ENNA: Software Effort Estimation Using Ensemble of Neural Networks with Associative Memory

Yigit Kultur
Bogazici University
Computer Engineering
34342, Bebek, Istanbul, Turkey
+90 542 660 19 56
yigit.kultur@boun.edu.tr

Burak Turhan
Bogazici University
Computer Engineering
34342, Bebek, Istanbul, Turkey
+90 212 359 72 27
turhanb@boun.edu.tr

Ayse Basar Bener
Bogazici University
Computer Engineering
34342 Bebek, Istanbul, Turkey
+90 212 359 72 26
bener@boun.edu.tr

ABSTRACT

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 research we used a relatively complex machine learning algorithm, neural networks, and showed that stable and accurate estimations are achievable with an ensemble using associative memory. Our experimental results revealed that our proposed algorithm (ENNA) achieves on the average PRED(25) = 36.4 which is a significant increase compared to Neural Network (NN) PRED(25) = 8.

Categories and Subject Descriptors

D.2.9 [**Software Engineering**]: Management – *cost estimation*

General Terms

Management, Economics, Theory.

Keywords

Effort Estimation, Cost Estimation, Neural Network, Multilayer Perceptron, Ensemble, Associative Memory, Bootstrap, Adaptive Resonance Theory, K nearest neighbors.

1. INTRODUCTION

Effort estimation in software project management has become an important task for companies. Over and under estimating the effort cause either waste of resources or they result in compromising the product quality. The critical success factors in a software development project are the budget, schedule and defect rate. Accurate effort estimation helps managers to optimize all three pillars.

Companies usually have 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 as the problem and estimation methods become more complex, it becomes harder 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, 2, 3, 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 they 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, 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 4 times better than standard NN. ENNA also achieves the minimum relative error rates compared to 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 way of efficient allocation of project resources with minimum manual labor.

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, ensemble of neural networks with associative memory (ENNA). In Section 4, datasets, evaluation criteria, validation methods are explained. Additionally, experimental results 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, 6, 7, 8, 9, 10, 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 to that, parametric models can only be calibrated manually. Machine learning methods are good alternatives to address these drawbacks. So far, many researchers have used machine learning methods in effort estimation [1, 2, 3, 4, 14, 15, 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 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. 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 [1, 2].

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 been used popularly for software effort estimation [1, 2, 3, 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 the answer of 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, ensemble of learners has been used in defect prediction [20].

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 past projects for further accuracy improvement? To find an answer, we used 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 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 [21]. 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, 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 causes large difference in the trained MLP. In other words, learning algorithm has high variance, which affects the learning performance negatively [19, 22]. Another problem is overtraining. 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 complexity and leads to poor generalization. Using a validation set solves the problem of overtraining. When the error on the validation set starts to increase beyond a certain point, the training is stopped [22]. However, this method is not applicable for 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 problems which

