Ensemble of neural networks with associative memory (ENNA) for estimating software development costs

Yigit Kultur a,*, Burak Turhan b, Ayse Bener a

aDepartment of Computer Engineering, Bogazici University, 34342 Istanbul, Turkey
bInstitute for Information Technology, National Research Council, Canada

A R T I C L E   I N F O

Article history:
Received 23 June 2008
Received in revised form 19 March 2009
Accepted 3 May 2009
Available online 10 May 2009

Keywords:
Software cost estimation
Neural network
Ensemble
Associative memory
Adaptive resonance theory
Wrapper

A B S T R A C T

Companies usually have limited amount of data for effort estimation. Machine learning methods have been preferred over parametric models due to their flexibility to calibrate the model for the available data. On the other hand, as machine learning methods become more complex, they need more data to learn from. Therefore the challenge is to increase the performance of the algorithm when there is limited data. In this paper, we use a relatively complex machine learning algorithm, neural networks, and show that stable and accurate estimations are achievable with an ensemble using associative memory. Our experimental results show that our proposed algorithm (ENNA) produces significantly better results than neural network (NN) in terms of accuracy and robustness. We also analyze the effect of feature subset selection on ENNA’s estimation performance in a wrapper framework. We show that the proposed ENNA algorithm that use the features selected by the wrapper does not perform worse than those that use all available features. Therefore, measuring only company specific key factors is sufficient to obtain accurate and robust estimates about software cost estimation using ENNA.

© 2009 Elsevier B.V. All rights reserved.

1. Introduction

In the last two decades, the software development industry has been growing exponentially in terms of revenues. However, the ubiquitous and around the clock software development made the industry highly competitive forcing companies to work under tight margins. Therefore, software development companies needed to find creative ways to keep their costs under control and make more profit. The major asset (cost) for a software company is its work force. In a given project, if the estimated effort is much more than the actual effort, then the resources are wasted. On the other hand, if the estimated effort is much less than the actual effort, then neither the deadline for the project is met nor the product achieves the desired quality level. Therefore, an accurate estimation of effort is essential for a software development company.

Companies usually have a small number of completed projects and consequently limited amount of effort data for estimating the effort of new projects. It is hard to make accurate estimations with scarce data because the more complex the problem and estimation methods become, the harder it is to learn effort function with small datasets.

In this paper, we investigate the use of neural networks for effort estimation. Neural networks (NN) have been widely used in previous research [1–4]. However, NN are complex models with too many parameters to estimate. Hence, relatively large datasets are required for accurate and stable estimates. Furthermore, NN are sensitive to training data, prone to get stuck in local minimums and do not explicitly use analogies in training data.

In order to overcome these drawbacks, we propose the ensemble of neural networks with associative memory (ENNA) where we benefit from bootstrapping and look-up tables. We compare our ENNA model with standard NN and another widely used method in effort estimation, regression trees, and on publicly available datasets. The results of our experiments show that ENNA not only improves the effort estimation performance in terms of MMRE, MdMRE and PRED(25), but also achieves robust and stable results with lower variances. As a proof of stability, we will show that PRED(25) results of ENNA are at least four times better than standard NN. ENNA also achieves the minimum relative error rates compared to the other models.

In practice, ENNA allows project managers to make accurate and stable effort estimations even with limited amount of data and it does not require any local calibration. Thus, it provides a mean for an efficient allocation of project resources with minimum manual labor.

We further analyze the effect of feature subset selection on the performance of ENNA using a wrapper framework. We show that using only a subset of dataset features produces equally well or better results while decreasing the complexity of our proposed model. Using less number of features reduces the number
parameters of the model and further overcomes the problem of limited amount of available cost data.

The remaining of this paper is structured as follows: in Section 2, background information about the effort estimation literature is given. In Section 3, multiple neural networks are combined to propose ensemble of neural networks (ENN) model. Furthermore, associative memory is combined with ENN to form the final model, an ensemble of neural networks with associative memory (ENNA). In Section 4, datasets, evaluation criteria, and validation methods are explained. Additionally, experimental results and the effect of feature subset selection are analyzed and threats to validity are discussed. In Section 5, the concluding remarks are stated and the future work is discussed.

