

Software Effort Estimation in Rapidly Changing Computing Environment

Eung S. Jun^a and Jae K. Lee^b

^a^bGraduate School of Management, Korea Advanced Institute of Science and Technology
207-43 Cheongryang, Seoul 130-012, Korea,
Tel.) +82-2-901-7681, Fax: +82-2-901-7680, E-mail: esjun@hanmail.net, jkleee@kgsm.kaist.ac.kr

Abstract

Since the computing environment changes very rapidly, the estimation of software effort is very difficult because it is not easy to collect a sufficient number of relevant cases from the historical data. If we pinpoint the cases, the number of cases becomes too small. However if we adopt too many cases, the relevance declines. So in this paper we attempt to balance the number of cases and relevance. Since many researches on software effort estimation showed that the neural network models perform at least as well as the other approaches, so we selected the neural network model as the basic estimator. We propose a search method that finds the right level of relevant cases for the neural network model. For the selected case set, eliminating the qualitative input factors with the same values can reduce the scale of the neural network model. Since there exists a multitude of combinations of case sets, we need to search for the optimal reduced neural network model and corresponding case set. To find the quasi-optimal model from the hierarchy of reduced neural network models, we adopted the beam search technique and devised the Case-Set Selection Algorithm. This algorithm can be adopted in the case-adaptive software effort estimation systems.

Keywords :

Software Effort Estimation; Reduced Neural Network Model; Case Set Selection Algorithm; Quasi-Optimal Reduced Model; Case-Selective Estimation System; Beam Search; Estimation Procedures;

Introduction

Accurate estimation of software effort is one of the key issues for the effective management of software projects. However, accurate estimation is very difficult, because software development is a labor-intensive task and an intangible creation process. Many researchers developed models for estimating software effort and assessed the factors that affect software development. Most of the software effort estimations are attempted using statistical models, case-based reasoning, and neural networks. A common problem with any of these models is that models are not flexible enough to keep up with the rapidly changing computing environment (Saiedian, Band & Barney, 1992; Vicinanza, Mukhopadhyay & Prietula, 1991).

Among the available estimation models, the neural network models performed at least as well as the other approaches (Hill, O'Conner & Remus, 1994), so in this research we select the neural network model as the estimator. To handle the estimation of the new breed of projects, we can estimate better by using only the relevant cases. But this means that the number of cases will be decreased. Thus we need to balance relevance and case availability.

To classify the case sets and corresponding neural models, we propose to use the qualitative input factors as criteria of case classification. The qualitative input factors that have the same values can be eliminated from the model, resulting in the reduced neural network models. So we need to seek the optimal or quasi-optimal case-selective neural network model among them. For this purpose, we devised the Case-Set Selection Algorithm based on the beam search technique with an adequate stopping rule and beam width. According to the paired t-test, we could prove that the quasi-optimal case-selective neural network model can significantly reduce errors more than the full neural network model.

Software Effort Estimation Models

There are four major approaches in software effort estimations: the statistical model, the knowledge-based model, the case-based reasoning model and the neural network model. Boehm et al. (2000) summarized that the rapid pace of change in software technology challenged all classes of techniques. The primary conclusion is that no single technique is best for all situations, and that a careful comparison of the results of several approaches is most likely to produce realistic estimates.

Statistical software effort estimation models adopt mathematical functions between the causing factors and resulting efforts. The statistical models require estimating the parameters based on the historical data. Among the statistical models, the COCOMO (Boehm, 1984a, 1988b) and function point (Albrecht & Gaffney, 1983) models are most widely known.

A limitation of the models that adopt the KDSI (Thousands of Delivered Source Instruction) as an independent variable is that the number of lines of code cannot be known in advance of the system development. And the number of lines of code is not meaningful because of different computing environments. Due to these reasons, the error rates of statistical models are as high as 50% ~ 100% (Bergeron & St-

Arnaud, 1992; Kermerer, 1987; Mukhopadhyay, Vicinanza & Prietula, 1992; Srinivasan & Fisher, 1995; Vicinanza, Mukhopadhyay & Prietula, 1991).

To overcome the limitation of statistical models, Artificial Intelligence (AI) techniques were widely attempted (Hill, O'Connor & Remus, 1994; Simon, 1986). Prietula et al. (1998) emphasized that qualitative improvements in estimation accuracy could come from the insight of experts. However, the qualitative knowledge-based approach has an inherent limitation in quantitative estimation of software effort. So major AI techniques, which are practically applicable, are case-based reasoning and neural network models.