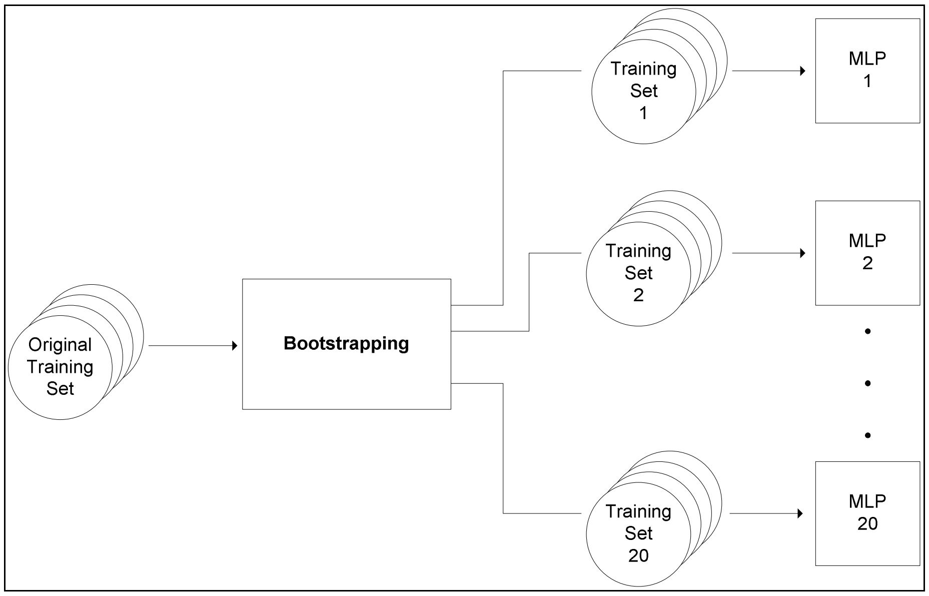

Figure 1. ENN training

are stated above. The main idea of the ensemble is training each MLP with a specific training set. Each training set is generated by randomly choosing training projects from the original training set, which 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 projects in each training set is equal to the number of 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 [22].

In this research, each MLP has one hidden layer with 8 hidden nodes and it is trained with Levenberg-Marquardt algorithm [23, 24]. Levenberg-Marquardt algorithm is a type of backpropagation algorithm and is very efficient in training moderate sized neural networks, which have up to several hundred weights [25]. The hidden nodes and the output node apply 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 Figure 1. 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 Equation 1:

$$\mathbf{PM} = \begin{bmatrix} \mathrm{PM}_1 \\ \mathrm{PM}_2 \\ . \\ . \\ \mathrm{PM}_{20} \end{bmatrix} \tag{1}$$

where PM_i is the person-month value estimated by ith MLP.

The estimations made by some MLP's may deviate much from the largest group of nearby estimations, because some MLP's would have fallen into 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. For this purpose, 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

