Optimal Pruned K-Nearest Neighbors: OP-KNN – Application to Financial Modeling

Q. Yu, A. Sorjamaa, Y. Miche, and A. Lendasse
Information and Computer Science Department
Helsinki University of Technology
IN-F Espoo, Finland

Eric Séverin
Laboratoire Economie
Management (LEM)
Lille cedex

A. Guillen
Department of Informatics
University of Jaén,
Jaén, Spain

F. Mateo
Department of Electronic Engineering
Polytechnic University of Valencia,
Valencia, Spain

Abstract

The paper proposes a methodology called OP-KNN, which builds a one hidden- layer feedforward neural network, using nearest neighbors neurons with extremely small computational time. The main strategy is to select the most relevant variables beforehand, then to build the model using KNN kernels. Multiresponse Sparse Regression (MRSR) is used as the second step in order to rank each k^{th} nearest neighbor and finally as a third step Leave-One-Out estimation is used to select the number of neighbors and to estimate the generalization performances. This new methodology is tested on a toy example and is applied to financial modeling.

1 Introduction

It is usual to have very long computational time for training a feedforward network using existing classic learning algorithms even for simple problems. Thus, Guang-Bin Huang in his paper [1] proposed an original algorithm called Extreme Learning Machine (ELM) for single-hidden layer feedforward neural networks (SLFN) which randomly chooses hidden nodes and analytically determines the output weights of SLFNs. The most significant characteristics of this method is that it tends to provide good generalization performance and a comparatively simple model at extremely high learning speed. But the remaining problem is the selection of the kernel, i.e. the activation function used between input data and the hidden layer.

In [8], Optimal Pruned Extreme Learning Machine (OP-ELM) has been proposed as an improvement of the original ELM. In this paper, a methodology similar to OP-ELM, called OP-KNN (for Optimal Pruned - k-Nearest Neighbors) is presented using KNN as the kernel. It has several notable achievements:

- keeping good performance while being simpler than most learning algorithms for feedforward neural network,
- the initial random step (ELM) is removed and replaced by a deterministic initialization (KNN),
- the computational time of OP-KNN being extremely low (lower than OP-ELM or any other algorithm). In our experiments, the computational time is less than a second (for a regression problem with 650 samples and 35 variables),
- for our application, Leave-One-Out (LOO) error is used both for variables selection [11] and OP-KNN complexity selection.

In the experimental Section, this paper deals with the explanation of corporate performance which is measured by ROA, ROE and Marris variables. We try to determine if the features of assets, the debt level or the cost structure have an influence on corporate performance. Our results highlight that the industry, the size, the liquidity and the dividend are the main determinants of corporate performance.

The main steps of the OP-KNN methodology are KNN, MRSR (for Multiresponse Sparse Regression) [7] and finally the LOO error validation [10], using PRESS statistic [3]. All these steps are detailed in the Section 2. To improve the methodology, a prior Variable Selection is performed to remove irrelevant input variables beforehand [11]. Section

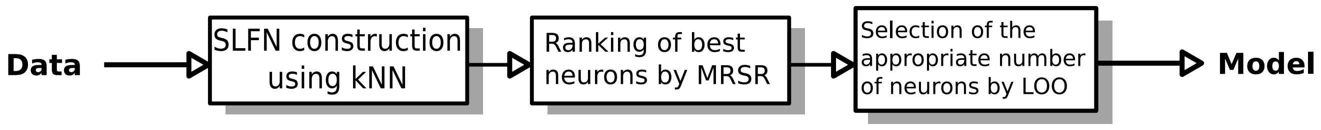

Data
SLFN construction using kNN
Ranking of best neurons by MRSR
Selection of the appropriate number of neurons by LOO
Model

Figure 1. The three steps of the OP-KNN algorithm.

3 shows the results on a toy example and on financial modeling.

2 Optimal Pruned – k-Nearest Neighbors

OP-KNN is similar to OP-ELM, which is an original and efficient way of training a Single-Layer Feedforward Network (SLFN). The three main steps of the OP-KNN are summarized in Figure 1.

