A Machine Learning Approach for Stock Price Prediction

Carson Kai-Sang Leung* Richard Kyle MacKinnon Yang Wang
University of Manitoba, Winnipeg, MB, Canada
kleung@cs.umanitoba.ca

ABSTRACT

Data mining and machine learning approaches can be incorporated into business intelligence (BI) systems to help users for decision support in many real-life applications. Here, in this paper, we propose a machine learning approach for BI applications. Specifically, we apply structural support vector machines (SSVMs) to perform classification on complex inputs such as the nodes of a graph structure. We connect collaborating companies in the information technology sector in a graph structure and use an SSVM to predict positive or negative movement in their stock prices. The complexity of the SSVM cutting plane optimization problem is determined by the complexity of the separation oracle. It is shown that (i) the separation oracle performs a task equivalent to maximum *a posteriori* (MAP) inference and (ii) a minimum graph cutting algorithm can solve this problem in the stock price case in polynomial time. Experimental results show the practicability of our proposed machine learning approach in predicting stock prices.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*data mining*; I.2.6 [**Artificial Intelligence**]: Learning—*parameter learning*; J.1 [**Computer Applications**]: Administrative data processing—*financial*

General Terms

Algorithms; Experimentation; Validation

Keywords

Business intelligence (BI), data mining, finance, graph labeling, machine learning, minimum graph-cuts, stock price prediction, structural support vector machine (SSVM), support vector machine (SVM)

*Corresponding author: C.K.-S. Leung.

IDEAS'14, July 07–09, 2014, Porto, Portugal.
 ACM 978-1-4503-2627-8/14/07 ...$15.00.
http://dx.doi.org/10.1145/2628194.2628211

1. INTRODUCTION

Data mining [4, 10, 11, 12, 18] and machine learning [13, 14, 17] approaches can be incorporated into business intelligence (BI) systems to help users for decision support in many real-life applications. One interesting BI application is to predict stock prices. In general, making predictions [3], including *stock price prediction*, is a difficult problem. 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 [6]. 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. Improvement in this area is desirable.

Algorithms that can predict stock prices more accurately have a huge financial incentive for professionals who have access to stock prices. Aside from the potential for creating multi-millionaires, such an algorithm would have other benefits as well. One such benefit is to root out bad investments that are destined to fail, reducing the chance of major disruptions and market crashes when they do. Another benefit is that a successful algorithm could be adapted to other domains with similar problem requirements.

Our key contribution of this paper is our proposal of a machine learning approach based on a structural support vector machine (SSVM), with maximum *a posteriori* (MAP) inference calculated using minimum graph cuts.

The remainder of this paper is organized as follows. Background is provided in Section 2. Section 3 describes our proposed SSVM with MAP inference calculated using minimum graph cuts. Specifically, we describe the graph structure, feature vectors, and training labels. Experimental results in Section 4 show the effectiveness (especially, accuracy) of our proposal. Finally, conclusions are given in Section 5.

2. BACKGROUND & RELATED WORKS

Structural support vector machines (SSVMs) allow us to classify non-linearly separated data using a linear hyperplane by means of a max-margin approach together with slack variables for each training example. The general problem can be expressed mathematically by the following equations [5]:

$$\min_{\mathbf{w},\xi\geq 0} \left(\frac{1}{2}||\mathbf{w}||^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i \right), \quad (1)$$

and

$$\forall i\, \forall \mathbf{y}\in\mathcal{Y}\backslash\mathbf{y}_i\colon \mathbf{w}^T\mathbf{\Psi}(\mathbf{x}_i,\mathbf{y}_i) \geq \mathbf{w}^T\mathbf{\Psi}(\mathbf{x}_i,\mathbf{y}) + \Delta(\mathbf{y}_i,\mathbf{y}) - \xi_i, \quad (2)$$

where (i) $\mathbf{w}$ is the parameter learnt by the SSVM during training, (ii) C is a parameter set by the user that describes the magnitude

of the penalty for a misclassification, (iii) n is the number of slack variables, (iv) ξ_i is a slack variable, (v) $\mathbf{x}$ are the set of inputs and $\mathbf{y}$ are their respective labels, (vi) $\mathbf{\Psi}(\mathbf{x}_i,\mathbf{y}_i)$ is a function relating feature vectors and labels, and (vii) $\Delta(\mathbf{y}_i,\mathbf{y})$ describes the actual distance between labels $\mathbf{y}_i$ and $\mathbf{y}$.

