Anais do IX Congresso Brasileiro de Redes Neurais / Inteligência Computacional (IX CBRN)
Ouro Preto 25–28 de Outubro de 2009

Uma Abordagem Morfológico-Posto-Linear para a Estimativa de Custo de Desenvolvimento de Software

Ricardo de A. Araújo[1,2]
[1]Information Technology Department, [gm][2] Intelligent Systems, Campinas, SP, Brazil
[2]Departamento de Sistemas e Computação, Universidade de Pernambuco, Recife, PE, Brasil.
ricardo@gm2.com.br
Adriano L. I. de Oliveira[2,3] **e Sergio C. B. Soares**[2,3]
[3]Centro de Informática, Universidade Federal de Pernambuco, Recife, PE, Brazil
{alio,scbs}@cin.ufpe.br

Abstract – Neste trabalho uma abordagem morfológico-posto-linear é apresentada para solucionar o problema de Estimativa de Custo de Desenvolvimento de Software (ECDS). Esta consiste de um modelo morfológico híbrido, que é uma combinação linear entre um operador não linear Morfológico-Posto (MP) e um operador linear de Resposta Finita ao Impulso (RFI), referido como filtro Morfológico-Posto-Linear (MPL). Um método de gradiente descendente para ajustar os parâmetros do filtro MPL (processo de aprendizagem), utilizando o algoritmo *Least Mean Squares* (LMS), e uma abordagem sistemática para superar o problema da não diferenciação do operador MP, são utilizadas para otimizar a robustez numérica do algoritmo de treinamento. Além disso, uma análise experimental é conduzida com a abordagem proposta utilizando a base de dados NASA, através de duas métricas e uma função de avaliação, para mostrar o desempenho acurado da abordagem proposta, e os resultados obtidos são comparados com os modelos recentemente apresentados na literatura.

Keywords: Filtro Morfológico-Posto-Linear, Morfologia Matemática, Operadores Morfológicos, Estimativa de Custo de Desenvolvimento de Software, Previsão, Engenharia de Software.

1 Introdução

Uma grande quantidade de softwares são desenvolvidos para empresas, indústrias e para a sociedade em geral, onde devido à demanda constantemente crescente, é necessário produzir softwares de alta qualidade, em tempo e dentro do orçamento de forma a garantir competitividade. O planejamento e o gerenciamento de projetos tem exigido cada vez mais a atenção do gerente de projetos, haja vista que sua ausência na fase anterior ao início do projeto implica em problemas significativos, como o tempo de conclusão do software, custo fora do esperado, desempenho inadequado, dentre outros [1].

De acordo com o *Standish Group's Chaos*, uma grande quantidade de projetos de software (em torno de 66%) são entregues com atraso, fora de orçamento e não são concluídos, onde a principal causa de tais problemas é devido a falha na Estimativa de Custo de Desenvolvimento de Software (ECDS) [2]. Portanto, uma metodologia acurada para prever de forma ótima tal custo é extremamente necessária. Assim, o ECDS representa uma importante área de pesquisa e uma tarefa muito relevante no planejamento e gerência de software [1, 3, 4].

Recentemente, métodos de aprendizagem de máquina têm sido aplicados com sucesso para solucionar problemas de estimação (previsão). Um trabalho interessante foi apresentado por Oliveira [1], que investigou a aplicação dos modelos *Linear Regression* (LR), *Support Vector Regression* (SVR) e redes neurais *Radial Basis Functions* (RBF) para solucionar o problema ECDS. Também, vale mencionar o trabalho de Braga et al. [4, 5], que investigou abordagens de aprendizagem de máquina para o ECDS. Tronto et al. [6] apresentou uma investigação de Redes Neurais Artificiais (RNAs), em particular as redes *MultiLayer Perceptron* (MLP), para solucionar o problema de ECDS. Elish [7] apresentou um método de previsão otimizado do custo do projeto de software utilizando Multiple Additive Regression Trees (MART). Koch e Mitlohner [8] desenvolveram uma nova abordagem para o ECDS utilizando *voting rules*.

Operadores não lineares tem sido extensamente aplicados na área de processamento de sinais. Uma classe importante de sistemas não lineares é baseada na Morfologia Matemática (MM) [9]. Operadores morfológicos são transformações não lineares que modificam as características geométricas dos sinais [9], sendo relacionadas com as operações básicas da teoria dos conjuntos e a geometria integral. Este tipo de operadores empregam sequências específicas de transformações de vizinhança a fim de medir características geométricas útil dos sinais [10]. Muitos trabalhos tem focado no projeto de sistemas morfológicos, mostrando que a MM pode solucionar com sucesso problemas complexos não lineares [11].

