ESTUDO PARA A CALIBRAÇÃO DE PARÂMETROS DE ALGORITMOS DE APRENDIZAGEM DE MÁQUINA UTILIZADOS NO CONTEXTO DE UMA ABORDAGEM PARA OTIMIZAÇÃO DE ARQUITETURA DE LINHA DE PROJETO DE SOFTWARE

Caio Vieira Arasaki (PIC/UEM), Aline Maria Malachini Miotto Amaral (Orientadora).
E-mail: ra127513@uem.br.

Universidade Estadual de Maringá, Centro de Tecnologia, Maringá, PR.

Área e subárea do conhecimento: Ciências Exatas e da Terra / Metodologia e Técnicas da Computação.

Palavras-chave: calibração de parâmetros; arquitetura de software; aprendizagem de máquina.

RESUMO

A Arquitetura de Linha de Produto (PLA) é um dos mais importantes artefatos de uma Linha de Produtos de Software. O projeto de PLA pode ser formulado como um problema de otimização interativo com muitos fatores conflitantes. Incorporar as preferências dos Decisores (DMs) durante o processo de busca pode ajudar os algoritmos a encontrar soluções mais adequadas para seus perfis. Abordagens interativas permitem ao DM avaliar soluções, orientando a otimização de acordo com suas preferências. No entanto, isto traz à tona problemas de fadiga humana causados pela quantidade excessiva de interações e soluções a avaliar. Usualmente para prevenir este problema é limitado o número de interações e soluções avaliadas pelo DM. Modelos de Machine Learning (ML) também foram utilizados para aprender a avaliar soluções de acordo com o perfil do DM e substituí-las após algumas interações. A seleção de *features* desempenha uma tarefa essencial, pois *features* não relevantes e/ou redundantes usados para treinar o modelo de ML podem reduzir a acurácia e a compreensibilidade das hipóteses induzidas pelos algoritmos de ML. Este trabalho tem como objetivo selecionar *features* de um modelo de ML usado para prevenir a fadiga humana em uma abordagem de design de PLA baseada em pesquisa interativa. Foram aplicados quatro seletores e através dos resultados foi possível reduzir 30% das *features*, obtendo uma acurácia de 99%.

INTRODUÇÃO

De acordo com Harman, M. e Jones, B. (2001) abordagens de Engenharia de Software Baseada em Pesquisa (SBSE) aplicam pesquisas técnicas para gerar um conjunto de soluções que resolvam problemas complexos de Engenharia de Software com base em parâmetros pré-definidos pelos Decisores (DMs). Conhecer as preferências do DM durante o processo de busca (interativo) pode auxiliar os algoritmos de busca a encontrar soluções mais adequadas ao perfil do DM. Ferreira

e cols. (2016) indica que as soluções obtidas nas abordagens SBSE são, em alguns casos, rejeitadas pelos DMs porque muitos aspectos do problema não podem ser modelados matematicamente através de funções objetivo. Dessa forma, é necessário incorporar nessas abordagens outros aspectos inerentes ao DM, como suas preferências individuais. Para tanto, alguns pesquisadores propuseram processos interativos para permitir a inclusão das preferências do DM durante o processo de busca. Abordagens interativas de SBSE são usualmente suportadas por algoritmos evolutivos em que sua execução envolve um grande número de interações e produz um grande número de soluções a serem avaliadas em cada interação. Assim, devido ao cansaço humano, a participação dos DMs para avaliar as soluções geradas em todas as interações torna-se inviável, devendo ser utilizadas estratégias para prevenir este problema. No contexto do projeto de Arquitetura de Linha de Produto (PLA), uma abordagem de otimização interativa, baseada no MOA4PLA (Multiobjective Optimization Approach for PLA), foi desenvolvida. Para evitar o esgotamento do DM esta abordagem utiliza: um algoritmo de agrupamento; um modelo de Machine Learning (ML). Através de um algoritmo de agrupamento, as soluções são categorizadas de acordo com suas semelhanças, e o DM pode avaliar apenas uma solução por categoria. Outrossim, quanto maior o número de interações com o DM, melhores serão os resultados do processo de busca, porém mais fatigado o DM pode ficar. Dessa forma, para reduzir o número excessivo de interações, foi desenvolvido um modelo de ML para aprender o perfil do DM enquanto ele interage, e assumir seu papel nas demais interações do processo de busca. Especificamente no contexto da abordagem de design de PLA, durante as interações com o DM o modelo ML foi treinado com um conjunto de dados composto por *features* obtidas das soluções de PLA avaliadas. A seleção de *features* desempenha um papel importante no processo de ML onde seu objetivo é escolher um subconjunto das *features* originais que descrevem um conjunto de dados de acordo com critérios importantes, removendo *features* irrelevantes e redundantes. Para tanto foi realizado um experimento quantitativo utilizando quatro seletores de *features* diferentes e os resultados demonstraram que é possível reduzir o conjunto de dados utilizado para treinar e testar o modelo mantendo a acurácia dos resultados em 99%. De acordo com Sutha, K. e Tamilselvi, J. J. (2015) a própria seleção de *features* permite a redução do tempo de treinamento quando o número de *features* é minimizado. Neste contexto, o principal objetivo deste trabalho foi minimizar o número de *features* sem prejudicar a acurácia do modelo.