In Equation (1), 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]. High values of C will impose a very large penalty on misclassified examples, causing the SSVM to select a lower margin hyperplane in order to avoid misclassifications. Conversely, low values of C will reduce the penalty on misclassified examples, allowing the SSVM to select a higher margin hyperplane. If C is set too high, the SSVM will over-fit the training data. On the other hand, if C is set too low, the SSVM will have a high error rate when assigning labels to the input data due to too many points being located within the margin.

Equation (1) can be solved via a *cutting plane algorithm* [5, 7]. 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 (ref. Equation (2)).

Equation (2) describes constraints imposed by each incorrect labelling $\mathbf{y}$ that allow the model parameter $\mathbf{w}$ to be learnt. The value of the slack variable ξ_i must be minimized such that the difference in score the SSVM calculates between the correct and incorrect label is greater than or equal to the loss function $\Delta(\mathbf{y}_i,\mathbf{y})$ minus ξ_i. For the training data, both $\mathbf{x}$ and $\mathbf{y}$ are observed. However, for the test data, only $\mathbf{x}$ is observed.

Tsochantaridis et al. [19] showed that, if the separation oracle can run in polynomial time, then the entire optimization problem can run in polynomial time. The complexity of the separation oracle is equivalent to solving a MAP inference problem with the addition of the constant loss function term [5]. This can be seen by considering Equation (3), which rearranges Equation (2) to show when a constraint is violated. Equation (4) calculates the most violated constraint while Equation (5) gives the general form of MAP inference [5].

$$\exists i\ \exists \mathbf{y} \in \mathcal{Y} \backslash \mathbf{y}_i\text{: } \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}_i,\mathbf{y}) - \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}_i,\mathbf{y}_i) + \Delta(\mathbf{y}_i,\mathbf{y}) - \xi_i > 0, \quad (3)$$

$$\widehat{\mathbf{y}} = \underset{\mathbf{y} \in \mathcal{Y}}{\operatorname{argmax}} \left(\mathbf{w}^T \mathbf{\Psi}(\mathbf{x}_i,\mathbf{y}) - \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}_i,\mathbf{y}_i) + \Delta(\mathbf{y}_i,\mathbf{y}) \right), \quad (4)$$

and

$$\underset{\mathbf{y} \in \mathcal{Y}}{\operatorname{argmax}} \ \left(\mathbf{w}^T \mathbf{\Psi}(\mathbf{x},\mathbf{y}) \right). \quad (5)$$

Fortunately, MAP inference is known to have a polynomial run time for the specific case of graphical model in which the potential functions at each node are known to be associative. An exact solution for this case can be obtained by taking minimum cuts in graphs [9].

3. OUR PROPOSED STRUCTURAL SVM (SSVM) WITH MAP INFERENCE CALCULATED USING MINIMUM GRAPH CUTS

In this section, we describe our proposed structural support vector machine (SSVM) with maximum *a posteriori* (MAP) inference calculated using minimum graph cuts. Specifically, we describe (i) the graph structure, (ii) feature vectors, and (iii) training labels of our SSVM.

3.1 Graph Structure

The stock market is generally 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 the stock price, 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. In this project, we use the companies appearing in the Information Technology sector of the S&P 500 Index [16].

