# Application du réseau de neurones RBFN à l'estimation des coûts de logiciels

Samir Mbarki
BP. 133, Faculté des sciences
Université Ibn Tofail, Kénitra
MAROC
mbarki@univ-ibntofail.ac.ma

Ali Idri
ENSIAS, BP. 713, Agdal
Université Mohamed V-Souissi
MAROC
idri@ensias.ma

Alain Abran
Ecole de technologie
supérieure, Montréal
CANADA
aabran@ele.etsmtl.ca

**RÉSUMÉ.** L'estimation des coûts de développement de logiciels par les réseaux de neurones a fait l'objet de beaucoup de travaux de recherches durant les années 90. Néanmoins, il y a encore une réticence marquée vis-à-vis de leur adoption comme une technique acceptable dans le domaine, principalement à cause de leur qualification de boîte noire. Notre contribution a pour objectif d'appliquer le réseau RBFN (Radial Basis Function Network) à l'estimation des coûts de logiciels. Le travail principal est de définir le choix des différents paramètres de l'architecture du réseau en concordance avec sa précision. La base des projets Cocomo'81 est utilisée pour l'apprentissage et les tests du réseau RBFN.

**ABSTRACT.** Software development effort estimation with the aid of artificial neural networks (ANN) attracted research interest at the beginning of the nineties. However, the lack of a natural interpretation of their estimation process have prevented them from been accepted as common practice in cost estimation. Indeed, they have generally been viewed with skepticism by a majority of the software cost estimation community. In this paper, we investigate the use of the Radial Basis Function Networks (RBFN) in the software cost estimation field. Our main work is to describe how we choose model parameters in agreement with the neural network accuracy. Our experiment is made on COCOMO'81 dataset.

**MOTS-CLÉS :** Modèles d'estimation des coûts, Réseau de neurones RBFN, Génie logiciel.

**KEYWORDS:** Cost estimation models, RBFN neural networks, Software engineering.

## 1. Introduction

L'estimation des coûts de développement de logiciels est le processus qui permet de prédire l'effort de développement requis pour la réalisation d'un projet logiciel. L'objectif des chercheurs et responsables informatiques est de fournir des estimations exactes et précises de l'effort de développement de projets logiciels à une étape prématurée du cycle de vie du logiciel. Pour atteindre cet objectif, plusieurs modèles d'estimation des coûts de logiciels ont été développés. Ces modèles d'estimation se répartissent généralement en deux catégories principales : les modèles paramétriques et les modèles non paramétriques. Les modèles paramétriques [2,3] nécessitent la connaissance de la fonction reliant l'effort de développement et les facteurs du coût. Ces modèles présentent des inconvénients pour le modèle d'estimation :

- nécessite une forme explicite (sous forme mathématique) de la relation entre la variable dépendante (coût à estimer) et les variables indépendantes (facteurs de coût).

- Il doit être calibré dans un nouveau contexte

Les modèles non paramétriques [1,14,15] sont apparus pour pallier à ces inconvénients. Ils sont fondés sur des approches de l'intelligence artificielle telles que les réseaux de neurones, le raisonnement à base des cas et les arbres de décision.

Dans ce travail, nous nous intéressons essentiellement à l'estimation des coûts de logiciels par les réseaux de neurones. Ces derniers se distinguent des autres outils de modélisation par leur aptitude de représenter n'importe quelle fonction (approximateurs universels) et leur capacité d'apprentissage automatique à partir de données historiques. En effet, l'apprentissage permet au modèle de s'adapter aux changements de son environnement. Ceci est plausible en estimation des coûts de logiciel suite à l'évolution croissante de l'industrie des logiciels. La plupart des travaux de recherche appliquant les réseaux de neurones à l'estimation des coûts de logiciels [7,14,15] avaient comme objectif l'évaluation de la précision des estimations de l'approche neuronique en la comparant avec celle des autres techniques. Dans l'ensemble de ces travaux, les auteurs ont utilisé le perceptron multi-couche (MLP), la fonction sigmoïde pour l'activation du réseau et l'algorithme de rétro-propagation pour l'apprentissage du réseau de neurones. Pour valider leurs modèles, les auteurs ont utilisé plusieurs bases de projets logiciels telle que Cocomo'81 [2] et Desharnais [5].

