2001 년 한국정보처리학회 추계 학술발표논문집 제 8 권 제 2 호

Case-Selective Neural Network Model and Its Application to Software Effort Estimation

Eung Sup Jun

Korea Advanced Institute of Science and Technology

Abstract

It is very difficult to maintain the performance of estimation models for the new breed of projects since the computing environment changes so rapidly in terms of programming languages, development tools, and methodologies. So, we propose to use the relevant cases for a neural network model, whose cost is the decreased number of cases. To balance the relevance and data availability, the qualitative input factors are used as criteria of data classification. With the data sets that have the same value for certain qualitative input factors, we can eliminate the factors from the model making reduced neural network models. So we need to seek the optimally reduced neural network model among them. To find the optimally case-selective neural network, we propose the search techniques and sensitivity analysis between data points and search space.

1. Introduction

The computing environment changes so rapidly in terms of programming languages (from COBOL to C++, 4th generation languages, object-oriented languages, visual tools, and GUI), development tools (with and without CASE tools and components), and platform (from the mainframe based computing to client-server based computing and Web based computing).

Among the available estimation models, the neural network models performed at least as good as the other approaches. However, even with the neural network, it is very difficult to maintain the estimation performance for the new breed of projects. So we need to use only the relevant cases for estimation, which means that the number of cases will be decreased. Thus we need to balance the relevance and data availability. To solve this problem, we propose to use the qualitative input factors as criteria of data classification. With the data sets that have the same value for certain qualitative input factors, we can eliminate the factors from the model building reduced neural network models. So we need to seek the optimally reduced neural network model among them. According to the paired t-test, we could prove that the optimally reduced neural network model can significantly reduce the error more effectively than the full neural network model which uses the all data with the all input factors. To find the optimally reduced model heuristically, we propose the search algorithms which selects the data set and associated reduced neural network model.

2. AI Models for software Effort Estimation

2.1. Case-Based Reasoning Models

The CBR model for the software effort estimation requires to collect past project cases, and to retrieve the most similar case to the target project according to a pre-defined similarity measure. However the modification of the past case's result identifying its difference to the target is in most cases very difficult. In the software effort estimation process, the modification process requires a numeric model that can tell the difference of numerically presented effort between the old case and new one. So the CBR approach alone cannot overcome the goal of software effort estimation.

Another issue to be resolved in the CBR approach is the scope quantitative and qualitative ones, and denoted as QT and QL. The factors I_5 – I_9 can be treated as either QT or QL depending upon whether we measure the value as a matter of degree or binary values. Let us regard these factors as QT in this study even though they have binary values. The same logic applies to I_{22} because the cases in the current case base have only two values: 3GL and 4GL (Generation Language). In the neural network model, the qualitative factors are represented as 0 or 1 (actually 0.001 or .999). So qualitative factors do not make a big difference from the quantitative factors in computation. A more important distinction in this study is that the cases with the same qualitative values can be grouped as a case set. Within a case set, we can eliminate the input factors that have the

of selecting past cases. It can be one, the closest case; or some, closer cases. If there is variance in the past software efforts for the same situations, using just one case is too risky. So we need a generalizing estimator based on the relevant past cases. A concern is how to decide the scope of relevant cases and the generalizing model. Let us call the first issue the *optimal selection of relevant cases* problem. If we select too few cases, the power of generalization will diminish. However, if we select too many cases, the irrelevant cases may distort the estimation. This is a theme that this paper attempts to balance. The generalization process may use either a neural network model or a statistical model.

2.2. Neural Network Models

Many studies have shown that the performance of the neural network model is at least as powerful as statistical models. However, the neural network also has the problem of *optimal selection of relevant cases* because the computing environment that significantly influences the software effort changes so rapidly. So we need to find the relevant similar cases that explain the specialty of the target project. If we use all of the past cases, the model becomes too dull for the special characteristics of a changed environment. However, we may not have a sufficient number of cases that are similar enough to use for estimation. Thus we need to balance the similarity and case point availability, which is what this research is pursuing. Let us call the neural network model with all possible input factors and all available cases, the *Full Neural Network Model*. To distinguish the neural network model with reduced relevant cases and reduced input factors (by eliminating the factors that have the same values) from the full model, let us call reduced ones the *Reduced Neural Network Models*. The goal of our research is to find an optimally reduced neural network model for software effort estimation.