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 (as the complexity is exponential in the tree-width of the graph). Fortunately, the collaboration relationship is an associative one. In other words, if two companies are collaborators, then 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: (Q1) collaboration between Companies A and B, (Q2) competition between Companies A and B. If query Q1 has more than 1.5 times the results of query Q2, an edge is added to the graph connecting the nodes for Companies A and B. This multiplier is necessary to avoid adding edges between companies which are not strong collaborators or competitors and thus have extremely similar result counts for both queries. This process was automated via the Bing Search API from the Windows Azure Marketplace [2]. The values on the edge vectors are initialized to one, requiring the SSVM to learn the labelling from the feature vectors at the nodes.

3.2 Feature Vectors

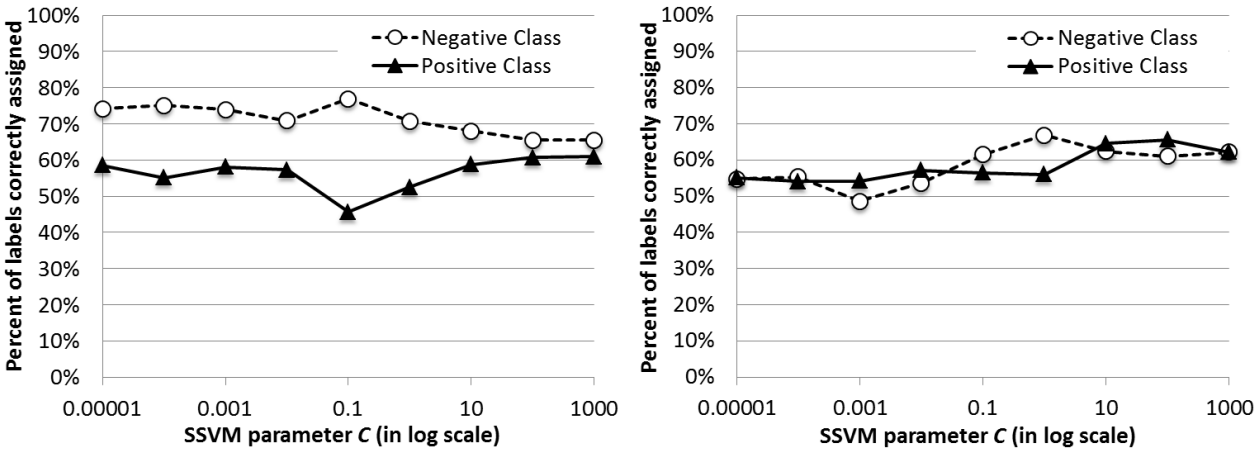
Each node in the graph has a 69-element feature vector containing fundamental financial information for the company represented by that node. Raw data for the feature vectors was obtained in comma-separated value (csv) format from Quandl via their API for stock fundamentals [15] (e.g., Open Financial Data Project data containing all financial ratios for companies such as Apple Inc. (AAPL), . . ., Yahoo! Inc. (YHOO)).

Stock fundamentals partly consist of both current and historical financial information for a company. Publicly traded companies in the USA are required to disclose their financial information in annual 10-K and quarterly 10-Q filings with the US 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 include (i) the number of shares outstanding, (ii) net margin, (iii) ratio of fixed assets to total assets, and (iv) 3-year standard deviation of stock price.

Some features such as “Book Value of Assets" could be valued in the hundreds of thousands 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 very difficult. To address this, we automated additional processing of the csv files to calculate the zero-mean unit-deviation normalized form of the features. This transformation is shown below:

$$x_f = \frac{x_i - \mu}{\sigma}, \quad (6)$$

where (i) x_f is the final value of the feature, (ii) x_i is the initial value of the feature, (iii) μ is the mean, and (iv) σ is the standard deviation.



(a) 2-fold cross validation

(b) 3-fold cross validation

Figure 1: Cross validation results for 2-fold and 3-fold tests

3.3 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 increases in stock price in the months after the release of financial data were labelled with the *positive class* while nodes whose companies saw decreases in stock price were labelled with the *negative class*. Historical stock price data was obtained for all companies from Yahoo! Finance [20] (e.g., Adobe Systems Incorporated (ADBE), . . ., Yahoo! Inc. (YHOO)).

