Software Project Risk Management Modeling with Neural Network and Support Vector Machine Approaches

Yong Hu
Guangdong University of Foreign Studies, Sun Yat-sen University, 510275, China
henryhu200211@163.com

Jiaxing Huang
Sun Yat-sen University, 510275, China
singforu@163.com

Juhua Chen
Sun Yat-sen University, 510275, China
isscjh@ mail.sysu.edu.cn

Mei Liu
University of Kansas, 66045, U.S.A
meiliu@ittc.ku.edu

Kang Xie†
Sun Yat-sen University, 510275, China
mnsxk@mail.sysu.edu.cn

Abstract

Software project development has high failure rate. Software project risk management may gain a high rate of return in investment. Establishing an intelligent risk evaluation model for project will be valuable in the analysis and control of project risks. In this paper, we employed neural network (NN) and support vector machine (SVM) approaches to establish a model for risk evaluation in project development. In the model, the input is a vector of software risk factors that were obtained through interview with 30 experts, and the output is the final outcome of the project. The data for modeling were collected from 120 real software projects through questionnaires. The experiment shows the model is valid. Interestingly, SVM is a powerful supervised learning method, and some believe that it is a more promising classification method that may someday supercede NN. In our study, the standard neural network model had lower prediction accuracy compared to SVM due to its tendency in finding local optima. However, after attempt in optimizing the neural network model with genetic algorithm, the experimental results showed that our enhanced model surpassed SVM in performance.

1. Introduction

High project failure rate is a major concern in software project development, and it is becoming more so as software projects grow in complexity. The research done by Standish Group on the 8,000 civilian and military software projects in America revealed that the success rates of software projects in 1996, 1998, 2000 and 2004 were 27%, 26%, 28% and 29%, respectively [1,2].

Table 1. Standish Group Survey on Software Project

	1994	1996	1998	2000	2004
Successful project	16%	27%	26%	28%	29%
Challenged project	31%	40%	28%	29%	53%
Failed project	53%	33%	46%	43%	18%

Intuitively, this implies that software project development consists of many risks that need to be managed in order to produce a successful outcome. Hence, managing the involved risks is of primary importance in software project development, especially in large-scale software projects. In [3], Charatte considered the large-scale software project management as risk management. According to the research by IEEE, 50%-70% of the risks can be found through inspection while 90% of the risks among them can be avoided. The rate of return on investment of risk management is 700% to 2000%, and the time-consuming and strenuous work in risk evaluation can be eased by software packages [4]. A study by Microsoft showed that the probability of completing projects on time with 5% risk management is 50-75% [5]. Thus, it is essential to manage the risks in software development. Predicting outcome of the risks in an early stage with an effective risk analysis method is one of the key risk management activities. In this paper, we attempt to study the relationship between software risk factors and project outcome.

2. Related Work and Methodology Selection

The U.S. Department of Defense (DOD) defines software risks as measurement indexes when the overall pre-established goals can not be achieved due to the limitation of the scheduled cost, time and technology [6]. In 1989, Boehm and other researchers introduced the concept of software risk management into the software circle and established the foundation of research in this field [7]. The risk management in software projects includes risk identification, risk analysis and risk control.

Numerous machine learning and data mining algorithms were exploited in risk analysis, which include Bayesian Belief Network (BBN) [8], Neural Network (NN) [9,10], Discriminant Analysis(DA) [11,12], Decision Tree (DT) [13], and etc. Although these methods have been applied to the analysis of software quality and risks, neural network is often better than Decision Tree and Discriminant Analysis [10,14], since NN has an excellent learning and analytical abilities in complex and nonlinear problems. In contrast to the BBN approaches, NN does not require the establishment of the relations and conditional probability table, which is relatively subjective and unreliable. Neural network is an effective solution when it is difficult to build the relationships.

