Uma Investigação de Modelos de Estimativas de Esforço em Gerenciamento de Projeto de Software

Iris Fabiana de Barcelos Tronto, José Demísio Simões da Silva, Nilson Sant'Anna

Laboratório Associado de Computação e Matemática Aplicada – Instituto Nacional de Pesquisas Espaciais (INPE)
Caixa Postal 515 – 12.201-970 – São José dos Campos – SP – Brazil

{iris_barcelos,demisio,nilson}@lac.inpe.br

Abstract. *Accurate estimation in software project management is critical. Fundamental measurements are size, effort, resources, cost, and time spent in the software development process. Underestimates can lead to time pressures that compromise full functional development and thorough testing of software. Likewise, overestimates can result in noncompetitive bids. In this paper, predictive artificial neural network and stepwise regression based models are investigated, with the goal of offering an alternative for those who do not believe in estimation models. The results presented in this paper compare the performance of both methods and indicate that these techniques are competitive with the methods of APF, SLIM, and COCOMO.*

Resumo. *Estimativas precisas em gerenciamento de projetos é um fator crítico. Medidas de tamanho, esforço, recursos, custo, e tempo despendidos no desenvolvimento de software são fundamentais. Valores subestimados de esforço podem fazer com que pressões de tempo comprometam todo o desenvolvimento funcional e até mesmo o teste de software. Por outro lado, valores superestimados podem resultar em projetos não competitivos. Neste artigo, modelos como redes neurais e regressão, são apontados como alternativas para aqueles que não acreditam em modelos de estimativas. Os resultados apresentados comparam o desempenho desses métodos e indicam que estas técnicas são competitivas com os métodos APF, SLIM e COCOMO.*

1. Introdução

O desenvolvimento contínuo das áreas de hardware e de software e o fenômeno da globalização da economia tem contribuído para o aumento da competitividade entre empresas produtoras e prestadoras de serviços de software. Assim, há uma crescente necessidade de se produzir software com maior qualidade, em tempo hábil e com baixo custo.

Um nível de qualidade e produtividade internacional pode ser conseguido através da gestão efetiva de seus processos de software, focalizando pessoas, produto, processo e projeto. Em relação a projeto, é preciso que haja planejamento e acompanhamento através de um conjunto de atividades, dentre as quais as estimativas podem ser consideradas fundamentais, pelo fato de fornecerem um guia para as demais atividades. As estimativas estão relacionadas ao esforço, aos recursos e ao tempo de desenvolvimento, necessários para se construir um software [Agarval 2001].

É amplamente aceito o fato de que as estimativas de tamanho de software são importantes para se determinar o esforço de projeto de software [Jones 1986], [Lai and Huang 2003], [Hasting and Sajeev 2001], [Briand and Wieczorek 2002]. Porém, de acordo com última pesquisa relatada pelo Ministério da Ciência e Tecnologia, no Brasil, em 2001 apenas 29% das empresas realizam estimativas de tamanho e 45,7% realizavam estimativa de esforço de software [MCT 2001]. Não há um estudo específico que identifique as causas do baixo índice de realização de estimativas de esforço, mas o nível de confiabilidade dos modelos pode ser uma das possíveis causas. Diante destes dados apresentados pelo MCT é evidente a necessidade de se ter uma abordagem alternativa para realizar estimativas de esforço, através da qual possa ter estimativas confiáveis com modelos de implementação simples.

Tendo em vista que realizar estimativas consistentes e precisas é um grande desafio para os gerentes de projeto, muitas pesquisas têm sido conduzidas a fim de construir, avaliar e recomendar técnicas de predição [Boehm 1981], [Albrecht and Gaffney 1983], [Shepperd and Schofield 1997], [Zhong et al. 2004], [Sentas et al. 2005], [Bisio and Malabocchia 1995], [Myrtveit et al. 2005], [Samson et al. 1997]. Essas técnicas se enquadram em três categorias gerais [Shepperd et al. 1996]

1) Julgamento de Especialistas – está técnica tem sido amplamente utilizada. Entretanto, os meios de se derivar uma estimativa não são explícitos e conseqüentemente não repetíveis. A opinião do especialista, embora seja sempre difícil de se quantificar, ela pode ser uma ferramenta de estimativa efetiva como um fator de ajuste para modelos algorítmicos [Gray et al. 1999].

2) Modelos Algorítmicos – estes modelos são os mais populares na literatura e precisam ser calibrados ou ajustados a circunstâncias locais.. Eles tentam representar o relacionamento entre esforço e uma ou mais características. Geralmente, o principal direcionador de custo utilizado em tais modelos é aquele referente ao tamanho do software (por exemplo, o número de linhas de código fonte, o número de Pontos por Função, o número de Pontos por Caso de Uso, o número de páginas, dentre outros). Dentre os modelos algorítmicos tem-se o Modelo COCOMO [Boehm 1981] e o Modelo SLIM [Putnam,1978].

3) Aprendizagem de Máquina – Na última década, as técnicas de aprendizagem de máquina têm sido utilizadas como um complemento ou uma alternativa para as duas categorias anteriores. Exemplos incluem modelos de lógica nebulosa [Kumar et al. 1994], árvores de regressão [Selby and Porter 1998], redes neurais artificiais [Srinivasan and Fisher, 1995] e raciocínio baseado em casos [Shepperd et al. 1996]. Um resumo útil destas técnicas é apresentado em [Gray e MacDonell 1997].

