Volume 2, Issue 4, April 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Application of Principal Component Analysis in Software Quality Improvement

Yajnaseni Dash*, Sanjay Kumar Dubey
Department of Computer Science & Engineering,
Amity School of Engineering & Technology,
Amity University

Abstract— **Statistical modeling technique has pivotal role in better understanding of the software development processes. Among them neural network techniques have enhanced predictive capability than most other statistical models. This paper explains the application of principal component analysis to neural network modeling as a way to improve predictability of neural network. The purpose of principal component analysis is to augment the performance of discriminant software quality models. The accurate neural training can be done by transferring the raw data into principal components. In this paper, the significance of principal components analysis is illustrated with the help of a commercial raw dataset and subsequently neural network modeling is described.**

Keywords— **principal components analysis, software quality, software metrics, object oriented, neural networks.**

I. INTRODUCTION

The Principal components analysis (PCA), a data reduction technique is used with the intention of reducing the dimensionality of a multivariate data. This is a method which not only reduces the complexity of data but also reduces the variables being used in the study. The analysis identifies the latent factors from a group of apparent important variables. In an extensive way it is a technique that analyses the correlation between variables and produces a few factors of the original data. Hence we can say that this method is more economical in the sense of storing the data and saving the memory space.

As we know that measuring the object oriented software is very crucial for the software engineers. The necessity of measurement is to reduce cost, effort and schedule for amplifying the system performance. Certain measures of the maintainability of software can be obtained by the use of available commercial tools. There are various software metrics in the object oriented system which can measure maintainability. The prediction of maintainability requires analysis of huge amount of data, hence this is time consuming and error prone. The main goal of PCA is to reduce that raw data by removing the correlations among the software metrics [1], thereby making the prediction model more stable.

II. SOFTWARE METRICS

Software metrics is defined as, "The continuous application of measurement-based techniques to the software development process and its products to supply meaningful and timely management information, together with the use of those techniques to improve that process and its products" [2].

Quality estimation means estimating maintainability or reliability of software [3]. Several researchers have aimed to specify maintainability by using various kinds of measures.

Chidamber & Kemerer (C&K) [4-6] have explored some of the object oriented design metrics for computing maintainability effort. These metrics includes WMC, DIT, NOC, CBO, RFC and LCOM. The four additional metrics such as MPC, DAC, NOM, SIZE2were proposed by Li and Henry [7]. SIZE1 is a traditional line of code size metrics. The maintainability is measured in terms of CHANGE metrics which informs the number of lines changed per class.

III. PRINCIPAL COMPONENT ANALYSIS

The data of a software system are highly interlinked. If there is existence of high correlation between the data sets, this indicates that these variables are aiming to describe the same goal and class property of the object to be measured. Similar is the situation with object-oriented system, where many metrics used to predict the maintainability are strongly correlated with each other. Suppose there are **n** number of classes and **m** number of metrics, then the total data set will be **n x m**. This will be a huge data set to analyse. Again the presence of several correlated data is merely duplicating the complexity for analysis.

Thus there is need of reduction of the data. To focus or analyse the minimum data, there is need of an effective tool to sort out the principal component, which will represent the original **n x m** data. Here comes the role of PCA i.e. principal component analysis.

PC analysis can be understood as:

1) It transforms raw metrics to variables that are not correlated to each other. So the PC data are uncorrelated or orthogonal.

2) It is a data reduction technique. It converts **n x m** data set to **n x p** data set, where **p<m**. This is by extracting linear combinations of the original data in such a way that they account for most of the original variation in the data set.

Thus Principal component analysis (PCA) is a standard technique to identify the underlying, independent/orthogonal dimensions that explain relations between the variables in a data set [8].

PC analysis is used to maximize the sum of squared loadings of each factor extracted in turn. In PC analysis a new variable (Pi) is constructed, called Principal Component (PC) out of a given set of variables Xj' s(j = 1,2,....,k).

$$P_1 = b_{11} X_1 + b_{12} X_2 + \ldots\ldots + b_{1K} X_K$$

$$P_1 = b_{21} X_1 + b_{22} X_2 + \ldots\ldots + b_{2K} X_K$$

.. ..

$$P_1 = b_{k1} X_1 + b_{k2} X_2 + \ldots\ldots + b_{KK} X_K$$

The factor b_{ij}s' are called loadings, and are find out in such a manner that the extracted PCs must be uncorrelated.