† Research supported by Guangdong Software Science Foundation (2005B70101096)., National Nature Science Foundation (70572053) (60673135)Corresponding author: Xie Kang mnsxk@mail.sysu.edu.cn

Third International Conference on Natural Computation (ICNC 2007)
0-7695-2875-9/07 $25.00 © 2007 IEEE

Furthermore, neural network has other advantages in software risk measurement [10,15,16]. First of all, neural network can deal with noisy data effectively. Since the project samples are acquired through the project stakeholders' memory, there will be a loss of information authenticity. Secondly, neural network is flexible in sample quantity. For instance, Khoshgoftaar and other researchers only collected 56 module samples to perform network training. Thirdly, neural network can deal with not only precise data but also fuzzy data. Neural network can process simultaneously quantitative and qualitative data.

Therefore, neural network is a good candidate for establishing our model.

Certainly, neural network also has disadvantages that back propagation (BP) neural network mainly uses gradient descent approach to acquire the minimum error value. As the initial values of BP networks are distributed randomly at the very beginning, it is more likely that the values are far from the global optimal location. Consequently, BP networks are powerful in local search and weak in global search. The networks are likely to converge in a local optimal location and fail to find the global optimal solution. This shortcoming considerably affects the performance of BP networks.

To ameliorate this weakness, genetic algorithm is introduced in this paper. Genetic algorithm is an optimization method that can be utilized to optimize neural networks in many ways, including weights, network structure and rule learning [17]. This paper employs genetic algorithm to determine the weights for neural networks.

Interestingly, SVM is a powerful supervised learning method. Comparing to NN, SVM uses quadratic optimization, which theoretically should avoid the inevitable local optima problem in NN [18]. It can be hypothesized that SVM is superior to NN in performance, and some believe that it is a more promising classification method that may someday supercede NN [19]. However, can SVM maintain its superiority when the local optima problem is solved in NN, especially when complex relationships exist in data? Although SVM and NN have been employed in many applications [20,21,22], so far we have not found any established SVM model that evaluates project-scale risk.

Therefore, in our study, we established software project risk evaluation models with standard NN, enhanced NN by GA and SVM, and then compared their performance.

3. Modeling

In our NN risk evaluation model, neural network is a set of connective input and output units, in which each connection is assigned with a weight. There are the input layer, the hidden layer and the output layer. In our model, the input layer is composed of the key risk factors. Similarly, input for the SVM model also consists of the key risk factors. The outputs of the predictive models represent the software outcome. In our model, the project outcome consists of six dimensions [23,24].

In risk management, risk factor identification is the foremost step. Risk identification method based on taxonomy proposed by CMU/SEI is advantageous in the overall and systematic identification of software project risks [25]. In their system, risks are classified into three main categories and 13 subcategories. The three main categories are product engineering, development environment and program restrictions. From these categories, they designed questionnaires with 194 questions. Linda Wallace and Mark Keil classified risks into six categories, which are team risk, organization environment risk, requirement risk, plan and control risk, user risk and complexity risk [26]. The model by SEI is too large to be applied conveniently, while Wallace only considered the risks in the requirement phase and failed to take into account the risks in other phases during a software lifecycle.

