Available online at www.sciencedirect.com
ScienceDirect
The Journal of Systems and Software 81 (2008) 1853–1867
The Journal of Systems and Software
www.elsevier.com/locate/jss

Software development cost estimation using wavelet neural networks

K. Vinay Kumar, V. Ravi *, Mahil Carr, N. Raj Kiran

Institute for Development and Research in Banking Technology, Castle Hills Road #1, Masab Tank, Hyderabad 500 057, AP, India

Received 30 May 2007; received in revised form 28 December 2007; accepted 28 December 2007
Available online 8 January 2008

Abstract

Software development has become an essential investment for many organizations. Software engineering practitioners have become more and more concerned about accurately predicting the cost and quality of software product under development. Accurate estimates are desired but no model has proved to be successful at effectively and consistently predicting software development cost. In this paper, we propose the use of wavelet neural network (WNN) to forecast the software development effort. We used two types of WNN with Morlet function and Gaussian function as transfer function and also proposed threshold acceptance training algorithm for wavelet neural network (TAWNN). The effectiveness of the WNN variants is compared with other techniques such as multilayer perceptron (MLP), radial basis function network (RBFN), multiple linear regression (MLR), dynamic evolving neuro-fuzzy inference system (DENFIS) and support vector machine (SVM) in terms of the error measure which is mean magnitude relative error (MMRE) obtained on Canadian financial (CF) dataset and IBM data processing services (IBMDPS) dataset. Based on the experiments conducted, it is observed that the WNN-Morlet for CF dataset and WNN-Gaussian for IBMDPS outperformed all the other techniques. Also, TAWNN outperformed all other techniques except WNN.

Keywords: Software development effort; Software cost estimation; Wavelet neural networks; Threshold accepting based wavelet neural network

1. Introduction

Software development has become an important activity for many modern organizations. In fact the quality, cost, and timeliness of developed software are often crucial determinants of an organization's success. There are significant financial and strategic implications for development projects in terms of activity scheduling and cost estimation. Software cost estimation is one of the most critical tasks in managing software projects. Development costs tend to increase with project complexity and hence accurate cost estimates are highly desired during the early stages development (Xu and Khoshgoftaar, 2004). A major problem of the software cost estimation is first obtaining an accurate size estimate of the software to be developed (Kitchenham et al., 2003). An important objective of the software engineering community has been to develop useful models that constructively explain the software development life cycle and accurately estimate the cost of software development.

In order to effectively develop software in an increasingly competitive and complex environment many firms use software metrics as part of their project management process. The field concerned with managing software development projects using empirical models is referred to as software metrics (Fenton and Pleeger, 1997). Software metrics are aspects of software development (either the software product itself, or the development process producing it) that can be measured. These metrics can be used as variables in models to predict some aspect(s) of the development process or product.

Estimating development effort and schedule, can include activities such as assessing and predicting system quality, measuring system performance, estimating user satisfaction and in fact any modeling task involving measurable

* Corresponding author. Tel.: +91 40 2353 4981x2042; fax: +91 40 2353 5157.

E-mail addresses: vinaykadiyam@gmail.com (K. Vinay Kumar), rav_padma@yahoo.com (V. Ravi), mahil.carr@gmail.com (M. Carr), nrajkiran@gmail.com (N. Raj Kiran).

0164-1212/$ - see front matter
doi:10.1016/j.jss.2007.12.793

attributes of interest within the software development sphere (Gray, 1999). However, the most researched area has been effort estimation as it carries the greatest promise of benefit for project management. Software effort estimates are crucial for estimating the amount of manpower needed for the project. This estimate determines staff allocation and schedule for a software project. Since human effort is the major cost driver in a software project, the effort estimate determines the budget of the project. Accurate effort estimates help software consultancies to make appropriate bids when quoting for tenders – a lower estimate than the actual will lead to a loss and an unreasonably high estimate will loose the bid. Such estimation models are developed using a set of measures that describe the software development process, product and resources such as developer experience, system size and complexity and the characteristics of the development environment, respectively. The output of the model is some measure of effort in terms of person hours (months or years).