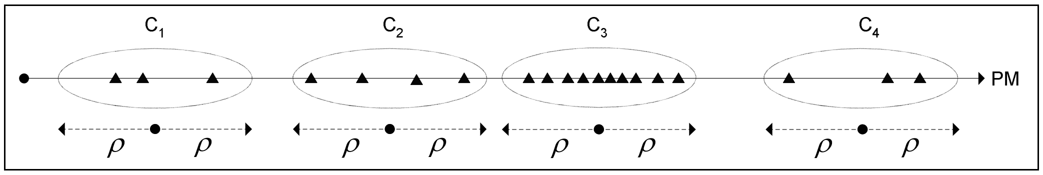

Figure 2. ART algorithm

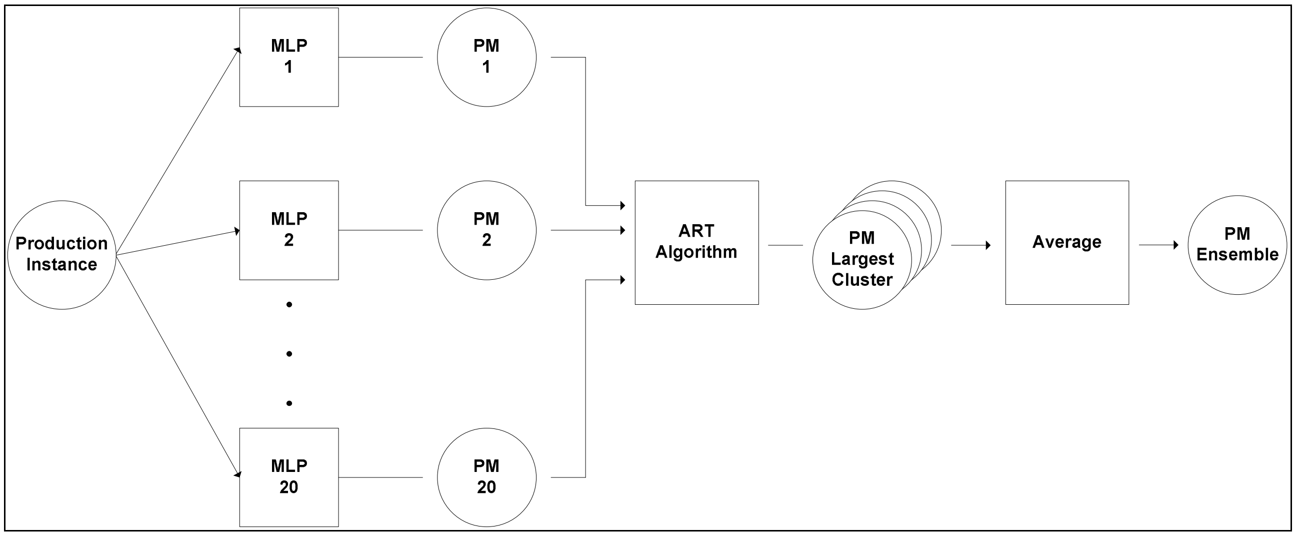

Figure 3. ENN simulation

(ART) algorithm, which follows such an incremental approach, is used in this research [26]. To enhance clarity, ART algorithm is simulated in Figure 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, average of the results in C3 is taken to provide the result of the ensemble. ρ, 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 Equation 2 to provide the effort estimated by the ensemble.

$$PM_{Ensemble} = \frac{\sum PM_i}{N_C} \tag{2}$$

where C is the largest cluster, N_C is the number of results in C

and $\forall\, PM_i \in C$.

The overall process of ensemble effort estimation is shown in Figure 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 Equation 2 to provide the estimated value by the ensemble, namely $PM_{Ensemble}$.

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 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, which stores 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 be reliably recalled [27]. In 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 Equation 2. In order to correct 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, 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 neighbours algorithm [22]. These nearest neighbors are given to the ensemble to get estimated effort values as in Equation 2. The difference between the actual and the estimated values gives the bias of the ensemble for those projects. The average of biases for nearest neighbors is taken to get the estimated bias for the new project as in Equation 3.

$$Bias_{Estimated} = \frac{1}{k} \sum_{i \in N_k} PM_{Actual}(i) - PM_{Ensemble}(i) \tag{3}$$

where N_k is the collection of k nearest neighbors, $PM_{Actual}(i)$ is the actual person-month value for project i and $PM_{Ensemble}(i)$ is the estimated person-month value for project i. Lastly, estimated bias value is added to the estimated effort of the ensemble to get the final estimated effort as in Equation 4.

$$PM_{Final} = PM_{Ensemble} + Bias_{Estimated} \tag{4}$$

The steps of ENNA effort estimation can be seen in Figure 4. In step 1, new project is given to the ensemble to get the estimated value, namely $PM_{Ensemble}$. 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,