Table 2. Taxonomy and Factors of Software Risks

Environment Complexity Risk
Environment complexity risk focuses on risks caused by the customer's IT infrastructure, management and the stability of the environment. For example, conflict of interest in key sectors of the customer department, low level of information resources, complexity and chaos in operation flow, change in project scope or resource because of change in top manager, and etc.
Project Requirement Complexity Risk
Project requirement complexity risk focuses on project characteristics, the complexity or change in project requirement, requirement including technical and non-technical complexity. For example, limitation in schedule and money, one of the largest projects attempted by the organization, project involving use of technology that has not been used in prior projects, project involving the use of new technology, large number of links to other systems, high level of technical complexity, bad expansibility in old system, inadequate estimation of required resources, new and/or unfamiliar subject matter for both users and developers, and etc.
Cooperation Risk
Cooperation risk focuses on cooperation in customer and supplier in whole project process For example, lack of user participation, project affecting a large number of user departments or units, lack of top manager support for the project(from customer), users' resistance to change, project personnel lacking required knowledge/skills, and etc.
Team Risk
Team risk focuses on people maturity, communication, training and management in project team. For example, inexperienced project manager, team members lacking specialized skills required by the project, frequent turnover within the project team, low morale, ineffective communication, top manager offering insufficient support, inadequate training, and etc.
Project Management Risk
Project management risk focuses on resources and project estimation, plan, control, quality assuring and supporting management. For example, project milestones unclearly defined, managing change improperly, project progress monitored loosely, lack of an effective project management methodology, and etc.
Engineering Risk
Engineering risk focuses on software engineering process activities to complete software product required by customer, including engineering requirement, design, development, testing, engineering document and those activities of validation and verification. For example, incorrect system requirements, unclear system requirements, choosing the wrong development strategy, inappropriate software technical design, lack of reusable code or data, lack of reusable designs or documentation, inappropriate software technical documentation, inadequate testing, lack of mechanism of validation and verification for product, and etc.

Third International Conference on Natural Computation (ICNC 2007)
0-7695-2875-9/07 $25.00 © 2007 IEEE

On the basis of the above literature and theories, we interviewed 30 experts, among whom there were 4 professors in software engineering or information system, and 26 project managers and experts in software development community, which includes software project managers, project technical leaders, customer managers and consultants. We establish our software project risk theory model and classify 64 risk factors into six categories, as shown in Table 2. Questionnaires were then designed based on the interviews.

4. Data Collection and Analysis

The data for this research were collected through questionnaires. In total, 300 questionnaires were given out, and 133 of which were returned. Among the returned questionnaires, 120 of which were found to be legitimate. The samples were collected from numerous cities in China such as Guangzhou, Shanghai, Shenzhen, Nanjing, and etc. People who were questioned work for different agencies including software engineering, communication, internet, transportation and government administration. Of all the people questioned, 71% were project managers, project technical leaders or customer manager while 25% were project team members. The detailed description is shown in Table 3.

Table 3. Project Samples Description

Attribute	Category	Percentage
Experience	8-15 years	18%
	4-7 years	46%
	1-3 years	16%
Position	Top manager	3%
	Project manager	25%
	Project technical leader	18%
	Project team member	25%
	Customer manager	25%
	Others	4%
Project Success Rate	Successful project	26%
	Challenged project	46%
	Failed project	28%

The overall project success rate is 26%, which is similar to the research done by the Standish Group that the success rate was between 27% and 29% among software projects in America in recent years. Hence, the samples are believed to be representative.

In general, learning is most reliable when the training examples follow a distribution similar to that of future test examples [27]. Therefore, constructing the model based on the described samples may be considered relatively reliable.

5. Experiments and Results

We randomly divide our dataset into two subsets, 100 samples for training, and 20 for testing. The original project outcome are standardized and divided into 3 ranks: successful projects (marked with 3), challenged projects or projects that are partially failed (marked with 2), and projects that are totally failed (marked with 1).

Experiments were performed to evaluate and analyze our proposed models. We start by predicting the risks with standard neural network. Then predictions were made using the combination of genetic algorithm and neural network. Finally, the two employed predictive models, SVM and NN are compared.

Before the experiments of NN, the input and output values are all scaled to the interval [0, 1].

In the experiment where standard NN is employed, a three-layer back-propagation (BP) NN is utilized. The numbers of neurons in the input, hidden and output layers are 64, 8 and 1, respectively. Linear transfer function was applied for the input layer neurons, and sigmoid transfer function was used for the neurons in the hidden and output layers.

In the experiment where NN is enhanced through the introduction of GA, real number coding is used on the training weights. The fitness function is the reciprocal of the error function used in BP. Roulette wheel approach was adopted as the selection procedure, while arithmetic crossover and perturbation mutation were used. In addition to the above genetic operators, elitist strategy is employed to speed up the convergence.

