Law Discovery from Financial Data using Neural Networks

Kazumi Saito, Naonori Ueda, Shigeru Katagiri
NTT Communication Science Laboratories
{saito,ueda,katagiri}@cslab.kecl.ntt.co.jp

Yutaka Fukai, Hiroshi Fujimaru, Masayuki Fujinawa
Tokyo-Mitsubishi Asset Management Ltd.
{y-fukai,fujimaru,m-fujinawa}@tmam.co.jp

Abstract

This paper describes an experimental study for discovering underlying laws of market capitalization using BS (Balance Sheet) items. For this purpose, we apply law discovery methods based on neural networks: RF5 (Rule Finder) discovers a single numeric law from data containing only numeric values, RF6 discovers a set of nominally conditioned polynomials from data containing both nominal and numeric values, and MCV regularizer is used to improve both the generalization performance and the readability. Our preliminary experimental results show that these methods are promising for discovering underlying laws from financial data.

1 Introduction

The discovery of a numeric law, e.g., Kepler's third law $T = kr^{3/2}$, from data is the central part of scientific discovery systems. By using such methods, we can expect to discover underlying laws in various types of scientific and business data including financial data. As a traditional AI (Artificial Intelligence) approach, the BACON systems (Langley, 1978; Langley et al., 1987) and many variants (Langley and Zytkow, 1989; Falkenhainer and Michalski, 1990; Nordhausen and Langley, 1993) have employed a recursive combination of two variables, but these systems have suffered from combinatorial explosion problems, troublesome preparations of appropriate model functions, and a lack of robustness against unknown samples. As an alternative to these conventional methods, we have proposed a connectionist approach that specially employs a generalized polynomial network structure referred to as a Rule Finder (RF) (see (Saito and Nakano, 1990)).

Among several implementations of the RF methods, we use two selections: RF5 (Saito and Nakano, 1997b) and RF6 (Nakano and Saito, 1999). We are indeed surrounded by a lot of numeric data, and RF5 is a method developed to effectively discover numeric laws from such numeric data. In contrast, RF6 is a method for coping with another type of data, which consists of both numeric and nominal values that can be easily observed in our daily life.

Moreover, data usually contain several irrelevant variables. Since these variables only work to overfit the training data, the generalization performance is generally degraded. Here the generalization means the performance with new data. In order to discover relevant weights of neural networks, we have proposed a new method, called *MCV regularizer* (Saito and Nakano, 2000), to learn a distinct squared penalty factor for each weight as a minimization problem using the cross-validation error. Since only the irrelevant weights are severely penalized, and these values are expected to be very small, the corresponding irrelevant variables will be negligible. In addition, this makes the interpretation of the discovered laws easy.

This paper describes an experimental study for discovering laws of market capitalization using BS (Balance Sheet) items. Section 2 briefly explains the law discovery methods used in this study. Section 3 reports our experimental results.

2 Law Discovery Methods

2.1 RF5 algorithm

Let $(x_1, \cdots, x_K, y)$ be a vector of variables describing each example, where x_k is a numeric explanatory variable and y is a numeric target variable. As a class of numeric formula $y(\boldsymbol{x}; \boldsymbol{\Theta})$, we consider a generalized polynomial, whose power values are not restricted to integers, expressed by

$$\begin{aligned} y(\boldsymbol{x}; \boldsymbol{\Theta}) &= w_0 + \sum_{j=1}^{J} w_j \prod_{k=1}^{K} x_k^{w_{jk}} \\ &= w_0 + \sum_{j=1}^{J} w_j \exp\left(\sum_{k=1}^{K} w_{jk} \ln x_k\right). \quad (1) \end{aligned}$$

Here each parameter w_j or w_{jk} is an unknown real number, and J is an unknown integer corresponding to the number of terms. $\mathbf{\Theta}$ is an M-dimensional parameter vector constructed by arranging parameters $w_j, j = 0, \cdots, J$, and $w_{jk}, j = 1, \cdots, J, k = 1, \cdots, K$. Note that Eq. (1) can be regarded as a feedforward computation of a three-layer neural network (Saito and Nakano, 1997b).