3. Full Neural Network Model

3.1 Input Factors for Software Effort Estimation

To select the 23 input factors and their values, we surveyed the opinions of 30 experts who have experiences of software development and maintenance. The input factors are identified as I_i, $i = 1,...,23$. The input factors can be classified into

same values, making a reduced neural network model as mentioned earlier. The case sets with a same qualitative value are denoted as D_j, $j=1,...,8$. Likewise, the case sets with two same qualitative factors can be denoted as D_{jk}, $j \neq k$, $j=1,...,8$, $k=1,...,8$, and so forth.

3.2. Performance of Full Model

The full neural network model has 23 input factors, one output of man-month, and one hidden layer with the nodes from 23/2 (we applied integer 12) to 2*23+1 (which is 47) with the increment of 5. The model is trained by the back-propagation algorithm starting from 500 epochs to 5,000 epochs with the increment of 500.

We have 50 cases in total. The cases are classified into two group of A and B, 25 cases each. When group A is used for training, group B is used for testing, and vice versa. So we have 50 test cases in total. The optimal performance was found with 12 hidden nodes at 3,000 epochs.

The popular measure of errors is MMRE (Mean Magnitude of Relative Error). This study adopted MMRE as the measure of errors.

The full model has the MMRE of 25.8%. This figure is not bad if we compare it with the ones by the statistical models (50%-772%) and CBR models (35%).

4. Reduced Neural Network Model

The reduced neural network models are the ones that have eliminated the qualitative input factors with the same values. So the number of input factors can be reduced from 23 (in the full model) to at most 15 in the software effort estimation model because we have 8 qualitative factors.

Now a question is how to find the optimal combination of eliminating input factors out of $\sum_{i=1}^{8} {}_8C_i = 255$ alternative combinations. As mentioned earlier, the demerit of the reduced model is the reduced number of data points at the cost of enhanced relevance. So just more reduction of the input factors does not necessarily mean the performance improvement. To measure the degree of reduction, we need to adopt the notion of similarity.

4.1. Measure of Similarity

Measure of similarity is widely used in the CBR to find the most similar case to the target. Let us denote *Case X* and *Y* as (2) of APPENDIX.

where $v(I_i^{(x)})$ means the value of input factor i in *Case X*. By comparing the values of each factor as (3) of APPENDIX,

the similarity between *Cases X* and *Y*, denoted as *SIM(X,Y)* can be computed by the equation in (4) of APPENDIX.

The similarity is proportional to the number of coinciding values even though different weights, w_i, can be applied to different factors as (4) of APPENDIX. Let us assume all w_i's are 1 in this study.

To denote $SIM(X,Y) = s$ in a simple notation, let us define *SL(s)* as (5) of APPENDIX. Here, *SL(s)* means that there are s input factors that have the same values.

4.2. Relevant Case Sets and Performance Evaluation Groups

Suppose the *SL(1)* situation exists between a target case and past cases in a case base. This means that at least one out of eight qualitative input factors has the same value. To define the *SL(1)* with the same value of input factor j, let us denote the situation as $SL(1|D_j)$ where D_j implies the data set with a same value in the qualitative factor j.

In this study, we have two alternative values in each qualitative factor. So we have two case sets in each $SL(1|D_j)$, one set for each value. For instance, suppose D_1 is the *development_type* which has two alternative values: *new_development* or *maintenance*. The two case sets in $SL(1|D_1)$ are :

$SL(1|\ v(development\_type) = new\_development)$ and

$SL(1|\ v(development\_type) = maintenance)$.