Nos últimos anos, têm sido realizados vários estudos comparativos utilizando essas três categorias de técnicas de estimativa, baseando-se no poder de predição [Gray and MacDonell 1997], [Briand et al. 2000], [Jeffery et al. 2001], [Shepperd et al 1996] [Angelis and Stamelos 2000], [Finnie et al. 1997]. Porém, como os conjuntos de dados empregados têm diferentes características (*outliers,* colinearidade, número de variáveis, número de observações, dentre outras), diferentes medidas de precisão e diferentes projetos comparativos, ainda não se chegou a uma convergência e há a dúvida sobre qual técnica ou quais técnicas se deve utilizar.

Existem vários fatores que podem e devem ser considerados na seleção de uma técnica de predição. A seleção de técnica deve ser dirigida pela necessidade e capacidade organizacional. Em termos de necessidade, o objetivo mais comum é maximizar a precisão da estimativa, dentre outras. Por exemplo, uma técnica que produz estimativas ligeiramente menos precisas pode ser preferida em detrimento de modelos mais robustos, em organizações que não tenham acesso a uma calibração local ou um conjunto de dados bem comportado. Em termos de capacidade organizacional, algumas técnicas de modelagem são mais complexas que outras e requerem experiência significativa para que elas sejam utilizadas efetivamente. Há benefícios de se empregar técnicas mais sofisticadas (potencialmente mais útil) para construir modelos de estimativas, porém isso somente fornecerá benefícios reais se as técnicas forem usadas apropriadamente.

Em um trabalho anterior [Barcelos Tronto et al. 2006], foi realizado um estudo comparativo dos resultados de estimativa de esforço utilizando as técnicas de análise de regressão e redes neurais. O esforço foi estimado baseado em uma única variável independente: tamanho de software - que é representado por uma variável numérica, denominada KDSI, do conjunto de dados COCOMO. Os resultados mostraram que o modelo de redes neurais rendeu melhores resultados que regressão, com estimativas mais precisas. Diferentes medidas de erro têm sido utilizadas por vários pesquisadores, mas neste projeto a principal medida para avaliar a precisão do modelo é a magnitude média do erro relativo – *Mean Magnitude of Relative Error* – MMRE. MMRE é a percentagem média de erros absolutos:

$$MMRE = \frac{\left(\sum_{i=1}^{n} \left| \frac{M_{est} - M_{act}}{M_{act}} \right| * 100 \right)}{n} \qquad (2)$$

em que n é o número de projetos; M_{act} é o esforço real observado; e M_{est} é o esforço estimado. Neste artigo, o objetivo é investigar como estas técnicas se comportam quando são utilizadas outras variáveis independentes, que possuem características diferentes (por exemplo, variáveis categóricas).

Para atingir este objetivo, foi realizado um estudo para contrastar os resultados obtidos utilizando uma rede neural do tipo perceptron de múltiplas camadas com o agoritmo de aprendizagem por retro-propagação do erro, com os resultados obtidos através de uma regressão múltipla *stepwise*, sobre o conjunto de dados COCOMO [Boehm 1981]. Este artigo está estruturado em cinco seções. A Seção 2 apresenta um pequeno resumo de estudos empíricos relacionados à estimativa de software. Uma breve descrição sobre redes neurais e modelos de regressão é apresentada nas seções 3 e 4, respectivamente. A Seção 5 apresenta um caso de estudo, em que são mostrados os modelos de regressão e de redes neurais gerados e uma análise de confiabilidade deles. Finalmente, na Seção 6 é apresentada uma discussão da significância dos resultados obtidos e sobre perspectivas de trabalhos futuros.

2. Modelos de Estimativa Aplicados à Engenharia de Software

Estimativas precisas e consistentes de recursos necessários para o desenvolvimento de um projeto de software são componentes determinantes no gerenciamento efetivo de

projetos de software. Apesar da extensiva pesquisa realizada nos últimos vinte anos, a comunidade de software ainda é significativamente desafiada quando se trata de estimativas efetivas de recursos. De uma forma geral, grandes esforços têm sido investidos no desenvolvimento de técnicas para remover ou reduzir a subjetividade no processo de estimativa de custo e de esforço de software. Exemplos destes trabalhos incluem modelos paramétricos e baseados em regressão, como Análise de Pontos por Função [Albrecht 1979], o modelo COCOMO [Boehm 1981; Boehm et al. 2000] e o modelo de Regressão Ordinal [Sentas et al. 2005].

Porém, outras técnicas para a análise exploratória de dados, como *clustering,* raciocínio baseado em casos e redes neurais artificiais têm se mostrado eficiente para realizar predições. Zhong et al (2004) descreve o uso de *clustering* para realizar estimativas de qualidade de software. Uma abordagem baseada em casos, chamada ESTOR, foi desenvolvida para estimativa de esforço de software [Vicinanza et al. 1990]. Vicinanza et al. mostraram que *ESTOR* é comparável a um especialista e significativamente mais preciso que o modelo para estimativas COCOMO ou Pontos por Função, sobre amostras restritas de problemas. Há pesquisas importantes que utilizam redes neurais na tentativa de produzir estimativas mais precisas de recursos [Gray and MacDonnell, 1999],[Witting e Finnie, 1997].

Em [Karunanithi et al. 1992] redes neurais são utilizadas para predizer confiabilidade de software, sendo que os experimentos são realizados com redes de Jordan e com um algoritmo de aprendizagem de correlação cascata.

Outro estudo importante, realizado por Samson et al. (1997) utiliza uma arquitetura *perceptron* de múltiplas camadas de forma a estimar o esforço de software. Eles utilizam o conjunto de dados COCOMO implementado por Boehm. O trabalho compara regressão linear, com uma única variável dependente, com uma abordagem de redes neurais. Mas, as duas abordagens parecem executar pobremente com um MMRE de 520,7% e 428,1%, respectivamente.