Dans un travail récent [12], nous avons étudié la possibilité de donner une interprétation compréhensible à un modèle d'estimation du coût fondé sur un perceptron à trois couches. Ce réseau de neurones a été transformé en un système à base de règles floues. Dans le présent travail, nous étudions un autre type de réseau de neurones : le réseau RBFN et l'appliquons à l'estimation des coûts de logiciels. Au cours de ce travail nous nous intéressons à l'étude de l'impact du choix des différents paramètres de l'architecture du réseau RBFN sur la précision des estimations.

Cet article est composé de cinq sections. Dans la deuxième section, nous exposons brièvement les réseaux de neurones RBFN. Dans la troisième section, nous discutons de l'application des réseaux RBFN à l'estimation des coûts de développement de logiciels. La quatrième section présente l'analyse des résultats de l'application du réseau de neurones RBFN à la base de données COCOMO'81. La dernière section sera consacrée à la conclusion et aux perspectives.

## 2. Le réseau de neurones RBFN : une vue d'ensemble

Les réseaux de neurones RBFN (Radial Basis Function Network) ont été développés par Moody et Darken [6]. Ils ont prouvé leur succès dans plusieurs domaines puisqu'ils peuvent approcher plusieurs types de fonctions [13]. Le réseau RBFN est un réseau feedforward composé de trois couches : une couche d'entrée, une couche cachée et une couche de sortie. La fonction d'activation de la couche cachée est une fonction radiale. La fonction d'activation la plus communément utilisée est la fonction gaussienne, définie par :

$$y_i(x) = \exp\left(-\left\|x - c_i\right\|^2 / \sigma_i^2\right) \qquad \text{(Equation 1)}$$

où $c_i$ et $\sigma_i$ sont respectivement le centre et le rayon de la fonction gaussienne et $\|\cdot\|$ dénote la distance euclidienne entre le vecteur d'entrée $x$ et le vecteur centre $c_i$. La sortie Z du réseau RBFN est déterminée par :

$$Z = \sum_{i=1}^{h} \beta_i y_i \qquad \text{(Equation 2)}$$

$y_i$ est l'activation du $i^{ème}$ neurone caché, calculée par l'équation 1.

## 3. Application du réseau de neurones RBFN à l'estimation des coûts de logiciels : cas de base de projets Cocomo'81

Le réseau RBFN produit une sortie (effort de développement) en propageant ses entrées initiales à travers les différents neurones jusqu'à la sortie. Chaque neurone d'une couche cachée calcule sa sortie en appliquant sa fonction d'activation à ses entrées. La figure 1 illustre l'application d'un tel réseau à l'estimation de l'effort de développement de logiciels.

La configuration d'un réseau de neurones RBFN optimal est une tâche difficile. En effet, dans le processus d'apprentissage du réseau, on sépare l'optimisation de la couche

cachée de celle des poids entre la couche cachée et la couche de sortie. Deux étapes séquentielles composent la configuration du réseau RBFN :

- La détermination des neurones de la couche cachée et de leurs paramètres. Cette partie d'apprentissage peut être réalisée par l'une des deux méthodes : appliquer une méthode ad hoc où les centres sont choisis parmi les données d'apprentissage ou appliquer des techniques de regroupement de données (clustering) de type K-means ou SOFM [6]. Dans l'ensemble, le but est de choisir les vecteurs centres comme points représentatifs de zones de grandes concentrations de données.

- La détermination des poids entre la couche cachée et la couche de sortie. Cette partie est réalisée par une technique d'apprentissage supervisé comme la méthode de descente de gradient ou par la méthode de régularisation [6].