So to measure the performance of the reduced model with $SL(1|D_1)$, we need to average the performances of case sets in $SL(1|D_1)$. In total, we have ${}_8C_1 * 2 = 16$ case sets for *SL(1)* as denoted by CS_D_j_a and CS_D_j_b, $j = 1,...,8$. Thus we have ${}_8C_1 = 8$ groups of average performance, denoted as $AP(D_j)$, $j = 1,...,8$.

The same logic applies to *SL(2)*. There are ${}_8C_2 * 2^2 = 112$ case sets and ${}_8C_2 = 28$ average performance groups. According to this notation, *SL(0)* corresponds to the full model.

Our concern in designing a reduced neural network is finding the optimal level of *SL(s)* and the best case set within *SL(s)* category. We need to devise a search method that can find the optimal *SL(s)*, denoted as *SL*(s)*.

5. Optimally Reduced Neural Network Model

5.1. Non-monotonicity

To find the optimal MMRE for the *SL(s)*, we need to compare the MMRE of alternative case sets in the *SL(s)*. For instance, MMREs of $SL(1|D_1)$ are : $SL(1|D_1) = 23.4\%$, $SL(1|D_2) = 25.4\%$, $SL(1|D_3) = 23.7\%$, $SL(1|D_4) = 23.0\%$, $SL(1|D_5) = 45.9\%$, $SL(1|D_6) = 28.9\%$, $SL(1|D_7) = 26.8\%$, $\mathbf{SL(1|D_8) = 20.7\%}$.

Thus $SL^*(1) = \min_j [(SL(1|D_j),\ j = 1,...,8)] = SL(1|D_8)$.

$SL(1|D_8)$ means that the factor I_{23} (whether to use or not use the CASE Tool, that corresponds to the data set D_8) will be eliminated from the input factors of the neural network model, and the model is estimated using the data set D_8.

In the same manner, we can discover $SL^*(2) = SL(2|D_3, D_8)$, $SL^*(3) = SL(3|D_1, D_3, D_8)$, and so forth. The optimal combinations of input factors for each $SL^*(s)$, $s=1,...,8$ are summarized in Table 1.

In this case, the optimally reduced neural network model is *SL*(5)* with the MMRE of $\min_s$ 18.6%. $[(SL^*(s), s = 1,...,8)] = SL^*(5) = SL(5|D_1,D_2,D_3,D_5,D_8)$

Recall that the MMRE of the full model was 25.8%, and note that the *SL*(s)* are not monotonic. So the issue is how to find the minimum *SL*(s)* efficiently, which is dealt in the next section.

5.2. Validation of the Performance of Reduced Model

To confirm whether the optimally reduced model *SL*(5)* significantly outperforms the full model, we have conducted a paired-t test.

- Null Hypothesis(H_0) : $MMRE_{SL(0)} = MMRE_{SL^*(5)}$
- Alternative Hypothesis(H_1) : $MMRE_{SL(0)} > MMRE_{SL^*(5)}$

Statistic Items	Significance Level	Decision
t-value = 1.875, P-value = 0.03	$\alpha = 0.05$	Fail to accept H_0
Degree of freedom = 98	($t_\alpha > 1.645$)	

The test confirms that the optimally reduced model significantly outperforms the full model at 5% of significance.

5.3. ADD and DROP Algorithms

Since the MMRE is not monotonic to s, we need to devise algorithms that can find the optimal (or approximate optimal) solution with a reduced search effort. In this study, we devise two algorithms: ADD and DROP algorithms. If we search all combinations of case sets, the *SL*(5)* can be found as optimum. However the heuristic ADD algorithm means to increment the similarity level a single qualitative factor at a time, while the DROP

algorithm decrement one at a time as depicted in Table 2.

The ADD algorithm actually starts with a full model, while the DROP algorithm starts with a purely quantitative model. Let us define the algorithms in general terms.

1) *Notations*

NL : Number of qualitative factors, which is 8 in this case.