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. In addition, some of the csv files inexplicably had Windows line endings while others had UNIX line endings so processing was also done to standardize this aspect across all csv files.

4. EXPERIMENTAL RESULTS

To evaluate our proposal, we conducted several experiments on a MacBookPro8,2 with a 2GHz Intel Core i7-2635QM processor and 8GB memory. The SSVM training and testing was performed using the dlib-C++ library with 4 threads [8]. The compiler used was clang-500.2.79 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. All results reported have been averaged over three runs.

4.1 Cross Validation

One of the important parts of training an SSVM is choosing the best value for the parameter C. As different datasets may behave differently, 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 SVM (or SSVM) using only a subset of those partitions. The trainer 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 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 $\langle$percentage of positive class labels correctly assigned, percentage of negative class labels correctly assigned$\rangle$ tuple. Ideally, the perfect values would be $\langle 1, 1 \rangle$, indicating that all nodes were correctly classified and no node was incorrectly classified. Results for 2-fold and 3-fold cross validation on the stock collaborators dataset are shown in Figure 1, which show that the two lines representing percentages of positive and negative class labels being correctly assigned are closely related. If the SSVM finds a hyperplane that favours negative class labels over positive class labels, we would expect to see (i) a higher percentage of negative class labels correctly classified and (ii) a lower percentage of positive class labels being correctly classified. The best numbers are obtained in the case of 3-fold cross validation, where the parameter C is set to 1000. In this case, the tuple returned is $\langle 62.22\%, 62.24\% \rangle$, indicating that—regardless of whether or not the assigned label indicated the positive class or the negative class—it was correct approximately 62% of the time. Note that, although the number of training examples used was not too large, 62% is clearly above the realm of random chance.

4.2 Test Data Accuracy

After choosing the value of C following the cross validation process, the SSVM was (i) trained with all the training examples and then (ii) tested for accuracy on those training data as well as on new data not used in any training. The results for this test are shown in Table 1. The SSVM was observed to have a much higher accuracy on the training samples than it did on the test samples.

For the training samples, the percentage of labels correctly classified was higher than 78%. This shows that (i) the SSVM has learnt the problem well enough to correctly classify the majority of nodes in the training examples and (ii) it has not over-fit the data (otherwise the accuracies would all be 100%).

The test samples may seem to elicit much lower accuracies, with all but one of them higher than 50%. One of the reasons for these lower numbers could be that the number of training examples was not large enough to truly learn all the details of the model. As future work, we plan to conduct more extensive experiments with larger training examples. Another reason is that the quality of the test samples used was much lower than the training samples. This is because the training samples used the most recent stock fundamental data and hence had relevant values for nearly all of their features. The test samples, on the other hand, used older data. Values

Table 1: SSVM accuracy for training and test data

Sample type & No.	Accuracy of SSVM predictions (correct labels)		
	% of total	# / Total positive class	# / Total negative class
Training sample 1	79.1045%	37 / 42	16 / 25
Training sample 2	79.1045%	27 / 35	26 / 32
Training sample 3	86.5672%	57 / 59	1 / 8
Training sample 4	100%	66 / 66	N/A
Training sample 5	98.4848%	0 / 1	65 / 65
Training sample 6	78.4615%	16 / 22	35 / 43
Training sample 7	78.125%	31 / 40	19 / 24
Test sample 1	68.254%	42 / 45	1 / 18
Test sample 2	50.8475%	22 / 31	8 / 28
Test sample 3	64.4068%	37 / 56	1 / 3
Test sample 4	73.2143%	1 / 11	40 / 45
Test sample 5	57.4074%	1 / 21	30 / 33
Test sample 6	37.7358%	20 / 53	N/A

for some features were not available for these older data, In some cases, entire stocks were missing and had to be excluded from the graph structure. However, on the positive side, some domain experts felt that our proposal led to a higher accuracy than many existing stock prediction software systems. As future work, we plan to conduct quantitative analyses for comparing the accuracy of our approach with existing systems.