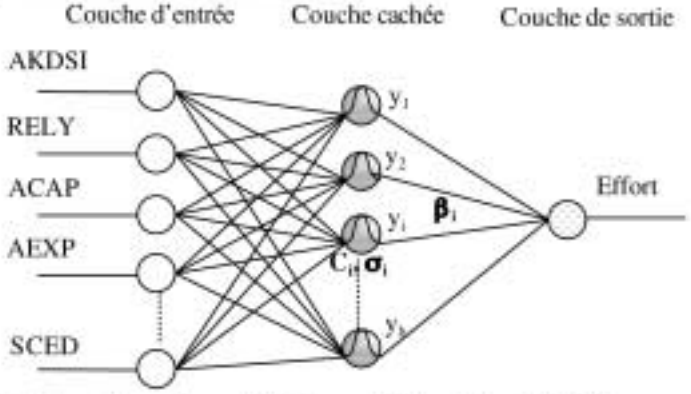


Figure 1. ***Architecture d'un réseau RBFN pour l'estimation de l'effort***

Hwang et al. [8,9] ont proposé une méthode d'apprentissage non supervisé APC-III pour la détermination de la couche cachée du réseau RBFN. La méthode APC-III (Adaptive Pattern Classifier) est une méthode de classification non supervisée comparable à la méthode k-means, SOFM, etc. Elle se distingue de ces méthodes par :

- L'algorithme associé est de type *one pass*, donc converge rapidement, et

- elle est adaptative, ce qui signifie que le nombre de clusters n'est pas fixé au préalable, ainsi, si de nouvelles données sont rajoutées à la base de données, APC-III est mise à jour en considérant uniquement les données nouvellement introduites.

L'algorithme APC-III [8,9] est un algorithme de clustering qui détermine en une seule itération les différents clusters à partir de projets historiques. Il calcule d'abord un rayon $R_0$ qui est proportionnel à la moyenne des distances minimales entre les projets logiciels :

$$R_0 = \alpha / N \sum_{i=1}^{N} \min_{i \neq j} \left( \left\| P_i - P_j \right\| \right) \qquad \text{(Equation 3)}$$

où N est la taille de la base des projets et $\alpha$ est une constante à choisir. Sa valeur est très critique pour le nombre de neurones cachés générés. Si $R_0$ est très petit le nombre de clusters générés sera élevé et vice versa. La deuxième étape de l'algorithme APC-III cherche pour chaque projet logiciel $P_i$, le premier cluster $C_j$ dont le centre $c_j$ a une distance euclidienne avec $P_i$ inférieure ou égale au rayon $R_0$. Cet ajout est accompagné d'un déplacement du centre $c_j$. Si $P_i$ n'appartient à aucun cluster, un nouveau cluster est crée dont le centre est $P_i$.

Notre expérimentation consiste à estimer l'effort de développement de logiciels en utilisant un réseau de neurones de type RBFN. Le réseau s'appliquera à la base de données COCOMO'81. Cette base de données contient 63 projets logiciels [2,3,4]. Chaque projet est décrit par 17 attributs : la taille du logiciel mesurée en KISL (Kilo Instruction Source Livrée), le mode de projet évalué par trois qualifications (organique, semi-détaché ou intégré), et les quinze autres attributs sont mesurés sur une échelle composée de six valeurs linguistiques ; 'très bas', 'bas', 'nominal', 'élevé', 'très élevé' et 'extra élevé'. Parmi ces 17 attributs, nous avons retenus la taille et 12 autres attributs. Les autres attributs ne sont pas pris en compte dans notre étude car leurs descriptions s'avèrent insuffisantes.

## 4. Analyse des résultats

Dans cette section, nous discutons les résultats obtenus de l'application du réseau de neurones RBFN aux projets de la base Cocomo'81. L'analyse des résultats repose sur le fait que la totalité de la base de données Cocomo'81 (les 63 projets logiciels) est utilisée à la fois pour l'apprentissage et les tests de notre réseau de neurones. Chaque projet est décrit par 13 attributs. Le nombre de neurones de la couche d'entrée est ainsi fixé à 13 neurones (facteurs influençant le coût d'un projet logiciel $P_i$). La couche de sortie est réduite à un seul neurone représentant le coût estimé. Le nombre de neurones de la couche cachée correspond au nombre de clusters générés par l'algorithme APC-III. Le tableau 1 présente le nombre de clusters générés par APC-III en fonction de la valeur de $\alpha$.