The variables with high loadings require some degree of interpretation. In order to find out these variables, and interpret the PCs, rotated components are considered. As the PCs are independent, orthogonal rotation is used. There are various strategies to perform such rotation and according to literature, varimax rotation is the most frequently used strategy. Eigenvalue or latent root is an integral part of PC. The sum of squared values of loadings describing the dimension is referred to as eigenvalue. The PCs with eigenvalue greater than 1 is usually taken for interpretation [9].

Given that **R** is a real symmetric matrix and that it has distinct roots, it can be stated as:

$$\mathbf{R = P.\Lambda.P^T}$$

In this, the eigenvalues $\boldsymbol{\lambda}$ lie on the diagonal of $\boldsymbol{\Lambda}$. PC_j is the eigenvector linked with $\boldsymbol{\lambda_j}$. Orthogonal linear combinations of the original data are presented by the coefficients of the m eigenvectors in PC. These linear combinations are the PC of **R**. Fraction of variance in the software measures is $\boldsymbol{\lambda_j/m}$, which is explained by PC_j. The diagonal of Λ forms a decreasing list that explains all of the data variance,

$$\boldsymbol{\lambda_1 > \lambda_2 > \ldots > \lambda_m.}$$

The total variance is the sum of the eigenvalues,

$$\sum_{i=1}^{m} \boldsymbol{\lambda_j = m \ [1]}$$

A large proportion of the total variance is usually explained by the first few PC. So to reduce the dimensionality of the problem without a substantial loss of the explained variance, restriction of analysis can be done to a relatively few PC. That is why most often a common stopping criterion is followed i.e. to consider PC with eigenvalues greater than one. In other way, the number of PC taken can be determined by the cumulative amount of variance associated with the PC [10]. For example, PCs those account for greater than 90% of the explained variance are included for interpretation.

$$\mathbf{PC = z.T}$$

In PC analysis **T** (in the above equation) describes the relationship between each of the **m** variables and the p s-significant PC. If **z** is the matrix of m standardized observed software measures, then this **z** can be reduced to **p** orthogonal PCs.

Most of the variations in the raw metric data are accounted to these components. Thus the software measures (metrics) are transformed in this manner to component/domain measures (metrics), and these domain measures are to be used as the input variables in neural-network modelling.

IV. NEURAL NETWORK MODELING

The Neural network is a network of interconnected neurons where information propagation occurs by firing electrical pulses via its connections. During the lifetime of a neuron, the connections or weights need to be adjusted [11].

The neural net consists of one input layer to feed raw data to network, one hidden layer to compute that data and one output layer for getting the computed results. The neural network is characterized by the weighted connections between neurons. [12]

If the connection between **i** and **j** neuron is **Wij**, then input value at neuron **i** will be the weighted sum of the entire inputs [13].

$$\mathbf{Net_i = \sum_j W_{ij} * Out_j - \theta_i}$$

Here $\boldsymbol{\theta_i}$ is the threshold of neuron **i**

The output of neuron **i** is the sigmoid function where **T** adjusts the temperature or gain of the function.

$$\mathbf{Out_i = \frac{1}{(1+e^{-Net_i/T})}}$$

The error of the neural network is defined by using the formula**:**

$$\frac{1}{2} \sum_{i=1}^{m} \mathbf{(d_i - Out_i)^2}$$

Where **M** = Number of output neurons,

$\mathbf{d_i}$ = Desired output for the i^{th} output of neuron

$\mathbf{Out_i}$ =Actual output.

The artificial neural network is employed for prediction of maintenance effort. All the domain metrics are used as input to the neural network. If the correlated raw object oriented software metrics used directly then the neural net models will not train properly [1]. The correct neural training can be done by transferring the raw data into principal components.

V. APPLICATION OF PCA IN NEURAL NETWORK MODELING

Various measures have been proposed by researchers to capture the quality of object oriented design and code [13-20]. The explored methods can be used to identify fault-proneness of the system with the help of artificial neural networks.

Several analyses also have been done to predict software quality by using statistical methods. ANN has seen an explosion of interest over the years, and is being successfully applied across a series of problem domains such as finance, medicine, engineering, geology and physics. Indeed, if there are problems of prediction, classification or control, neural networks will be an efficient asset to solve them [11]. Some of the applications of PCA to neural network modelling are listed in the Table 1.