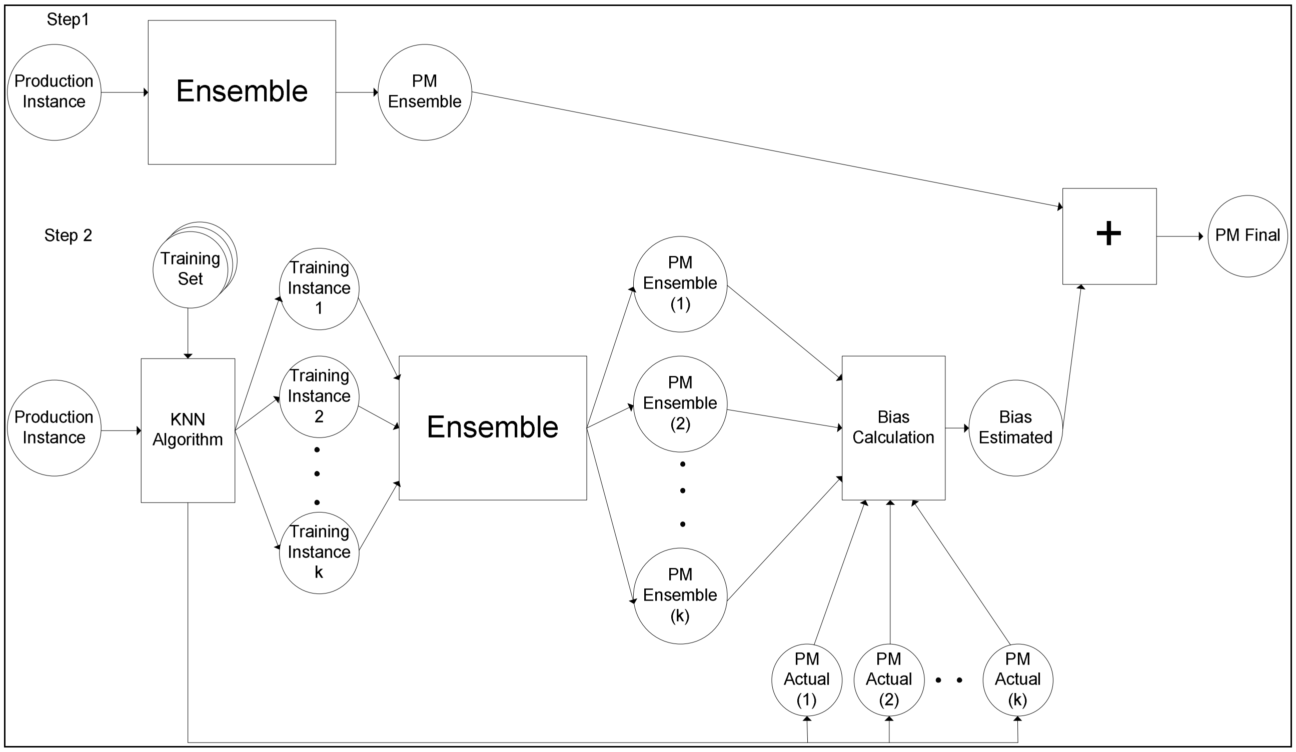

Figure 4. ENNA simulation

$PM_{Ensemble}(i)$. Actual efforts for the projects, $PM_{Actual}(i)$, are already known since they are past projects. Therefore, Equation 3 gives the estimated bias for the new project, $Bias_{Estimated}$. Finally, $Bias_{Estimated}$ is added to $PM_{Ensemble}$ to provide PM_{Final} as in Equation 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 are COCOMO based, one of them is COCOMO II based and the remaining one is Function Points based.

COCOMO based datasets are NASA dataset, NASA 93 dataset and USC dataset [28]. 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. NASA dataset includes 60 NASA projects from the 1980's and 1990's. These projects are sequencing, avionics and mission planning projects. NASA 93 dataset consists of 93 NASA projects. These projects are from 1970's and 1980's. USC dataset contains 63 projects, which were analyzed by Barry Boehm to introduce COCOMO [28].

Softlab Data Repository (SDR) is collected at Bogazici University Software Engineering Research Laboratory (SoftLab) from software development organizations in Turkey [29]. 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 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, which were implemented in 2000's. Therefore, effects of new methodologies in software engineering are more visible in this data set.

Desharnais dataset consists of 77 projects from the late 1980's [28, 30]. 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. 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 such criteria that have become standards for model evaluation. These criteria are MMRE, MdMRE and PRED(L) [31].

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 Equation 5 below:

$$\mathrm{MMRE} = \frac{1}{N}\sum^{N}\frac{\left|\mathrm{PM}_{\mathrm{Final}} - \mathrm{PM}_{\mathrm{Actual}}\right|}{\mathrm{PM}_{\mathrm{Actual}}} \qquad (5)$$

where N is the total number of estimations $\mathrm{PM}_{\mathrm{Final}}$ is the estimated effort value and $\mathrm{PM}_{\mathrm{Actual}}$ 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 ±L% of the actual value. The equation for PRED(L) is in Equation 6 as below:

$$\mathrm{PRED(L)} = \frac{k}{N} \qquad (6)$$

where k is the number of estimations that fall within ±L% of the actual value and N is the total number of estimations.

It is suggested that an acceptable value of L is 25 or less [31]. Therefore, PRED(25) is used in these experiments.

4.3 Validation Method

For evaluating the performance of models, holdout method with random sub sampling is used [32]. 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, average and standard deviation of the evaluation criteria are reported. Observing extreme points provides additional information about the limits of a model.