Tableau 1. ***Nombre de clusters en fonction de*** $\alpha$

| $\alpha$ | Nombre de clusters |
|---|---|
| 0,4/0,45/0,46/0,47/0,49/0,5/ | 62/61 59/60/59 58/56 55/55 54/ |
| 0,52/0,6/0,62/0,66 | 52/51 50/49 48 50//49 48 47 |
| 0,70/0,75/0,78/0,80 | /45 47 48/43 45 46/42 39 41/39 38 37 40 |

Nous remarquons que le nombre de *clusters* décroît en fonction de α. Ceci est dû au fait que $R_0$ est monotone croissante en fonction de α. Pour chaque valeur de α, nous avons varié l'ordre de présentation des 63 projets du COCOMO'81 dans l'algorithme APC-III. En effet, les *clusters* fournis par l'algorithme APC-III dépendent de l'ordre de présentation des projets puisque celui-ci influence les déplacements des centres de ces *clusters*. Nous notons aussi que l'impact de l'ordre de présentation des projets sur la classification générée par APC-III devient plus important quand α est grande (cas de α supérieur à 0,66 dans le tableau 1).

Le choix de la meilleure classification à utiliser dans le réseau RBFN n'est pas immédiat. En effet, dans notre cas, ce choix doit répondre à deux objectifs :

- la classification adoptée doit permettre au réseau RBFN de fournir des estimations acceptables aux coûts des nouveaux projets logiciels, et

- la classification doit être cohérente, c'est-à-dire les projets d'un *cluster* donné sont nécessairement suffisamment similaires.

En réponse aux deux objectifs, nous avons mené une étude empirique qui consiste à répéter plusieurs expérimentations avec un réseau RBFN utilisant à chaque fois, l'une des classifications du tableau 1. Les poids entre la couche cachée et la couche de sortie, $\beta_j$ sont mis à jour par l'algorithme de rétro-propagation de l'erreur avec un taux d'apprentissage égale à 0,03. Le réseau RBFN converge rapidement avec un nombre d'itérations d'apprentissage ne dépassant pas 6000 et une erreur maximale de l'ordre de $10^{-3}$. La précision des estimations du réseau RBFN est évaluée par deux indicateurs MMRE (Mean Magnitude of Relative Error) et Pred(0,25) qui représente le pourcentage des projets logiciels ayant une Erreur Relative (MRE, définie par $MRE = \left|(Effort_{reel} - Effort_{estimé})/Effort_{réel}\right|$) au dessous de 25%. La figure 2 montre les variations de ces deux indicateurs en fonction de la classification adoptée par le réseau RBFN en fonction de α. Nous constatons que la précision des estimations du réseau RBFN est meilleure quand α est inférieure ou égale à 0,49 (MMRE 30% et Pred(0,25) 70%). En revanche, quand α est supérieure à 0,49, MMRE devient grande bien que les valeurs de Pred(0,25) soient satisfaisantes. En effet, l'indicateur de précision MMRE est très sensible aux variations des estimations.

Ainsi, quand la valeur de α augmente, la classification générée par l'algorithme APC-III est moins cohérente. Cette non cohérence influence négativement la précision des estimations du réseau de neurones RBFN. Afin de vérifier cette hypothèse, nous avons analysé les clusters fournis par l'algorithme APC-III moyennant les mesures de similarité précédemment définies dans [10,11]. Nous avons constaté qu'à partir de α supérieur ou égale à 0,49, certains clusters contiennent des projets qui ont des degrés de similarité faibles. En tant qu'illustration, nous traitons le cas de α égale à 0,5 : un nouveau cluster est formé et composé des projets $P_{13}$ et $P_{15}$ auxquels s'ajoute le projet $P_{57}$ quand α est égale à 0,66. L'évaluation de la similarité entre ($P_{13}$, $P_{15}$), d'une part, et de celle entre ($P_{13}$,$P_{57}$), d'autre part, indique que, dans le premier cas, les deux projets $P_{13}$ et $P_{15}$ ont six similarités individuelles égales à zéro parmi douze alors que, dans le deuxième cas, les deux projets $P_{13}$

et $P_{27}$ en ont sept. Les MRE respectives aux projets $P_{13}$ et $P_{27}$ sont nettement grandes 100.50% et 118.64%.