MATERIAIS E MÉTODOS

Este projeto, buscou responder à seguinte questão de pesquisa:: é possível reduzir o número de *features* usadas no modelo de ML sem perder sua performance? O conjunto de dados utilizado possuí arquivos de onze DMs com treze *features* cada para serem avaliadas. Utilizando a ferramenta de aprendizagem de máquina WEKA e o algoritmo K* no conjunto de dados citado anteriormente, foi performados testes utilizando quatro seletores de *features* que a ferramenta oferece: Correlation-Based Selector with Best-First Search; Correlation; Information Gain; Gain Ratio. A partir

dos resultados obtidos de cada seletor foram construídos diversos subconjuntos de *features* os quais foram avaliados por meio de quatro métricas: Acurácia (Acc); Precisão (Prec); Recall (Rec); F1-Score (F1). Ao fim foram comparadas as métricas obtidas da performance de cada subconjunto com as métricas obtidas da performance do conjunto de *features* original.

RESULTADOS E DISCUSSÃO

Em uma análise inicial dos resultados dos seletores, foi descartada uma *feature* do conjunto original, pois ela obteve pontuação nula em todos os testes, restando apenas doze *features* a serem consideradas nos demais testes. Em seguida as *features* foram ordenadas em ordem decrescente de pontuação, em que essa pontuação foi obtida somando as pontuações retornadas por cada seletor na respectiva *feature* e feita a divisão por quatro, ou seja, a quantidade de seletores. A partir dessa organização foram construídos oito subconjuntos nomeados como conjuntos A. O primeiro subconjunto foi construído a partir de uma análise considerando cinco *features* bem pontuadas em diferentes seletores. Para o segundo subconjunto foi feita a união do primeiro subconjunto à *feature* mais bem pontuada das remanescentes e nomeado como A1. O terceiro subconjunto foi construído da união do subconjunto A1 à próxima feature mais bem pontuada das remanescentes e nomeado como A2. Assim os subconjuntos subsequentes foram construídos da mesma forma incremental. Essa abordagem apresentou performance satisfatória apenas no subconjunto A7, entretanto esse subconjunto possui doze *features*, assim não satisfaz o objetivo de redução de *features*.

Subconjunto	Acc(%)	Prec(%)	Rec(%)	F1(%)
A	-2,76	-1,87	-1,29	-1,60
A1	-1,19	-0,81	-0,54	-0,67
A2	-0,28	-0,36	-0,06	-0,15
A3	-0,28	-0,23	-0,06	-0,15
A4	-0,28	-0,23	-0,06	-0,15
A5	-0,29	-0,23	-0,06	-0,15
A6	-0,28	-0,23	-0,06	-0,14
A7	0	0	0	0

Para uma segunda abordagem foi construído um subconjunto nomeado como B que contém todas as oito *features* que obtiveram pontuação diferente de nula nos testes do seletor Correlation-Based Selector with Best-First Search. Também foram construídos outros quatro subconjuntos onde foi realizada a união do subconjunto B com uma única *feature* que obteve pontuação nula nesse seletor. O subconjunto Bd obteve performance satisfatória utilizando apenas nove *features*, assim atingiu também o objetivo de redução de *features* com 30% de redução *features*.

Subconjunto	Acc(%)	Prec(%)	Rec(%)	F1(%)
B	-0,28	-0,23	-0,06	-0,15

Ba	-0,28	-0,23	-0,06	-0,15
Bb	-0,28	-0,23	-0,06	-0,15
Bc	-0,28	-0,22	-0,06	-0,14
Bd	0	0	0	0

Para uma última abordagem foram construídos subconjuntos a partir do subconjunto B e retirando uma única *feature* por vez, resultando em oito subconjuntos de sete *features* cada, satisfazendo o objetivo de redução de *features*. Entretanto eles falharam em manter a performance.

Subconjunto	Acc(%)	Prec(%)	Rec(%)	F1(%)
Ca	-2,81	-2,66	-0,71	-1,70
Cb	-0,28	-0,23	-0,06	-0,15
Cc	-0,28	-0,23	-0,06	-0,15
Cd	-0,29	-0,23	-0,06	-0,15
Ce	-1,31	-1,14	-0,35	-0,75
Cf	-1,19	-0,81	-0,54	-0,67
Cg	-0,28	-0,23	-0,06	-0,15
Ch	-0,28	-0,23	-0,06	-0,15

CONCLUSÕES

A calibração de parâmetros de algoritmos de aprendizagem de máquina utilizados no contexto de otimização de arquitetura de linha de projeto de software se mostrou uma área de estudo ampla. Utilizando diversos seletores de *features* pudemos obter dados que possibilitaram a construção de diferentes subconjuntos menores que o original com métricas semelhantes. É conclusivo que a questão levantada ao início do projeto pode ser respondida, visto que foi obtido subconjuntos que apresentam redução de *features* sem que ocorra perda de performance do modelo de aprendizagem de máquina, com destaque ao subconjunto que, com nove *features* obteve performance similar e proporcionou uma redução de 30% na quantidade de *features* a serem consideradas pelo modelo.

REFERÊNCIAS

HARMAN, M.; JONES, B. Search-based software engineering Information and Soft Technology. **ACM Computing Surveys (CSUR)**, v. 43, n. 14 p. 833-839, 2001.

FERREIRA, FN.; ARAÚJO, A. A.; NETO, A. D. B.; SOUZA, J. T. Incorporating user preferences in ant colony optimization for the next release problem. **Applied Software Computing**, v. 49, p. 1283-1296, 2016.

SUTHA, K.; TAMILSELVI, J. J.; A Review of Feature Selection Algorithms for Data Mining Techniques. **International Journal on Computer Science and Engineering**, v. 7, 2015.