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

4.4 Experiment Analysis

In this research, multiple neural networks are used to form 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 ensemble of neural networks with associative memory (ENNA). K nearest neighbors algorithm is used to implement associative memory concept. The basic building block of the proposed model is single neural network (NN). Therefore, NN is also evaluated so that the reader may observe whether the proposed ideas provide 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 [33] machine learning method, is also evaluated.

In these experiments, we keep the training set as large as possible for 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 neighbours algorithm seeks for the projects that are most similar 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 neighbors algorithm is run with k = N/10. These parameters are listed in Table 1.

Table 1. Experiment parameters

Dataset	Training	Test	Total	k
NASA	50	10	60	6
NASA 93	78	15	93	9
USC	53	10	63	6
SDR	20	4	24	2
Desharnais	64	13	77	8

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, which is one of the most accurate machine learning methods [33]. Additionally, the worst case performances of ENN and ENNA are always better than the best case performance of NN. In addition to these, standard deviation of ENN and ENNA are small and this fact 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.

4.5 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 [34], we used three other datasets to ensure external validity. These datasets are USC, SDR and 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 the most important cultural differences, which have direct impact on the software development effort [35].

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.

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.

Table 2. Evaluation results

		MMRE (%)			MdMRE (%)			PRED(25) (%)		
		Best	Worst	Average (Std Dev)	Best	Worst	Average (Std Dev)	Best	Worst	Average (Std Dev)
NASA	RT	48.03	82.47	66.02 (11.64)	37.76	64.34	46.14 (7.99)	30.00	10.00	24.00 (8.43)
	NN	122.14	385.29	184.46 (82.22)	89.63	118.75	97.97 (8.00)	20.00	0.00	5.00 (8.50)
	ENN	49.23	66.13	55.71 (5.11)	32.50	47.39	41.82 (4.40)	40.00	30.00	32.00 (4.22)
	ENNA	36.04	55.66	47.66 (6.62)	26.07	41.10	33.17 (4.34)	50.00	30.00	38.00 (6.33)
NASA 93	RT	334.97	532.68	394.34 (65.11)	51.71	67.10	58.94 (5.83)	26.67	6.67	20.67 (6.63)
	NN	232.13	1,925.40	527.69 (502.77)	94.19	197.69	107.76 (31.86)	13.33	0.00	6.00 (4.92)
	ENN	50.49	72.76	62.28 (7.56)	39.10	59.64	49.54 (8.31)	40.00	26.67	30.67 (4.66)
	ENNA	46.43	64.86	54.35 (7.03)	27.05	41.44	36.91 (4.19)	40.00	26.67	32.67 (3.78)
USC	RT	176.02	918.37	323.23 (217.49)	76.02	311.88	135.72 (75.20)	10.00	0.00	1.00 (3.16)
	NN	1,934.80	14,829.00	5584.90 (4848.40)	96.71	2,717.20	657.39 (914.76)	0.00	0.00	0.00 (0.00)
	ENN	73.42	127.16	105.52 (20.46)	54.10	78.71	67.83 (8.80)	20.00	10.00	18.00 (4.22)
	ENNA	56.99	129.06	85.44 (25.83)	49.25	71.61	61.42 (7.12)	40.00	20.00	24.00 (6.99)
SDR	RT	120.07	882.45	434.77 (241.59)	56.61	796.51	205.64 (266.72)	25.00	0.00	10.00 (12.91)
	NN	315.17	12,774.00	2274.10 (3791.30)	87.67	17,000.00	1892.00 (5309.60)	25.00	0.00	15.00 (12.91)
	ENN	36.25	83.40	49.85 (13.47)	30.41	49.82	38.60 (6.27)	50.00	25.00	47.50 (7.91)
	ENNA	22.23	53.45	39.88 (10.95)	10.90	44.42	29.10 (11.92)	75.00	25.00	55.00 (15.81)
DESHARNAIS	RT	68.07	251.80	119.36 (58.74)	39.74	62.73	47.67 (7.64)	30.77	15.39	23.85 (5.68)
	NN	83.28	291.93	154.20 (69.00)	66.01	140.20	84.91 (23.57)	23.08	0.00	14.62 (8.47)
	ENN	52.52	64.36	57.85 (3.42)	47.40	53.64	50.06 (2.21)	30.77	15.39	23.08 (5.13)
	ENNA	42.05	55.76	49.82 (5.05)	35.59	49.03	44.13 (4.57)	46.15	23.08	33.08 (7.30)