2.1 Single-hidden Layer Feedforward Neural Networks (SLFN)

The first step of the OP-KNN algorithm is the core of the original ELM: the building of a single-layer feed-forward neural network. The idea of the ELM has been proposed by Guang-Bin Huang *et al.* in [1].

In the context of a single hidden layer perceptron network, let us denote the weights between the hidden layer and the output by $\mathbf{b}$. Activation functions used with the OP-KNN differ from the original SLFN choice since the original sigmoid activation functions of the neurons are replaced by the *k*-Nearest Neighbors, hence the name OP-KNN. For the output layer, the activation function remains as a linear function.

A theorem proposed in [1] states that the activation functions, output weights $\mathbf{b}$ can be computed from the hidden layer output matrix $\mathbf{H}$: the columns $\mathbf{h_i}$ of $\mathbf{H}$ are the corresponding output of the k-nearest-neighbors. Finally, the output weights $\mathbf{b}$ are computed by $\mathbf{b} = \mathbf{H}^{\dagger}\mathbf{y}$, where $\mathbf{H}^{\dagger}$ stands for the Moore-Penrose inverse [6] and $\mathbf{y} = (y_1, \dots, y_M)^T$ is the output.

The only remaining parameter in this process is the initial number of neurons N of the hidden layer.

2.2 *k*-Nearest Neighbors

The k-Nearest Neighbors (KNN) model is a very simple, but powerful tool. It has been used in many different applications and particularly in classification tasks. The key idea behind the KNN is that similar training samples have similar output values. In OP-KNN, the approximation of the output is the weighted sum of the outputs of the k-nearest neighbors. The model introduced in the previous subsection becomes:

$$\hat{y}_i = \sum_{j=1}^{k} b_j y_{P(i,j)} \qquad (1)$$

where $\hat{y}_i$ represents the output estimation, $P(i,j)$ is the index number of the j^{th} nearest neighbor of sample $\mathbf{x}_i$ and b is the results of the Moore-Penrose inverse introduced in the previous subsection.

2.3 Multiresponse Sparse Regression (MRSR)

For the removal of the useless neurons of the hidden layer, the Multiresponse Sparse Regression proposed by Timo Similä and Jarkko Tikka in [7] is used. It is an extension of the Least Angle Regression (LARS) algorithm [2] and hence is actually a variable ranking technique, rather than a selection one.

The main idea of this algorithm is the following: denote by $\mathbf{T} = [\mathbf{t}_1 \dots \mathbf{t}_p]$ the $n \times p$ matrix of targets, and by $\mathbf{X} = [\mathbf{x}_1 \dots \mathbf{x}_m]$ the $n \times m$ regressors matrix. MRSR adds each regressor one by one to the model

$$\mathbf{Y}^k = \mathbf{X}\mathbf{W}^k, \qquad (2)$$

where $\mathbf{Y}^k = [\mathbf{y}_1^k \dots \mathbf{y}_p^k]$ is the target approximation by the model and $\mathbf{W}^k$ the weight matrix. This weight matrix has k nonzero rows at kth step of the MRSR. With each new step a new nonzero row, and a new regressor to the total model, is added.

An important detail shared by the MRSR and the LARS is that the ranking obtained is exact in the case where the problem is linear. In fact, this is the case, since the neural network built in the previous step is linear between the hidden layer and the output. Therefore, the MRSR provides the exact ranking of the neurons for our problem.

Details on the definition of a cumulative correlation between the considered regressor and the current model's residuals and on the determination of the next regressor to be added to the model can be found in the original paper about the MRSR [7].

MRSR is hence used to rank the kernels of the model: the target is the actual output y_i while the ”variables” considered by MRSR are the outputs of the k-nearest neighbors.