In the SVM experiment, we used LIBSVM, an open source SVM package [28]. The input values are scaled to the interval [0, 1]. RBF function is selected as the kernel function, and leave-one-out cross validation (LOOCV) method was used together with the grid search method to search for the optimal combination of the penalty and kernel parameters.

The standard BP neural networks can predict the outcome of software projects with 70% in accuracy. SVM on the other hand achieved higher accuracy of 80%. However, after introducing the genetic algorithm on the basis of the standard BP neural networks, the precision can reach 85%. The result details are shown in Table 4.

Because standard BP network is powerful in local search but weak in global search, it is the least effective in prediction of the three models. After introduction of genetic algorithm, the improvement in accuracy is obvious. This improvement can be attributed to GA that alleviated the local optimal solution search in neural networks. The prediction results indicate that the enhanced neural network model is a more effective prediction model compared to SVM in this particular study.

6. Conclusions

In this paper, we employed neural network (NN) and support vector machine (SVM) approaches to establish a model for risk evaluation in project development. First, we established the software project risk theory model by studying literatures on the software risk management and interviewing experts. As a result, we classified the risks into 6 categories: environment complexity risk, project requirement complexity risk, cooperation risk, team risk,

Third International Conference on Natural Computation (ICNC 2007)
0-7695-2875-9/07 $25.00 © 2007 IEEE

project management risk and engineering risk. The risk factors are used as an input for the neural network and SVM models to analyze the software risks and predict outcomes of software projects. The standard NN was enhanced by introducing GA because neural networks are powerful in local search but weak in global search while GA is effective in global search. Results of the experiments showed that after introducing GA to the NN training process, the enhanced neural network software risk evaluation model can be improved notably and achieve higher accuracy when compared to the SVM model. The above experiments imply that when complex relationships exist in data, the enhanced neural network model may perform better than SVM.

Table 4. Actual and Test Results Comparison of Methods in Experiments

Sample		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Precision
Actual Result		L	L	L	L	M	M	M	M	M	M	M	M	M	M	H	H	H	H	H	H	Comparison
NN	Test Result	L	L	M	L	M	L	M	M	H	L	M	H	M	M	H	H	H	L	H	H	
	Precision	T	T	F	T	T	F	T	T	F	F	T	F	T	T	T	T	T	F	T	T	70%
GA-NN	Test Result	L	L	L	L	M	M	M	H	H	M	M	M	M	M	H	H	H	H	H	M	
	Precision	T	T	T	T	T	T	T	F	F	T	T	T	T	T	T	T	T	T	T	F	85%
SVM	Test Result	L	L	L	L	M	M	M	M	M	H	M	M	M	M	H	H	H	M	L	M	
	Precision	T	T	T	T	T	T	T	T	T	F	T	T	T	T	T	T	T	F	F	F	80%

H: successful project, M: challenged project, L: totally failed project, T: true, F: false

References

[1] The Standish Group, extreme chaos[R], www.standishgroup.com，2001.

[2] The Standish Group, 2004 Third Quarter Research Report[R]，www.standishgroup.com，2004

[3] Robert N. Charette, Large-Scale Project Management Is risk Management [J], IEEE Software, 1996,7.

[4] Lister T. Interview with Tim Lister[J]. IEEE Software, 1997, 14 (3):18-19.

[5] Steve McConnell, Software Project Survival Guide[M], Microsoft Press, 1997

[6] Orna Raz, Mary Shaw. Software Risk Management And Insurance[C], Proceedings of the 23rd International Conference on Software Engineering (Workshop on Economics-Driven Software Engineering Research), 2001.

[7] Boehm B W. Software Risk Management. Los Alamitos: IEEE Computer Science Press, 1989.

[8] Sunita Chulani, Barry Boehm. Bayesian Analysis of Empirical Software Engineering Cost Models. IEEE Transactions on Software Engineering. 1999, 24(4):573-583.

