Representação da Diversidade de Componentes Latentes em Exercícios de Programação para Classificação de Perfis

Márcia G. de Oliveira [1], Nátaly A. Jiménez Monroy [2], Paula Daher Ximenes [2], Elias Oliveira [1]

[1] Programa de Pós-Graduação em Informática (PPGI)
Universidade Federal do Espírito Santo
Vitória – ES – Brasil

[2]Departamento de Estatística
Universidade Federal do Espírito Santo
Vitória – ES – Brasil

clickmarcia@gmail.com, elias@acm.org

Abstract. *Using the factorial analysis technique, a set of factors or latent components can be discovered from quantitative variables to characterize programs written by students. Representing each program by a set of latent components, we call this representation of student profile. In order to classify students profiles, this paper proposes a non-random selection method based on Graph Clustering to select representative profiles from the diversity of latent components to build a training set for classification models. The results achieved demonstrate that the combination of factorial analysis with Graph Clustering improves outcomes of profiles classification.*

Keywords: *Factorial analysis, Programming Learning, Classification of Profiles, Graph Clustering.*

Resumo. *Através da técnica de análise fatorial é possível descobrir um conjunto de fatores ou componentes latentes a partir de variáveis quantitativas que caracterizam programas desenvolvidos por alunos. Ao representar cada programa por um conjunto de componentes latentes, chamamos essa representação de perfil de estudante. Com o objetivo de classificar perfis de estudantes, este trabalho propõe um método de seleção não-aleatória baseado em Clustering em Grafo para selecionar perfis representativos da diversidade de componentes latentes como conjunto de treino de modelos de classificação. Os resultados alcançados demonstram que a combinação da técnica de análise fatorial com Clustering em Grafo melhora os resultados de classificação de perfis.*

Palavras-chave: *Análise fatorial, Aprendizagem de programação, Classificação de perfis, Clustering em Grafo.*

1. Introdução

A programação de computadores é um conhecimento que demanda extensa prática de exercícios para ser de fato aprendido. No entanto, essa prática deve ser assistida por um professor de forma que seus *feedbacks* possibilitem a evolução dos estudantes em seu processo de aprendizagem. A prática assistida de programação, porém, torna-se um

desafio quando se considera o grande esforço de professores para avaliar individualmente muitos exercícios, especialmente em cursos massivos [Pieterse 2013].

Embora muitas propostas de tecnologias de avaliação automática de exercícios de programação tenham sido desenvolvidas [Oliveira and Oliveira 2015], ainda estamos longe de uma tecnologia que de fato avalie como um professor avalia.

Podemos, no entanto, criar estratégias semi-automáticas de avaliação que possibilitem qualificar a aprendizagem de programação aplicando muitos exercícios, sem necessariamente atribuir uma nota exata para cada exercício. Uma solução para isso seria, por exemplo, extrair de programas escritos por alunos, atributos que nos permitam representar e classificar perfis por níveis de aprendizagem. Nesse caso, a ideia seria estabelecer faixas de *scores* representando classes de perfis de estudantes.

Seguindo essa ideia, o trabalho de [Oliveira et al. 2014] propõe uma estratégia de mapeamento e classificação de perfis de alunos a partir de programas em Linguagem C. O mapeamento de perfis é uma representação de programas por variáveis chamadas componentes de habilidades medidas pela contagem de atributos de código-fonte e por *flags* indicadores de execução correta. Já as componentes latentes representam em fatores as relações entre as componentes de habilidades. Através de uma seleção aleatória de amostras representadas por componentes latentes, um classificador é treinado e, a partir dessa aprendizagem, classifica outras amostras por faixas de *scores* de classes.

Este trabalho estende a proposta de classificação de perfis de [Oliveira et al. 2014] ao propor um método de seleção não-aleatória de amostras de treino para classificação de perfis através da combinação das técnicas de análise fatorial e de *Clustering em Grafo*. A análise fatorial é utilizada para redução de dimensionalidade e o *Clustering em Grafo*, para a seleção de amostras que representem a diversidade de componentes latentes.

Integrado a sistemas de sistemas de apoio à prática de programação como o *PCodigo* [Oliveira et al. 2015], o método de seleção e classificação de perfis pode ser utilizado por professores para fornecer *feedbacks* mais imediatos para seus alunos.