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 from 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, which 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, which have 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. Since we focused on learning oriented models, parametric models are out of the scope of this research. Therefore, we did not compare the proposed approach with parametric models such as COCOMO and COCOMO II. In a previous research, we have already proved that learning oriented models make similar or better estimations than parametric models do [33]. 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 do not claim that the proposed model is unrivalled. However, our empirical results show that it may guide the practitioners to estimate project effort as a secondary tool for decision making.

Understandability is an important criterion to judge a model, because the user of the model may want to focus on the facts that affect the result. However, a NN stores the experience of past projects internally and it is impossible to induce rules from a NN. Going forward RT's instead of NN's may be used as Ensemble of Regression Tress with Associative Memory (ERTA) to address the issue of understandability.

In this research, we used many datasets each consisting of less than 100 projects. Another future direction would be to analyze the sensitivity of our model to the size of the dataset.

6. ACKNOWLEDGEMENTS

This research is supported in part by Bogazici University Research Fund under grant number BAP-06HA104.

7. REFERENCES

[1] Srinivasan, K. and Fisher, D., 1995. Machine Learning Approaches to Estimating Software Development Effort, IEEE Transactions on Software Engineering, Vol. 21, No. 2 (February 1995), Pages: 126-137

[2] Younghee, K. and Keumsuk L., 2005. A Comparison of Techniques for Software Development Effort Estimating, System Integration

[3] Venkatachalam, A. R., 1993. Software Cost Estimation Using Artificial Neural Networks, Proceedings of 1993 International Joint Conference on Neural Networks, Vol. 1, (October 1993), Pages: 987-990

[4] Finnie, G. R. and Wittig, G. E., 1996. AI Tools for Software Development Effort Estimation, Proceedings of International Conference on Software Engineering: Education and Practice (SE:EP '96), Pages: 346

[5] Ahn, Y., Suh, J., Kim, S. and Kim, H., 2003. The Software Maintenance Project Effort Estimation Model Based on Function Points, Journal of Software Maintenance and Evolution: Research and Practice, Vol. 15, Issue 2 (April 2003), Pages: 71-85

[6] Adrangi, B. and Harrison, W., 1987. Effort Estimation in a System Development Project, Journal of Systems Management, Vol. 36, Issue 8, Pages: 21-23

[7] Banker, R. D., Chang, H. and Kemerer, C. F., 1994. Evidence on Economies of Scale in Software Development, Information and Software Technology, Vol. 36, Issue 5, Pages: 275-282.

[8] Benediktsson, O., Dalcher, D. and Reed, K., 2003. COCOMO-Based Effort Estimation for Iterative and Incremental Software Development, Software Quality Journal, Vol. 11, Issue 4, Pages: 265-281

[9] Harrison, W. and Adrangi, B., 1987. The Role of Programming Language in Estimating Software Development Costs, Journal of Management Information Systems, Vol. 3, Issue 3, Pages: 101-110

[10] Bielak, J., Improving Size Estimates Using Historical Data, 2000. IEEE Software, Vol. 17, Issue 6 (November-December 2000), Pages: 27-35

[11] Kaplan, H. T., 1991. The Ada COCOMO Cost Estimating Model and VASTT Development Estimates vs. Actuals, Vitro Technical Journal, Vol. 9, Issue 1, Pages: 48-60

[12] Jorgensen, M., and Shepperd, M., 2007. A Systematic Review of Software Development Cost Estimation Studies, IEEE Transactions on Software Engineering, Vol. 33, No. 1 (January 2007), Pages: 33-53