Neste trabalho uma abordagem morfológico-posto-linear é apresentada para solucionar o problema ECDS. Esta consiste de um modelo morfológico híbrido, composto de uma combinação linear entre um operador não linear Morfológico-Posto (MP) [12] e um operador linear de Resposta Finita ao Impulso (RFI) [11], referido como filtro Morfológico-Posto-Linear (MPL) [11]. Um método de gradiente descendente para ajustar os parâmetros do filtro MPL (processo de aprendizagem), utilizando o algoritmo *Least Mean Squares* (LMS) [11], e uma abordagem sistemática para superar o problema da não diferenciação do operador MP, são utilizados para otimizar a robustês numérica do algoritmo de treinamento do filtro MPL.

Além disso, uma análise experimental é conduzida com a abordagem MPL proposta utilizando a base de dados NASA [1], onde duas métricas de desempenho relevantes e uma função de avaliação são utilizadas para medir o desempenho do modelo proposto, e os resultados obtidos são comparados aos resultados previamente publicados na literatura.

2 O Problema da Estimativa de Custo de Desenvolvimento de Software

O problema da Estimativa de Custo de Desenvolvimento de Software (ECDS) consiste em construir uma base de dados (composta por produtos, processos e recursos) e quantificar seus atributos subjetivos em valores. Desta forma, o termo estimativa (ou previsão) é aplicado quando se usa para prever o valor de um atributo no tempo futuro, onde no caso particular da ECDS, o atributo a ser predito é o custo [4, 5]. Um exemplo típico de ECDS é utilizar atributos que caracterizam o projeto de software para prever (estimar) o custo, em programadores por mês, para concluir o desenvolvimento do software, e consequentemente possibilitar a previsão do tempo necessário para entregar o projeto [1].

Portanto, a fim de atender às novas demandas de desenvolvimento, manutenção e customização de software, a equipe do projeto tem que estimar quanto tempo é necessário para concluir o projeto do software e do custo para a empresa. De acordo com Oliveira [1], os principais fatores de risco do projeto do software é o cronograma e o esforço (custo) para terminá-lo, onde as particularidades dos requisitos do projeto do software, da equipe do projeto e da tecnologia empregada dificulta o processo de estimativa de custo.

Segundo estas e outras peculiaridades de cada projeto, verifica-se que a estimativa dos custos e do tempo de desenvolvimento de software só é possível estimar com precisão quando o projeto estiver concluído [1, 4, 5]. No entanto, é necessário realizar estimativas antes do término do projeto, onde há um grande número de técnicas e métodos que podem ser empregadas para estimar essas variáveis. Desta forma, este trabalho tenta prever (estimar), o custo de desenvolvimento de software utilizando uma abordagem morfológico-posto-linear.

3 Filtro Morfológico-Posto-Linear (MPL)

3.1 Preliminares

Definição 1 – Função de Posto: a r-ézima função de posto do vetor $\underline{t} = (t_1, t_2, \ldots, t_n) \in \mathbb{R}^n$ é o r-ézimo elemento do vetor $\underline{t}$ ordenado de forma decrescente ($t_{(1)} \geq t_{(2)} \geq \ldots \geq t_{(n)}$). Esta é denotada por [11]

$$\mathcal{R}_r(\underline{t}) = t_{(r)}, \quad r = 1, 2, \ldots, n. \tag{1}$$

Por exemplo, dado o vetor $\underline{t} = (3, 0, 5, 7, 2, 1, 3)$, sua 4-ézima função de posto é $\mathcal{R}_4(\underline{t}) = 3$.

Definição 2 – Função de Amostra Unitária: a função de amostra unitária é dada por [11]

$$q(v) = \begin{cases} 1, & \text{se } v = 0, \\ 0, & \text{caso contrário.} \end{cases} \tag{2}$$

em que $v \in \mathbb{R}$.