The CBR model for software effort estimation requires collecting past project cases, and retrieving the case(s) most similar to the target project according to a pre-defined similarity measure. Up to this point, the CBR is a very effective approach. However the modification of past cases identifying their differences from the target, is in most circumstances very difficult. Lee et al. (1998) have developed the process of rule- and constraint-based modification for the real world construction project-planning network. The modification of the past case's result identifying its difference to the target in terms of man-month as a quantitative unit is in most cases very difficult. In the software effort estimation process, the modification process requires a numeric model that can tell the difference of numerically presented effort between the old case and new one. So the CBR approach alone cannot fulfill the goal of software effort estimation.

The neural network with hidden layers allows the nonlinear mapping function between the causing input factors and output results. To handle the many qualitative input factors, the neural network model is more suitable than statistical models, and many studies have shown that the performance of the neural network model is at least as powerful as statistical models (Hill, O'Connor & Remus, 1994). Many researchers have applied the neural network models for software effort estimation expecting that they can outperform statistical models. Shukla (2000) demonstrated substantial improvement in prediction accuracy by using the neuro-genetic approach as compared to both a regression-tree-based conventional approach and neural network approach by back-propagation algorithm.

However, the neural network model as well as others has the problem of *selecting the quasi-optimal case set* because the computing environment that significantly influences the software effort changes so rapidly. If we use all of the past cases, the model becomes too dull for the special characteristics of the new environment. So we need to use the relevant similar cases that can explain the specialty of the target project. However, we may not have a sufficient number of cases that are similar enough to use for estimation. Thus we need to balance similarity and case point availability. Let us call the neural network model that uses all possible

input factors and all available cases, the *Full Neural Network Model* (in short, *Full Model*). To distinguish the neural network model that uses reduced relevant cases and reduced input factors (by eliminating the qualitative factors that have the same values) from the Full Model, let us call it the *Reduced Neural Network Model* (in short, *Reduced Model*). The goal of our research is to find the best relevant case set and corresponding quasi-optimal reduced neural network model for software effort estimation (Jun & Lee, 2001). Although we develop this approach for the software effort estimation in this paper, this approach can be used for other dynamic environments too.

Full Neural Network Model for Software Effort Estimation

Although our goal is finding the optimally reduced neural network model, let us start with the full neural network model first because we need to compare their performances.

Factors for Software Effort Estimation

There are numerous variables that can influence the software efforts required to complete a project (Boehm, 1988; Blackburn & Scudder, 1996; Deephouse, Mukhopadhyay, Goldenson & Kellner, 1996; Finnie, Wittig & Petkov, 1993; Maxwell, Wassenhove & Dutta, 1996; Rasch, Cuccia & Amer, 1995; Rasch & Tosi, 1992; Redmond-Pyle, 1996; Roberts, Gibson, Fields & Rainer, 1998; Subramanian & Zarnich, 1996). Four categories of factors that are used in various models are selected: *project requirement, characteristics of products, staff skill level, and computing platform*. We selected 23 input factors and their values as listed in Table 1, based on surveyed opinions of 30 experts who have experiences of software development and maintenance.

In Table 1, the input factors are identified as I_i , $i = 1, \dots, 23$. The input factors can be categorized into quantitative and qualitative, as denoted QT and QL in the fourth column. The qualitative factors may be used either as input variables or as classifiers of case sets.

Table 1 appears here.

Full Neural Network Model

We designed a full type of neural network model for software effort estimation as shown in Figure 1. The full neural network model has 23 input factors in Table 1, one output in man-months, and one hidden layer with the nodes from 23/2 (we applied integer 12) to $2*23+1$ (which is 47) with an increment of 5. Since there is no concrete theory of determining the number of hidden nodes, we tried between $I/2$ and $2I + 1$ nodes with an increment of 5 (where I is the number of inputs) (Hect-Nielsen, 1990; Venkatachalam,

1993; Bjornson & Barney, 1999). The model is trained by the back-propagation algorithm starting from 5,000 epochs to 30,000 epochs with an increment of 5,000.

We have collected 112 cases mainly from two companies: Korea Electric Power Cooperation and Samsung Electronics. Eighty-two cases are used for training, and 30 cases for testing. Among the 42 trials of Full Models, the optimal performance was found with 17 hidden nodes at 15,000 epochs. To measure the errors, we adopted the popular measurement: *Mean Magnitude of Relative Error (MMRE)* as defined in (1) (Kermerer, 1987).

$$MMRE = \left(\sum_{i=1}^n \left| \frac{E(Y_i) - Y_i}{Y_i} \right| \times 100 \right) / n \quad (1)$$

Y_i implies actual man-months spent for the Project i , and $E(Y_i)$ the estimated effort. n is the number of test cases. The MMREs of 42 trials ranged between 17.2% and 22.8 %, with the minimum value at 17.2%.