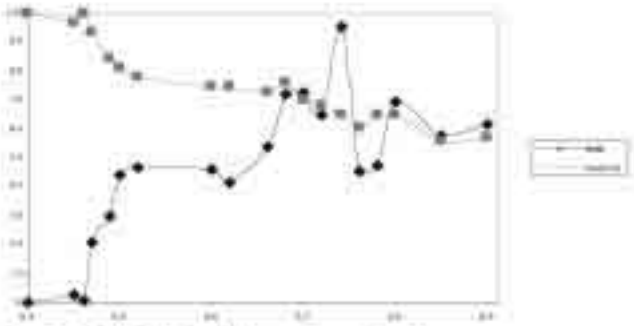

Figure 2. ***Variation du MMRE Pred(25) en fonction de*** $\alpha$

La précision des estimations d'un réseau RBFN ne dépend pas uniquement du nombre de neurones de la couche cachée, voire du paramètre $\alpha$ ; mais aussi de la variance $\sigma_i$ utilisée par la fonction d'activation des neurones de la couche cachée (Equation 1). Dans nos expérimentations précédentes, nous avons utilisé une même valeur $\sigma_i$ égale à 30 pour les neurones de la couche cachée. Cependant, dans la littérature, la valeur $\sigma_i$ doit être choisie de telle manière à recouvrir uniformément l'espace des entrées [9]. Dans notre cas, ce choix ne nous paraît pas adéquat. En effet,

Un recouvrement uniforme de tout l'espace de entrées implique que le réseau RBFN pourra générer une estimation au coût d'un nouveau projet même s'il ne présente aucune similarité satisfaisante avec les projets logiciels historiques.

Le choix des valeurs $\sigma_i$ pour recouvrir uniformément tout l'espace des entrées n'est pas unique.

Dans le cas de la base de projets COCOMO'81, les centres des clusters, générés par l'algorithme APC-III, sont souvent très éloignés les uns des autres. Ainsi, les valeurs de $\sigma_i$ doivent être assez grandes pour assurer un recouvrement uniforme de tous les projets du COCOMO'81. Cependant, comme nous l'illustrons ci-dessus, des valeurs assez grandes de $\sigma_i$ mènent souvent à des estimations imprécises dans le cas des projets Cocomo'81.

La stratégie que nous adoptons pour le choix des valeurs $\sigma_i$ se fonde essentiellement sur deux hypothèses : premièrement, une seule valeur est assignée à toutes les $\sigma_i$ ; deuxièmement, cette valeur dépendra du rayon $R_0$. Pour valider notre proposition, nous évaluons la précision d'un réseau RBFN en variant la valeur de $\sigma_i$ pour une même classification. L'analyse des résultats de la figure 3 montre que si $\sigma_i$ est au voisinage de $R_0$, la précision des estimations fournies par le réseau RBFN sont satisfaisantes. Cependant, si $\sigma_i$ est loin de $R_0$, le réseau génère des estimations moins précises. La marge de tolérance autour de $R_0$ dépend de la classification obtenue.

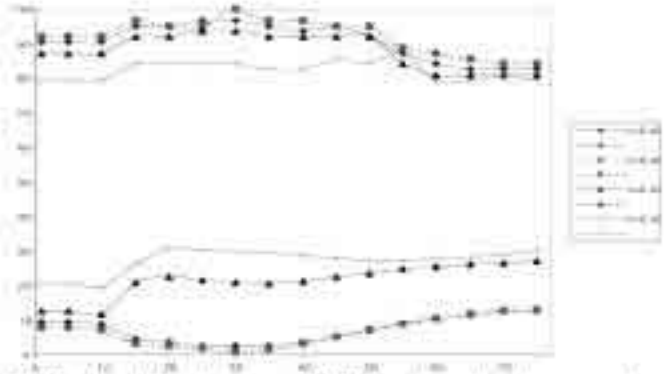

Figure 3. ***MMRE et Pred(25) du réseau RBFN en fonction de σ pour les quatre valeurs de λ: 0,45, 0,46, 0,47 et 0,49.***

## 5. Conclusion et perspectives

Dans cet article, nous avons étudié l'application du réseau de neurones RBFN à l'estimation des coûts de logiciels. La configuration d'un tel réseau constitue la tâche la plus difficile de son utilisation. En effet, le processus d'apprentissage s'effectue en deux étapes séquentielles : détermination des paramètres de la couche cachée par l'algorithme de classification non supervisé APC-III. Ensuite, l'algorithme de descente de gradient est appliqué pour la détermination des poids entre la couche cachée et la couche de sortie.