Srinivasan e Fisher (1994) também relatam o uso de uma rede neural com um algoritmo de retro propagação do erro. Eles encontraram um MMRE = 70% resultante da aplicação da rede neural, o que significa um bom resultado quando comparado com outros modelos como COCOMO, SLIM e Pontos por Função. Entretanto, eles não deixam claro como o conjunto de dados foi dividido para treinar e validar o modelo. Aparentemente existe uma inconsistência no cálculo do MMRE para o modelo gerado.

Khoshgoftaar et al (2000) apresenta um estudo de caso realizado sobre dados de projetos de software de tempo real para estimar a testabilidade de cada módulo a partir de medidas estáticas de código fonte. Eles consideram redes neurais uma técnica promissora para construir modelos de predição, porque elas são capazes de modelar relacionamentos não lineares.

Finalmente, nos últimos anos, pôde-se perceber que o interesse pela utilização de redes neurais tem aumentado, em geral para projetar soluções para problemas de estimativas, classificação, controle, dentre outros. Redes Neurais têm sido aplicadas, com sucesso, em vários domínios de problemas, em áreas tais como medicina, engenharia, geologia e física. Elas podem ser usadas como modelos de predição porque são técnicas de modelagem capazes de modelar funções complexas.

Neste trabalho, a metodologia de redes neurais é utilizada com o objetivo de predizer o esforço de desenvolvimento de software (em homens-hora) a partir do tamanho do projeto (dado pela quantidade de linhas de código fonte) e de outros atributos direcionadores de custo e de esforço. Foi realizada uma análise comparativa entre o modelo de regressão e o modelo de redes neurais calibrados e testados neste estudo.

3. Redes Neurais

Redes Neurais artificiais – RNA, são sistemas paralelos inspirados na arquitetura de redes neurais biológicas, que compreende algumas unidades interconectadas (neurônios artificiais). O neurônio computa uma soma pesada de suas entradas e gera uma saída, se a soma excede um certo limite. Esta saída, então, torna uma entrada estimulante (positiva) ou inibitória (negativa) para outros neurônios na rede. O processo continua até que uma ou mais saídas sejam geradas.

A Figura 1 mostra um neurônio artificial que computa a soma pesada de suas *n* entradas e gera uma saída de *y*. A rede neural é o resultado de um arranjo de tais unidades em camadas, que são interconectadas umas as outras. As arquiteturas resultantes resolvem problemas através da aprendizagem das características dos dados disponíveis relacionados ao problema. Existem muitos algoritmos diferentes de aprendizagem. Os algoritmos *feed-forward multilayer perceptrons* são os mais comumente usados em RNA, embora existam redes neurais mais sofisticadas. Na maioria das vezes, as arquiteturas de múltiplas camadas são treinadas com o algoritmo de aprendizagem por retro-propagação do erro, que requer uma função de ativação diferençável.

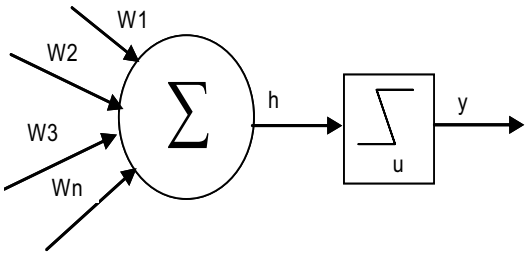

Figure 1. Um neurônio de McCulloch e Pitts

A rede é iniciada com pesos randômicos e, gradualmente, aprende os relacionamentos implícitos em um conjunto de dados de treinamento através do ajuste de seus pesos. Dentre os vários algoritmos de treinamento disponíveis, o de aprendizagem por retro-propagação do erro é o mais usado pelos pesquisadores de métricas de software.

Em geral, a maioria dos estudos relacionados com o uso de RNA para predizer esforço de desenvolvimento de software tem como principal objetivo comparar a precisão dos modelos. Por exemplo, Witting e Finnie (1997) exploram o uso de uma rede neural de múltiplas camadas sobre os conjuntos de dados de Desharnais e da Associação de Métricas de Software Australiana ASMA. Para o conjunto de dados de Desharnais, eles dividiram os projetos três vezes, randomicamente, em conjuntos de teste (contendo 10 projetos) e de treinamento (contendo 71 projetos). Os resultados obtidos dos três conjuntos de validação são agregados e mostram um alto nível de

precisão (MRE – *Magnitude of Relative Error* – para o conjunto de dados de Desharmais = 27% e o MRE para ASMA = 17%). Porém, alguns valores que representam possíveis *outliers* foram excluídos.

Entretanto, outros fatores também precisam ser investigados, por exemplo, o valor exploratório é igualmente importante. Neste trabalho, além de se analisar a precisão das predições, também se investiga a adequação da abordagem para construir sistemas de predição de esforço de software.

4. Regressão

Regressão tenta encontrar um relacionamento entre um ou mais parâmetros de predição (variáveis independentes) e uma variável dependente, minimizando o quadrado dos erros para a gama de eventos do conjunto de dados. Alguns pesquisadores têm defendido a construção de modelos locais simples, por exemplo, Kok et al.(1990), usa este tipo de abordagem. A filosofia é, essencialmente, resolver problemas de predição locais antes de tentar construir modelos universais. O sistema de predição resultante tem a forma:

$$Y_{ext} = \beta_0 + \beta_1 X_1,...,\beta_n X_n \quad (1)$$

Onde $\acute{Y}$ é o valor estimado e X_1, …,X_n são variáveis independentes, por exemplo, o tamanho do projeto (dado como a quantidade de linhas de código), o tamanho da base de dados, a capacidade do analista, a confiabilidade requerida do software. De acordo com os resultados encontrados, a variável que representa o tamanho do projeto influencia significativamente a estimativa de esforço.