Figure 1 appears here.

Case-Selective Reduced Neural Network Models

In designing the case-selective reduced neural network models, we need to define the input nodes and select the relevant case set for learning. In selecting the relevant case set, we need to consider the sensitivity of the sample-size effect.

Reduced Neural Network Models

As defined earlier, the reduced neural network models are the ones that have eliminated the qualitative input factors that have the same values. So in the software effort estimation model, the number of input factors can be reduced from 23 (in the full model) to at most 15 because there are 8 qualitative factors.

Now the question is how to find the optimal combination by eliminating input factors out of $\sum_{s=1}^8 {}_8C_s = 255$ alternative

combinations. Obviously the demerit of the reduced model is the reduced number of case points at the cost of enhanced relevance. So, just more reduction of the input factors does not necessarily mean the performance improvement.

Measure of Similarity and Case Sets Hierarchy

The similarity can be measured as the weighted sum of the distance of the factors in the cases. For the classification of case sets, we use the qualitative factors as the variables of similarity measurement. All weights are assumed to be equal. For instance, the similarity level s means that s qualitative variables have the same values with the new project to be estimated.

Assume that there are two values in each qualitative variable. Then the possible case set in each level of similarity s is ${}_8C_s * 2^s$.

2^s . With 8 qualitative variables, there are $\sum_{s=1}^8 {}_8C_s * 2^s = 6,560$

combinations of value sets. Our concern is how to organize the case sets. Due to the relationship between the case set and its corresponding reduced model, we may just select either the case set or the reduced model, because the other part will be determined automatically.

One way of organizing the case sets is by the hierarchy of similarity level, regarding the level as the depth of the tree as depicted in Figure 2. The notation is formally defined in section 5, but tentatively $D_s(\cdot)$ means the case set of similarity level s , and $v_i, i=1, \dots, 8$ means the qualitative variables that have the same values. The subscripts a and b imply the illustrative values of the variables. The number in the bubble of the case set is the available sample size usable for the training. Note that the sample size diminishes as the level of similarity becomes deeper.

Figure 2 appears here.

Sample Size Effect and Pruning Policy

Hu et al. (1999) claimed that neural network predictions are quite accurate even when the sample size is relatively small. Twomey and Smith (1998) demonstrated that when the sample size is small, the error of a network is more sensitive to the construction of training sample. Statistically, the sample size still affects the precision of the neural network model.

When the sample size is too small, the credibility of the model will become too low. So we have to determine the cutting point by deciding an appropriate lower bound as an acceptable sample size. This process is somewhat judgmental depending upon the available data and acceptable precision level. In this study, we have evaluated the effect of sample size as depicted in Figure 3. Since the error level becomes quite flattened above 7, we selected 7 as the cutting point of these models.

Figure 3 appears here.

Search for the Quasi-Optimal Case-Selective Neural Network Model

Non-monotonicity and Beam Search

To develop a search technique named the *Case-Set Selection Algorithm*, the first concern is its monotonicity and global optimality. According to our test, the MMRE of the best-reduced model in each similarity level is not monotonic to s . This implies that we have to determine a rule for when to stop

after a certain number of iterations of local optima. To reflect this feature, we compare the sensitivity of iterations, and select the most adequate stopping rule.

The second concern is the selection of the search scheme. We seek a scheme that is computationally affordable with acceptable precision. For this purpose, we used the Beam Search technique, and evaluated the sensitivity of beam widths. Beam search is a graphic algorithm with limited beam width for pruning (Sabuncuoglu & Bayiz, 1999).

Notations for Case Set Selection Algorithm

To develop the Case-Set Selection Algorithm based on the beam search technique with the adequate stopping rule and beam width, let us define the notations formally and specify the algorithm using the notation.