[9] T M Khoshgoftaar, D L Lanning. A Neural Network Approach for Early Detection of Program Modules Having High Risk in the Maintenance Phase. Journal of Systems Software, 1995,29:85-91

[10] Taghi M. Khoshgoftaar, Edward B. Allen. Application of Neural Networks to Software Quality Modeling of a very Large Telecommunications System[J], 1997, 8. IEEE Transactions on Neural Networks.

[11] J. C. Munson, T. M. Khoshgoftaar, The detection of fault-prone programs, IEEE Trans. Software Eng., 1992 May. vol. 18, pp. 23–433.

[12] V. Rodriguez and W. T. Tsai, Evaluation of software metrics using discriminant analysis, J. Inform. Software Technol., 1987. May vol. 29, no. 3, pp.245–251.

[13] R. W. Selby and A. A. Porter, Learning from examples: Generation and evaluation of decision trees for software resource analysis, IEEE Trans. Software Eng., 1988. Dec. vol. 14, pp. 1743–1756.

[14] T. M. Khoshgoftaar, A. S. Pandya, and D. L. Lanning, Application of neural networks for predicting faults, Ann. Software Eng., 1995.vol. 1, pp. 141–154.

[15] D. K. H. Chua,P. K. Loh, Y. C. Kog, E. J. Jaselskis, Neural Network for Coutruction Project Success, Expert Systems with Applications, vol 3. No. 4 1997.

[16] Richard J. Roiger, Michael W. Geatz, Data maing: A Tutorial-Based Primer, Person Education, 2003, PP78.

[17] Bornholdt, S., Graudenz, D. General Asymmetric Neural Networks and Structure Design by Genetic Algorithms. Neural networks, 5:327-334.

[18] Vapnik, V. (Ed.) The Nature of Statistical Learning Theory, Springer, New York, 1995.

[19] Zhou Zhihua, Cao Cungen, Neural Networks and Application, Beijing: Tsinghua University Press, 2004.

[20] Farhad Dadgostar, Abdolhossein Sarrafzadeh, Chao Fan, Liyanage De Silva, Chris Messom, Modeling and Recognition of Gesture Signals in 2D Space: A Comparison of NN and SVM Approaches, Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence, Nov. 2006. pp. 701-704.

[21] Patrick Conrad and Mike Foedisch, Performance Evaluation of Color Based Road Detection Using Neural Nets and Support Vector Machines, Proceedings of the 32nd Applied Imagery Pattern Recognition Workshop, 2003, pp. 157-160.

[22] Sven F. Crone, Stefan Lessmann and Swantje Pietsch, Forecasting with Computational Intelligence - An Evaluation of Support Vector Regression and Artificial Neural Networks for Time Series Prediction, 2006 International Joint Conference on Neural Networks Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada July 16-21, 2006. pp. 3159-3166.

[23] Delone, W. H., Mclean, E. R. Information System Success: The Quest for the Dependent Variables. Information Systems Research. 1992, 3 PP60-95.

[24] Hu Yong，Xiao JingHua，Pang JiaFeng，Xie Kang, A Research on the Appraisal Framework of E-Government

Project Success，The Proceedings International Conference of Electronic Commerce 2005, 2005, 8.

[25] Marvin J Carr, S L Konda, I Monarch, F C Ulrich, C F Walker. Taxonomy-Based Risk Identification. Software Engineer Institute Technical Report SEI-93-TR-006, Pittsburgh, PA. Software Engineering Institute, 1993.

[26] Wallace L. The Development of an Instrument to Measure Software Project Risk. Doctoral Dissertation, Georgia State University, 1999.

[27] Tom M. Mitchell, Machine Learing, MeGraw-Hill Companies, Inc. 1997

[28] Chih-Chung Chang and Chih-Jen Lin, LIBSVM: a library for support vector machines, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

Third International Conference on Natural Computation (ICNC 2007)
0-7695-2875-9/07 $25.00 © 2007 IEEE