Para apresentar o método proposto, este trabalho foi organizado conforme a ordem a seguir. Na Seção 2, descrevemos as técnicas utilizadas e os trabalhos relacionados. Na Seção 3, apresentamos a arquitetura de funcionamento da classificação de perfis. Na Seção 4, detalhamos os procedimentos, os materiais e os resultados de aplicação do método. Na Seção 5, concluimos com as considerações finais e propostas de trabalhos futuros.

2. Fundamentação Teórica

O método desenvolvido neste trabalho combina as técnicas de análise fatorial e de *Clustering em Grafo* para seleção não-aleatória de amostras de treino de modelos de classificação de perfis. A ideia desse método é baseada na proposta de representação e classificação de perfis de [Oliveira et al. 2015] e [Oliveira et al. 2014] e na estratégia de clusterização de plágios em exercícios de programação de [Moussiades and Vakali 2005].

2.1. A Análise Fatorial

A análise fatorial é uma técnica de análise multivariada que consiste na redução da dimensionalidade de um conjunto de dados, encontrando grupos homogêneos de variáveis a partir de um grande número de variáveis. Esses grupos homogêneos, também chamados

de *fatores*, são formados por variáveis muito correlacionadas entre si. O ideal é que os fatores sejam independentes uns dos outros. Portanto, a análise fatorial é uma técnica que busca explicar ao máximo a informação contida nos dados, usando a menor quantidade possível de dimensões [Oliveira et al. 2014].

Frequentemente as influências dos k fatores costumam ser divididas em *comuns* e *específicas*. Por exemplo, há fatores altamente informativos que são comuns para todas as p componentes de $\mathbf{X}$ e fatores que são específicos apenas para algumas componentes. Nesse caso, a matriz $\mathbf{X}$ pode expressar-se como

$$\mathbf{X} = \mathbf{QF} + \mathbf{U} + \mu,$$

onde $\mathbf{Q}$ é uma matriz $(p \times k)$ de coeficientes (não-aleatórios) dos fatores comuns $\mathbf{F}$ (de dimensão $(k \times 1)$), $\mathbf{U}$ é uma matriz $(p \times 1)$ dos fatores (aleatórios) específicos e μ é o vetor de médias de $\mathbf{X}$.

Assumindo que um modelo fatorial com k fatores foi encontrado razoável, isto é, a maioria das variações das p variáveis em $\mathbf{X}$ foram explicadas por k fatores fixos, podemos interpretar os fatores F_l, $l = 1,\ldots,k$ calculando suas correlações com as variáveis X_j, $j = 1,\ldots,p$, para obter a matriz $\mathbf{P}_{XF}$. Essa correlação é dada por

$$P_{XF} = D^{-1/2}Q,$$

onde $D = diag\{\sigma_{X_1X_1},\ldots,\sigma_{X_pX_p}\}$, sendo $\sigma_{X_jX_j}$ a variância de X_j. Usando essa informação podemos determinar quais das variáveis originais $X_1,\ldots,X_p$ influenciam nos fatores não observados $F_1,\ldots,F_k$ [Anderson 1984, Morrison 1990].

2.2. Clustering em Grafo

Os grafos são estruturas formadas por um conjunto de vértices e de arestas. Em um grafo, os vértices são nós ou representações de objetos e as arestas são conexões entre pares de vértices. O *Clustering em Grafo* é a tarefa de agrupar os vértices de um grafo em *clusters* de forma que haja muitas arestas dentro dos *clusters* e poucas entre eles [Schaeffer 2007].

Um corte de grafo é o particionamento dos vértices desse grafo em dois *clusters*. As arestas de corte são aquelas que possuem vértices em dois *clusters*. O mínimo corte consiste em achar um particionamento em que a soma dos pesos das arestas de corte seja o mínimo possível.

Um exemplo de *Clustering em Grafo* é apresentado na Figura 1, onde Cl_1, Cl_2 e Cl_3 são os *clusters* separados pelos cortes mínimos P_1 e P_2.