5. CONCLUSIONS

Data mining and machine learning approaches can be incorporated into business intelligence (BI) systems to help users for decision support in many real-life applications. In this paper, we focused on an interesting application of stock price prediction. Specifically, we used minimum graph cuts as parts of a cutting plane algorithm to solve the optimization problem of the structural support vector machine (SSVM). This allowed the SSVM to learn a prediction model for a complex graph input with multiple edges per node (representing complex relationships between companies that affect the stock price by using feature vectors that contain fundamental financial information). The resulting model was applied to the problem of stock price prediction, where the positive and negative class labels corresponded to increasing and decreasing stock prices respectively. We used 3-fold cross validation to determine that an appropriate value for the SSVM parameter C was 1000. Accuracy of the SSVM on the training samples—in terms of correctly classified nodes—was higher than 78%, suggesting the model was learnt successfully without over-fitting. This shows the effectiveness of our machine learning approach—using SSVM classification augmented with minimum graph cuts—in a practical BI application of stock price prediction.

Acknowledgements. This project is partially supported by NSERC (Canada) and University of Manitoba.

6. REFERENCES

[1] A. Ben-Hur and J. Weston. A user's guide to support vector machines. In O. Carugo and F. Eisenhaber (eds.), *Data Mining Techniques for the Life Sciences*, pp. 223–239. Springer, 2010.

[2] Bing Search API. Windows Azure Marketplace. http://datamarket.azure.com/dataset/bing/search

[3] N.K. Chowdhury and C.K.-S. Leung. Improved travel time prediction algorithms for intelligent transportation systems. In *Proc. KES 2011, Part II*, pp. 355–365.

[4] A. Cuzzocrea, C.K.-S. Leung, and R.K. MacKinnon. Mining constrained frequent itemsets from distributed uncertain data. *FGCS*, 37: 117–126, July 2014.

[5] T. Finley and T. Joachims. Training structural SVMs when exact inference is intractable. In *Proc. ICML 2008*, ACM, pp. 304–311. ACM.

[6] D. Hodges. Is your fund manager beating the index? *MoneySense*, 15(7):10, Dec. 2013/Jan. 2014.

[7] T. Joachims, T. Finley, and C.-N.J. Yu. Cutting-plane training of structural SVMs. *Machine Learning*, 77(1): 27–59, Oct. 2009.

[8] D.E. King. Dlib-ml: a machine learning toolkit. *JMLR*, 10: 1755–1758, July 2009.

[9] V. Kolmogorov and C. Rother. Minimizing nonsubmodular functions with graph cuts-a review. *IEEE TPAMI*, 29(7): 1274–1279, July 2007.

[10] C.K.-S. Leung, F. Jiang, and Y. Hayduk. A landmark-model based system for mining frequent patterns from uncertain data streams. In *Proc. IDEAS 2011*, pp. 249–250. ACM.

[11] C.K.-S. Leung, F. Jiang, L. Sun, and Y. Wang. A constrained frequent pattern mining system for handling aggregate constraints. In *Proc. IDEAS 2012*, pp. 14–23. ACM.

[12] C.K.-S. Leung, S.K. Tanbeer, B.P. Budhia, and L.C. Zacharias. Mining probabilistic datasets vertically. In *Proc. IDEAS 2012*, pp. 199–204. ACM.

[13] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[14] K.P. Murphy. *Machine Learning: a Probabilistic Perspective*. MIT Press, 2012.

[15] Quandl Stock Index. http://www.quandl.com/stocks

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

[17] C. Sammut and G.I. Webb (eds.) *Encyclopedia of Machine Learning*. Springer, 2010.

[18] S.K. Tanbeer, C.K.-S. Leung, and J.J. Cameron. Interactive mining of strong friends from social networks and its applications in e-commerce. *JOCEC*, 24(2-3): 157–173, 2014

[19] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *JMLR*, 6: 1453–1484, Sept. 2005.

[20] Yahoo! Finance - Canada. http://ca.finance.yahoo.com