2) *ADD Algorithm*

1) In the case of the full neural network model, $SL^*(0) = SL(0)$. All qualitative factors remain in the input factors.
2) Compute the MMREs of $SL(1|D_i), i = 1,\ldots,NL$. Find $SL^*(1) = \min_i [SL(1|D_i), i = 1,\ldots,NL] = SL(1|Dq_1)$ where q_1 denotes i that satisfies $SL^*(1)$.
3) Compute the MMREs of $SL(2|Dq_1, D_i, i = 1,\ldots,NL, i \neq q_1)$. Find $SL^*(2) = \min_i [SL(2|Dq_1, D_i, i = 1,\ldots,NL, i \neq q_1)] = SL(2|Dq_1, Dq_2)$ where q_2 denotes the i that satisfies $SL^*(2)$ in addition to q_1.
4) Repeat the addition of qualitative factors one at a time until a stopping rule (a threshold of MMRE) is satisfied or the qualitative factors are exhausted.

Now let us define the DROP Algorithm which is the reverse of the ADD Algorithm.

3) *DROP Algorithm*

1) Compute the most reduced neural network model, $SL^*(NL)$. All qualitative factors are eliminated from the input factors.
2) Compute the MMREs of $SL(NL-1| D_i, i = 1,\ldots, NL$ and $i \neq j)$ for $j=1,\ldots,NL$. Find $SL^*(NL-1) = \min_i [SL(NL-1| D_i, i = 1,\ldots,NL$ and $i \neq j)] = SL(NL-1| D_i, i = 1,\ldots, NL, i \neq r_1)$ where r_1 denotes the i that is dropped from $SL^*(NL)$.
3) Compute the MMREs of $SL(NL-2| D_i, i = 1,\ldots,NL, i \neq r_1 \neq j)$. Find $SL^*(NL-2) = \min_{i,j} [SL(NL-2| D_i, i = 1,\ldots,NL, i \neq j \neq k), j = 1,\ldots,NL, k = 1,\ldots,NL] = SL(NL-2| D_i, i = 1,\ldots, NL, i \neq r_1 \neq r_2)$ where r_2 denotes i that is dropped from $SL^*(NL-1)$ in addition to r_1.
4) Repeat the elimination of the qualitative factors one at a time until a stopping rule is satisfied or all qualitative factors are exhausted.

5.4. Performance of Heuristic Algorithms and Stopping Rules

In our experiment, both ADD and DROP algorithms find the optimally reduced neural network model even though there is no guarantee that it will be found. To find the $SL^*(5)$, the ADD algorithm computed 31 neural network models, while the DROP algorithm computed 22 neural network models in our experiment. Note that these figures are far smaller than all combinations of 255. However, the smaller number for the DROP algorithm than that for ADD algorithm cannot be interpreted as a general phenomenon. It depends upon the characteristics of the case set we have and the position of optimal point.

6. Conclusion

To enhance the performance of software effort estimation models under the rapidly changing computing environment, we have attempted to use only the relevant cases and to reduce the qualitative input factors of neural networks that have the same values. To find the optimally reduced neural network, we have devised the ADD and DROP algorithms that can heuristically find the (near) optimal models. We have proved that the optimally reduced neural network model can perform significantly better than the original neural network model. These algorithms can be used not only for the software effort estimation, but also for any neural networks that have both quantitative and qualitative input factors.

REFERENCES

[1] Azuma, M., Mole, D. Software Management Practice and Metrics in the European Community and Japan: Some Results of a Survey. Systems Software, 1994.

[2] Blackburn, J.D., and Scudder, G.D. Improving Speed and Productivity of Software Development: A Global Survey of Software Developers. IEEE Transactions on Software Engineering, Vol. 22, No. 12, December 1996.

[3] Deephouse, C.; Mukhopadhyay, T.; Goldenson, D.R.; and Kellner, M.I. Software Process and Project Performance. Journal of Management Information Systems, Vol. 12, No. 3, 1996.

[4] Finnie, G.R., Wittig, G.E., and Petkov, D.I. Prioritizing Software Development Productivity Factors Using the Analytic Hierarchy Process. Systems Software, Vol.22, 1993, pp129-139.