TABLE I
STUDIES ON PCA AND NEURAL NETWORK

Sr. no.	Author	Reference	Year	Description
1	Khoshgaftaar *et al.*	[13]	1994	They explored the potential of PCA in improving Neural Network Predictions of Software Quality. It was found that the PCA resulted in quicker training of neural network.
2	Lanubile *et al.*	[19]	1995	They compared models for identifying fault-prone software components. They applied various modelling techniques in their study including principal component analysis, discriminant analysis, logistic regression, logical classification models, layered neural networks, and holographic networks. They found that no model was able to discriminate between components with faults and components without faults.
3	Khoshgaftaar *et al.*	[1]	1997	They applied neural networks for software quality modelling of a very large telecommunications system. They have used PCA to remove the high correlation of the software metrics.
4	Khoshgaftaar and Szabo	[14]	2000	They have used neural networks to predict software faults during testing. Ten softwares product measures were compared using two neural networks i.e. raw data analysis and principal component analysis. It was found that PCA has better predictive quality.
5	Briand *et al.*	[20]	2001	They investigated quality factors in object-oriented designs by applying various techniques such as principal component analysis, logistic regression and univariate regression analysis on an open multi-agent development environment: LALO (language agents Logical Object) which consists of 90 classes and 40K SLOC.
6	Quah *et al.*	[8]	2003	Neural Networks models were used for software quality prediction using object-oriented metrics. They have used two neural networks (WARD and GRNN) and proposed that GRNN has better predictability than WARD network. In this study they have used PCA with varimax rotation method for QUES (Quality Evaluation System) and HMI (Human Machine Interface) systems.
7	Thwin and Quah	[15]	2005	They have applied neural networks for software maintainability prediction and also used PCA. They have taken OO metrics as independent variable and used two neural network models namely ward neural network and general regression neural network (GRNN). They concluded that GRNN network model can predict more accurately than Ward network model. The data systems used are QUES and UIMS (User Interface Systems).
8	Aggarwal *et al.*	[9]	2006	They have applied artificial neural network (ANN) to predict maintainability using object-oriented metrics. In their study maintenance effort was used as dependent variable and principal component of 8 object oriented metrics as independent variable. They found that ANN method is very useful in software quality prediction.
9	Aggarwal *et al.*	[16]	2007	They investigated the effect of design metrics on fault proneness in object oriented systems. They have collected data from Java applications which contained 136 classes. They have used PCA and extracted up to 6 principal components. Their proposed model predicted faulty classes with more than 80% accuracy.
10	Katiyar *et al.*	[21]	2010	They estimated software quality by applying object-oriented metrics through artificial neural network. They have used principal components of eight OO metrics as the independent variables and found that ANN method was valuable for constructing software quality model.
11	Shaik *et al.*	[17]	2011	They investigated the result of object oriented design software metrics on fault proneness in object oriented systems. They have applied principal component method to object oriented metrics to get the fault proneness. They have extracted up to 6 components and found that the number of dimensions capture in PCA is much lower than the number of metrics.
12	Zhong *et al.*	[18]	2011	They tried to predict software quality by applying principal components analysis and wavelet neural network and genetic algorithm. They stated that model combining principal components analysis with wavelet neural network and genetic algorithm can obtain better prediction accuracy than the models using the neural network of back propagation and generalized regression neural network.

VI. CONCLUSION

Improvement in the software development processes and the software quality needs a higher degree of accuracy during prediction. Several statistical models including artificial neural networks are now becoming a part of software engineering for the prediction and measurement. This process of estimation is being overwhelmed with large number of software metrics, huge number of executable statements and lines of codes. Furthermore, these data often tend to have a higher degree of correlations among themselves which makes the prediction process more complex. To get around these problems, principal component analysis has shown a way to most of the researchers. On one hand it reduces the multidimensional problem of the huge data by converting it to a small orthogonal datasets, on the other hand it finds out the factorial relationship and most of the variance of the original dataset. The resultant orthogonal domain metrics provides a swift base for the easier prediction by neural networks. Thus principal component analysis has incredible importance in evaluating the original data and predicting the software quality using neural network models.

REFERENCES

[1] T.M. Khoshgaftaar, E.D. Allen, J.P. Hudepohl and S.J. Aud "Application of neural networks to software quality modeling of a very large telecommunications system," IEEE Transactions on Neural Networks, Vol. 8, No. 4, pp. 902--909, 1997.

[2] P. Goodman, "Practical Implementation of Software Metrics", McGraw Hill, London, 1993.