2.4 Leave-One-Out (LOO)

Since the MRSR only provides a ranking of the kernels, the decision over the actual best number of neurons for the model is taken using a Leave-One-Out method. One problem with the LOO error is that it can get very time consuming if the dataset tends to have a high number of samples. Fortunately, the PRESS (or PREdiction Sum of Squares) statistics provide a direct and exact formula for the calculation of the LOO error for linear models. See [3, 4] for details on this formula and implementations:

$$\epsilon^{\text{PRESS}} = \frac{y_i - \mathbf{h}_i \mathbf{b}}{1 - \mathbf{h}_i \mathbf{P} \mathbf{h}_i^T}, \tag{3}$$

where $\mathbf{P}$ is defined as $\mathbf{P} = (\mathbf{H}^T\mathbf{H})^{-1}$ and $\mathbf{H}$ the hidden layer output matrix defined in subsection 2.1.

The final decision over the appropriate number of neurons for the model can then be taken by evaluating the LOO error versus the number of neurons used (properly ranked by MRSR already).

2.5 Discussion on the Advantages of the OP-KNN

In order to have a very fast and still accurate algorithm, each of the three presented steps have a special importance in the whole OP-KNN methodology. The K-nearest neighbor ranking by the MRSR is one of the fastest ranking methods providing the exact best ranking, since the model is linear (for the output layer), when creating the neural network using KNN.

Without MRSR, the number of nearest neighbor that minimizes the Leave-One-Out error is not optimal and the Leave-One-Out error curve has several local minima instead of a single global minimum. The linearity also enables the model structure selection step using the Leave-One-Out, which is usually very time-consuming. Thanks to the PRESS statistics formula for the LOO error calculation, the structure selection can be done in a small computational time.

3 Experiments

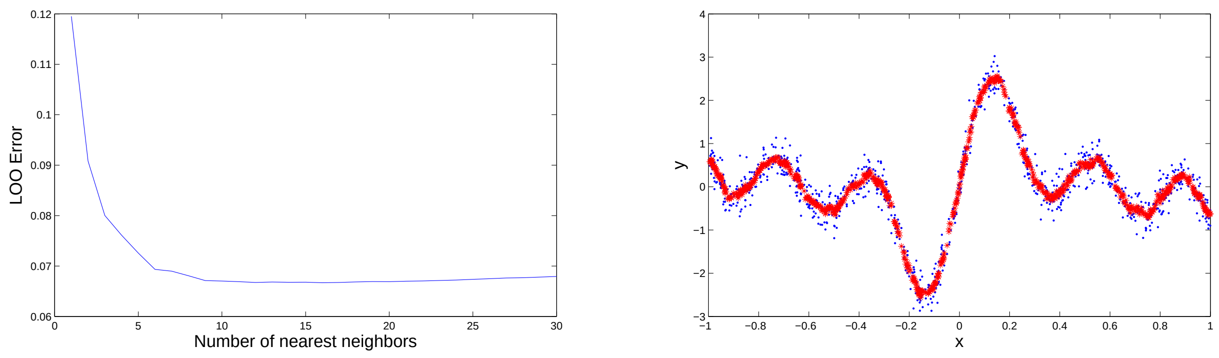
3.1 Sine in one dimension

In this experiments, a set of 1000 training points are generated (and represented in Fig. 2B), the output is a sum of two sines. This single dimension example is used to test the method without the need for variable selection beforehand. The Fig. 2A shows the LOO error for different number of nearest neighbors and the model built with OP-KNN using the original dataset. This model approximates the dataset accurately, using 18 nearest neighbors; and it reaches a LOO error close to the noise introduced in the dataset which is 0.0625. The computational time for the whole OP-KNN is one second (using Matlab© implementation).