Aplicando a função de amostra unitária a um vetor $\underline{v} = (v_1, v_2, \ldots, v_n) \in \mathbb{R}^n$, leva a um vetor de função de amostra unitária ($Q(\underline{v})$), dado por [11]

$$Q(\underline{v}) = [q(v_1), q(v_2), \ldots, q(v_n)]. \tag{3}$$

Definição 3 – Vetor Indicador de Posto: o r-ézimo vetor indicador de posto $\underline{c}$ de $\underline{t}$ é dado por [11]

$$\underline{c}(\underline{t}, r) = \frac{Q\left((z \cdot \underline{1}) - \underline{t}\right)}{Q\left((z \cdot \underline{1}) - \underline{t}\right) \cdot \underline{1}'}, \tag{4}$$

em que $z = \mathcal{R}_r(\underline{t})$, $\underline{1} = (1, 1, \ldots, 1)$ e o símbolo $'$ representa a transposição.

Por exemplo, dado o vetor $\underline{t} = (3, 0, 5, 7, 2, 1, 3)$, seu 4-ézimo vetor indicador de posto é $\underline{c}(\underline{t}, 4) = \frac{1}{2}(1, 0, 0, 0, 0, 0, 1)$.

Definição 4 – Função de Posto Suave: a r-ézima função de posto suave é dada por [11]

$$\mathcal{R}_{r,\sigma}(\underline{t}) = \underline{c}_\sigma(\underline{t}, r) \cdot \underline{t}', \tag{5}$$

com

$$\underline{c}_\sigma(\underline{t}, r) = \frac{Q_\sigma\left((z \cdot \underline{1}) - \underline{t}\right)}{Q_\sigma\left((z \cdot \underline{1}) - \underline{t}\right) \cdot \underline{1}'}, \tag{6}$$

em que c_σ é uma aproximação da função de posto $\underline{c}$ e $Q_\sigma(\underline{v})$=$[q_\sigma(v_1), q_\sigma(v_2), \ldots, q_\sigma(v_n)]$ é uma função de impulso suave (onde $q_\sigma(v)$ é do tipo $sech^2(v/\sigma)$ ou $exp\left[-\frac{1}{2}(v/\sigma)^2\right]$) e $\sigma \geq 0$ é uma parâmetro de escala.

O termo $\underline{c}_\sigma$ é uma aproximação do vetor indicador de posto $\underline{v}$. Utilizando ideias da teoria dos conjuntos difusos, $\underline{c}_\sigma$ pode também ser interpretada como um vetor de função de pertinência [11]. Por exemplo, se o vetor $\underline{t} = (3, 0, 5, 7, 2, 1, 3)$, $q_\sigma(v) = sech^2(\frac{v}{\sigma})$ e $\sigma = 0.5$ então sua 4-ézima função de posto suave é

$$\underline{c}_\sigma(\underline{t}, 4) = \tfrac{1}{2}(0.9646, 0, 0.0013, 0, 0.0682, 0.0013, 0.9646),$$

enquanto sua função de posto é $\underline{c}(\underline{t}, 4) = \frac{1}{2}(1, 0, 0, 0, 0, 0, 1)$.

3.2 Definição do Filtro MPL

Definição 5 – Filtro MPL: Seja $\underline{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ um sinal de entrada sobre um n-ézimo ponto sobre uma janela deslizante e seja y a saída do filtro. Então, o filtro MPL é definido como um sistema invariante ao deslocamento com regra de transformação local do tipo $\underline{x} \rightarrow y$, que é dada por [11]

$$y = \lambda\alpha + (1 - \lambda)\beta, \tag{7}$$

com

$$\alpha = \mathcal{R}_r(\underline{x} + \underline{a}) = \mathcal{R}_r(x_1 + a_1, x_2 + a_2, \ldots, x_n + a_n), \tag{8}$$

e

$$\beta = \underline{x} \cdot \underline{b}' = x_1 b_1 + x_2 b_2 + \ldots + x_n b_n, \tag{9}$$

em que $\lambda \in \mathbb{R}$, $\underline{a}$ e $\underline{b} \in \mathbb{R}^n$. Os termos $\underline{a} = (a_1, a_2, \ldots, a_n)$ e $\underline{b} = (b_1, b_2, \ldots, b_n)$ representam os coeficientes do filtro MP e os coeficientes do filtro linear RFI, respectivamente. O termo $\underline{a}$ é geralmente referido como "elemento estruturante" porque para $r = 1$ ou $r = n$ o filtro posto se torna uma dilatação ou erosão morfológica, respectivamente, por uma função estruturante igual a $\pm\underline{a}$ sobre o seu suporte [11].