Notations

D : Case set (or Data set)

s : Similarity level (which means the number of matching qualitative variables)

D_s : Case set at the similarity level of s

D_0 : Case set at the similarity level of zero implies the original case set

V : Variable set

v_j : Variable j

$V = [v_1, \dots, v_p]$: variable set with p variables

U_j : Value set of variable j

$U_j = \{u_{j1}, u_{j2}, \dots, u_{jq}\}$

U^o : Value set of the target project

$U^o = \{u_{j1}^o, u_{j2}^o, \dots, u_{jq}^o\}$

$D_s(V, U)$: Case sets hierarchy for all variables and values

$D_s(V = U^o)$: Case sets hierarchy that matches the values of the target project

$D_s(v_j = u_j^o)$: A case set which matches a set of s variables with the target project

e.g., $D_2(v_1 = \text{"New development"}, v_3 = \text{"Online"})$

$n[D_s(v_j = u_j^o)]$: Sample size of the case set $D_s(v_j = u_j^o)$

$NN[.]$: Neural Network Model

$NN[D_s(v_j = u_j^o)] = NN_s[v_j = u_j^o]$: Neural network model

with the case set $D_s(v_j = u_j^o)$

e.g., $NN_2[v_1 = \text{"New Development"}, v_3 = \text{"Online"}]$

$NN_0[v_j = u_j^o] = NN_0[\{.\}]$: Full neural network model with

the case set $D_0(v_j = u_j^o)$

$NN[D_s(.)] = NN_s[.]$: Neural Network Models Set with all combinations of similarity levels s against the target project

$MRE(NN_s[v_j = u_j^o]) = MRE_s(v_j = u_j^o)$: Mean Magnitude of the Relative Error of the neural network model with the case

set $D_s(v_j = u_j^o)$

$MRE(\{.\})$: The set of MREs

$OPEN$ set : The current neural network set that can be expanded for search

$LAST$ set : The neural network set that was in the $OPEN$ a step before. $LAST$ does not expand, but needs to be kept for comparison.

$CLOSED$ set : the neural network set that cannot be expanded any more.

$TEMP$: The expanded set from the current $OPEN$ set.

$BEST$: The best neural network model found so far.

$EXPAND[OPEN]$: A command that request to expand the neural models in the $OPEN$ set of similarity level s to those of level $s+1$.

$MIN_w\{MRE(\{Set\})\}$: Select the w neural network models with the minimal MRE values in the Set .

Case Set Selection Algorithm

The Case-Set Selection Algorithm is developed based on the beam search technique. It starts with the Full Model and expands toward the higher level of similarity and keeps the w -best reduced models in each similarity level. The stopping condition is determined by counting the occurrence of local minima. The algorithm can be specified as follows using the notations defined above:

0. Setting

Lower bound of sample size : $LB_n \leftarrow n^o$

Width of Beam Search : $w \leftarrow w^o$

Iteration counter of local minimal point that stops the search : $z \leftarrow z^o$

1. Initialization

$s := 0$

$LAST = \{.\}$

$CLOSED = \{.\}$

$OPEN = \{NN_0[v_j = u_j^o]\}$

SOLVE $NN_0[v_j = u_j^o]$

$Progress \leftarrow \{.\}$; Flag that checks whether the error is increased or not.

$BEST \leftarrow \{MRE_0(v_j = u_j^o)\}$

2. Expansion

$s \leftarrow s+1$

$TEMP \leftarrow EXPAND[OPEN]$

Prune the case set in $s+1$ that does not satisfy $n[D_{s+1}(v_j = u_j^o)] > n^o$

$CLOSED \leftarrow LAST$

$LAST \leftarrow OPEN$

SOLVE $TEMP$

$OPEN \leftarrow MIN_w\{MRE(TEMP)\}$

```
TEMP ← { }
BEST ← MINi{MRE(BEST), MRE(OPEN)}
```

3. Comparison

```
IF MINi{MRE(LAST)} ≥ MINi{MRE(OPEN)}
  Progress ← {"Non-increased Error"}
  GOTO the Expansion routine
ELSE
  Progress ← {"Increased Error"}
  z ← z-1
  GOTO the Check the Stopping Condition routine.
```

4. Check the Stopping Condition

```
IF z = 0
  STOP the iteration
ELSE
  GOTO the Expansion routine.
```

Performance of the Quasi-Optimal Reduced Model

The performance of the quasi-optimally reduced neural network models is evaluated for the 30 test cases. We can see that the average error of the full model 17.2% is reduced to 12.13%. According to the paired t-test between the two groups, the quasi-optimal case-selective reduced model has significantly less error than the full model (t-value = 8.271, p-value = 0.000). In this case, the quasi-optimal model has sacrificed only 0.28% of errors on average from the true optimal model, and has discovered 26 true optimal models out of 30 test cases. The fitness measured by the degree of determination of the quasi-optimal model ("sum of errors after the quasi-optimal model" / "sum of errors with the full model") turns out 0.761. These results verify that the quasi-optimal neural network model can significantly outperform the full model.