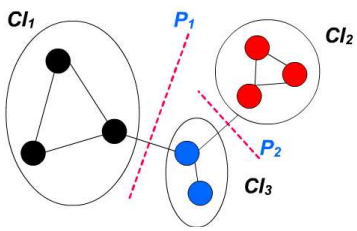

Figura 1. Clustering em Grafo

Neste trabalho, utilizamos o algoritmo de *Clustering em Grafo* do *software Cluto* [Karypis 2003]. A medida de similaridade escolhida foi o *cosseno*. Já os parâmetros de

corte utilizados foram o *edgeprune=0.7* e o *vtxprune=0.4*. O primeiro parâmetro separa certas arestas do grafo dos vizinhos mais próximos que tendem a conectar vértices pertencentes a diferentes *clusters*. Já o segundo parâmetro separa certos vértices do grafo de vizinhos mais próximos que não se agrupam em qualquer *cluster*.

2.3. Trabalhos Relacionados

O *PCodigo*, um sistema de apoio à prática assistida de programação desenvolvido por [Oliveira et al. 2015], representa perfis de estudantes a partir de programas em Linguagem C. Esses perfis são vetores cujas dimensões são formadas a partir da contagem de atributos da Linguagem C e por *flags* indicadores de compilação e de execução.

Para resolver o problema de se obter *scores* que melhor representem as classes de perfis de estudantes no *PCodigo*, [Oliveira et al. 2014] propõem um método de seleção aleatória de amostras de cada *cluster* gerado pelo algoritmo *Bissecting-kmeans* [Karypis 2003]. Em seguida, os *scores* dos demais perfis de cada *cluster* são preditos e uma classe de nível de apredizagem é associada a cada perfil.

O *PDetect* é um sistema que faz análise e clusterização de plágios em exercícios de programação [Moussiades and Vakali 2005]. Para isso, o *PDetect* recebe códigos-fontes, mapeia-os em um conjunto de *tokens* de uma linguagem de programação, identifica os pares suspeitos de plágios e os reúne em *clusters*, formados a partir de um parâmetro de corte, que é o *Coeficiente de Jaccard* [Moussiades and Vakali 2005].

O trabalho de [Moussiades and Vakali 2005] utilizou o *Clustering em Grafo* para separar agrupamentos de plágios de exercícios de programação. Seguindo essa ideia para classificar os perfis gerados pelo *PCodigo*, estendemos a proposta de [Oliveira et al. 2014] combinando a análise fatorial e o *Clustering em Grafo* para seleção não-aleatória dos conjuntos de treino dos modelos de classificação de perfis de estudantes de programação.

3. Método de Representação da Diversidade de Componentes Latentes

A Figura 2 ilustra a arquitetura do método de representação, de seleção e de classificação de perfis deste trabalho. O módulo de *Mapeamento de Perfis* recebe programas de computador em Linguagem C e os transforma em vetores cujas dimensões são chamadas de componentes de habilidades (C_i). Cada componente de habilidade é quantificada pela frequência de ocorrência de palavras reservadas, símbolos, operadores e funções da Linguagem C ou por *flags* que indicam se um programa funcionou ou não [Oliveira et al. 2014].

O módulo de *Análise Fatorial*, conforme a Figura 2, redimensiona os perfis gerados pelo módulo de *Mapeamento de Perfis* extraindo as componentes latentes através das relações entre as componentes de habilidades C_i [Oliveira et al. 2014].

No módulo de *Seleção de Diversidade*, todos os perfis representados pelas componentes latentes F_j são submetidos ao algoritmo de *Clustering em Grafo* para seleção não-aleatória de treino do modelo de classificação de perfis. Em seguida, é executada a função de *Separação Treino/Teste* que seleciona as amostras de treino do modelo de classificação e as amostras que serão classificadas por esse modelo.

