Expert Systems with Applications 38 (2011) 2187–2194

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Functional networks as a novel data mining paradigm in forecasting software development efforts

Emad A. El-Sebakhy *

Medical Artificial Intelligence (MEDai), Inc., An Elsevier Company, Millenia Park One, 4901 Vineland Road, Suite 450, Orlando, FL 32811, USA

ARTICLE INFO

Keywords:
Support vector machines
Neural networks
Type I fuzzy logic inference system
Data mining
Regression
Functional networks
Software development effort

ABSTRACT

This paper proposes a new intelligence paradigm scheme to forecast that emphasizes on numerous software development elements based on functional networks forecasting framework. The most common methods for estimating software development efforts that have been proposed in literature are: line of code (LOC)-based constructive cost model (COCOMO), function point (FP) based on neural networks, regression, and case-based reasoning (CBR). Unfortunately, such forecasting models have numerous of drawbacks, namely, their inability to deal with uncertainties and imprecision present in software projects early in the development life-cycle. The main benefit of this study is to utilize both function points and development environments of recent software development cases prominent, which have high impact on the success of software development projects. Both implementation and learning process are briefly proposed. We investigate the efficiency of the new framework for predicting the software development efforts using both simulation and COCOMO real-life databases. Prediction accuracy of the functional networks framework is evaluated and compared with the commonly used regression and neural networks-based models. The results show that the new intelligence paradigm predicts the required efforts of the initial stage of software development with reliable performance and outperforms both regression and neural networks-based models.

1. Introduction

Software intelligence maintainability is defined as the ease of finding and correcting errors in the software. One of the most important issues encountered during the process of software development is a company's ability to accurately estimate the efforts of the initial stage of development. Inaccurate estimation of development efforts lowers the proficiency of the project, wastes the company's budget, and can result in failure of the entire project. This has promoted much research during the past three decades, to identify the factors that influence development effort (Ahmed, Saliu, & AlGhamdi, 2005; Boehm's HYPERLINK Data, 1981; Finnie & Wittig, 1996; Heiat, 2002; Huang, Ho, Ren, & Capretz, 2007; Kadoda, Cartwright, & Shepperd, 2000a; Kadoda, Cartwright, Chen, & Shepperd, 2000b; Martin, Pasquier, Yanez, & Tornes, 2005; Park & Baek, 2007; van Koten & Gray, 2006). Many existing research papers have proposed various effort estimation techniques. Unfortunately no estimation technique has proved consistently accurate. For this reason there has been growing interest in recent years in exploring a variety of both data mining and machine learning schemes either as a complement or an alternative to existing models.

One of the critical factors in software product quality is identifying how efficiently a software development process has been defined and deployed. A high-quality product could be produced by an individual's effort once. But, without a well-defined development process, reproducing a software product of the same quality as its prior ones cannot be guaranteed. To maintain consistent quality of software, the need for a well-defined software development process is inevitable. The most common techniques that have been utilized in software effort estimation are: artificial neural networks (Heiat, 2002; Jun & Lee, 2001; Park & Baek, 2007), neuro-fuzzy logic inference systems (Ahmed et al., 2005; Martin et al., 2005), Bayesian statistics (van Koten & Gray, 2006), and case-based reasoning (Kadoda et al., 2000a, 2000b).

Recently, functional network has been introduced by Castillo (1998), Castillo, Cobo, Gutierrez, and Hadi (1999), El-Sebakhy (2004), El-Sebakhy, Faisal, El-Bassuny, Azzedin, and Al-Suhaim (2006) and El-Sebakhy, Hadi, and Faisal (2007) as a generalization of the standard neural network. This new computational intelligence paradigm is still a new framework and it was utilized once in the area of software engineering by El-Sebakhy (2008) to identify the software reliability. The results have shown that this novel approach is reliable and more accurate than the most common

* Tel.: +1 321 281 4546; fax: +1 321 281 4486.
E-mail addresses: e.el-sebakhy@elsevier.com, eae6@cornell.edu

0957-4174/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2010.08.005

statistical and machine learning techniques, namely, feedforward neural networks and multiple linear regressions.

The motivation behind this research is to investigate the capabilities of functional networks as a new intelligence paradigm scheme to predict and identify the software development efforts. The methodology and training algorithm process are explained in details below. The implementations were carried out on representative datasets related to the target systems. In addition, the comparative studies between the new intelligence scheme versus the existing models presented in van Koten and Gray (2006), which include regression-based, neural networks-based, and Bayesian belief network-based models, in terms of their performance measures values, as recommended in the literatures. Comparative studies will be carried out to measure the capabilities of this new paradigm in predicting the software development efforts and evaluate its performance against the most common and recently published data mining schemes.

