimizing technical analysis rules for stock picking and portfolio composition," *Expert Systems with Applications*, **38**(11): 14072-14085, Oct. 2011.

[14] D. Hodges, "Is your fund manager beating the index? (hint: probably not)," *MoneySense*, Dec 2013-Jan. 2014.

[15] C.-M. Hsu, "A hybrid procedure for stock price prediction by integrating self-organizing map and genetic programming," *Expert Systems with Applications*, **38**(11): 14026-14036, Oct. 2011.

[16] D.E. King, "Dlib-ml: a machine learning toolkit," *Journal of Machine Learning Research*, **10**: 1755-1758, July 2009.

[17] V. Kolmogorov & C. Rother, "Minimizing nonsubmodular functions with graph cuts–a review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **29**(7), 1274-1279, 2007.

[18] C.K. Leung, "Big data mining and analytics," in *Encyclopedia of Business Analytics and Optimization*, pp. 328-337, Feb. 2014

[19] C.K. Leung, A. Cuzzocrea, & F. Jiang, "Discovering frequent patterns from uncertain data streams with time-fading and landmark models," *Transactions on Large-Scale Data- and Knowledge-Centered Systems,* **8**: 174-196, 2013.

[20] C.K. Leung, R.K. MacKinnon, & Y. Wang, "A machine learning approach for stock price prediction," in *Proc. IDEAS 2014*, pp. 274-277.

[21] C.K. Leung, S.K. Tanbeer, & J.J. Cameron, "Interactive discovery of influential friends from social networks," *Social Network Analysis and Mining*, **4**: art. 154, Dec. 2014.

[22] B.G. Malkiel, "The efficient market hypothesis and its critics," *Journal of Economic Perspectives*, **17**(1): 59-82, 2003.

[23] Microsoft. Bing Search API. http://datamarket.azure.com/dataset/bing/search.

[24] C. Park, "When does the dividend–price ratio predict stock returns?" *Journal of Empirical Finance*, **17**(1): 81-101, 2010.

[25] C.-H. Park & S.H. Irwin, "What do we know about the profitability of technical analysis?" *Journal of Economic Surveys*, **21**(4): 786-826, Sept. 2007.

[26] M.J. Pring, *Technical Analysis Explained* (5th. ed.) McGraw-Hill Education, New York, NY.

[27] S. Richardson İ. Tuna, & P. Wysocki, "Accounting anomalies and fundamental analysis: a review of recent research advances," *Journal of Accounting and Economics*, **50**(2-3): 410-454, Dec. 2010.

[28] B. Schölkopf, A.J. Smola, R.C. Williamson & P.L. Bartlett, "New support vector algorithms," *Neural Computation*, **12**(5): 1207-1245, 2000.

[29] W. Shen, X. Guo, C. Wu, & D. Wu, "Forecasting stock indices using radial basis function neural networks optimized by artificial fish swarm algorithm," *Knowledge-Based Systems*, **24**(3): 378-385, April 2011.

[30] S & P Dow Jones Indices. S&P 500 Information Technology. http://spindices.com/indices/equity/sp-500-information-technology-sector.

[31] S.K. Tanbeer, C.K. Leung, & J.J. Cameron, "Interactive mining of strong friends from social networks and its applications in e-commerce," *Journal of Organizational Computing and Electronic Commerce*, **24**(2-3): 157-173, 2014.

[32] I. Tsochantaridis, T. Joachims, T. Hofmann, & Y. Altun, "Large margin methods for structured and interdependent output variables," *Journal of Machine Learning Research*, **6**: 1453-1484, Sept. 2005.

[33] P. Wongbangpo & S.C. Sharma, "Stock market and macroeconomic fundamental dynamic interactions: ASEAN-5 countries," *Journal of Asian Economics*, **13**(1): 27-51, Jan.-Feb. 2002.

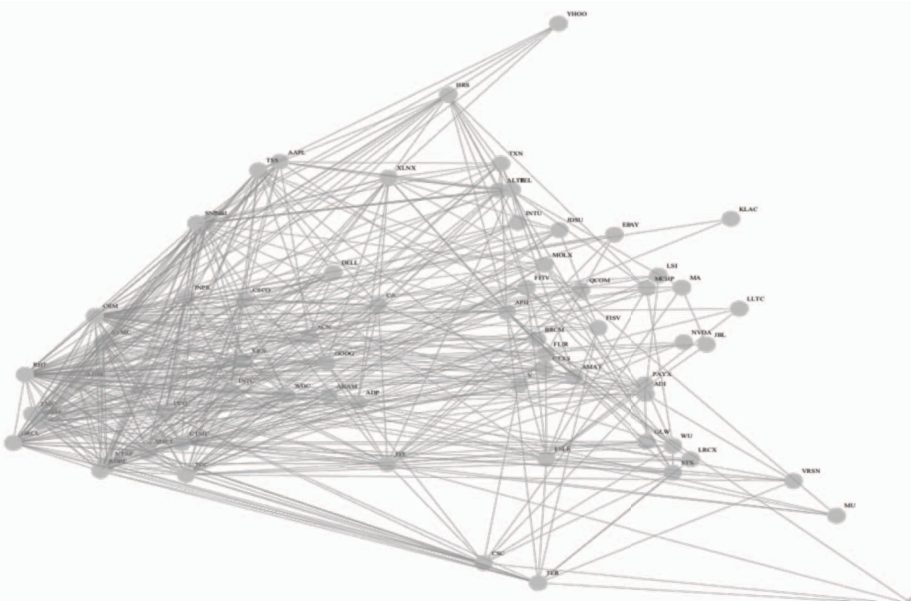

Figure 1. Visualization of our graph structure.