Let $D = \{(\boldsymbol{x}^\mu, y^\mu), \mu = 1, \cdots, N\}$ be a set of training data, where N is the number of examples. Here we assume that each training example $(\boldsymbol{x}^\mu, y^\mu)$ is independent and identically distributed. Now, our ultimate goal in discovering laws is defined as a problem of minimizing the generalization error, that is, finding the optimal estimator $\mathbf{\Theta}^*(D)$ that minimizes

$$\mathcal{G}(\mathbf{\Theta}^*) = E_D E_T \left(y^\nu - y(\boldsymbol{x}^\nu; \mathbf{\Theta}^*(D))\right)^2, \quad (2)$$

where $T = (\boldsymbol{x}^\nu, y^\nu)$ denotes test data independent of the training data D.

The *least-squares estimate* of $\mathbf{\Theta}^*$, denoted by $\widehat{\mathbf{\Theta}}$, minimizes the error sum of squares

$$\mathcal{E}_1(\mathbf{\Theta}) = \frac{1}{2}\sum_{\mu=1}^{N} \left(y^\mu - y(\boldsymbol{x}^\mu; \mathbf{\Theta})\right)^2. \quad (3)$$

In order to efficiently and constantly obtain good learning results, we employ a second-order learning algorithm called *BPQ* (Saito and Nakano, 1997a). By adopting the quasi-Newton method (Gill et al., 1981; Luenberger, 1984) as a basic framework, we calculate the descent direction on the basis of a partial BFGS update and then efficiently calculate a reasonably accurate step-length as the minimal point of a second-order approximation.

2.2 RF6 algorithm

Let $(q_1, \cdots, q_{K_1}, x_1, \cdots, x_{K_2}, y)$ be a vector of variables describing an example, where q_k is a nominal explanatory variable. For each q_k we introduce a *dummy variable* q_{kl} defined by

$$q_{kl} = \begin{cases} 1 & \text{if } q_k \text{ matches the } l\text{-th category}, \\ 0 & \text{otherwise}, \end{cases} \quad (4)$$

where $l = 1, \cdots, L_k$, and L_k is the number of distinct categories appearing in q_k. As a true model governing data, we consider the following set of multiple rules

$$\begin{cases} if \bigwedge_{q_{kl} \in Q^{(1)}} q_{kl} = 1 \quad then \quad y = y(\boldsymbol{x}; \mathbf{\Theta}^{(1)}), \\ \cdots \quad \cdots \quad \cdots \quad \cdots, \\ if \bigwedge_{q_{kl} \in Q^{(R)}} q_{kl} = 1 \quad then \quad y = y(\boldsymbol{x}; \mathbf{\Theta}^{(R)}) \end{cases} \quad (5)$$

where $Q^{(r)}$ denotes a set of dummy variables corresponding to the r-th nominal condition and R is the number of rules.

We introduce a framework to learn nominal conditions from data.

$$g(\boldsymbol{q}; \boldsymbol{V}^{(r)}) = \exp\left(\sum_{k=1}^{K_1}\sum_{l=1}^{L_k} v_{kl}^{(r)} q_{kl}\right), \quad (6)$$

where $\boldsymbol{V}^{(r)}$ denotes a vector of parameters $v_{kl}^{(r)}$. By setting the adequate values of $\boldsymbol{V}^{(r)}$, we can see that the following formula can closely approximate the final output value from a set of multiple nominally conditioned polynomials, defined by Eq. (5).

$$F(\boldsymbol{q}, \boldsymbol{x}; \mathbf{\Psi}) = \sum_{r=1}^{R} g(\boldsymbol{q}; \boldsymbol{V}^{(r)})\; y(\boldsymbol{x}; \mathbf{\Theta}^{(r)}). \quad (7)$$

Here $\mathbf{\Psi}$ consists of all parameters $\boldsymbol{V}^{(r)}, \mathbf{\Theta}^{(r)}, r = 1, \cdots, R$. With an adequate number J, the following can completely describe Eq. (7),

$$y(\boldsymbol{q}, \boldsymbol{x}; \mathbf{\Theta}) = w_0 + \quad (8)$$
$$\sum_{j=1}^{J} w_j \exp\left(\sum_{k=1}^{K_1}\sum_{l=1}^{L_k} v_{jkl}\; q_{kl} + \sum_{k=1}^{K_2} w_{jk} \ln x_k\right).$$

Eq. (8) can also be regarded as the feedforward computation of a three-layer neural network.