Case-Selective Estimation System : SOFFORT

System Architecture and Estimation Procedures

The architecture of a prototype system, SOFFORT, and its procedure for software effort estimation is designed as Figure 4.

Figure 4 appears here.

SOFFORT is composed of three sub-systems; *case base*, *knowledge base*, and *estimation model base*. Case base has the past project cases represented in frames (or objects). Knowledge base has rules for grouping similar cases, post-estimation adjusting factors, and case management. The estimation model base has two types of neural network models: full and reduced NN models.

The procedure of case-selective software effort estimation is

as follows:

Step 1) Define the characteristics of target software project estimation.

There are four types of features in a S/W project: project features, product features, staff features, and technology features. A frame represents the target project.

Step 2) Retrieve the best similar case group.

The similarity measure was used to group similar cases to the target project. It is necessary to find the best similarity level and input factors.

Step 3) Select the architecture of best neural network model.

In the similarity level of D_0 , all cases are used irrespective of the similarity level. However, input factors of the NN model are determined according to the level of similarity from D_1 to D_8 .

Within the same case group, the qualitative input factors with same values to those of the target project are eliminated from the candidate input factors for the NN model because they have the same values in the same case group. Thus, the number of input nodes for the NN model becomes reduced accordingly.

Step 4) Estimate the required software effort for the target project.

It requires a lot of endeavor to learn and test data. Through the enumeration, we find a best NN model architecture along with an appropriate case group which can be used for estimation.

Model Adjustment to Added New Cases

As new projects in the up-to-date computing environment are collected, the neural network models need to be adjusted because the added cases may change the structure of optimal models. First of all, the new cases need to be assigned to the relevant case group. The performance of the new cases can be tested with the currently optimal NN model. If the error is within the acceptable limit, we may continue to use the pair of relevant case group and the currently optimal NN model as the best choice. However if it is not the case, we have to adjust to the change and recover a new optimal NN model with a relevant case group. In this study, we suggest two adjustment strategies: *local search* and *restructuring*. The local search approach is triggered when the errors for the new cases are too big. In this case, we have to find another model and case set for the estimation. The search process for this can be a variation of Case Set Selection Algorithm. We may locally search forward or backward from the current optimal models.

Restructuring is necessary when a significant number of new projects have been accumulated and it has caused the nature of case base fundamentally. In this case, it may be impossible to find the optimal neural network model by local search. So it is necessary to restructure the relevant case group and to

find a new optimal reduced NN model based on the current case base.

Conclusion

To enhance the performance of software effort estimation models under the rapidly changing computing environment, we have attempted to use only the relevant cases and reduce the qualitative input factors of neural networks that have the same values. To find the quasi-optimal case-selective reduced neural network model, we have devised the Case-Set Selection Algorithm based on the beam search technique. According to the test results with 30 cases, the result by the quasi-optimal model significantly outperformed the original full model. By developing a decision support system that can implement this approach, the software effort can be estimated adaptive to the addition of new cases from the up-to-date computing environment. We propose two adjustment strategies: local search and restructuring approaches. The Case-Set Selection Algorithms can be used not only for the software effort estimation, but also for any neural networks models that need dynamic case adaptation.