3.3 Algoritmo de Treinamento do Filtro MPL

Pessoa and Maragos [11] mostraram que o principal objetivo no projeto do filtro MPL é especificar um conjunto de parâmetros $(\underline{a}, \underline{b}, r, \lambda)$ de acordo com os requerimentos do projeto. Entretanto, em vez de se utilizar o parâmetro posto inteiro r diretamente na definição das equações do filtro MPL (7-9), eles argumentaram que é possível trabalhar com uma variável real ρ implicitamente definida através da seguinte reescala [11]

$$r = round\left(n - \frac{n-1}{exp(-\rho)}\right), \tag{10}$$

onde $\rho \in \mathbb{R}$, n é a dimensão do vetor do sinal de entrada $\underline{x}$ sobre a janela deslizante e round$(\cdot)$ denota a operação de truncamento simétrica tradicional. Desta forma, o vetor de peso a ser utilizado para a tarefa de projeto do filtro é definida por [11]

$$\underline{w} \equiv (\underline{a}, \underline{b}, \rho, \lambda). \tag{11}$$

O projeto do filtro MPL é visto como um processo de aprendizagem, onde seus parâmetros são iterativamente ajustados. A abordagem usual para ajustar adaptativamente o vetor $\underline{w}$, e consequentemente projetar o filtro, é definir uma função de custo $J(\underline{w})$, estimar seu gradiente $\nabla J(\underline{w})$, e atualizar o vetor $\underline{w}$ através da formula iterativa

$$\underline{w}(i+1) = \underline{w}(i) - \mu_0 \nabla J(\underline{w}), \tag{12}$$

onde $\mu_0 > 0$ (geralmente referenciado como tamanho do passo) e $i \in \{1, 2, \ldots\}$. O termo μ_0 é responsável pela regulagem entre a estabilidade e a velocidade de convergência do procedimento de aprendizagem ou ajuste dos parâmetros do filtro. A iteração da Equação 12 inicia com uma vetor inicial $\underline{w}(0)$ e finaliza quando uma condição de parada é encontrada. Esta abordagem é conhecida como método do gradiente descendente [11].

A função de custo J deve refletir a qualidade da solução encontrada pela configuração dos parâmetros do sistema. Uma função de custo J, por exemplo, pode ser qualquer função de erro, como

$$J[\underline{w}(i)] = \frac{1}{M} \sum_{k=i-M+1}^{i} e^2(k), \tag{13}$$

onde $M \in \{1, 2, \ldots\}$ é um parâmetro de memória e $e(k)$ é o erro instantâneo, dado por

$$e(k) = d(k) - y(k), \tag{14}$$

onde $d(k)$ e $y(k)$ são a saída desejada e a saída do filtro, respectivamente, para a amostra de treinamento (k). O parâmetro de memória M controla a suavidade do processo de atualização. Se o sinal não possui ruído, recomenda-se utilizar $M = 1$ [11]. Entretanto, quando se usa $M > 1$, o processo de atualização tende a reduzir a influência do ruído em sinais ruidosos durante o processo de treinamento [11].

Consequentemente, o algoritmo de treinamento do filtro MPL é então definido por [11]

$$\underline{w}(i+1) = \ \underline{w}(i) + \frac{\mu}{M} \sum_{k=i-M+1}^{i} e^2(k) \frac{\partial y(k)}{\partial \underline{w}}, \tag{15}$$

em que $\mu = 2\mu_0$ e $i \in \{1, 2, \ldots\}$. Das Equações (7), (8), (9) e (11), o termo $\frac{\partial y(k)}{\partial \underline{w}}$ [11] pode ser calculado por

$$\frac{\partial y}{\partial \underline{w}} = \left(\frac{\partial y}{\partial \underline{a}}, \frac{\partial y}{\partial \underline{b}}, \frac{\partial y}{\partial \rho}, \frac{\partial y}{\partial \lambda} \right) \tag{16}$$

com

$$\frac{\partial y}{\partial \underline{a}} = \lambda \frac{\partial \alpha}{\partial \underline{a}}, \tag{17}$$

$$\frac{\partial y}{\partial \underline{b}} = (1 - \lambda)\underline{x}, \tag{18}$$

$$\frac{\partial y}{\partial \rho} = \lambda \frac{\partial \alpha}{\partial \rho}, \tag{19}$$