O conjunto das amostras de treino, isto é, o conjunto R_D da Figura 2, é formado pelas amostras que não foram clusterizadas pelo algoritmo de *Clustering em Grafo*, isto é, pelas amostras que não se agrupam a qualquer dos *k clusters* ou cujos vértices estejam

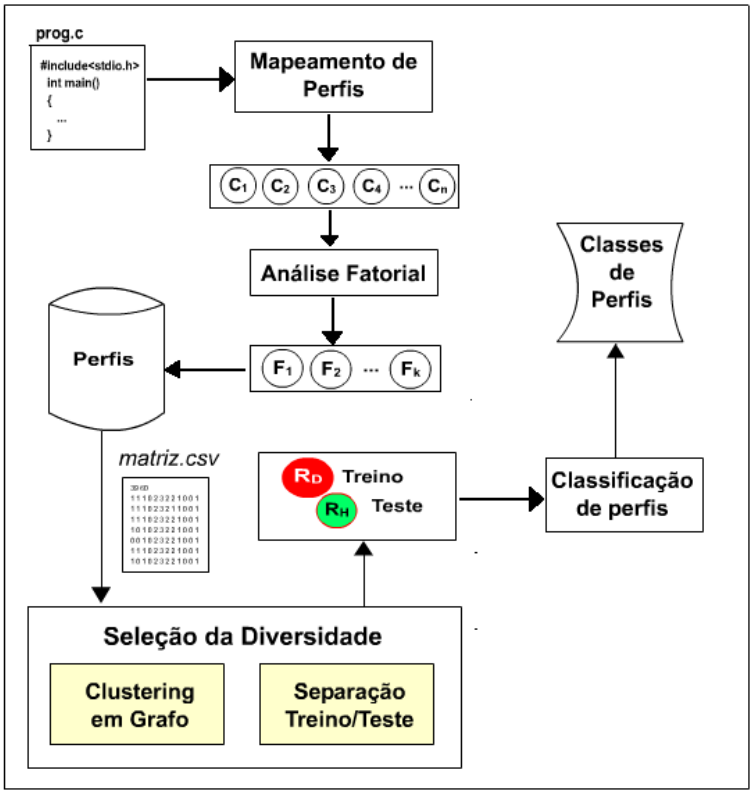

Figura 2. Mapeamento, seleção e classificação de perfis

entre dois *clusters*. O conjunto R_D representa, portanto, as amostras com maior diversidade das componentes latentes. Já o conjunto R_H das amostras a serem classificadas é formado pelos *clusters* gerados pelo algoritmo de *Clustering em Grafo*. Esses *k clusters* são agrupamentos homogêneos das amostras com similaridade acima de 70% entre si.

A saída do módulo de *Seleção de Diversidade* são dois arquivos contendo perfis de treino e de teste, respectivamente. O *Treino* é pré-classificado por um professor e é utilizado na aprendizagem do modelo de classificação. Já o *Teste* contém as amostras que serão classificadas pelo módulo de *Classificação de Perfis*.

No módulo de *Classificação de Perfis*, utilizamos modelos de regressão linear para predição dos *scores* que representam as classes de cada perfil do conjunto de teste. Os valores dos *scores* de referência das *Classes de Perfis*, em uma escala de 0 a 5, são 0.5 (ruim), 2.0 (regular), 2.75 (médio), 4.0 (bom) e 5.0 (Excelente). Dessa forma os estudantes são classificados por *scores* iguais ou próximos a esses valores.

Em resumo, a metodologia deste trabalho visa aproximar-se o máximo possível do *score* de classe atribuído pelo professor a um exercício de programação.

4. Experimentos e Resultados

A experimentação do método deste trabalho foi realizada através dos seguintes passos:

1. Separação das bases experimentais
2. Representação das bases em matrizes
3. Aplicação da análise fatorial nas bases experimentais

4. Seleção dos conjuntos de treino dos modelos de classificação
5. Classificação de Perfis
6. Apresentação dos resultados

Os procedimentos de cada um desses passos de experimentação são descritos em detalhes nas subseções a seguir.

4.1. Separação das Bases

Para a realização dos experimentos deste trabalho, utilizamos duas bases, a *Base-A* e a *Base-B*. A *Base-A* é formada por 100 exemplos de programas em Linguagem C desenvolvidos por alunos em resposta a um exercício de programação. A *Base-A* foi a mesma base utilizada nos experimentos de [Oliveira et al. 2014].

A *Base-B*, por sua vez, foi gerada em condições controladas, pois as amostras obtidas são programas em Linguagem C desenvolvidos por alunos para uma questão de prova aplicada em uma turma de programação de uma universidade.