2.3 MCV regularizer

To improve both the generalization performance and the readability, we consider a distinct penalty factor for each weight. Hereafter, an M-dimensional vector $\boldsymbol{\lambda}$ is defined by $\boldsymbol{\lambda} = (\lambda_1, \cdots, \lambda_M)^T$, and an M-dimensional diagonal matrix $\mathbf{\Lambda}$ is defined by $diag(\mathbf{\Lambda}) = (\exp(\lambda_1), \cdots, \exp(\lambda_M))$, where $\boldsymbol{a}^T$ denotes a transposed vector of $\boldsymbol{a}$. Here, since the penalty factor must be non-negative, we adopted $\exp(\lambda_m)$, instead of a standard parameterization λ_m. As a result, the discovery of laws subject to Eq. (1) can be defined as the following learning problem in neural networks.

$$\mathcal{E}(\mathbf{\Theta}) = \mathcal{E}_1(\mathbf{\Theta}) + \frac{1}{2}\mathbf{\Theta}^T \mathbf{\Lambda} \mathbf{\Theta}. \quad (9)$$

In our own experiments (Saito and Nakano, 1997c), the combination of the squared penalty term and the BPQ algorithm drastically improves the convergence performance in comparison to the other combinations, while at the same time bringing about excellent generalization performance.

We introduce an objective function for penalty factors derived from the procedure of *cross-validation*, which is frequently used to evaluate the generalization performance of learned networks (Bishop, 1995). The procedure of cross-validation divides the data D at random into S distinct segments $(G_s,\ s = 1, \cdots, S)$; it uses $S - 1$ segments for training, and uses the remaining one for the test. This process is repeated

Table 1: target and explanatory variables

names	meanings
y	market capitalization
x_1	current assets
x_2	property and equipment
x_3	total assets
x_4	current liabilities
x_5	long-term debt
x_6	total liabilities

S times by changing the remaining segment, and the generalization performance is evaluated by using the following MSE (mean squared error) over all S test results.

$$MSE_{CV} = \frac{1}{N}\sum_{s=1}^{S}\sum_{\nu\in G_s}\left(y^\nu - y(\boldsymbol{x}^\nu;\widehat{\boldsymbol{\Theta}}_s)\right)^2 \quad (10)$$

Here $\widehat{\boldsymbol{\Theta}}_s$ denotes the optimal weights obtained by minimizing the following objective function for weights

$$\mathcal{E}_s(\boldsymbol{\Theta}_s) = \frac{1}{2}\sum_{\mu\notin G_s}(y^\mu - y(\boldsymbol{x}^\mu;\boldsymbol{\Theta}_s))^2 + \frac{1}{2}\boldsymbol{\Theta}_s^T\boldsymbol{\Lambda}\boldsymbol{\Theta}_s. \quad (11)$$

The extreme case of $S = N$ is known as the *leave-one-out* method, which is often used for a small data size. Note that Eq. (10) is regarded as a reasonable approximation to Eq. (2) for a given data set D.

According to the *implicit function theorem*, since $\widehat{\boldsymbol{\Theta}}_s$ can be regarded as a vector consisting of implicit functions of $\boldsymbol{\lambda}$, Eq. (10) can be defined as the objective function for penalty factors. Thus, we can calculate the $\widehat{\boldsymbol{\lambda}}$ that minimizes Eq. (10). Then, with $\widehat{\boldsymbol{\lambda}}$, we calculate the $\widehat{\boldsymbol{\Theta}}$ that minimizes Eq. (9). Finally, $\widehat{\boldsymbol{\Theta}}$ is adopted as the final weight vector of the discovered law (Saito and Nakano, 2000).