$$\frac{\partial y}{\partial \lambda} = (\alpha - \beta), \tag{20}$$

onde

$$\frac{\partial \alpha}{\partial \underline{a}} = \underline{c} = \frac{Q\left((\alpha \cdot \underline{1}) \ - \underline{x} - \underline{a}\right)}{Q\left((\alpha \cdot \underline{1}) \ - \underline{x} - \underline{a}\right) \cdot \underline{1}'}, \tag{21}$$

$$\frac{\partial \alpha}{\partial \rho} = 1 - \frac{1}{n} Q\left((\alpha \cdot \underline{1}) \ - \underline{x} - \underline{a}\right) \cdot \underline{1}', \tag{22}$$

onde n é a dimensão de $\underline{x}$ e $\alpha = \mathcal{R}_r(\underline{x} + \underline{a})$.

4 Métricas de Desempenho

A primeira métrica utilizada é capaz de identificar precisamente o desvio percentual do modelo, referida como *Mean Magnitude of Relative Error* (MMRE), que é definida por

$$\text{MMRE} = \frac{1}{N} \sum_{i=1}^{N} \frac{|\text{target}_i - \text{output}_i|}{\text{target}_i}. \tag{23}$$

em que N representa a quantidade de padrões, target_i e output_i representam, respectivamente, a saída desejada e o valor previsto para o padrão i.

A segunda métrica é definida como o percentual de previsões que estão no intervalo de um determinado percentual do valor real conhecido, referida como PRED, que é dada por

$$\mathrm{PRED(x)} = \frac{100}{N}\sum_{i=1}^{N} D_i, \tag{24}$$

onde

$$D_i = \begin{cases} 1, & \text{se } (\mathrm{MMRE}_i) < \frac{x}{100} \\ 0, & \text{caso contrário.} \end{cases}, \tag{25}$$

em que o termo x é igual a 25 (então a métrica PRED é referenciada por PRED(25)).

A fim de prover um modelo de previsão mais robusto, uma Função de Avaliação (FA) é construída, que é uma combinação de duas métricas: MMRE and PRED(25). A FA proposta é dada por

$$\mathrm{FA} = \frac{\mathrm{PRED}(25)}{1+\mathrm{MMRE}}. \tag{26}$$

5 Simulações e Resultados Experimentais

A base de dados de projeto de software NASA é utilizada para avaliação do modelo proposto. Esta é composta de dois atributos independentes, referidos como *Developed Lines* (DL) e *MEthodology* (ME), e um atributo dependente (que será estimado), referido como *Software Development Cost* (SDC). O atributo Dl é computado pela quantidade de linhas de código do software (o tamanho do programa que considera ambas as linhas de código fonte e o código re-usável de outros projetos, dando desta forma uma medida satisfatória do tamanho do software). Vale mencionar que o atributo DL inclui a quantidade de linhas de código fonte com os seus respectivos comentários (novas linhas com comentários) incrementados de 20% de linhas re-usáveis [1]. O atributo ME é dado de acordo com as metodologias aplicadas no desenvolvimento do software. Alguns fatores que são incluídos para determinação do atributo ME são: i) a utilização de planos de teste formais, ii) documentação formal, e iii) treinamento formal [1]. Finalmente, o SDC é calculado utilizando o critério de programadores por mês para conclusão do projeto. Note que o SDC inclui ambos os tempos de programação e gerenciamento do projeto, os quais são medidos do começo da fase de projeto até o teste de aceitação, incluindo horas de programação, gerenciamento e suporte [1].

A base de dados foi normalizada no intervalo $[0, 1]$. Foi utilizado o procedimento *Leave-One-Out Cross-Validation* (LOOCV) para determinar o erro de generalização do modelo proposto, de acordo com Shin e Goel [3]. O procedimento LOOCV é bastante simples. A base de dados é particionada em k subconjuntos (k representa a quantidade de amostras da base de dados). O conjunto de treinamento é definido por $k-1$ amostras do subconjunto, e o conjunto de teste é definido pelo k-ézimo subconjunto (utilizado para o propósito de previsão). Este processo é repetido k vezes, onde para cada experimento se utiliza um projeto de software distinto como conjunto de teste.