Table 1. The meaning of variables

index	Variable	Meaning
1	Sector	Industry
2	Transaction	Number of shares exchanged during the year
3	Rotation	Security turnover rate
4	Vrif Rotation	Not useful
5	Net dividend	Amount of dividend for one share during the year
6	Effectifs	Number of employees
7	CA	Sales
8	II	Other assets
9	AMORII	Dotations on other assets
10	IC	Property, plant and equipement
11	AMORIC	Dotations on property, plant and equipement
12	IF	Not useful
13	AI	Fixed assets
14	S	Stocks or inventories
15	CCR	Accounts receivables
16	CD	Not useful
17	L	Cash in hands and at banks
18	AC	Total of current assets
19	CPPG	Total of capital of group (in book value)[a]
20	PRC	Not useful
21	FR	Accounts payables
22	DD	Not useful
23	DEFI	Financial debt
24	Debt-1AN	Debt whose maturity is inferior to 1 year
25	Debt+1AN	Debt whose maturity is superior to 1 year
26	TD	Total Debt
27	CA	Sales
27	CPER	Cost of workers
28	CPO	Not useful
29	DA	Dotations on amortizations
30	REXPLOI	Operating income before tax
31	CFI	Interests taxes
32	RFI	Financial income
33	RCAI	Operating income before tax + Financial income
34	REXCEP	Extraordinary item
35	IS	Taxes from State
Output1	ROA	net income / total assets
Output2	ROE	net income / capital
Output3	MARRIS	Market to book

[a]By construction the total debt is equal to Total assets



(a) (b)

Figure 2. Sine Toy example

3.2 Financial Modeling

In this experiment, we use the data [11] related to 200 French companies during a period of 5 years. 35 input variables are used, these input variables are financial indicators that are measured every year (for example debt, number of employees, amount of dividends, . . .). The target variables are

- The ROA defined as the ratio between the net income and the total assets.
- The ROE represents the ratio between the net income and the capital.
- The Marris (or Q ration) is calculated by dividing the market value of shares by the book value of shares

Table 1 shows the real meaning in financial field about all the variables we have used.

All these three targets are tested one by one using OP-KNN; Variable Selection (VS) and Variable Selection+Scaling are performed beforehand [11] for comparison. The results are listed in Tables 2 to 7 where we can see the LOO error are decreased almost half with variable selection step for each cases.

The minimum LOO error is achieved when using OP-KNN on the scaled selected input variables as expected. Moreover, the final LOO error reaches roughly the same stage as the value we estimated while doing variable selection [9]. Thus, for this financial dataset, this methodology not only build the model in a simple and fast way, but also prove the accuracy of our previous selection algorithm [9, 11].

It should be noted that on the example of this financial data, the integration of OP-KNN and Variable Selection phase shows the best efficiency and accuracy, meanwhile it selected the most important variables and build the model with them. The computational time for the whole OP-KNN is one second for each output.

Table 2. VS+Scaling result: ROA

index	Variable	Scaling value
14	S	1
19	CPPG	1
33	RCAI	1
34	REXCEP	0.8
12	IF	0.8
13	AI	0.6
17	L	0.6
31	CFI	0.5
30	REXPLOI	0.4
35	IS	0.4
26	TD	0.3
29	DA	0.3
21	FR	0.1
25	Debt+1AN	0.1
16	CD	0
LOO result	0.2906	0.2496

4 Conclusions

In this paper, we proposed a methodology OP-KNN based on Extreme Learning Machine which gives better performance than the OP-ELM or any other algorithms for the financial modeling we have tested. Using KNN as a kernel, the MRSR algorithm and the PRESS statistic are all required for this method to build an accurate model.

Besides, to do the prestep Variable Selection is clearly a wise choice to raise the interpretability of variables and

Table 3. VS+Scaling result: ROE

index	Variable	Scaling value
33	RCAI	0.8
35	IS	0.8
19	CPPG	0.6
18	AC	0.5
13	AI	0.3
31	CFI	0.3
34	REXCEP	0.3
2	Transaction	0.2
LOO result	0.5596	0.5208

Table 4. VS+Scaling result: Marris

index	Variable	Scaling value
23	DEFI	1
11	AMORIC	0.9
20	PRC	0.9
13	AI	0.7
18	AC	0.6
29	DA	0.6
9	AMORII	0.5
24	Debt-1AN	0.5
25	Debt+1AN	0.5
27	CPER	0.4
31	CFI	0.4
16	CD	0.4
17	L	0.2
32	RFI	0.2
26	TD	0.1
34	REXCEP	0.1
15	CCR	0
LOO result	0.9084	0.8270