3 Law Discovery from Financial Data

3.1 Experimental data

Our experiments used 953 companies listed on the first section of the Tokyo Stock Exchange, where banks, and insurance, securities, and recently listed companies were excluded. The market capitalization of each company was calculated by multiplying the shares of outstanding stocks with the stock price at the end of October, 1999. As the first step in this study, we selected six fundamental items from all of the BS items, and the values of these items were also calculated at the end of October, 1999. Table 1 summarizes target and explanatory variables.

3.2 Experimental settings

Before the analysis, the following scaling was applied to the variables: $\widetilde{y} = (y - avg(y))/std(y)$, and $\ln\widetilde{x}_k = (\ln x_k - avg(\ln x_k))/std(\ln x_k),\ k = 1,\cdots,6$. When a distinct squared penalty factor is adopted for each weight, this penalty term is known to be consistent with any linear scaling of input and/or output variables. Here consistency requires that we obtain equivalent networks that differ only by the scaling of the weights as given (Bishop, 1995).

In the experiments, the initial values for the weights w_{jk} were independently generated according to a normal distribution with a mean of 0 and a standard deviation of 1; the initial values for the weights w_j were set to 0. The initial values for the penalty factors $\boldsymbol{\lambda}$ were set to $\mathbf{0}$, i.e., $\boldsymbol{\Lambda}$ was set to the identical matrix. The iteration was terminated when the gradient vector was sufficiently small, i.e., $\max_m\{\|\partial/\partial\theta_m\ \mathcal{E}(\boldsymbol{\Theta})\|\} < 10^{-6}$ for learning over $\boldsymbol{\Theta}$; $\max_m\{\|\partial/\partial\lambda_m\ MSE_{CV}(\boldsymbol{\lambda})\|\} < 10^{-6}$ for learning over $\boldsymbol{\lambda}$. The cross-validation error was calculated by using the leave-one-out method, i.e., $S = N$.

Table 2: RF5+MCV results

	MSE		MSE_{CV}	
	avg	std	avg	std
$J=1$	0.2878	0.0031	0.3090	3.0814
$J=2$	0.2350	0.0029	0.2437	2.7761
$J=3$	0.2314	0.0026	0.2516	2.6004

3.3 RF5+MCV result

The number of hidden units was changed from 1 to 3 ($J = 1, 2, 3$). Table 2 shows the performance of the learning results. The MSE values were minimized when $J = 3$, while the MSE_{CV} values were minimized when $J = 2$. This indicates the suitable number of hidden units is considered to be 2. When $J = 2$, an example of the discovered laws was as follows:

$$\begin{aligned} y = & -25055.171654 \\ & +6.983186x_1^{+0.000008}x_2^{+0.013135}x_3^{+0.967862} \\ & \quad x_4^{+0.000000}x_5^{-0.000001}x_6^{-0.000001} \\ & -8.205773x_1^{+0.000005}x_2^{+0.000005}x_3^{+0.592110} \\ & \quad x_4^{+0.000002}x_5^{-0.006089}x_6^{+0.378892}, \end{aligned} \quad (12)$$

where the weight values were rounded off to the seventh decimal. These weight values were transformed to correspond to the original scale of variables. When the weight values were rounded off to the second decimal, the discovered law was as follows:

$$y = -25055.2 + 7.0x_3^{1.0} - 8.2x_3^{0.6}x_6^{+0.4}. \quad (13)$$

Note that existing numeric discovery methods cannot find such laws, as described in Eq. (13), without preparing some appropriate prototype functions. This point is an important advantage in RF5 over existing methods.