Para todos os experimento, a seguinte inicialização de parâmetros foi utilizada: os coeficientes dos filtros MPL ($\underline{a}$ e $\underline{b}$, respectivamente) foram normalizados no intervalo $[-0.5, 0.5]$. Os parâmetros do filtro MPL λ e ρ estão no intervalo $[0, 1]$ e $[-2, 2]$, respectivamente. O filtro MPL foi treinado via algoritmo LMS utilizando um fator de convergência $\mu = 0.01$. Os critérios de parada para o algoritmo LMS são uma quantidade máxima de épocas de (10^4), o aumento no erro de validação (*generalization loss*) ($Gl > 5\%$) e a diminuição no erro do conjunto de treinamento (*process training*) ($Pt < 10^{-6}$).

A seguir serão apresentados os resultados obtidos com o modelo MPL proposto. A fim de estabelecer um estudo de desempenho, resultados previamente publicados literatura com os modelos *Linear Regression* (LR) [1], *Support Vector Regression* (SVR) [1] (SVR-Linear e SVR-RBF), redes neurais *Radial Basis Functions* (RBF) [1], *Bagging* [4,5] and GA-SVR-RBF and GA-SVR-Linear [4,5], fora examinados no mesmo contexto e sob as mesmas condições experimentais. A Tabela 1 apresenta os resultados obtidos (do conjunto de teste) pelos modelos previamente apresentados na literatura, bem como os alcançados pelo modelo MPL proposto

De acordo com a Tabela 1, os modelos LR e RBF obtiveram desempenho semelhantes (FA$\sim$59), bem como os modelos SVR-Linear, Bagging, GA-SVR-Linear e GA-SVR-RBF também obtiveram desempenho semelhantes (FA$\sim$80), onde para a métrica PRED(25) os modelos GA-SVR-Linear e GA-SVR-RBF obtiveram um desempenho superior (PRED(25)=94.44). Em relação o modelo MPL proposto neste trabalho, é possível notar que este superou (em termos do MMRE e FA), tendo um desempenho sgnificantemente superior (0.1112 e 84.9892, respectivamente), todos os modelos analisados. Pode-se observar que para a métrica PRED(25) o modelo proposto obteve desempenho igual aos modelos GA-SVR-Linear e GA-SVR-RBF.

A principal vantagem do modelo proposto, além do seu desempenho, é a habilidade de determinar o percentual de utilização de ambas as componentes linear e não linear, bem como é bastante atrativo devido a sua simplicidade, em termos de complexidade computacional, em relação aos outros modelos analisados neste trabalho. A Tabela 2 provê uma

Tabela 1: Desempenho de previsão.

Modelo	PRED(25)	MMRE	FA
LR	72.22	0.2330	58.5726
RBF	72.22	0.1907	60.6534
SVR-Linear	88.89	0.1650	79.3004
SVR-RBF	83.33	0.1800	70.6186
Bagging	88.89	0.1639	79.3725
GA-SVR-Linear	**94.44**	0.1624	81.2457
GA-SVR-RBF	**94.44**	0.1636	81.1619
MPL	**94.44**	**0.1112**	**84.9892**

Tabela 2: Valores reais e previstos.

Projeto	Real	Previsão	PRED(25)	MMRE
1	115.8	122.65	1.0	0.0591
2	96.00	78.30	1.0	0.1844
3	79.00	74.36	1.0	0.0587
4	90.8	93.11	1.0	0.0254
5	39.6	35.03	1.0	0.1155
6	98.4	92.83	1.0	0.0566
7	18.9	18.75	1.0	0.0080
8	10.3	6.18	0.0	0.3996
9	28.5	30.94	1.0	0.0855
10	7.00	7.58	1.0	0.0834
11	9.00	10.45	1.0	0.1610
12	7.30	9.01	1.0	0.2339
13	5.00	4.44	1.0	0.1121
14	8.40	8.40	1.0	0.0005
15	98.70	106.96	1.0	0.0837
16	15.60	17.08	1.0	0.0948
17	23.90	19.38	1.0	0.1893
18	138.30	131.46	1.0	0.0495

análise detalhada, em termos de desempenho de previsão, para cada projeto de software utilizando o modelo MPL proposto, de acordo com o procedimento LOOCV.

De acordo com a Tabela 2, a métrica PRED(25), que indica se a previsão está dentro do percentual de 25% do valor real, mostra que o modelo MPL proposto é capaz de prever eficientemente o custo de 17 de 18 projetos de software. Em relação ao desvio do modelo, dado pela métrica MMRE, é possível notar que esta fica em torno de 10%, indicando que a previsão gerada tem um precisão em torno de 90%.