A *Base-B* é formada por 39 amostras de programas e é considerada difícil por ser uma base pequena e com alta diversidade de soluções.

A especificação do exercício das duas bases é a mesma e consiste em construir um programa em Linguagem C para identificar o campeão e o vice-campeão de um campeonato de futebol a partir das pontuações obtidas por três times.

4.2. Representação das bases

As amostras de programas da *Base-A* e da *Base-B* foram representadas em vetores com 60 dimensões chamadas componentes de habilidades, que representam a frequência de ocorrência de palavras-chave, operadores, funções e estruturas da Linguagem C e *flags* que informam se um programa funcionou (valor 1) ou não (valor 0) [Oliveira et al. 2015]. Em seguida, os vetores da *Base-A* foram reunidos na matriz M_{100x60} e os da *Base-B*, na matriz N_{39x60}.

4.3. Aplicação da Análise Fatorial

A técnica de análise fatorial foi aplicada nas bases experimentais deste trabalho utilizando o método das componentes principais com *Rotação Varimax* e adotando como critério de decisão do número de fatores os autovalores maiores que 1 e com base no *scree-plot* (Figura 3). O processo de análise fatorial da *Base-A* e a seleção de seus doze fatores são explicados em detalhes por [Oliveira et al. 2014]. Neste trabalho, apresentamos apenas o processo de análise fatorial da *Base-B*.

Das 60 variáveis (ou componentes de habilidades) da *Base-B*, 29 tiveram valores iguais a zero em todas as amostras e nove variáveis foram utilizadas somente uma vez pelos alunos. Essas variáveis foram retiradas da análise, restando apenas 22 variáveis.

Um modelo com cinco fatores mostrou-se adequado explicando 81% da variabilidade dos dados originais. Na Tabela 1, são apresentados os autovalores e as porcentagens da variabilidade explicada e acumulada.

A Figura 3 mostra o *scree plot*, isto é, a relação entre autovalores e fatores, utilizado para determinar a quantidade de fatores. Observe que, a partir do Fator 5, o autovalor

Autovalores e Variabilidade

	Fator 1	Fator 2	Fator 3	Fator 4	Fator 5
Autovalores	8,449	4,762	1,661	1,560	1,382
Variabilidade	38,406	21,647	7,550	7,092	6,284
Variabilidade Acumulada	38,406	60,053	67,602	74,694	80,978

Tabela 1. Autovalores e porcentagem da variabilidade explicada

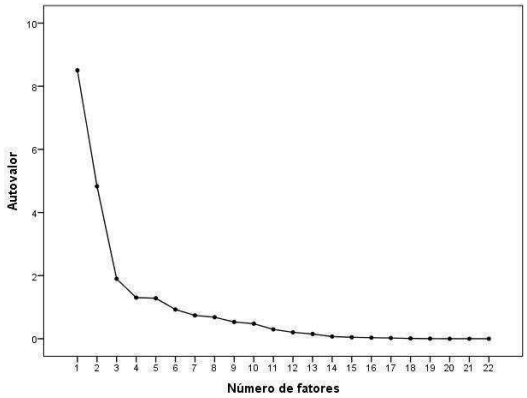

Figura 3. Scree plot para determinar a quantidade de fatores

descresce até estabilizar-se, o que mostra que o aumento do número de fatores não melhorará a variabilidade.

Com tais resultados, buscou-se identificar quais variáveis mais influenciaram em cada fator, ou seja, aquelas que possuem maiores cargas fatoriais. Dessa forma, foi possível nomear cada fator conforme os rótulos de componentes latentes da Tabela 2.

Componentes Latentes

Fator	Características principais	Componente Latente
1	*opmais* (+), *opmult* (*), *opmenos* (-)	Operação Aritmética
2	*ccase*, *cbreak*, *celse*, *opmaior* (>)	Seleção, comparação
3	*opmenor* (<), *opatrib* (=)	Comparação, Atribuição
4	*cscanf*	Entrada de dados
5	*opmenorigual* ($\geq$), *cdo*	Comparação, Repetição

Tabela 2. Análise de componentes latentes da Base-B

4.4. Seleção do Conjunto de Treino

O *Clustering em Grafo* foi aplicado nas bases experimentais em dois passos. Primeiro, obtivemos do processo de *clustering* dois conjuntos: o de amostras heterogêneas (não-clusterizadas) e o de amostras homogêneas (clusterizadas com similaridade acima de 70%). Segundo, se o processo anterior não gerasse um número satisfatório de amostras heterogêneas para formar o treino do classificador, realizamos o *re-clustering* informando ao algoritmo de *clustering* para particionar a base em dois *clusters*. O *cluster* mais heterogêneo desse particionamento é o *cluster* selecionado para treinar o classificador.

