Optimizing a Pseudo Financial Factor Model with Support Vector Machine and Genetic Programming

Matthew Butler and Vlado Kešelj

Faculty of Computer Science, Dalhousie University,
6050 University Ave., Halifax, NS, Canada B3H 1W5
mbutler@cs.dal,ca, vlado@cs.dal.ca

Abstract. We compare the effectiveness of Support Vector Machines (SVM) and Tree-based Genetic Programming (GP) to make accurate predictions on the movement of the Dow Jones Industrial Average (DJIA). The approach is facilitated though a novel representation of the data as a pseudo financial factor model, based on a linear factor model for representing correlations between the returns in different assets. To demonstrate the effectiveness of the data representation the results are compared to models developed using only the monthly returns of the inputs. Principal Component Analysis (PCA) is initially used to translate the data into PC space to remove excess noise that is inherent in financial data. The results show that the algorithms were able to achieve superior investment returns and higher classification accuracy with the aid of the pseudo financial factor model. As well, both models outperformed the market benchmark, but ultimately the SVM methodology was superior in terms of accuracy and investment returns.

Key words: support vector machines, genetic programming, financial forecasting, principle component analysis

1 Introduction

We concentrate on and compare the results from a SVM and tree-based GP, performed in LIBSVM [1] and lilgp [2], respectively. To demonstrate the effectiveness of using the pseudo financial factor model the algorithm outputs are compared to models developed from using only the monthly changes of the inputs. The information that is used to generate the predictions is macro-economic data such as information on inflation and corporate bond ratings. The relationship between market movements and macro-economic data is not linear or monotonic. To assist in modeling these relationships a financial factor model is created that represents correlations between the market and each indicator in the input set. The combination of financial factor modeling with machine learning was explored by Azzini and Tettamanzai [3], where they implemented an evolving neural network to create a factor model to explain the returns in the DJIA. The

canonical form of a linear financial factor model is shown below:

$$r_i = b_{i1} \cdot f_1 + b_{i2} \cdot f_2 + \ldots + b_{im} \cdot f_m + \epsilon_i$$

r_i is the return on asset i, m is the number for factors, b_{ij} is the change in return of asset i per unit change in factor j, f_j is the change in return of factor j, and ϵ_i is the portion of the return in asset i not related to m factors. Traditionally, a financial factor model would be used to explain the returns of an asset by an equation and when the model output and the actual return begin to diverge, appropriate investments are made under the assumption that the two will converge again in the near future. As the name of the paper suggests, we are not using the equation in the traditional sense but changing the left hand side of the equation to be a class rather than a price level. A class of 1 indicates the DJIA will rise over the next month and -1 suggests it will fall over the same time period. The new pseudo equation is thus:

$$r_i = b_{i1} \cdot f_1 + b_{i2} \cdot f_2 + \ldots + b_{im} \cdot f_m$$

where $r_i \in \{1, -1\}$.

2 Data Description and Preprocessing

The data used to train the models was based on macro-economic data that was utilized by Enke and Thawornwong [4], where they created a market prediction model that outperformed the S&P 500 market index with the aid of a multilayer perceptron. The monthly changes in the indicators were combined with their respective β to create the inputs. The β calculation is shown below:

$$\beta(DJIA, X_i) = \frac{cov(DJIA, X_i)}{var(X_i)}$$

The β value is an indication of how much the market would move based on a unit movement of 1 in a given indicator; it was calculated on a rolling 10-year period.

Both algorithms were trained on data from 1977 to 2001 and then tested for 84 months or 7 years up until June 2008. The justification for the extended training period was to expose each model to market reactions during each stage of the business cycle. Initially the data is projected into principle component (PC) space where 99% of the variance is equated for, than the data is projected back into attribute space with a proper rank and excess noise removed.