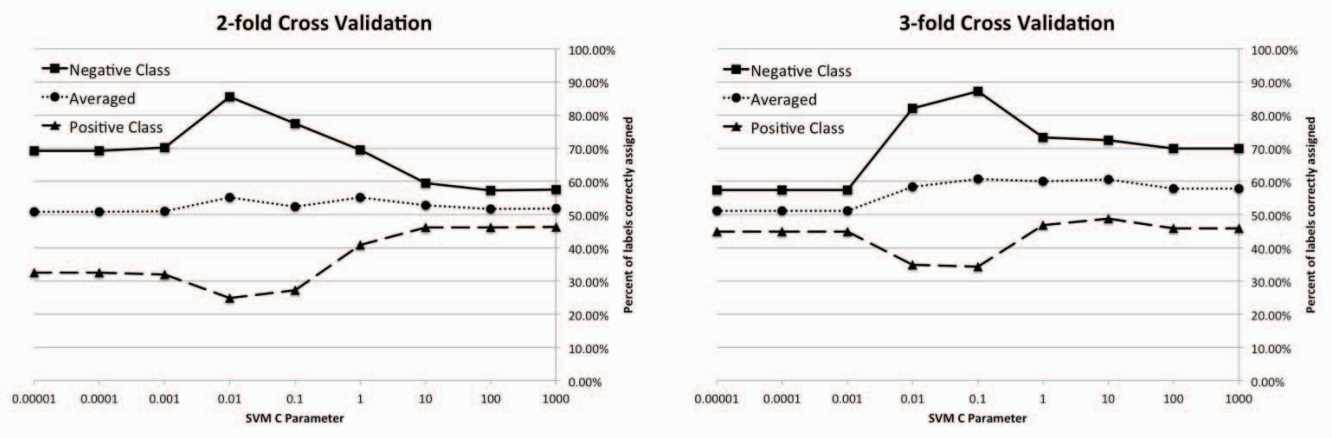

Figure 2. Cross validation results for 2- and 3-fold test.

TABLE I. STRUCTURAL SVM 2-FOLD CROSS VALIDATION DATA

Value of *C*	2-fold Cross Validation Raw Data									
	Percentage Positive Class Correctly Classified (Scale from 0 to 1)					*Percentage Negative Class Correctly Classified (Scale from 0 to 1)*				
	Run 1	*Run 2*	*Run 3*	*Average*	*Standard Deviation*	*Run 1*	*Run 2*	*Run 3*	*Average*	*Standard Deviation*
0.00001	0.325893	0.325893	0.325893	0.325893	0	0.692042	0.692042	0.692042	0.692042	0
0.0001	0.325893	0.325893	0.325893	0.325893	0	0.692042	0.692042	0.692042	0.692042	0
0.001	0.319196	0.319196	0.319196	0.319196	0	0.702422	0.702422	0.702422	0.702422	0
0.01	0.247768	0.25	0.247768	0.248512	0.001052175	0.858131	0.851211	0.858131	0.855824	0.003262119
0.1	0.267857	0.285714	0.265625	0.273065	0.008990255	0.778547	0.768166	0.775087	0.773933	0.004315824
1	0.412946	0.408482	0.404018	0.408482	0.003644841	0.681661	0.695502	0.709343	0.695502	0.011301129
10	0.457589	0.46875	0.459821	0.462053	0.004822134	0.595156	0.595156	0.595156	0.595156	0
100	0.462054	0.457589	0.464286	0.461310	0.002784239	0.570934	0.577855	0.570934	0.573241	0.003262591
1000	0.464286	0.459821	0.464286	0.462798	0.002104821	0.570934	0.595156	0.560554	0.575548	0.014498077