Na experimentação do método, a *Base-A*, sendo maior, foi clusterizada no primeiro passo. Já a *Base-B*, por não formar um conjunto heterogêneo no primeiro passo, foi dividida em dois *clusters* no segundo passo.

A Figura 4 apresenta os gráficos dos *clusters* 0 e 1 formados pela aplicação do *Clustering em Grafo* na *Base-B*. Nesses gráficos, cada linha contém o rótulo da representação vetorial de um exercício de programação e seu *score* de classe. As colunas são os cincos fatores ou componentes latentes da *Base-B* gerados pela análise fatorial. A cor vermelha indica valores positivos, a cor verde, valores negativos e a cor branca, valor nulo. Quanto mais intensa é a cor, maior é o valor do seu fator.

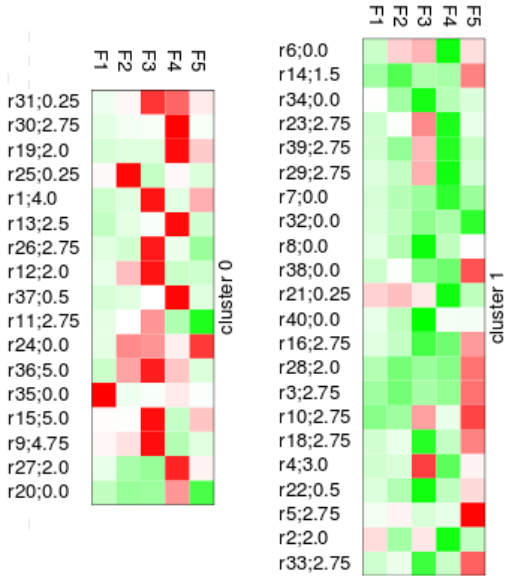

Figura 4. Clustering em Grafo da Base-B

Nos gráficos de *Clustering em Grafo* da Figura 4, observa-se que a cor vermelha predomina no *Cluster 0* e é mais presente entre os fatores F_i, o que significa que o *Cluster 0* representa melhor a diversidade das componentes latentes para formar o treino do classificador de perfis. Uma confirmação disso está na variedade de *scores* de classes nas linhas do *Cluster 0* e nos resultados da *Base-B* apresentados na Subseção 4.6.

4.5. Classificação de Perfis

A classificação de perfis foi realizada por modelos de regressão linear para obter valores próximos dos *scores* de classes. Esses modelos foram treinados com as amostras heterogêneas geradas no passo 1 ou com o *cluster* mais heterogêneo gerado no passo 2 do processo de *Clustering*.

4.6. Resultados

A Tabela 3 apresenta os resultados de aplicação do método de seleção não-aleatória de treino de classificação na *Base-A*. Sendo a *Base-A* uma base grande, o algoritmo de *Clustering em Grafo* formou o conjunto de treino com um número razoável de amostras não clusterizadas, isto é, com as amostras com maior diversidade de componentes latentes.

Os resultados da *Base-A* nas colunas *sAF* e *cAF* da Tabela 3 indicam a aplicação do método sem e com a análise fatorial, respectivamente. A linha *Qtd.Treino/Teste* informa, respectivamente, quantas amostras de cada base foram selecionadas para treinar o modelo de classificação e quantas amostras o modelo classificou automaticamente.

Para avaliar o erro de predição de *scores* da *Base-A* e da *Base-B*, utilizamos as medidas de *Erro Médio*, *Erro Mínimo* e *Erro Máximo*. Nas tabelas 3 e 4, quanto menor os valores dessas medidas, melhor é o desempenho do classificador.