A Figura 1 apresenta os valores reais (linha sólida) e os valores previstos gerados pelo modelo MPL proposto (linha tracejada) para a base de dados NASA (conjunto de teste).

6 Conclusão

Neste trabalho uma abordagem Morfológico-Posto-Linear (MPL) foi apresentada para solucionar o problema de Estimativa de Custo de Desenvolvimento de Software (ECDS). Duas métricas relevantes foram utilizadas para medir o desempenho do modelo MPL proposto. Também, uma Função de Avaliação (FA) foi projetada com estas métricas a fim de otimizar a descrição da base de dados.

Uma validação experimental foi executada utilizando a base de dados NASA, mostrando toda a robustez do modelo MPL proposto através de uma comparação com os resultados previamente encontrados na literatura. Esta investigação experimental indica um melhor e mais consistente desempenho global de previsão do modelo proposto. Vale mencionar que dentre as vantagens do modelo proposto, além de seu desempenho de previsão acurado, é sua simplicidade computacional quando comparado aos outros modelos analisados neste trabalho.

Trabalhos futuros considerarão o desenvolvimento de estudos adicionais para formalizar as propriedades do modelo pro-

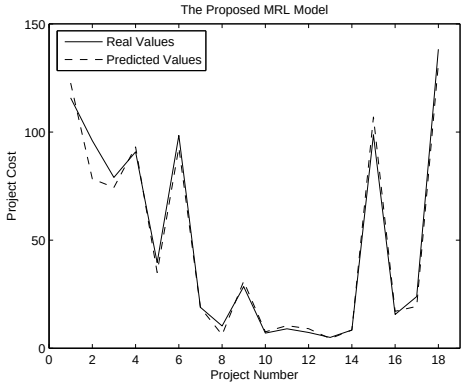

Figura 1: Resultados de previsão para a base de dados NASA (conjunto de teste).

posto, bem como a utilização de outras bases de dados para confirmação de eficiência do modelo proposto.

Agradecimentos

Este trabalho foi apoiado em parte pelo Instituto Nacional de Ciência e Tecnologia para Engenharia de Software (INES), financiado pelo CNPq e FACEPE, processos 573964/2008-4 e APQ-1037-1.03/08.

Referências

[1] A. L. I. Oliveira. "Estimation of software project effort with support vector regression". *Neurocomputing*, vol. 69, no. 13-15, pp. 1749–1753, 2006.

[2] R. Charette. "Why software fails". *IEEE Spectrum*, vol. 42, no. 9, pp. 42–49, 2005.

[3] M. Shin and A. L. Goel. "Empirical data modeling in software engineering using radial basis functions". *IEEE Trans. Software Eng.*, vol. 26, no. 6, pp. 567–576, 2000.

[4] P. L. Braga, A. L. I. Oliveira, G. H. T. Ribeiro and S. R. L. Meira. "Software effort estimation using machine learning techniques with robust confidence intervals". In *IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, 2007.

[5] P. L. Braga, A. L. I. Oliveira and S. R. L. Meira. "A GA-based feature selection and parameters optimization for support vector regression applied to software effort estimation". In *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*, pp. 1788–1792, New York, NY, USA, 2008. ACM.

[6] I. F. de B. Tronto, J. D. S. da Silva and N. Santanna. "An investigation of artificial neural networks based prediction systems in software project management". *The Journal of Systems and Software*, vol. 81, pp. 356–367, 2008.

[7] M. O. Elish. "Improved estimation of software project effort using multiple additive regression trees". *Expert Systems with Applications*, 2008.

[8] S. Koch and J. Mitlohner. "Software project effort estimation with voting rules". *Decision Support Systems*, vol. 46, pp. 895–901, 2009.

[9] P. Maragos. "A Representation Theory for Morphological Image and Signal Processing". *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 11, pp. 586–599, 1989.

[10] N. R. Harvey and S. Marshall. "The Use of Genetic Algorithms in Morphological Filter Design". *Signal Processing Image Communication*, vol. 8, pp. 55–71, 1996.

[11] L. F. C. Pessoa and P. Maragos. "MRL-Filters: A General Class of Nonlinear Systems and Their Optimal Design for Image Processing". *IEEE Transactions on Image Processing*, vol. 7, pp. 966–978, 1998.

[12] P. Salembier. "Adaptive rank order based filters". *Signal Process.*, vol. 27, no. 1, pp. 1–25, 1992.