2. Background

Many researchers proposed parametric models for effort estimation [5–11]. These models define effort as a function of variables such as lines of code, function points, personnel capabilities and process maturity. COCOMO, COCOMO II and Function Points Analysis are the most popular parametric models [12].

Despite their popularity, parametric models have important drawbacks [13]. Parametric models are unable to deal with exceptional conditions such as personnel, teamwork and match between skill-levels and tasks. In addition, parametric models can only be calibrated manually. Machine learning methods are good alternatives for addressing these drawbacks. So far, many researchers have used machine learning methods in effort estimation [1–4,14–16]. These methods focus on learning from past projects to provide estimations for future projects. Case-based reasoning, regression trees, artificial neural networks, and genetic algorithms are some of the most popular machine learning methods in the effort estimation domain [12].

Case-based reasoning is the process of estimating the effort of a current project by using the actual efforts of similar past projects. Estimation by analogy is a form of case-based reasoning. The steps of this method are identification of the project as a new case, retrieval of similar projects from the repository, and effort estimation by using knowledge of similar projects. The similarity of projects can be assessed via various approaches including nearest neighbor algorithms, manual human expert guidance, and goal directed preference [14].

Regression tree is a hierarchical model where a local region is identified in a sequence of recursive splits in a small number of steps [17]. Software project attributes are used to split projects into smaller groups and this process is recursively repeated to form a regression tree [12].

Genetic algorithms treat software effort equations as chromosomes. Instantly, a number of random effort equations are generated. Thereafter, a new population of effort equations is created from previous effort equations by applying genetic operations to the most accurate equations until an accurate enough effort equation is derived [15,16].

Neural networks are universal approximators [18]. In other words, a neural network can learn any function. Since software effort can be defined as a function of several project characteristics, neural networks have the ability of learning this function and estimating the efforts of new software projects. Previously, neural networks have popularly been used for software effort estimation [1–4].

Despite their popularity in software effort estimation, neural networks have several drawbacks. Firstly, neural networks are unstable structures. In other words, small changes in the training set cause large difference in the trained neural network [19]. Secondly, overtraining of neural networks has a negative impact on prediction accuracy. These drawbacks may be eliminated by combining multiple neural networks. Therefore, we look for an answer to the following question: Can we improve software effort estimation accuracy by using an ensemble of neural networks rather than a single one? Using an ensemble of neural networks and combining their estimations in a robust way provide the answer. Previously, the ensemble of learners has been used in defect prediction [20]. There are also novel uses of neural networks structures in machine learning to improve the learning performance of the network [21]. However, in this research we wanted to construct a neural network for software cost estimation problem such that our aim was to get a stable structure with high prediction accuracy and low standard deviation.

Neural networks are memoryless structures. They store project information as weights during training and once they are trained, they do not need those projects any more. Naturally, the following question appeared: Can we use the past projects for further accuracy improvement? To find an answer, we used the past projects to decrease the bias of the ensemble. Since the “associated” past projects are retrieved and used, we use the term “associative memory”.

3. Ensemble of neural networks with associative memory (ENNA)

In the first subsection, the mechanism of ensemble of neural networks (ENN) is explained. Afterwards, the concept of associative memory is combined with ENN to form an ensemble of neural networks with associative memory (ENNA).

3.1. Ensemble of neural networks (ENN)

Neuroscientists have discovered that individual neural networks are very noisy and are unable to perform a certain task alone [22]. The neural networks in the visual cortex, which provide the ability of sight, can be given as an example. Examining the activity of a single visual neural network is not enough to reconstruct the visual scene that the owner of the brain is looking at. Therefore, a population of visual neural networks provides the visual scene.

There is a similar issue for multilayer perceptrons (MLP). If the function is complex, a MLP may not learn the function perfectly in all regions of the input space. Therefore, a significant bias may appear for regions of the input space in which the function is not perfectly trained. In addition to this, small changes in the training set of a MLP cause a large difference in the trained MLP. In other words, learning algorithm has high variance, which affects the learning performance negatively [19,23]. Another problem is over-training. Initially, all weights in MLP are close to zero and consequently have little effect. As the training continues, the most important weights start moving away from zero and they become utilized. However, if the training continues further, “all” weights, including the less important ones, move away from zero and become utilized. This increases the complexity and leads to a poor generalization. Using a validation set solves the problem of over-training. When the error on the validation set starts to increase beyond a certain point, the training is stopped [23]. However, this method is not applicable for the software effort estimation domain in which data sets are very small. All instances are essential for training and they should be used for training.