[13] Kemerer, C. F., 1987. An Empirical Validation of Software Cost Estimation Models, Communications of the ACM, Vol. 30, Issue 5 (May 1987), Pages: 416-429

[14] Shepperd, M. and Schofield, C., 1997. Estimating Software Project Effort Using Analogies, IEEE Transactions on Software Engineering, Vol. 23, No. 12 (November 1997), Pages: 736-743

[15] Burgess, C. J. and Lefley, M., 2001. Can genetic programming improve software effort estimation? A

comparative evaluation", Information and Software Technology 43, Pages: 863-873

[16] Shan, Y., McKay, R. I., Lokan, C. J. and Essam, D. L., 2002. Software Project Effort Estimation Using Genetic Programming, International Conference on Communications, Circuits and Systems and West Sino Expositions, Vol. 2, Pages: 1108-1112

[17] Breiman, L., Friedman J., Olshen, R. A. and Stone C. J., 1984. Classification and regression trees, Wadsworth

[18] Hornik, K., Stinchcombe, M. and White, H., 1989. Multilayer Feedforward Networks are Universal Approximators, Neural Networks, Vol. 2, Issue 5, Pages: 359-366

[19] German, S., Bienenstock, E. and Doursat R., 1992. Neural Networks and the Bias/Variance Dilemma, Neural Computation, Vol. 4, Issue 1 (January 1992), Pages: 1-58

[20] Oral, A. D. and Bener, A., 2007. Defect Prediction for Embedded Software, ISCIS 2007, Ankara, Turkey, November 7-9 2007

[21] Abeles, M., 1991. Corticotronics: Neural circuits of the cerebral cortex, Cambridge University Press, New York

[22] Alpaydın, E., 2004. Introduction to Machine Learning, MIT Press, 2004

[23] Levenberg, K., 1944. A Method for the Solution of Certain Non-Linear Problems in Least Squares, The Quarterly of Applied Mathematics 2, Pages: 164-168

[24] Marquardt, D., 1963. An Algorithm for Least-Squares Estimation of Nonlinear Parameters, SIAM Journal on Applied Mathematics 11, Pages: 431–441.

[25] Hagan, M. T. and Menhaj, M. B., 1994. Training Feedforward Networks with the Marquardt Algorithm, IEEE Transactions on Neural Networks, Vol. 5, Issue 6 (November 1994), Pages: 989-993

[26] Carpenter, G. A. and Grossberg, S., 1988. The ART of Adaptive Pattern Recognition by a Self-Organizing Neural Network, IEEE Computer, Vol. 21, Issue 3 (March 1988), Pages: 77-88

[27] Kohonen, T., 1984. Self-Organization and Associative Memory, Springer Series In Information Sciences, Vol. 8.

[28] Boetticher, G., Menzies, T. and Ostrand T., 2007. PROMISE Repository of empirical software engineering data http://promisedata.org/ repository, West Virginia University, Department of Computer Science

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

[30] Desharnais, J. M., 1989. Analyse statistique de la productivitie des projets informatique a partie de la technique des point des function, University of Montreal, Masters Thesis

[31] Conte, S. D., Dunsmore, H. E. and Shen, V. Y., 1986. Software Engineering Metrics and Models, Benjamin-Cummings Publishing, Menlo Park CA

[32] Kohavi, R., 1995. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection, Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, Pages: 1137-1143.

[33] Baskeles, B., Turhan, B. and Bener A., 2007. Software Effort Estimation Using Machine Learning Methods, ISCIS 2007, Ankara, Turkey, November 7-9 2007

[34] Basili, V. R., McGarry, F. E., Pajerski, R. and Zelkowitz, M. V., 2002. Lessons Learned from 25 years of process improvement: The Rise and Fall of the NASA Software Engineering Laboratory, Proceedings of the 24th International Conference on Software Engineering, Pages: 69-79

[35] Hersleb, J. D. and Moitra, D., 2001. Global software development, IEEE Software, Vol. 18, Issue 2, (March-April 2001), Pages: 16-20