A desvantagem com esta técnica é sua vulnerabilidade para valores extremos (*outliers)*, embora tenham sido utilizadas com sucesso, técnicas de regressão, que são menos sensíveis a tais problemas [Briand and Wieczorek 2002]. Outro problema é o impacto de co-linearidade – a tendência de variáveis independentes estarem fortemente correlacionadas umas com as outras – sobre a estabilidade de um sistema de predição do tipo regressão. Por exemplo, neste trabalho, verifica-se que a precisão das estimativas pode estar longe do desejado quando se aplica uma regressão linear múltipla sobre as variáveis categóricas do conjunto de dados COCOMO, sem um pré-processamento específico para as variáveis categóricas.

5. Dados, Experimentos e Resultados

A análise empírica abordada neste estudo, foi realizada sobre uma base de dados de projetos de software que tem sido utilizado em estudos anteriores na literatura de engenharia de software [Sentas et al. 2005], [Srinivasan e Fisher 1995], [Samson et al. 1997]. Este conjunto de dados foi selecionado porque ele está disponível para uso público e foi utilizado para descrever e testar um dos mais importantes métodos na área, que é o modelo COCOMO [Boehm,1981], [Boehm et al. 2000]. O conjunto de dados consiste de 63 projetos, incluindo instâncias de projetos de software de negócio, científico e de sistema, escritos em uma variedade de linguagens de programação que incluem COBOL, PLI, HMI e FORTRAN. As variáveis multiplicadoras de esforço de desenvolvimento de software (e uma breve descrição delas) consideradas neste estudo são apresentadas na Tabela 1. KDSI é uma variável do tipo numérico contínuo enquanto

que as demais são variáveis categóricas em que cada categoria é representada por um valor numérico.

Tabela 1. Multiplicadores de esforço de desenvolvimento de software

Variável	Descrição
RELY	Confiabilidade requerida do software
DATA	Tamanho da base de dados
CPLX	Complexidade do produto
TIME	Restrição de tempo de execução
STOR	Restrição de armazenamento principal
VIRT	Volatilidade da máquina virtual
TURN	Tempo de execução da máquina
ACAP	Capacidade do analista
AEXP	Experiência com aplicações
PCAP	Capacidade do programador
VEXP	Experiência com máquina virtual
LEXP	Experiência com linguagem de programação
MODP	Uso de práticas modernas de programação
TOOL	Uso de ferramentas de software
SCED	Cronograma de desenvolvimento requerido
RVOL	Volatilidade dos requisitos
KDSI	Número de linhas de código ajustado

Todos os 63 projetos foram utilizados na análise realizada. A variável dependente, considerada neste estudo, é ACT MM (quantidade de esforço total, em número de homens /hora, despendido no desenvolvimento do projeto). De uma maneira geral, para cada um dos experimentos os dados históricos são divididos em amostras usadas para treinar o sistema de aprendizagem e amostras separadas para testar a precisão do classificador treinado para estimar o esforço de desenvolvimento.

Estimativas de precisão das predições obtidas a partir de um conjunto de dados de treinamento são sempre otimistas. De forma a investigar a precisão dos modelos de redes neurais e de regressão múltipla construídos como parte deste estudo, utilizou-se o mesmo procedimento que em [Kitchenham 1998]. Kitchenham desenvolveu um procedimento simples para investigar a precisão das predições realizadas pelo seu modelo sobre a base de dados COCOMO. Baseando-se neste processo, omitiu-se um subconjunto de projetos (o conjunto de dados de teste), depois se desenvolveu um modelo com os projetos restantes (o conjunto de dados de treinamento) e finalmente avaliou-se a precisão das predições do modelo sobre o conjunto de dados de teste. Desta forma, foram criados seis diferentes pares de conjuntos de treinamento e de teste. Cada conjunto de treinamento foi construído removendo-se todo sexto projeto, a começar (da primeira vez) do primeiro projeto. Assim, o conjunto de dados de aprendizagem 1 foi

construído removendo-se os projetos: 1, 7, 13, 19, 25, 31, 37, 43, 49, 55 e 61; o conjunto de dados de aprendizagem 2 foi construído removendo-se os projetos: 2, 8, 14, 20, 26, 32, 38, 44, 50, 56, 62 e assim sucessivamente. Cada conjunto de dados removidos representa o respectivo conjunto de dados de teste.

Sabendo-se que todos os 63 projetos da base de dados COCOMO foram utilizados para construir o modelo, dos seis pares de conjuntos criados três conjuntos de dados de aprendizagem contêm 52 projetos e os outros três contêm 53 projetos.

Para um propósito comparativo entre os métodos algorítmicos e os métodos de predição de aprendizagem de máquina, foram conduzidos experimentos utilizando análise de regressão múltipla *stepwise* e redes neurais artificiais, sobre cada um desses conjuntos de dados.

A rede neural foi implementada com 17 entradas (a 18ª variável é o esforço a ser estimado, dado pela variável ACT MM), 5 neurônios na primeira camada, 3 neurônios na segunda camada e um neurônio de saída para estimar o esforço (variável de saída). Os dados foram todos normalizados em relação aos valores máximos que cada variável pode assumir. As variáveis de entrada são listadas na Tabela 1. A fase de treinamento foi repetida várias vezes, em uma busca pela melhor rede para resolver o problema. Por outro lado, várias arquiteturas de redes neurais foram experimentadas e os resultados apresentados neste artigo correspondem à rede neural com o melhor desempenho de generalização. As predições obtidas a partir de RNA (depois do treinamento sobre os seis conjuntos de dados de treinamento) usando os seis conjuntos de teste são mostradas na Tabela 2, sendo que "obs" representa os valores reais e "est" os valores estimados.