Using an ensemble of MLP’s and combining the results of individual MLP’s may provide a solution to the above-mentioned problems. The main idea of the ensemble is training each MLP with a specific training set. Each training set is generated through a random choice of training projects from the original training set that
contains all past projects. After each project is chosen, it is replaced back to the original set. Therefore, a project may appear more than once in the generated specific training set. The number of the projects in each training set is equal to the number of the projects in the original training set. This method is called bootstrapping and it is considered as the best way of forming specific training sets for domains with very small datasets such as software effort estimation [23].

In this research, each MLP has one hidden layer with eight hidden nodes and it is trained with the Levenberg–Marquardt algorithm [24,25]. The Levenberg–Marquardt algorithm is a type of backpropagation algorithm and is very efficient in training moderate sized neural networks that have up to several hundred weights [26]. The hidden nodes and the output node apply a hyperbolic tangent sigmoid function to the weighted sum of its inputs. The ensemble consists of 20 MLP’s that have these attributes.

The training process is shown in Fig. 1. The original training set is the set of past projects whose features and actual effort values are known. Training sets are generated by bootstrapping and each MLP is trained with the corresponding training set.

Now, ENN is ready for estimating the effort of new projects. When a new project is given as input, each MLP in the ensemble returns its own result as in Eq. (1):

\[
PM = \begin{bmatrix}
PM_1 \\
PM_2 \\
\vdots \\
PM_{20}
\end{bmatrix}
\]  

where \(PM_i\) is the person-month value estimated by \(i\)th MLP.

The estimations made by some MLP’s may deviate greatly from the largest group of nearby estimations because some MLP’s would have fallen into the local minimum and some others would have been inaccurately trained because of the random individual training set. Therefore, a simple average of all estimations would not be an accurate result. Instead, it would be good practice to detect the largest group of nearby results and return the average of this group. In other words, the results that are not a member of this group should be ignored, i.e. treated as outliers. To this end, clustering is used. At that moment, the number of result groups is not known. Therefore, the clustering algorithm should start with one single cluster, add new clusters as they are needed and delete the empty clusters. Adaptive resonance theory (ART) algorithm, which follows such an incremental approach, is used in this research [27]. To enhance clarity, ART algorithm is simulated in Fig. 2. The ensemble consists of 20 MLP’s and provides 20 estimations for the given new project. Each MLP gives a result that is a scalar effort value in person-months. ART algorithm divides these results into four clusters. The largest cluster is C3. Therefore, the average of the results in C3 is taken to provide the result of the ensemble. \(\rho\), namely vigilance, indicates the distance between the center and the border of the cluster.

By using ART algorithm the largest group of results is detected. The average of the results in the largest group is taken as in Eq. 2 to provide the effort estimated by the ensemble.

\[
PM_{\text{Ensemble}} = \frac{\sum PM_i}{N_C}
\]

where \(C\) is the largest cluster, \(N_C\) is the number of the results in \(C\), and \(\forall PM_i \in C\).

The overall process of ensemble effort estimation is shown in Fig. 3. A new project, namely Production Instance, is fed to ENN. Each MLP evaluates the given project and provides an estimated effort value. \(PM_i\) is the estimated effort value of MLP\(_i\). At this moment, there are 20 estimated values. These values are clustered by using ART algorithm and the average of the largest cluster is calculated as in Eq. (2) to provide the estimated value by the ensemble, namely \(PM_{\text{Ensemble}}\).

### 3.2. Associative memory

Using an ensemble rather than a single MLP and combining the results of MLP’s by using a robust method provide a model for an accurate software effort estimation. However, there may still be considerable bias of this model. Therefore, the concept of associative memory is used for estimating and correcting the bias.

An associative memory is a system that stores the mappings of specific input representations to specific output representations.
The word "associative" comes from the fact that this system "associates" two patterns such that when one is encountered subsequently, the other can reliably be recalled [28]. In the software effort estimation domain, the effort database of past projects corresponds to the associative memory.