Table 5. Normalized result for output 1

	LOO error	Number of the Nearest Neighbors selected (and order)
all the variables	0.7331	8 (1 2 3 7 23 24 32 16)
	0.7443	4 (1 2 3 7)
VS	0.2424	8 (1 3 2 6 7 10 18 13 12 4)
	0.2435	6 (1 3 2 6 7 10)
VS+Scaling	0.2216	7 (1 3 2 6 5 21 8)
	0.2229	6 (1 3 2 6 5 8 7)

Table 6. Normalized Result for output 2

	LOO error	Number of the Nearest Neighbors selected (and order)
all the variables	0.9250	3 (1 8 2)
	0.9246	2 (1 2)
	0.9230	3 (1 2 5)
VS	0.5437	5 (1 7 3 10 2)
	0.5525	2 (1 2)
VS+Scaling	0.5142	4 (2 1 4 8)
	0.4951	11 (2 1 4 48 11 12 23 36 9 45 42)

Table 7. Normalized Result for output 3

	LOO error	Number of the Nearest Neighbors selected (and order)
all the variables	0.9990	1 (63)
	0.9942	2 (2 1)
VS	0.7559	4 (1 4 13 15)
	0.7594	2 (1 4)
VS+Scaling	0.7058	6 (1 20 4 28 38 32)
	0.7068	3 (1 20 4)
	0.7088	2 (1 4)

increase the efficiency of the built model.

We test our methodology on 200 French industrial firms listed on Paris Bourse (Euronext nowadays) within a period of 4 years (1991-1995). Our results highlight that the first ten variables are the best combination to explain corporate performance. Afterward, the new variables do not allow improving the explanation of corporate performance. For example, we show that the company size is a variable that improves performance. Furthermore, it is interesting to notice that the discipline of market allows to put pressure on firms to improve corporate performance.

References

[1] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew (2006), Extreme learning machine: Theory and applications, *Neurocomputing*, p. 489-501.

[2] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani (2004), Least angle regression, *Annals of Statistics*, **vol. 32** p. 407-499.

[3] R.H. Myers, (1990), *Classical and Modern Regression with Applications*, Duxbury, Pacific Grove, CA, USA

[4] G. Bontempi, M. Birattari, H. Bersini (1998), Recursive lazy learning for modeling and control, *European Conference on Machine learning*, p. 292-303.

[5] W.T. Miller, F.H. Glanz, L.G. Kraft (1990), Cmas: An associative neural network alternative to backpropagation, *Proceeding of the IEEE*, **vol. 70**97-102 p. 1561-1567.

[6] C.R. Rao, S.K. Mitra (1972), Generalized Inverse of Matrices and its Applications, *John Wiley & Sons Inc*,

[7] T. Similä, J. Tikka (2005), Multiresponse sparse regression with application to multidimensional scaling, *Proceedings of the 15th International Conference on Artificial Neural Networks*, **Part II** p. 97-102.

[8] Y. Miche, P. Bas, C. Jutten, O. Simula, A. Lendasse (2008), A methodology for building regression models using Extreme Learning Machine: OP-ELM, *European Symposium on Artificial Neural Networks*, p. 23-25.

[9] Q. Yu, E. Séverin, A. Lendasse (2007), Feature Selection Methodology for Financial Modeling, *Computational Methods for Modeling and Learning in Social and Human Science*, Brest, France.

[10] A. Lendasse, V. Wertz, M. Verleysen (2003), Model Selection with Cross-Validations and Bootstraps - Application to Time Series Prediction with RBFN Models, *Joint International Conference on Artificial Neural Networks*, Istanbul, Turkey.

[11] Q. Yu, E. Séverin, A. Lendasse (2007), Variable Selection for Financial Modeling, *Computing in Economics and Finance*, Montréal, Canada.