References

- [1] Bergeron, F., & St-Arnaud, J.Y. 1992. Estimation of Information Systems Development Efforts. *Information and Management*, 22, 239-254.
- [2] Bjornson, C., & Barney, D.K. 1999. Identifying significant model inputs with neural networks Tax court determination of reasonable compensation. *Expert Systems with Applications*, 17, 13-9.
- [3] Blackburn, J.D., & Scudder, G.D. 1996. Improving Speed and Productivity of Software Development: A Global Survey of Software Developers. *IEEE Transactions on Software Engineering*, 22(12), 875-885.
- [4] Boehm, B.W. 1988. Understanding and Controlling Software Costs. *IEEE Transactions on Software Engineering*, 14(10), 1462-1477.
- [5] Boehm, B.W., Abts, C., & Chulani, S. 2000. Software Development Cost Estimation Approaches-A Survey. *Annals of Software Engineering* 10(1), 177-205.
- [6] Deephouse, C., Mukhopadhyay, T., Goldenson, D.R., & Kellner, M.I. 1996. Software Process and Project Performance. *Journal of Management Information Systems*, 12(3), 187-205.
- [7] Finnie, G.R., Wittig, G.E., & Petkov, D.I. 1993. Prioritizing Software Development Productivity Factors Using the Analytic Hierarchy Process. *Systems Software*, 22, 129-139.
- [8] Hect-Nielsen, R. 1990. *Neuro Computing*. Addison-Wesley Publishing Co., Inc.
- [9] Hill, T., O'Connor, M.L., & Remus, M. 1994. Artificial Neural Network Models for Forecasting and Decision Making. *International Journal of Forecasting*, 10, 5-15.
- [10] Hu, M.Y., Zhang, G., Jiang C.X., & Patuwo, B.E. 1999. A Cross-Validation Analysis of Neural Network Out-of-Sample Performance in Exchange Rate Forecasting. *Decision Sciences*, 30(1), 197-216.
- [11] Jun, E.S., & Lee, J.K. 2001. Quasi-Optimal Case-Selective Neural Network Model for Software Effort Estimation. *Expert Systems with Application*, 21(1), 1-14.
- [12] Kermerer, C.F. 1987. An Empirical Validation of Software Cost Estimation Models. *Communication Of the ACM*, 30(5), 416-429.
- [13] Lee, K.J., Kim, H.W., Lee, J.K., & Kim, T.H. 1998. FASTrak-APT: Case and Constraint-Based Construction Project Planning System. *AI Magazine*, 19(1), 13-24.
- [14] Maxwell, K.D., Wassenhove, L.V., & Dutta, S. 1996. Software Development Productivity of European Space, Military, and Industrial Applications. *IEEE Transactions on Software Engineering*, 22(10), 706-718.
- [15] Mukhopadhyay, T., Vicinanza, S.S., & Prietula, M.J. 1992. Examining the Feasibility of a Case-Based Reasoning Model for Software Effort Estimation. *MIS Quarterly*, 4(1), 155-171.
- [16] Prietula, M., Feltovich, P., & Marchak, F. 1989. A Heuristic Framework for Assessing Factors Influencing Knowledge Acquisition. *Proceedings of the Twenty-Second Hawaii International Conference on Systems Sciences*, 419-426.
- [17] Rasch, R.H., Cuccia, A.D., & Amer, T. 1995. The Relationship between Software Project Characteristics, Case Technology and Software Development Productivity. *Journal of Information Technology Management*, 6(1), 1-11.
- [18] Redmond-Pyle, D. 1996. Software Development Methods and Tools: Some Trends and Issues. *Software Engineering Journal*, 99-103.
- [19] Roberts, T.L., Gibson, M.L., Fields, K.T., & Rainer, K. 1998. Factors that Impact Implementing a System Development Methodology. *IEEE Transactions On Software Engineering*, 24(8), 640-649.
- [20] Sabuncuoglu, I., & Bayiz, M. 1999. Job shop scheduling with beam search. *European Journal of Operational Research*, 118, 390-412.
- [21] Saiedian, H., Band, M., & Barney, D. 1992. The Strengths and Limitations of Algorithmic Approaches to Estimating and Managing Software Costs. *International Business Schools Computing Quarterly*, Spring, 21-22.
- [22] Shukla, K.K. 2000. Neuro-genetic prediction of software development effort. *Information and Software Technology*, 42(10), 701-713.

- [23] Simon, A.H. 1986. Whether Software Engineering Needs to Be Artificially Intelligent. IEEE Transactions on Software Engineering , 12(7), 726-732.
- [24] Srinivasan,K., & Fisher,D. 1995. Machine Learning Approaches to Estimation Software Development Effort. IEEE Transactions On Software Engineering, 21(2), 126-137.
- [25] Subramanian, G.H., & Zarnich, G.E. 1996. An Examination of Some Software Development Effort and Productivity Determinants in ICASE Tool Projects. Journal of Management Information Systems, 12(4), 143-160.
- [26] Twomey, J.M., & Smith, A.E. 1998. Bias and Variance of Validation Methods for Function Approximation Neural Networks Under Conditions of Sparse Data. IEEE Transactions on Systems, Man and Cybernatics.
- [27] Vicinanza, S. S., Mukhopadhyay, T., & Prietula, M.J. 1991. Software-Effort Estimation: An Explolatory Study of Expert Performance. Information Systems Research, 2(4), 243-262.