3.4 RF6+MCV result

Since laws of market capitalization can differ according to the type of industry, the 33 classifications of the Tokyo Stock Exchange were used as a nominal variable. In this experiment, to understand the effect of the nominal variable intuitively, we fixed the number of hidden units at 1. An example of the discovered laws was as follows:

$$\begin{aligned} y \;=\; & 12944.3 + 1.9\exp(-1.4q_1 - 1.3q_2 - 1.0q_3 \\ & -0.4q_4 - 0.6q_5 - 0.7q_6 - 0.5q_7 + 0.2q_8 \\ & -1.0q_9 - 0.6q_{11} - 0.9q_{12} - 0.3q_{13} - 1.6q_{20} \\ & -0.8q_{21} - 0.7q_{22} - 0.9q_{23} - 1.2q_{24} - 1.1q_{26} \\ & -1.0q_{31} - 0.9q_{32}) x_2^{+0.7} x_3^{+1.0} x_6^{-0.7}, \end{aligned} \tag{14}$$

where the above q_k denotes the k-th type of industry, and the weight values are rounded off to the second decimal. Here the MSE and MSE_{CV} values of the discovered law were 0.2365 and 0.2668, respectively. This formula can be easily decomposed into a set of rules. Since the second term on the right hand side of Eq. (14) is always positive, a weight for q_k indicates the price setting tendency of the k-th type of industry for similar BS situations. More specifically, Eq. (14) tells us that the pharmaceutical industry (q_8) is likely to have a high setting due to the largest weight (0.2), while the electric power or gas industry (q_{20}) is likely to have a low setting due to the smallest weight (-1.6).

4 Conclusion

This paper has reported an experimental study for discovering underlying laws of market capitalization using BS items. Namely, both a single numeric law and a set of nominally conditioned polynomials were discovered by using RF5 and RF6 algorithms. Our preliminary experimental results showed that these methods are promising for discovering laws from financial data. Currently, we are investigating possible interpretations of the laws discovered in experiments.

References

Bishop, C. (1995). *Neural networks for pattern recognition.* Oxford Press.

Falkenhainer, B. and Michalski, R. (1990). Integrating quantitative and qualitative discovery in the ABACUS system. In Kodratoff, Y. and Michalski, R., editors, *Machine learning: an artificial intelligence approach, Volume* III, pages 153–190. Morgan Kaufmann, San Mateo, CA.

Gill, P., Murray, W., and Wright, M. (1981). *Practical optimization.* Academic Press, London.

Langley, P. (1978). Bacon.1: a general discovery system. In *Proc. 2nd National Conferenee of the Canadian Soeiety for Computational Studies of Intelligenee*, pages 173–180.

Langley, P., Simon, H., Bradshaw, G., and Zytkow, J. (1987). *Scientific discovery: computational explorations of the creative process.* MIT Press, Cambridge, MA.

Langley, P. and Zytkow, J. (1989). Data-driven approaches to empirical discovery. *Artificial Intelligence*, 40:283–312.

Luenberger, D. (1984). *Linear and nonlinear programming.* Addison-Wesley, Reading, MA.

Nakano, R. and Saito, K. (1999). Discovery of a set of nominally conditioned polynomial. In *Proc. 2nd Int. Conference on Discovery Science, LNAI 1721*, pages 287–298, Tokyo, Japan.

Nordhausen, B. and Langley, P. (1993). An integrated framework for empirical discovery. *Machine Learning*, 12(1/2/3):17–48.

Saito, K. and Nakano, R. (1990). Rule extraction from facts and neural networks. In *Proc. Int. Neural Network Conference*, pages 379–382, Paris, France.

Saito, K. and Nakano, R. (1997a). BFGS update and efficient step-length calculation for three-layer neural networks. *Neural Computation*, 9(1):123–141.

Saito, K. and Nakano, R. (1997b). Law discovery using neural networks. In *Proc. 15th Int. Joint Conference on Artificial Intelligence*, pages 1078–1083, Nagoya, Japan.

Saito, K. and Nakano, R. (1997c). Second-order learning algorithm with squared penalty term. In *Advances in Neural Information Processing Systems 9*, pages 627–633, Denver, CO.

Saito, K. and Nakano, R. (2000). Law discovery using neural networks. In *Proc. 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Kyoto, Japan (to appear).