There are many models and tools used in software cost estimation that provide invaluable information regarding efforts and expenditure to the management to bid for a project (Kitchenham et al., 2003). The most commonly used methods for predicting software development effort have been based on linear-least-squares regression such as COCOMO (Boehm, 1981; Fenton and Pleeger, 1997; Pressman, 1997). As such, COCOMO is extremely susceptible to local variations in data points (Miyazaki et al., 1994). Additionally, the models have failed to deal with implicit nonlinearities and interactions between the characteristics of the project and effort (Gray, 1999). Software cost estimation models are deemed to be acceptably accurate if they yield estimates with 25% mean relative error to the actual and this must be true at least 75% of the time. There is always scope for developing effort estimation models with better predictive accuracy (Kemerer, 1987).

In recent years, a number of alternative modeling techniques have been proposed. Existing datasets have their performance examined with some success including those in Gray and MacDonell (1997). Alternative models include artificial neural networks (Verkatachalm, 1993), analogy-based reasoning, regression trees and rule induction models. Gray and MacDonell (1997) applied fuzzy logic to software metric models for development effort estimation. They outlined the use of fuzzy logic for defining software metrics as linguistic variables and for modeling process. They made comparison of results obtained from an elementary fuzzy inference system with other techniques such as linear regression and neural network techniques and found that it outperformed. Gray (1999) presented several different predictive model-building techniques such as robust statistical procedures, various forms of neural network models, fuzzy logic, case-based reasoning and regression trees. He also described a simulation-based study on the performance of these empirical modeling techniques using size and effort software metric dataset and observed that M-estimation regression outperformed all other parametric and non-parametric techniques. Xu and Khoshgoftaar (2004) presented an innovative fuzzy identification software cost estimation modeling technique, which is an advanced fuzzy logic technique that integrates fuzzy clustering, space projection, fuzzy inference and defuzzification. Based upon their case study on the COCOMO'81 database it was observed that the fuzzy identification model provided significantly better cost estimations than the three COCOMO models, i.e. basic, intermediate and detailed. Many researchers have applied the neural networks approach to estimate software development effort (Hughes, 1996; Jorgerson, 1995; Samson et al., 1997; Schofield, 1998; Seluca, 1995; Heiat, 2002; Srinivasan and Fisher, 1995; Wittig and Finnie, 1997). Most of their investigations have focused more attention on the accuracy of the other cost estimation techniques such as COCOMO and Function Point Analysis. Idri et al. (2002) have also done research on estimating software cost using the neural networks approach and fuzzy if–then rules on the COCOMO'81 dataset. The use of software effort estimations by means of analogy have been evaluated and confirmed in several studies (Angelis and Stamelo, 2000; Jorgenson et al., 2003; Shepperd and Schofield, 1997). Wolverton (1974) explained the use of estimation by analogy and described the similarities and differences of existing software cost estimating techniques. Mukhopadhyay et al. (1992) utilized analogy for software effort estimation by retrieving the most similar cases. Their results showed that the analogy approach is more accurate and consistent than the function point and COOCMO models. Chiu and Huang (2007) used adjusted analogy-based software effort estimation based on similarity distances between pairs of projects. They demonstrated that applying effort adjustment to the analogy-based software effort estimations is a feasible approach to improve estimating using the three distance metrics. In addition, they demonstrated that the proposed adjusted analogy-based estimations are also compatible to the widely used estimation models of ANN, CART and OLS.

The rest of the paper is organized as follows: In Section 2, the techniques applied to software cost estimation are described briefly. Section 3 introduces wavelet neural networks. Section 4 describes the datasets and data preparation for our empirical study, while Section 5 presents our experimental methodology and compares the estimation performances with the most often used software effort estimation methods in the literature. Finally, Section 6 summarizes our work and outlines further directions.

2. Brief overview of the techniques employed

2.1. Multilayer perceptron (MLP)

Multilayer perceptrons (MLPs) are feed-forward neural networks trained with the standard back propagation algorithm. They are supervised networks so they require a desired response to be trained. They learn how to trans-