Tabela 2. Esforço estimado através de modelos de redes neurais

Conjunto1		Conjunto2		Conjunto3		Conjunto4		Conjunto5		Conjunto6	
obs	est	obs	est	obs	est	obs	est	obs	est	obs	est
2040	401,9	1600	2246,3	243	265,4	240	544,9	33	111,9	43	162,6
8	224,8	1075	858,4	423	258,1	321	249,5	218	165,9	201	135,1
79	254,5	73	269,5	61	399,8	40	466,9	9	165,0	11400	6077,8
6600	10570,3	6400	10543,9	2455	523,2	724	293,3	539	119,0	453	281,9
523	432,8	387	229,2	88	223,6	98	266,5	7,3	118,2	5,9	94,5
1063	448,3	702	3781,9	605	360,7	230	201,0	82	103,7	55	104,3
47	234,0	12	191,3	8	154,3	8	135,5	6	79,5	45	86,0
83	287,1	87	244,0	106	179,2	126	191,4	36	82,5	1272	2864,4
156	293,4	176	301,5	122	384,2	41	226,3	14	95,1	20	101,9
18	254,1	958	378,1	237	212,0	130	492,3	70	95,4	57	145,2
50	237,6	38	181,4	15	147,8						

Vários experimentos foram realizados e seis modelos foram calibrados, utilizando a técnica de regressão múltipla e o método *stepwise forward*, com um nível de significância igual a 0,05. Cada um desses modelos foi treinado sobre seu respectivo

conjunto de dados de treinamento, considerando ACT MM como a variável dependente e tomando-se todas as variáveis mostradas na Tabela 1 como variáveis independentes. A Tabela 3 mostra os modelos de regressão finais encontrados.

Os experimentos foram realizados usando o módulo GRM - General Regression Models, implementado pelo pacote de software *STATISTICA*. Foi implementado o algoritmo *subset model-building* para encontrar o melhor modelo a partir de um número de possíveis modelos. A estatística conhecida como R^2_ajustado, dos subconjuntos (modelos), permitiu comparações diretas e a escolha do "melhor" subconjunto entre os vários subconjuntos encontrados. Para cada conjunto de dados de treinamento escolheu-se o modelo (dentre dez modelos gerados) que tem o maior valor de R^2_ajustado. Conseqüentemente, foram obtidos seis modelos. Apesar dos seis conjuntos de treinamento serem derivados de uma mesma base de dados de projetos (a base de dados COCOMO), os respectivos modelos encontrados, são constituídos de variáveis diferentes entre si.

Todos os modelos encontrados possuem algumas variáveis com um valor de β considerado não significativo. Como o objetivo é obter uma maior precisão de estimativa, aplicou-se uma regressão múltipla *stepwise forward* tomando como variáveis independentes, aquelas contidas no modelo. Esta regressão foi utilizada para cada um dos seis modelos de forma que somente as variáveis com um valor de β significativo fossem consideradas no modelo. A regressão *stepwise forward* constrói um modelo de predição, adicionando-se ao modelo (a cada estágio) a variável com a maior correlação parcial com a variável dependente, considerando-se todas as variáveis presentes no modelo. O objetivo é encontrar um conjunto de variáveis independentes que maximizam a estatística F. F avalia se as variáveis independentes, quando consideradas juntas, são significativamente associadas com a variável dependente. O critério utilizado para adicionar uma variável é se ela aumenta o valor de F para a regressão, por alguma quantidade *k*. Quando uma variável reduz F, também por alguma quantidade *w*, ela é removida do modelo. As variáveis independentes que compõem cada modelo são apresentadas na Tabela 3.

Tabela 3. Modelos de regressão múltipla

Modelo	Equação de Regressão
Modelo 1	ACT MM = -2937,57453363+16,1182817KDSI+2410,01478707*CPLX
Modelo 2	ACT MM =-5558,849858+ 7,980057*KDSI+ 5654,065476*DATA
Modelo 3	ACT MM=-3709,2565+3757,96154*MODP+9,05218371*KDSI
Modelo 4	ACT MM=-8835,5111+5396,35474*DATA+3638,30337*MODP+7,77161528*KDSI
Modelo 5	ACT MM =-3682,0398+3802,92414*MODP+8,92926545*KDSI
Modelo 6	ACT MM=-4074,3481+1394,65051*RELY+2308,47763*VEXP +2535,60316*MODP-2129,0629*TOOL+7,29998110*KDSI

As estimativas dos modelos de regressão múltipla (obtidos da calibração sobre os conjuntos de treinamento) usando os conjuntos de teste são mostradas na Tabela 4.