Table 1 - Definition of Input Factors and Case Groups

(QT : Quantitative, QL : Qualitative)

Categories	Input Factors		Values	Type		Qualitative Variables
				QT	QL	
Project Requirement	I_1	Required Training and Education	Average Required Man-Months	*		
	I_2	Size	No. of Modules	*		
	I_3	Development Type	New Development		*	v_1
			Maintenance			
I_4	Urgency	Normal		*	v_2	
		Urgent				
Characteristics of Product	I_5	Processing Type	Batch		*	v_3
			Online			
	I_6	<i>Used Algorithms :</i> -Database Access -I/O Process -Math./Statistic Process -AI based Process	-Relative Fraction Value			
	I_7		Percentage	*		
	I_8		Percentage			
	I_9		Percentage			
	I_{10} I_{11} I_{12}	<i>Application Task Structure</i> -Strategic Planning -Tactical Management -Operational Control	-Relative Fraction Value			
Percentage			*			
Percentage						
Staff Level	I_{13} I_{14}	<i>Beginner :</i> -Number -Experience of Similar Application Development	No. of Persons	*		
			Average Months of Experience			
	I_{15} I_{16}	<i>Junior :</i> -Number -Experience of Similar Application Development	No. of Persons	*		
			Average Months of Experience			
	I_{17} I_{18}	<i>Senior :</i> -Number -Experience of Similar Application Development	No. of Persons	*		
			Average Months of Experience			
Computing Platform	I_{19}	Network Type	Centralized Network		*	v_4
			Distributed Network			
	I_{20}	Reuse of Modules	Module Reuse		*	v_5
			No Module Reuse			
	I_{21}	Use of Formal Methodology	Formal Methodology		*	v_6
			Informal Methodology			
	I_{22}	Language	3GL		*	v_7
			4GL			
I_{23}	Use of CASE Tools	CASE Tool		*	v_8	
		No CASE Tool				

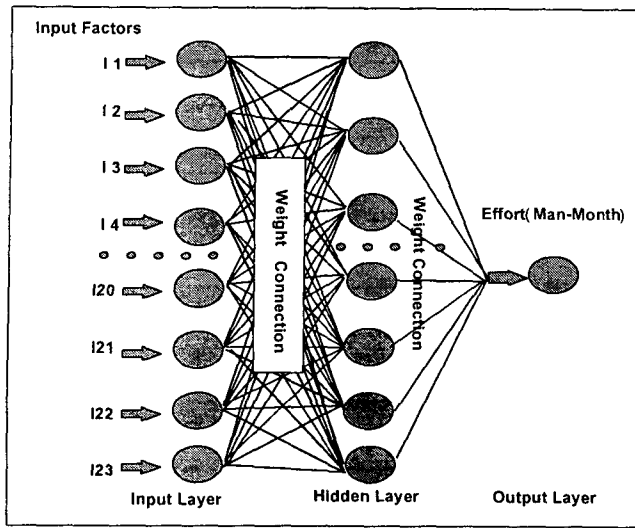


Figure 1 - Full Neural Network Model for Software Effort Estimation:

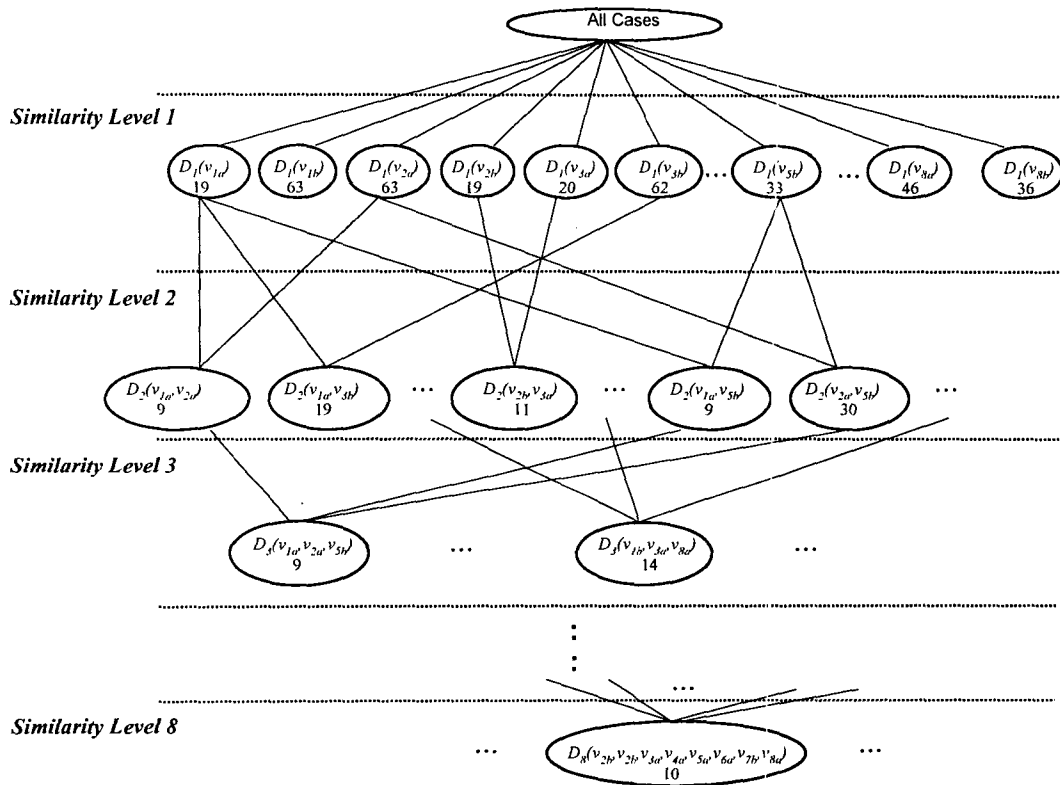


Figure 2 - Relevant Case Set Hierarchy

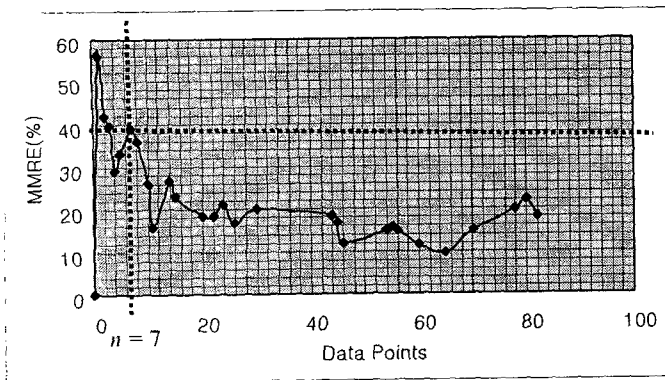


Figure 3 - Effect of Sample Size

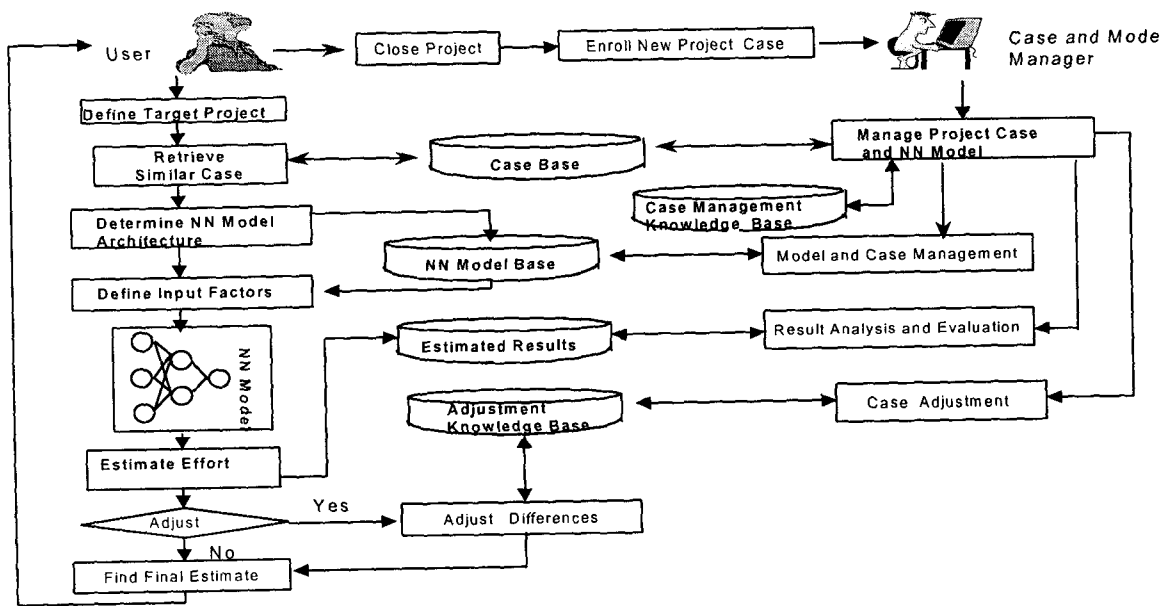


Figure 4 - The Architecture and Procedure of SOFFORT