[5] Maxwell, K.D.; Wassenhove, L.V.; and Dutta, S. Software Development Productivity of European Space, Military, and Industrial Applications. IEEE Transactions on Software Engineering, Vol.22, No. 10,October 1996.

[6] Rasch, R.H., and Tosi, H.L. Factors Affecting Software Developers' Performance: An Integrated Approach. MIS Quarterly, September 1992.

[7] Redmond-Pyle, D., Software Development Methods and Tools: Some Trends and Issues. Software Engineering Journal, March 1996.

[8] Roberts Jr, T.L.; Gibson, M.L.; Fields, K.T.; and Rainer Jr, K. Factors that Impact Implementing a System Development Methodology. IEEE Transactions On Software Engineering, Vol.24, No.8, August 1998, pp.640-649.

[9] Saiedian,H.; Band, M.; and Barney, D. The Strengths and Limitations of Algorithmic Approaches to Estimating and Managing Software Costs. International Business Schools Computing Quarterly, Spring , 1992, pp. 21-22.

[10] Venkatachalam, A.R. Software Cost Estimation Using Artificial Neural Networks. Proceedings of 1993 International Joint Conference on Neural Networks, July 1993, pp. 987-990.

Appendix

Table 1. Number of Data Points by Similarity Level

Similarity Level *SL*(s)*	Optimal Case Sets for Each *SL*(s)*	Cases Sets with the SameValues	No. of Data Points	MMRE (%) of Case Set	Average Performance by MMRE(%)	Ratio of Available Case Sets
SL(5)*	(D1,D2,D3,D5,D8)	{d1a,d2a,d3a,d5a,d8a}	11	15.9	18.6	$5/2^5$=0.17
		{d1b,d2a,d3a,d5a,d8a}	9	32.5		
		{d1b,d2a,d3b,d5a,d8a}	7	12.2		
		{d1b,d2a,d3b,d5a,d8b}	1	NA		
		{d1b,d2b,d3b,d5a,d8b}	12	17.3		
		{d1b,d2b,d3b,d5b,d8b}	10	18.3		

Table 6. Best MMRE by All Combination of Case Sets

Similarity Level	Data Sets	Added Factor by ADD Algorithm	Dropped Factor By DROP Algorithm	MMRE (%)
SL(0)*	{ }	Starting point	D_8	25.8
SL(1)*	$\{D_8\}$	D_8	D_3	20.7
SL(2)*	$\{D_3,D_8\}$	D_3	D_1	21.3
SL(3)*	$\{D_1,D_3,D_8\}$	D_1	D_2	20.8
SL(4)*	$\{D_1,D_2,D_5,D_8\}$	D_2	D_5	19.9
SL(5)*	$\{D_1,D_2,D_3,D_5,D_8\}$	D_5	D_7	18.6
SL(6)*	$\{D_1,D_2,D_3,D_5,D_7,D_8\}$	D_7	D_4	19.5
SL(7)*	$\{D_1,D_2,D_3,D_4,D_5,D_7,D_8\}$	D_4	D_6	20.0
SL(8)*	$\{D_1,D_2,D_3,D_4,D_5,D_6,D_7,D_8\}$	D_6	Starting point	21.6

$$\left.\begin{array}{l} Case\ \ X\ :\ Case^{(x)}(v(I_1^{(x)}),\ldots,v(I_{23}^{(y)})) \\ Case\ \ Y\ :\ Case^{(y)}(v(I_1^{(y)}),\ldots,v(I_{23}^{(y)})) \end{array}\right\} \qquad (2)$$

$$f(I_i) = \begin{cases} 1 & v(I_i^{(x)}) = v(I_i^{(y)}) \\ 0 & v(I_i^{(x)}) \neq v(I_i^{(y)}) \end{cases} \qquad (3)$$

$$SIM(X,Y) = \sum_{i=1}^{23} w_i f(I_i) \qquad (4)$$

$$SIM(X,Y) = s \ \Leftrightarrow\ SL(s) \qquad (5)$$