When a new project is given to the ensemble, the estimated effort is returned as in Eq. (2). In order to correct the bias, the bias of the model for the new project must be estimated.

The key idea here is that the model bias is similar for similar projects. Therefore, the similar past projects in the effort database
are used for estimating the bias of the model for the new project. These projects are retrieved from the database using \( k \) nearest neighbor algorithm [23]. These nearest neighbors are given to the ensemble to get estimated effort values as in Eq. (2). The difference between the actual and the estimated values gives the bias of the ensemble for those projects. The average of biases for the nearest neighbors is taken to get the estimated bias for the new project as in Eq. (3).

\[
\text{Bias}_{\text{Estimated}} = \frac{1}{k} \sum_{i=1}^{k} (\text{PM}_{\text{Actual}}(i) - \text{PM}_{\text{Estimated}}(i))
\]

where \( N_k \) is the collection of \( k \) nearest neighbors, \( \text{PM}_{\text{Actual}}(i) \) is the actual person-month value for project \( i \), and \( \text{PM}_{\text{Estimated}}(i) \) is the estimated person-month value for project \( i \). Lastly, the estimated bias value is added to the estimated effort of the ensemble to get the final estimated effort as in Eq. (4).

\[
\text{PM}_{\text{Final}} = \text{PM}_{\text{Ensemble}} + \text{Bias}_{\text{Estimated}}
\]

The steps of ENNA effort estimation can be seen in Fig. 4. In step 1, new project is given to the ensemble to get the estimated value, namely \( \text{PM}_{\text{Ensemble}} \). In step 2, projects that are similar to the new project are retrieved from the effort database and given to the ensemble to get estimated values, \( \text{PM}_{\text{Ensemble}}(i) \). Actual efforts for the projects, \( \text{PM}_{\text{Actual}}(i) \), are already known, since they are the past projects. Therefore, Eq. (3) gives the estimated bias for the new project, \( \text{Bias}_{\text{Estimated}} \). Finally, \( \text{Bias}_{\text{Estimated}} \) is added to \( \text{PM}_{\text{Ensemble}} \) to provide \( \text{PM}_{\text{Final}} \) as in Eq. (4).

### 4. Experimental results

#### 4.1. Datasets

In this research, five datasets are used for evaluating the performance of the proposed model. Three of them is COCOMO II based, and the last one is function points based.

COCOMO based datasets are NASA dataset, NASA 93 dataset and USC dataset [29]. In these datasets, each project is described with 15 effort multipliers, lines of code value and actual effort value. Effort multipliers can be grouped into four categories as product attributes, hardware attributes, personnel attributes and project attributes. Actual effort is the goal value and is measured in person-months. The NASA dataset includes 60 NASA projects from the 1980’s and 1990’s. These projects are sequencing, avionics and mission planning projects. The NASA 93 dataset consists of 93 NASA projects. These projects are from 1970’s and 1980’s. The USC dataset contains 63 projects that were analyzed by Barry Boehm to introduce COCOMO [29].

Softlab Data Repository (SDR) is collected at Bogazici University Software Engineering Research Laboratory (SoftLab) from software development organizations in Turkey [30]. SDR is a COCOMO II based dataset and it consists of 24 projects. For each project in this dataset, there are 17 cost drivers, 5 scale drivers, lines of code value and actual effort value. Cost drivers can be grouped into four categories as product factors, platform factors, personnel factors and project factors. Scaling factors define an organizational behavior such as process maturity and team cohesion. Actual effort is the goal value and is measured in person-months. SDR dataset consists of projects that were implemented in 2000’s. Therefore, the effects of new methodologies in software engineering are more visible in this data set.

The Desharnais dataset consists of 77 projects from the late 1980’s [29, 31]. These projects are commercial projects that were implemented in a Canadian software house. Each project is defined with team experience, manager experience, year that the project...
ended, basic logical transactions in the system, number of entities
in the data model, non adjusted function points, adjustment factor,
adjusted function points, programming language and actual effort
value. The actual effort is the goal value and is measured in person-
hours.