Despite the importance of software maintenance, it is unfortunate that little work has been done as regards developing predictive models for software maintainability, particularly object-oriented software system, which is evident in the fewer number of software maintainability prediction models, that are currently found in the literature. Quality is an important factor in the success of any software product. Defective software systems can cause software failures, increase maintenance cost, and decrease customer satisfaction. This in turn increases the interest in effective software defect prediction models. Defect prediction of object-oriented classes helps software engineers focus their quality assurance activities and assists managers allocate effort and resources more efficiently.

The rest of this paper is organized as follows. Section 2 of this paper provides a brief literature review. The proposed functional networks intelligent system paradigm is described in detail in Section 3. Implementation of the proposed models is discussed in Section 4. Section 5 contains discussions of results while Section 6 presents conclusions and recommendations.

2. Literature review

Software development effort estimation is an important factor for scheduling and determining the cost of software project; many researchers have done empirical experiments using different machine learning techniques in order to get an accurate estimation of software development effort. The most common computational intelligence scheme that have been utilized so far by numerous researchers in literature is the standard multilayer perceptron feedforward neural networks to estimate software development effort; for instance, the authors in De Barcelos Tronto, Simíes da Silva, and Sant'Anna (2008), Zhou and Leung (2007) and Quah and Thwin (2003) examine the performance of back-propagation neural networks in estimating software development effort by evaluating the performance of neural networks effort estimation models on actual project data. They concluded that despite the restrictions of the project dataset, neural networks have shown their ability to provide an adequate effort estimation model. It has shown competitive results when compared to multiple regressions, SLIM, COCOMO, and function points. A primary advantage of a neural network approach is that it is adaptable and nonparametric.

To define a well-organized and efficient software development process for an organization is one thing. It is totally another thing to customize an organizational process to a specific project, which is known as 'process tailoring'. So far, process tailoring has been conducted by few well-experienced software engineers using their own heuristic methods. Hence, if a project team does not have experienced process engineers, they have no choice but to deploy an organizational process as is. Even if they have a few process engineers with some experience, tasks related to process tailoring are knowledge intensive and time-consuming. Assuming that they had their own customized process by individual efforts, this high dependence on individual experience may make them hesitate to answer questions regarding rationales for their tailoring results.

It is hard to tailor a process without process engineers' knowledge. It is equally difficult to automate the steps of process tailoring like a mechanical process since the software development process varies with project environment. Even though it cannot be fully automated, we contend that it could have a more systematic and objective method for tailoring a process than existing heuristic ones so that it can be preserved as an asset and reused within an organization. We have studied the process a novice-level process engineer goes through until he or she becomes an expert. While it is difficult to develop a system that would exactly replicate how the process engineer would tailor a process using prior knowledge based on his or her experience, we have found an alternative way to substitute the process by using learning theory, especially an artificial neural network model. The basic idea is that the more knowledge is learned, the more steps of process tailoring can be automated. The purpose of our study is to provide a mechanism that can reconfigure a generic process to an optimized one for a given project environment by reusing knowledge acquired by process engineers from process tailoring experience. The expected benefit of using this mechanism is to reduce the duration of the process filtering phase, which is one of the three tailoring phases. It also provides reasonable rationales for the tailoring results and facilitates the reuse of process tailoring knowledge acquired from prior project experience to successive projects.

The authors in Martin et al. (2005) utilized the neuro-fuzzy inference systems to estimate the software development effort and compare the obtained estimated values with the statistical regression methodology. In addition, the authors in Ahmed et al. (2005) they presented a transparent fuzzy logic based framework, equipped with training and adaptation algorithms for development effort prediction. Their framework allows contribution from experts, and also enables the prediction technique to model and adapt to the environment of the prediction problem. Furthermore, the author in Ryder (1998) used fuzzy modeling techniques incorporated in the COCOMO model and the Function Points model. The main advantage of this approach is that it handles uncertainty of the data. The author in Huang et al. (2007) combines fuzzy logic, feedforward neural networks, and adaptive neuro-fuzzy inference systems; then they select the best performance of these approaches.

Bayesian belief network and case-based reasoning (CBR) model was used by Kadoda et al. (2000a, 2000b), Pendharkar, Subramanian, and Rodger (2005) and Quah and Thwin (2003) to provide an estimate of software development effort. The authors proved that their technique can be utilized to provide a point forecast for a software development effort. Their results outperform the most common exiting schemes. The key factor in selecting a software estimation model is accuracy of its predictions. According to Leung and Fan (2002) many studies have attempted to evaluate the performance of software effort estimation models. However, many of them were found to be not very accurate (Kemerer, 1987). In a study by Kemerer using COCOMO, FP, SLIM, and Estimacs, most models showed large estimation errors, ranging from a MAPE of 57–800% (Kemerer, 1987). As it is shown in El-Sebakhy et al. (2007) most of the above proposed approaches have numerous of drawbacks. In addition, each modeling scheme has its own advantages and shortcomings, but as yet no model has proved to be successful at effectively and consistently predicting software development effort. Therefore, we investigate the capability of the functional networks intelligence system paradigm in predicting