3 Trading Strategy and Results

3.1 Trading Strategy

The experiment is setup as a semi-active trading strategy where at the beginning of each month a prediction is made as to whether or not the DJIA will contract or

expand over the coming month. If the prediction is for the market to go up, than the model will take a long position; conversely, if the market is predicted to fall, than a short position will be taken. Several financial instruments are available to short the DJIA, but essentially they all profit from market contractions.

3.2 Testing Results

The best model for both algorithms from each data set, determined by the training results, was supplied the out-of-sample data that spanned 84 months from 2001 up until June of 2008. In Table 1 we display the testing results for each algorithm. The investment returns are based off an initial investment of $1000 and for simplicity reasons transaction costs are ignored. Reported in the results is the Sharpe Ratio, which is a gauge of how much additional return the trading system generates for the extra risk it is exposed to—the higher the Sharpe ratio the better the risk-adjusted performance.

Table 1. Testing results for each algorithm and data set

	SVM		**GP**	
	Factors	**No Factors**	**Factors**	**No Factors**
Overall Accuracy	69.05%	59.52%	66.67%	57.72%
Precision (contractions)	68.60%	60.00%	63.41%	55.55%
# of contraction predictions	35	25	41	27
Precision (expansions)	69.40%	59.30%	69.76%	57.89%
Yearly investment yield (%)	21.70%	4.13%	16.00%	4.04%
Cumulative return ($)	$4505	$1335	$3042	$1327
Excess return to the market[1]	20.60%	3.03%	14.90%	2.94%
Sharpe ratio[2]	3.994	0.822	2.813	0.795

The testing results in Table 1 clearly show the advantages of using the financial factor model to create the inputs for the SVM and GP algorithms, where the overall accuracy and investment return were superior.

4 Discussion and Conclusions

In this study we compared the effectiveness of a novel data representation to optimize SVM and GP trading models to make accurate predictions on the movement of the DJIA. In each of the performance measures the algorithms achieved superior performance when the inputs reflected the pseudo financial factor model. Precision for contraction predictions was of particular interest in

[1] DJIA yearly investment return over testing period was 1.10%.
[2] The risk-free rate in the calculation was replaced by the market rate.

this study due to the trading strategy. Since we are investing directly in the DJIA and also using it as the benchmark the only way to overperform is to avoid market contractions. This can be done in one of two ways, exiting the market and investing in a risk-free rate or alternatively short-selling the market to profit directly from its decline. The later is a much more aggressive approach and was the one utilized in our study; therefore incorrect market contractions will lead to the investment losing money while the market is increasing. As a result a precision for contraction predictions of less that 50% will most likely lead to inferior investment returns.

The effectiveness of using the factor model could be explained by the fact that the algorithms are given more information about the problem with this type of data representation. Not only is the model training on the returns of the indicators but they are also supplied a ratio that describes the relationship between said indicator and the market. This enables the algorithm to have a more complete picture and therefore is able to create a more robust market model. Each of the models presented in this paper were able to outperform the DJIA, however the non-financial factor models did so by a much smaller margin. Ultimately the SVM proved to be the most effective in terms of risk and return, it's Sharpe ratio was the highest reflecting the most efficient use of the extra risk the model took on to achieve the excess returns. The obtained results for investment returns are not entirely accurate as transaction costs were ignored. However, because the trading strategy was semi-active and only made trades on a month to month basis, and only if required, the transaction costs would be less inhibitory to overall profits than that of other more active trading approaches.

References

1. Chang, C.C., Lin, C.J.: LIBSVM: A library for support vector machines (2001) Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.
2. Punch, B., Zongker, D.: Michigan State University GARAGe (1998) Web site of Genetic Algorithms Research and Applications Group (GARAGe), `http://garage.cse.msu.edu/`.
3. Azzini, A., Tettamanzi, A.: A neural evolutionary approach to financial modeling. In: GECCO 06: Proceedings of the 2006 Conference on Genetic and Evolutionary Computation. (2006) 1605–1612
4. Enke, D., Thawornwong, S.: The use of data mining and neural networks for forecasting stock market returns. Expert Systems with Applications **29** (2005) 927–940