Tabela 4. Esforço estimado através dos modelos de regressão múltipla

Conjunto1	Conjunto2	Conjunto3	Conjunto4	Conjunto5	Conjunto6
570,8	2986,9	905,4	2293,3	1176,5	-119,4
-416,3	-68,5	-17,9	15,3	299,5	286,4
582,3	-220,1	-254,2	227,9	137,8	2976,2
15042,7	2837,8	1991,6	718,8	790,6	661,3
807,9	478,3	-542,6	1095,7	-202,3	-154,2
278,4	2630,3	72,6	474,4	1149,7	-83,7
-702,5	214,9	-233,4	258,6	-174,1	-240,9
92,5	560,8	338,4	857,9	706,5	2161,0
577,7	286,7	1041,1	812,3	168,2	-373,7
-1149,1	-28,6	560,3	-250,1	-33,9	564,9
285,3	-171,4	-537,2			

Diferentes medidas de erro têm sido utilizadas por vários pesquisadores, mas neste projeto a principal medida para avaliar a precisão do modelo é a magnitude média do erro relativo – *Mean Magnitude of Relative Error* – MMRE. MMRE é a percentagem média de erros absolutos:

$$MMRE = \frac{\left(\sum_{i=1}^{n} \left| \frac{M_{est} - M_{act}}{M_{act}} \right| * 100 \right)}{n} \qquad (2)$$

em que n é o número de projetos; M_{act} é o esforço real observado; e M_{est} é o esforço estimado.

Neste artigo, o MMRE é adotado como indicador de desempenho de predição, uma vez que ele é amplamente utilizado e nos permite comparar os resultados aqui obtidos com aqueles obtidos por outros pesquisadores. A Tabela 5 apresenta os valores de MMRE obtidos através das estimativas realizadas sobre os seis conjuntos de dados de teste, utilizando os modelos de redes neurais artificiais, os modelos de regressão linear simples (somente com uma variável independente, KDSI) e os modelos de regressão múltipla gerados como resultado deste estudo.

Outros pesquisadores têm utilizado o R^2_ajustado ou o coeficiente de determinação para indicar a percentagem de variação na variável dependente que é "explicada" em termos das variáveis independentes. Neste estudo realizou-se uma análise de regressão linear para calibrar as predições realizadas utilizando redes neurais e regressão múltipla. Para tanto se considerou M_{est} como a variável independente e M_{act} como a variável dependente. O valor de R^2 indica a quantidade de variação nos valores reais de esforço, considerada por causa de um relacionamento linear com os valores estimados. Valores de R^2 próximos de 1.0 sugerem um forte relacionamento linear e aqueles próximos de 0.0 sugerem que não há relacionamento.

A Tabela 5 apresenta, além dos valores de MMRE, os valores de R^2 resultantes da regressão linear de M_{est} e M_{act} para os modelos de regressão múltipla, regressão

linear simples e de redes neurais. Também são mostrados os resultados obtidos por Srinivasan e Fisher (1995) utilizando um modelo de rede neural com retro-propagação do erro e os resultados obtidos por Kemerer (1987) com os modelos COCOMO-Básico, Pontos por Função e SLIM. Esses resultados indicam que as predições realizadas utilizando os modelos de redes neurais e de regressão linear simples têm um forte relacionamento linear com os valores de esforço de desenvolvimento reais observados para os projetos usados para teste. Sobre esta dimensão R^2, o desempenho do modelo de redes neurais é menor que o desempenho de SLIM nos experimentos de Kemerer e o de regressão linear simples usando somente a variável KDSI como dependente, mas melhor que os modelos de regressão múltipla *stepwise*. Em termos de MMRE, o modelo de redes neurais executa notavelmente melhor quando comparado com as outras abordagens e com os modelos de regressão múltipla. Regressão múltipla executou pobremente em todos os experimentos, com exceção daquele realizado utilizando o conjunto de dados 2, que é constituído de dados de teste bem comportados e que todos os outros modelos também tiveram seu melhor desempenho.

Para cada um dos experimentos, foram comparados os esforços estimados e os observados. Por exemplo, os resultados das estimativas utilizando a abordagem de regressão múltipla para o conjunto 1 mostram que a magnitude média do erro relativo MMRE é 1372. Enquanto que o valor de MMRE para as estimativas utilizando redes neurais, para este mesmo conjunto de dados de teste é igual a 506. Ao comparar esses resultados confirma-se que para projetos com valores maiores de KDSI o melhor ajuste é fornecido pelo modelo de redes neurais.

Tabela 5. Uma comparação de abordagens de estimativa de esforço

	Conjunto1		**Conjunto2**		**Conjunto3**		**Conjunto4**		**Conjunto5**		**Conjunto6**	
	MMRE	**R^2**	**MMRE**	**R^2**	**MMRE**	**R^2**	**MMRE**	**R^2**	**MMRE**	**R^2**	**MMRE**	**R^2**
Redes Neurais	506	0,89	278	0,89	353	0,46	384	-0,12	559	-0,05	278	0,87
Regressão Múltipla	1372	0,14	354	0,33	865	0,55	843	- 0,12	1526	-0,01	706	0,62
Regressão Simples	335	0,93	231	0,42	358	0,77	288,5	0,70	795	0,87	291	0,49
	MMRE						**R^2**					
APF	103						0.58					
SLIM	772						0.89					
COCOMO básico	610						0.70					

Porém, pode ser visto que alguns dos resultados obtidos com regressão múltipla para projetos menores predizem esforço negativo. Isto torna óbvio que estimativas são todas sujeitas a erros e devem ser interpretadas com cautela. É interessante notar que os resultados de regressão múltipla sobre estes conjuntos de dados executaram mais pobremente quando comparado com modelos anteriores utilizando somente uma única variável dependente: KDSI. Em termos de MMRE, modelos de redes neurais executam notavelmente melhor que a regressão múltipla e levemente melhor que regressão simples. Como pode ser visto na tabela acima, redes neurais executa bem e certamente melhor que a regressão sobre a maioria dos pontos. O ajuste de regressão para projetos

pequenos é pior que aqueles alcançados através de redes neurais. Isto não é surpreendente quando se considera as limitações de regressão linear.