4.2. Evaluation criteria

Evaluating the effort estimation accuracy of a model is based on
comparing the actual effort with the estimated effort. Therefore,
we used the criteria that have become standards for model evalu-
ation. These criteria are MMRE, MdMRE and PRED(L) [32].

MMRE is the mean magnitude of relative error and represents
the average of the absolute values of relative errors. The equation
of MMRE is in Eq. (5) below:

\[
\text{MMRE} = \frac{1}{N} \sum_{i=1}^{N} \frac{|PM_{\text{Final}} - PM_{\text{Actual}}|}{PM_{\text{Actual}}}
\]  

(5)

where \( N \) is the total number of estimations, \( PM_{\text{Final}} \) is the estimated
effort value, and \( PM_{\text{Actual}} \) is the actual effort value.

MMRE is very sensitive to large deviations in the estimation.
Therefore, the median of the absolute values of relative errors
(MdMRE) is also used.

PRED(L) reports the percentage of the estimates that fall within
\( \pm L\% \) of the actual value. The equation for PRED(L) is in Eq. (6) as
below:

\[
PRED(L) = \frac{k}{N}
\]

(6)

where \( k \) is the number of estimations that fall within \( \pm L\% \) of the ac-
tual value and \( N \) is the total number of estimations.

```
F = Ø  // Start with no features
S = Ø  // Start with no stale features
min_MMRE = MAX
max_stale = 5
stale = 0
for i=1:number_of_features
    for j=1:number_of_features // Initialize MMRE matrix at each step
        MMRE(i,1) = MAX
        MMRE(i,2) = 0
    for all fk ∉ F // Calculate MMRE for all possible cases
        MMRE(k, 1) = train and test ENNA(F ∪ fk)
    sort(MMRE, 1)  // Sort calculated MMRE values
    if(MMRE(1,1) < min_MMRE)  // If accuracy improvement is obtained
        min_MMRE = MMRE(1,1)
        F = F ∪ fk
    else  // If no accuracy improvement is not obtained
        if(stale < max_stale)  // Stale addition
            stale = stale + 1
            F = F ∪ fk
            S = S ∪ fk
        else  // Remove stale features and stop
            F = F – S
            Break
```

Fig. 5. Wrapper algorithm (adopted from [37]).
It is suggested that an acceptable value of \( L \) is 25 or less [32]. Therefore, PRED(25) is used in these experiments.

4.3. Validation method

For evaluating the performance of the models, the holdout method with random sub-sampling is used [33]. A number of test instances are chosen randomly from the original set and the remaining instances form the training set. A model is trained and tested with mutually exclusive training set and test set respectively. This process is repeated 25 times and the best case, the worst case, the average and the standard deviation of the evaluation criteria are reported. Observing extreme points provide additional information about the limits of a model.

In order to check for statistical significance of the results, \( t \)-tests with 95\% confidence interval are conducted.

4.4. Experiment analysis

In this research, multiple neural networks are used to form an ensemble of neural networks (ENN). For this purpose, each MLP is trained on different training sets obtained by bootstrapping method and their estimations are filtered by ART algorithm. Thereafter, associative memory is combined with ENN to form an ensemble of neural networks with associative memory (ENNA). \( k \) nearest neighbor algorithm is used to implement the associative memory concept. The basic building block of the proposed model is a single neural network (NN). Therefore, NN is also evaluated so that the reader may observe whether the proposed ideas provide an accuracy improvement.

Comparing the proposed models, namely ENN and ENNA, with only NN does not provide enough objectivity. Therefore, regression tree (RT), which is a popular [12] and efficient [34] machine learning method, is also evaluated.

In these experiments, we keep the training set as large as possible for a better training. For a dataset of size \( N \), about \( N/6 \) of the projects are used for testing whereas the remaining projects are used for training. \( k \) nearest neighbor algorithm seeks for the projects that are the most similar ones to the new project. However, effort estimation datasets are small by nature and consequently there exist a few similar projects. Therefore, \( k \) is kept small because large \( k \) may decrease the accuracy. \( k \) nearest neighbor algorithm is run with \( k = N/10 \). These parameters are listed in Table 1.

Five datasets are used for evaluating the performances of RT, NN, ENN and ENNA. Evaluation results are listed in Table 2.