TABLE II. STRUCTURAL SVM 3-FOLD CROSS VALIDATION DATA

Value of *C*	3-fold Cross Validation Raw Data									
	Percentage Positive Class Correctly Classified (Scale from 0 to 1)					*Percentage Negative Class Correctly Classified (Scale from 0 to 1)*				
	Run 1	*Run 2*	*Run 3*	*Average*	*Standard Deviation*	*Run 1*	*Run 2*	*Run 3*	*Average*	*Standard Deviation*
0.00001	0.448661	0.448661	0.448661	0.448661	0	0.574394	0.574394	0.574394	0.574394	0
0.0001	0.448661	0.448661	0.448661	0.448661	0	0.574394	0.574394	0.574394	0.574394	0
0.001	0.448661	0.448661	0.448661	0.448661	0	0.574394	0.574394	0.574394	0.574394	0
0.01	0.352679	0.339286	0.352679	0.348215	0.006313521	0.813149	0.823529	0.823529	0.820069	0.004893179
0.1	0.352679	0.337054	0.339286	0.343006	0.006900039	0.865052	0.871972	0.878893	0.871972	0.005650565
1	0.466518	0.470982	0.466518	0.468006	0.00210435	0.737024	0.733564	0.726644	0.732411	0.004315378
10	0.491071	0.488839	0.482143	0.487351	0.003793671	0.723183	0.726644	0.723183	0.724337	0.001631531
100	0.444196	0.46875	0.462054	0.458333	0.01036363	0.695502	0.692042	0.709343	0.698962	0.007474917
1000	0.457589	0.457589	0.462054	0.459077	0.002104821	0.698962	0.698962	0.698962	0.698962	0

TABLE III. STRUCTURAL SVM ACCURACY FOR TRAINING AND TEST DATA

Sample Type & Number	Accuracy of Structural SVM Predictions (Correct Labels)								
	% of Total			*# / Total Positive Class*			*# / Total Negative Class*		
	Run 1	*Run 2*	*Run 3*	*Run 1*	*Run 2*	*Run 3*	*Run 1*	*Run 2*	*Run 3*
Training 0	86	80	73	53/56	43/45	41/45	4/10	10/21	7/21
Training 1	74	67	86	43/45	25/34	55/58	6/21	19/32	1/7
Training 2	65	88	97	31/34	56/58	0/0	12/32	1/7	61/63
Training 3	94	94	73	61/62	61/62	8/25	0/3	0/3	37/37
Training 4	66	76	66	15/25	42/43	29/33	26/37	2/15	10/26
Training 5	66	89	83	30/33	50/54	42/43	9/26	0/2	6/15
Training 6	81	75	96	40/43	3/15	54/54	7/15	35/36	0/2
Training 7	91	N/A	91	51/54	N/A	3/8	0/2	N/A	46/46
Test 0	72	65	73	45/58	38/56	45/56	2/7	5/10	3/10
Test 1	76	70	55	0/0	0/0	32/34	48/63	44/63	4/32
Test 2	51	60	91	24/30	3/25	59/62	5/27	34/37	0/3
Test 3	85	47	44	1/8	3/33	17/30	45/46	25/26	8/27
Test 4	72	58	69	2/15	24/30	2/15	35/36	9/27	33/36
Test 5	N/A	81	N/A	N/A	1/8	N/A	N/A	43/46	N/A

TABLE IV. REGULAR SVM PARAMETER VALUES AND CROSS VALIDATION ACCURACIES

Test Run	Percentage of Labels Correctly Assigned			
	ν	γ	*% Positive Class*	*% Negative Class*
Run 1	0.00467289	0.0032776	69.7143	52.1429
Run 2	0.000125988	4.29085	72.2581	66.3636
Run 3	0.322483	0.0159246	73.8462	65.7143

TABLE V. REGULAR SVM ACCURACY FOR TRAINING AND TEST DATA

Sample Type & Number	Accuracy of Regular SVM Predictions (Correct Labels)								
	% of Total			*# / Total Positive Class*			*# / Total Negative Class*		
	Run 1	*Run 2*	*Run 3*	*Run 1*	*Run 2*	*Run 3*	*Run 1*	*Run 2*	*Run 3*
Training 0	82	85	98	35/44	40/50	33/34	19/22	16/16	32/32
Training 1	77	70	97	38/47	28/43	34/35	13/19	18/23	29/30
Training 2	80	71	98	34/43	34/48	35/35	19/23	12/17	27/28
Training 3	86	57	100	40/44	25/41	33/33	16/21	12/24	29/29
Training 4	83	64	100	39/44	31/48	33/33	13/18	6/10	26/26
Training 5	88	68	98	41/44	31/46	29/30	11/15	7/10	28/28
Training 6	78	67	100	32/43	29/35	33/33	13/15	5/16	23/23
Training 7	86	N/A	95	36/43	N/A	32/33	12/13	N/A	21/21
Test 0	51	47	55	30/58	26/56	33/56	3/7	5/10	3/10
Test 1	33	32	58	0/0	0/0	32/34	21/63	20/63	6/32
Test 2	49	53	72	24/30	13/25	47/62	4/27	20/37	0/3
Test 3	50	59	54	7/9	23/33	26/30	21/47	12/26	5/27
Test 4	55	61	76	13/14	22/30	5/15	15/35	13/27	34/36
Test 5	N/A	66	N/A	N/A	2/8	N/A	N/A	35/46	N/A

TABLE VI. MARKET EVALUATION OF OUR SSVM MODEL

Test #	Using the SSVM Predictions			Tracking the S&P 500 Information Technology Index		
	Investment	*Balance*	*Return on Investment*	*Investment*	*Balance*	*Return on Investment*
1	2.90M	3.38M	16.83%	2.90M	2.38M	–17.76%
2	3.61M	3.98M	10.34%	3.61M	3.30M	–8.69%
3	3.05M	3.59M	17.60%	3.05M	3.33M	9.20%