Este experimento ilustra dois pontos. Por um lado, nenhum dos modelos executa particularmente bem para estimativas de esforço de desenvolvimento de software, particularmente sobre a dimensão MMRE. Mas, por outro lado, a abordagem de redes neurais é competitiva com modelos tradicionais. Em geral, embora MMRE seja alto no caso de todos os modelos, um valor alto de R^2 sugere que ao calibrar um modelo de predição em um novo ambiente, o modelo de predição ajustado pode ser usado com segurança. Considerando-se a dimensão R^2, o método de redes neurais fornece um ajuste significativo para os dados.

6. Conclusão

Este artigo compara o método de redes neurais com abordagens tradicionais para estimativa de esforço de software. Vários experimentos foram realizados, aplicando-se redes neurais e análise de regressão múltipla sobre o conjunto de dados COCOMO de Boehm, para estimar o esforço a partir do tamanho do software e de outros atributos multiplicadores de esforço. Os resultados das estimativas de redes neurais se comparam favoravelmente com aqueles obtidos a partir da regressão múltipla e até mesmo da regressão linear simples para estimativa de esforço a partir do tamanho [Barcelos Tronto et al. 2006].

De uma maneira geral, os resultados obtidos nesta pesquisa mostram que os modelos de redes neurais tiveram um desempenho melhor que os modelos de regressão múltipla para modelar as complexidades envolvidas no domínio de estimativa de esforço de software. A rede neural executou melhor que os modelos de regressão sobre estes conjuntos de dados e pode-se saber porque isso acontece. Sabe-se que o tamanho do software (dado pela variável KDSI) é a variável com maior significância na estimativa de esforço. Considerando o conjunto de dados COCOMO, existem duas observações com valores de esforço muito grandes, que estão fora de todas as proporções de seus tamanhos; além disso, existe um valor de esforço que é pequeno em relação ao tamanho do software. Assim, uma função linear de tamanho não teria muito sucesso para realizar estimativas para tais valores. Por outro lado, uma tentativa de resolver estes *outliers* poderia influenciar negativamente na precisão de regressão para realizar estimativas para outras observações. Uma vez que redes neurais não são limitadas a funções lineares, elas podem lidar melhor com observações que estão fora da reta idealizada.

O melhor dos modelos de regressão múltipla, calibrados nesta pesquisa teve um MMRE igual a 354 e o pior teve um MMRE igual a 1526. Já os modelos de redes neurais se mostraram capazes de capturar efetivamente os parâmetros que influenciam no esforço de desenvolvimento, sendo que o melhor modelo de redes neurais tem um MMRE igual a 277 e o pior tem um MMRE igual a 559. As variáveis categóricas do conjunto de dados COCOMO (todas aquelas da Tabela 1, exceto KDSI), consideradas nesta pesquisa, podem ser responsáveis por tal variação. Acredita-se que resultados melhores possam ser encontrados se for realizado um pré-processamento mais rigoroso dos dados, por exemplo, utilizando-se de recursos de inteligência artificial para realizar a transformação de dados e outras técnicas de seleção de atributos, análise de resíduos, dentre outros.

Possivelmente, um modelo de regressão se mostre mais vantajoso quando é utilizado um conjunto de dados mais homogêneo, sem *outliers.* Mas, rede neural também executa melhor sobre tais conjuntos de dados.

Embora a abordagem de redes neurais tenha demonstrado ter vantagens significativas em certas circunstâncias, ela não substitui regressão e deve ser considerada como outra poderosa ferramenta para ser usada na calibração de modelos de esforço de software. Conclui-se que existe um forte relacionamento entre o sucesso de uma técnica particular e o tamanho do conjunto de treinamento, a natureza da função custo e das características do conjunto de dados (*outliers*, colinearidade, número de características, dentre outras) e que a melhor técnica pode não ser a idéia certa a seguir. Essa premissa é investigada neste trabalho, em que uma variedade de experimentos revela que existe uma considerável variação no desempenho destes sistemas quando a natureza dos dados históricos muda. Por exemplo, para alguns dos conjuntos de dados apresentados neste artigo, a regressão linear simples tem melhor desempenho que os modelos de redes neurais para o mesmo conjunto de dados.

Conseqüentemente, novos experimentos estão sendo conduzidos de forma a combinar as técnicas de redes neurais e de regressão para treinar e testar modelos de predição de esforço de software, sobre outros conjuntos de dados. Por exemplo, sobre a base de dados ISBSG (*International Software Benchmarking Standard Group*), que é constituída de um conjunto de dados contendo informações de desenvolvimento de projetos, resultantes do uso de técnicas mais modernas de desenvolvimento de software. O objetivo é melhorar o desempenho do modelo de redes neurais obtido neste trabalho e obter um modelo que possa ser utilizado com segurança no desenvolvimento de software.

Referências

Agarval, R. (2001). Estimating software projects, Software Engineering Notes, v.26, p. 60-57.

Albrecht, A.J. (1979). Measuring application development productivity, Proc. IBM Application Development Symposium, p. 83-92.

Albrecht, A.J. and Gaffney, J.R. (1983). Software function, source lines of code and development effort prediction: a software science validation, IEEE Transactions on Software Engineering, v. 9, n. 6, p. 639-648.

Angelis, L. and Stamelos, I. (2000). A simulation tool for efficient analogy based cost estimation, Empirical Software Engineering, n.5, 35-68.

Barcelos Tronto, I. F., Simões da Silva, J.D. and Sant'Anna, N. (2006). Comparison of Artificial Neural Network and Regression Models in Software Effort Estimation, In: Symposium Brazilian of Neural Networks, Ribeirão Preto, Brazil (submitted).