For all datasets, NN has the worst estimation accuracy. On the other hand, ENN achieves a considerable improvement in accuracy. This shows that an ensemble of inaccurate models can make accurate estimations. Furthermore, ENNA outperforms ENN, which shows the positive effect of associative memory. Both ENN and ENNA are better than RT that is one of the most accurate machine learning methods [34]. Additionally, the worst case performances of ENN and ENNA are always better than the best case performance of NN. In addition, the standard deviation of ENN and ENNA are small, which implies that they are stable models.

\( t \)-Test results support the evaluation results. For the majority of cross comparisons between these models, ENN outperforms RT and NN. Furthermore, ENNA is better than RT, NN and ENN.

The proposed model provided similar results for all datasets. This fact shows that the proposed models can be applied to a wide range of software projects from different countries.

4.5. Feature subset selection (wrapper)

Wrapper is a supervised feature subset selection algorithm [38] and it is previously discovered that it may improve the accuracy of other cost estimation techniques [37,39]. Wrapper starts with no feature and adds them one by one until no further accuracy improvement is obtained. At each step, the target model is trained and tested by using feature subset and the feature whose addition results in the smallest error is selected. If the addition of new feature with smallest error does not provide any accuracy improvement, it is still added and named as a slack feature. If the number of stale features reaches a predetermined number, the algorithm stops and the slack features are removed from the feature set. In this research, maximum slack number is decided to be five as in the previous effort estimation studies with wrapper [37,39]. The wrapper algorithm is shown in Fig. 5. We use ENNA in the wrapper experiments in order to see any improvements in its performance. Performance measure is selected to be MMRE. The effect of feature subset selection by using Wrapper is listed in Table 3. The performances of the models either stay the same or increase when the dimensionality of the datasets are reduced with the Wrapper algorithm. The results in Table 3 are from \( t \)-test significance tests with \( \alpha = 0.05 \). Prior research in software cost estimation studied the effect of feature subset selection with wrapper for parametric cost models and concluded that it can increase the performance of the cost models [37,39]. Our results show that it is also applicable to, at least, our proposed ENNA model that is a neural network variant. The selected features vary in each dataset; however using 3–4 features is sufficient to produce accurate results.

4.6. Threats to validity

In this research, five datasets are used for evaluating the accuracy of models. Two of them consist of projects that are implemented in NASA software development centers. Although NASA datasets are considered to be relevant to the general software engineering industry [35], we used three other datasets to ensure external validity. These datasets are the USC, SDR and the Desharnais datasets. The projects in these datasets were implemented via methods that are organizationally and culturally different from NASA datasets. Attitudes toward hierarchy and sense of time are