Au cours de ce travail, nous avons utilisé la base des projets Cocomo'81 pour la configuration de notre réseau RBFN. la classification des projets logiciels choisie devait répondre à deux objectifs principaux : permettre au réseau de fournir des estimations acceptables aux coûts des nouveaux projets logiciels et la classification doit être cohérente. Afin de satisfaire le premier objectif, nous avons mené une étude empirique dont le but est de choisir la classification qui donne une meilleure précision du réseau RBFN. Pour vérifier le deuxième objectif, nous avons employé des mesures de similarité précédemment définit dans [10.11]. La validation du modèle neuronale s'est achevée par le choix de la variance des fonctions d'activation. Nous proposons dans un futur travail d'étudier l'interprétation d'un réseau de neurones RBFN en estimation des coûts de logiciels et mener une étude comparative entre les deux approches : le réseau MLP et le réseau RBFN.

## Bibliographie


[1] R. Bisio, F. Malabiecchia, 'Cost estimation of software projects through case based reasoning', in International Conference on Case Based reasoning, Sesimbra, Portugal, 1995

[2] B.W. Boehm, *Software Engineering Economics*, Prentice-Hall, 1981.

[3] B.W. Boehm, and *al.*, "Cost Models for Future Software Life Cycle Processes: COCOMO 2.0", *Annals of Software Engineering on Software Process and Product Measurement*, Amsterdam, 1995.

[4] D.S. Chulani, "Incorporating Bayesian Analysis to Improve the Accuracy of COCOMO II and Its Quality Model Extension", Ph.D. Qualifying Exam Report, USC, February, 1998.

[5] J.M. Desharnais, 'Analyse statistique de la productivité des projets de développement en informatique à partir de la technique des points de fonction', M sc. Thesis, Université du Québec, Montréal, 1988

[6] S. Haykin, Neural Networks : A Comprehensive Foundation, IEEE PRESS, 1994.

[7] R.T Hughes, 'An Evaluation of Machine Learning Techniques for Software Effort Estimation', University of Brighton, 1996

[8] Y. S. Hwang, S. Y. Bang, "A neural network model APC-III and its application to unconstrained handwritten digit recognition", Proceeding of International Conference on Neural Information Processing, pp 1500-1505.

[9] Y. S. Hwang, S. Y. Bang, "An Efficient Method to Construct a Radial Basis Function Network Classifier", Neural Networks, vol. 10, n° 8, 1997, pp. 1495-1503.

[10] A. Idri, and A. Abran, "Towards A Fuzzy Logic Based Measures For Software Project Similarity", *Sixth Maghrebian Conference on Computer Sciences. Fes*, Morocco, November, 2000. pp. 9-18

[11] A. Idri, T. M. Khoshgoftaar, A. Abran, , "Estimating Software Project Effort by Analogy based on Linguistic values", 8th IEEE International Software Metrics Symposium", Ottawa, Canada, 4-7 juin 2002, pp. 21-30

[12] A. Idri, S. Mbarki, A. Abran, "L'interprétation d'un réseau de neurones en estimation du coût de logiciels", Actes du 6ème Colloque Africain sur la recherche en Informatique (CARI'02), 14-17 octobre 2002, pp. 221-228.

[13] J. Park, I.W. Sandberg, 'Approximation and radial basis function network', Neural Computation, 5, 1993, pp. 305-316

[14] C. Schofield, 'Non-Algorithmic Effort Estimation Techniques', Tech. Report TR98-01, March, 1998

[15] C. Serluca, 'An Investigation into Software Effort Estimation using a Back-propagation Neural Network, M.Sc. Thesis, Bournemouth University, 1995