[3] Y. Dash, S.K. Dubey and A. Rana, "Maintainability Measurement in Object Oriented Paradigm", International Journal of Advanced Research in Computer Science (IJARCS), Vol.3, no.2, pp. 207-213, April 2012.

[4] S. R. Chidamber and C. F. Kemerer,"Towards a Metrics Suite for Object Oriented design". Proc. Conference on Object-Oriented Programming: Systems, Languages and Applications (OOPSLA'91), Published in SIGPLAN Notices, vol 26 no. 11, pp.197-211, 1991.

[5] S. R. Chidamber and C. F. Kemerer, "A metrics suite for object oriented design." IEEE Trans. Software Eng., vol. 20, no. 6, pp. 476–493, 1994.

[6] S. R. Chidamber, D. Darcy and C. F. Kemerer, "Managerial use of Metrics for Object-Oriented Software: An Exploratory Analysis", IEEE Transactions on Software Engineering, vol.24 no.8, pp. 629-639, 1998.

[7] W. Li and S. Henry, "Object-Oriented Metrics that Predict Maintainability", Journal of Systems and Software, vol 23, no.2, pp.111-122, 1993.

[8] J. T. S. Quah, M. M. T. Thwin, "Application of Neural Networks for Software Quality Prediction Using Object-Oriented Metrics, Proceedings of the International Conference on Software Maintenance (ICSM'03), IEEE Computer Society, 2003.

[9] K. K. Aggarwal, Y. Singh,A. Kaur and R. Malhotra, "Application of Artificial Neural Network for Predicting Maintainability using Object-Oriented Metrics, World Academy of Science, pp. 140-144, 2006.

[10] T.M. Khoshgaftaar and R.M. Szabo, "Using Neural Networks to Predict Software Faults During Testing, IEEE Transactions on Reliability, vol. 45, no. 3, pp. 456-462, Sept. 1996.

[11] Y. Dash and S.K. Dubey, "Quality Prediction in Object Oriented System by Using ANN: A Brief Survey", International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE), Vol. 2, no. 2, Feb. 2012.

[12] F. Nielsen, Neural Networks – algorithms and applications. 2001 Available at: http://www.glyn.dk/download/Synopsis.pdf. Accessed on Dec.2011.

[13] T.M. Khoshgaftaar and R.M. Szabo, "Improving Neural Network Predictions of Software Quality Using Principal Components Analysis", IEEE, pp. 3295-3300, 1994.

[14] T. M. Khoshgoftaar, E. B. Allen, Zhiwei Xu, "Predicting testability of program modules using a neural network", In Proc. of 3rd IEEE Symposium on Application-Specific Systems and Software Engineering Technology, pp.57-62, 2000.

[15] M. M. T. Thwin,T. S. Quah, "Application of neural networks for software maintainability prediction using Object-oriented metrics", Journal of systems and software, Vol.76, No.2, pp.147-156, 2005.

[16] K. K. Aggarwal, Y. Singh, A. Kaur and R. Malhotra, "Investigating effect of design metrics on fault proneness of object oriented systems, Journal of Object Technology,vol.6, no. 10, pp. 127-141, 2007.

[17] A. Shaik, N. Satyanarayana, M Huzaifa, N. Shaik, M. Z. Naveed, S. V. A. Rao and C. R. K. Reddy. "Invetsigate the result of object oriented design software metrics on fault proneness in object oriented systems: A case study," Journal of emerging trends in computing and emerging sciences, Vol. 2, no. 4, pp. 201-208, 2011.

[18] C. Zhong, Q. Hu, F. Yang and M. Yin. "Software Quality Prediction Method with Hybrid Applying Principal Components Analysis and Wavelet Neural Network and Genetic Algorithm,' International Journal of Digital Content Technology and its Applications, Vol. 5, no. 3, pp.225-234, 2011.

[19] F. Lanubile, A. Lonigro, G. Visaggio, "Comparing models for identifying fault-prone software components", In: Proc. of the 7th Int'l. Conf. Software Eng. and Knowledge Eng., pp. 312–319, June 1995.

[20] L.C. Briand, J. Wüst, and H. Lounis, "Replicated Case Studies for Investigating Quality Factors in Object-Oriented Designs," Empirical Software Engineering. International Journal (Toronto, Ont.), 6(1), pp.11–58. 2001.

[21] N. Katiyar and Y. H. Siddiqui, "Estimation of Software Quality using Object-Oriented Metrics through Artificial Neural Network," VSRD-TNTJ, Vol. I (3), pp.110-118, 2010.