Bisio, R. and Malabocchia, F. (1995). Cost estimation of software projects through case base reasoning, 1st Intl. Conf. on Case-Based Reasoning Research & Development, Springer-Verlag, p.11-22.

Boehm, B.W. (1981). *Software engineering economics*, Prentice-Hall, Englewood Cliffs, NJ.

Boehm, W., Horowitz, E., Madachy, R., Reifer, D., Clark, B.K., Steece, B., Brown, A.D. and Abts, C. (2000). *Software Cost Estimation with COCOMOII,* Prentice-Hall.

Briand, L.C., Langley, T. and Wieczorek, I. (2000). A replicated assessment and comparison of common software cost modeling techniques, Pro. ICSE 2000, Limerick, Ireland, 377-386.

Briand, L.C. and Wieczorek, I. (2002). Software resource estimation, Encyclopedia of Software engineering, n.2, p. 1160-1196.

Finnie, G.R. Witting, G.E. and Desharnais, J-M. (1997). A comparison of software effort estimation techniques: Using function points with neural networks, case-based reasoning and regression models, Journal of Systems and Software, 39, 281-289.

Gray, A.R. and MacDonell, S.G. (1997). A comparison of model building techniques to develop predictive equations for software metrics. Information and Software Technology, 39, 425-437.

Gray, A.R., MacDonell, S.G. and Shepperd, M. (1999). Factors systematically associated with errors in subjective estimates of software development effort: the stability of expert judgment, Proc. IEEE 6th Metrics Symposium.

Hasting, T.E. and Sajeev, A.S.M. (2001). A vector based approach to software size measurement and effort estimation," IEEE Transactions on Soft. Engineering, v. 27, n.4.

Jeffery, R., Ruhe, M. and Wieczorek, I. (2001). Using public domain metrics to estimate software development effort, Proc. IEEE 7th Metrics Symposium, London, UK, 16-27.

Jones, C. 1986. *Estimating Software Costs,* McGraw-Hill.

Karunanitthi, N., Whitley, D. and Malaiya, Y.K. (1992). Using neural networks in reliability prediction, IEEE Software, v. 9, n.4, p.53-59.

Kemerer, C.F. (1987). An empirical validation of software cost estimation models, Comunication of ACM, v.30, p.416-429.

Khoshgoftaar, T. M., Allen, E.B. and Xu, Z. (2000). Predicting testability of program modules using a neural network, Proc. 3rd IEEE Symposium on Application-Specific Systems and Sof. Eng. Technology, p. 57-62.

Kitchenham, B. (1998). A procedure for analyzing unbalanced datasets, IEEE Transactions on Software Engineering, 24, 278-301.

Kok, P., Kitchenham, B.A. and Kirakowski, J. (1990). The MERMAID approach to software cost estimation, Espirit Technical Week.

Kumar, S. Krishna, B.A. and Satsangi, P.S. (1994). Fuzzy systems and neural networks in software engineering project management. Journal of Applied Intelligence, 4: 31-52.

Lai, R. and Huang, S. (2003). A model for estimating the size of a formal communication protocol specification and its implementation, IEEE Transaction on Software Engineering, v.29, n. 1, p. 46-62.

MCT - Ministerio da Ciência e Tecnologia, (2001). Qualidade e Produtividade no setor de software," In: http://www.mct.gov.br/Temas/info/Dsi/Quali2001/2001Tab40.htm, Tabela 40 – Práticas de Engenharia de Software no Desenv. e Manutenção de Software.

Myrtveit, I., Stensrud, E. and Shepperd, M. (2005). Reliability and validity in comparative studies of software prediction models, IEEE Transaction on Soft. Engineering, v.31, n. 5, p. 46-62.

Putnam, L.H.(1978). A general empirical solution to the Macro Sizing and Estimating Problem, IEEE Transaction on Software Engineering, 4, 345-361.

Samson, B., Ellison, D. and Dugard, P. (1997). Software cost estimation using albus perceptron (CMAC), Information and Software Technology, v.39, p. 55-60.

Selby, R.W. and Porter, A.A. (1998). Learning from examples: generation and evaluation of decision trees for software resource analysis. IEEE Trans on Software Engineering, 14, 1743-1757.

Sentas, P., Angelis, L., Stamelos, I. and Bleris, G. (2005). Software productivity and effort prediction with ordinal regression, Journal Information and Software Technology, n. 47, p.17-29.

Shepeerd, M.J., Shofield, C. and Kitchenham, B. (1996). Effort Estimation Using Analogy. Proc. ICSE-18, Berlin.

Shepperd, M. and Schofield, C. (1997). Estimating software project effort using analogies, IEEE Transactions on Software Engineering, v.23, n.12, p.736-743.

Srinivazan, K. and Fisher, D. (1995). Machine learning approaches to estimating software development effort, IEEE Transactions on Software Engineering, v.21, n.2, p.126-137.

Vicinanza, S., Prietula, M.J. and Mukhopadhyay, T. (1990). Case-based reasoning in software effort estimation," In Proc.11th Int. Conf. Info. Syst, p.149-158.

Zhong, S., Khoshgoftaar, T.M. and Seliya, N. (2004). Analysing software measurement data with clustering techniques, IEEE Intelligent Systems, p.20-27.

Witting, G. and Finnie, G. (1994). Using artificial neural networks and function points to estimate 4GL software development effort, Journal Information and Systems, v. 1, n.2, p. 87-94.

Witting, G. and Finnie, G. (1997). Estimating software development effort with connectionist models, Information and Software Technology, v. 39, p. 369-476.