<table>
<thead>
<tr>
<th>Dataset</th>
<th>MMRE (%)</th>
<th>MdMRE (%)</th>
<th>PRED(25) (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NASA</td>
<td>SIMILAR</td>
<td>SIMILAR</td>
<td>SIMILAR</td>
</tr>
<tr>
<td>RT</td>
<td>SIMILAR</td>
<td>SIMILAR</td>
<td>SIMILAR</td>
</tr>
<tr>
<td>NN</td>
<td>SIMILAR</td>
<td>BETTER</td>
<td>BETTER</td>
</tr>
<tr>
<td>ENN</td>
<td>BETTER</td>
<td>BETTER</td>
<td>BETTER</td>
</tr>
<tr>
<td>ENNA</td>
<td>BETTER</td>
<td>BETTER</td>
<td>BETTER</td>
</tr>
<tr>
<td>NASA 93</td>
<td>SIMILAR</td>
<td>SIMILAR</td>
<td>SIMILAR</td>
</tr>
<tr>
<td>RT</td>
<td>SIMILAR</td>
<td>SIMILAR</td>
<td>SIMILAR</td>
</tr>
<tr>
<td>NN</td>
<td>SIMILAR</td>
<td>BETTER</td>
<td>BETTER</td>
</tr>
<tr>
<td>ENN</td>
<td>SIMILAR</td>
<td>SIMILAR</td>
<td>SIMILAR</td>
</tr>
<tr>
<td>ENNA</td>
<td>SIMILAR</td>
<td>SIMILAR</td>
<td>SIMILAR</td>
</tr>
<tr>
<td>USC</td>
<td>SIMILAR</td>
<td>SIMILAR</td>
<td>SIMILAR</td>
</tr>
<tr>
<td>RT</td>
<td>SIMILAR</td>
<td>BETTER</td>
<td>BETTER</td>
</tr>
<tr>
<td>NN</td>
<td>SIMILAR</td>
<td>BETTER</td>
<td>BETTER</td>
</tr>
<tr>
<td>ENN</td>
<td>BETTER</td>
<td>SIMILAR</td>
<td>SIMILAR</td>
</tr>
<tr>
<td>ENNA</td>
<td>SIMILAR</td>
<td>BETTER</td>
<td>BETTER</td>
</tr>
<tr>
<td>SDR</td>
<td>SIMILAR</td>
<td>SIMILAR</td>
<td>SIMILAR</td>
</tr>
<tr>
<td>RT</td>
<td>SIMILAR</td>
<td>SIMILAR</td>
<td>SIMILAR</td>
</tr>
<tr>
<td>NN</td>
<td>SIMILAR</td>
<td>BETTER</td>
<td>BETTER</td>
</tr>
<tr>
<td>ENN</td>
<td>BETTER</td>
<td>SIMILAR</td>
<td>SIMILAR</td>
</tr>
<tr>
<td>ENNA</td>
<td>BETTER</td>
<td>BETTER</td>
<td>BETTER</td>
</tr>
<tr>
<td>DESHARNAIS</td>
<td>SIMILAR</td>
<td>SIMILAR</td>
<td>SIMILAR</td>
</tr>
<tr>
<td>RT</td>
<td>SIMILAR</td>
<td>SIMILAR</td>
<td>SIMILAR</td>
</tr>
<tr>
<td>NN</td>
<td>SIMILAR</td>
<td>SIMILAR</td>
<td>SIMILAR</td>
</tr>
<tr>
<td>ENN</td>
<td>SIMILAR</td>
<td>SIMILAR</td>
<td>SIMILAR</td>
</tr>
<tr>
<td>ENNA</td>
<td>SIMILAR</td>
<td>BETTER</td>
<td>BETTER</td>
</tr>
</tbody>
</table>
the most important cultural differences that have direct impact on the software development effort [36].

5. Conclusion and future work

Accurate estimation of software effort is essential in the software development industry. However, it is hard due to the scarcity of project data. Parametric models such as Cocomo and FPA have been used in the industry. Since these models needed local calibration effort, learning based models such as machine learning models have become popular.

In this research, a new machine learning method for software effort estimation is proposed and evaluated. In this model an ensemble is used rather than a single MLP. When a new project is given to the model for estimation, each MLP provides its own result. These results are divided into clusters and the average of the largest cluster is returned. Additionally, variance is decreased because the result of the ensemble is the average of the results in the largest cluster. In addition to the use of ensemble approach in this model, associative memory is used for bias correction. The bias of the ensemble for similar projects is calculated and added to the result of the ensemble to provide the final result.

The proposed models, ENN and ENNA are shown to be accurate effort estimators. Furthermore, their accuracies stay in a narrow region as observed in the best case and the worst case values. This fact shows that the proposed models are stable and reliable. We have also checked the external validity to examine whether the proposed models can be used for a wide range of software projects from different countries.

We have focused on a practical problem that is the difficulty of learning effort estimators from limited amount of data. We aimed at increasing the performance of machine learning models that allow automatic calibration that has to be carried out manually in parametric models. We have empirically shown that our proposed models are both more accurate and stable than standard NN. The practical impact of ENNA is to allow project managers to make both accurate and stable effort estimations, which leads to better project planning and efficient allocation of scarce resources. We have also show that feature subset selection can improve the performance of ENNA that should be preferred when there is a limited amount of data.

In future research, we will focus on applying ENNA on other domains that require solutions to regression problems in order to check the external validity in other problem domains.

Acknowledgements

This research is supported in part by Tubitak EEEAG under Grant Number 108E014. The authors also extend their gratitude to Ms. Neslihan Demirkol who professionally edited the manuscript.

References

[27] Software Research Laboratory, Bogazici University, SDR Dataset for Cost Estimation, <http://softlab.boun.edu.tr/?q=resources>.