Classificação de Perfis - Base-A

Avaliação	sAF	cAF
Qtd. Treino/Teste	53/47	**36/64**
Predição de scores		
Erro Médio	0.8000	**0.7954**
Erro Mínimo	0	0.0200
Erro Máximo	2.3000	**1.7800**

Tabela 3. Resultados de classificação de perfis - Base-A

Através dos valores em negrito na Tabela 3, podemos observar que as medidas de *Erro Médio* e *Erro Máximo* de predição diminuíram com a aplicação da análise fatorial.

A *Entropia* e a *Pureza* são medidas de avaliação da homogeneidade de *clusters*. Na Tabela 4, apresentamos a *Entropia* e a *Pureza* dos dois *clusters* (0 e 1) formados a partir da *Base-B*. Quanto maior a *Entropia* e menor a *Pureza*, mais heterogêneo é um *cluster*, isto é, maior é a sua diversidade de características.

Classificação de Perfis - Base-B

	Cluster 0		Cluster 1	
Avaliação	sAF	cAF	sAF	cAF
Qtd. Treino/Teste	28/11	**17/22**	11/28	22/17
Avaliação de Clusters				
Entropia	0.782	**0.7870**	0.220	0.497
Pureza	0.429	0.5880	0.909	0.545
Predição de scores				
Erro Medio	1.6990	**0.9022**	2.7142	1.9111
Erro Minimo	0	0	0	0.25
Erro Maximo	5.000	**1.9200**	5	4.75

Tabela 4. Resultados de classificação de perfis da Base-B

Na Tabela 4, os resultados da *Base-B*, que é a base mais difícil, evidenciam que a combinação da análise fatorial com o *Clustering em Grafo* para selecionar as amostras com maior diversidade de componentes latentes melhoram os resultados de classificação de perfis. Observa-se que, na Tabela 4, o *Cluster 0*, que, pelas medidas de Entropia e Pureza, é o mais heterogêneo, apresenta melhores resultados do que o *Cluster 1*.

Com a análise fatorial, apenas com 17 amostras de treino, o *Cluster 0* apresentou os melhores resultados de predição para classificar 22 amostras. Sem a análise fatorial, o número de amostras de treino aumentaria para 28 para classificar 11 amostras.

Em relação ao trabalho de [Oliveira et al. 2014], melhoramos os resultados da *Base-A* ao reduzir o erro máximo de classificação e a quantidade de amostras de treino necessárias para treinar o modelo de classificação. Além disso, o novo método tem a vantagem de realizar uma seleção não-aleatória de amostras de treino conforme a diversidade das componentes latentes.

Esses resultados nos levam a concluir que a combinação da análise fatorial com o *Clustering em Grafo* para selecionar as amostras mais heterogêneas da *Base-A* ou o

cluster mais heterogêneo da *Base-B* (Figura 4), melhora os resultados de classificação de perfis reduzindo o erro de predição dos *scores* de classes.

5. Considerações Finais

Este trabalho apresentou um método de seleção não-aleatória de treino para classificação de perfis de estudantes de programação. Os resultados mostram que a combinação das técnicas de análise fatorial e de *Clustering em Grafo*, representando a diversidade de componentes latentes, melhora os resultados de classificação perfis.

Como trabalhos futuros a partir deste, sugerimos que a estratégia de seleção não-aleatória de treino de modelos de classificação seja avaliada em outros sistemas de aprendizagem supervisionada como, por exemplo, os sistemas de recomendação.

Os estudos apresentados neste trabalho representam uma passo importante para melhorar a aprendizagem de modelos de classificação de perfis de estudantes. Dessa forma, ao selecionar um número menor de amostras representando a diversidade de perfis, reduzimos esforços de professores na atribuição de *scores* às amostras de exercícios de programação que formarão o conhecimento do classificador.

Já para a prendizagem de programação, o método proposto integrado ao sistema *PCodigo* de [Oliveira et al. 2015] oferece possibilidades de *feedbacks* mais imediatos para estudantes e uma prática de programação melhor assistida por professores.

Referências

Anderson, T. W. (1984). *An Introduction to Multivariate Statistical Analysis*. Wiley, New York, NY, second edition.

Karypis, G. (2003). Cluto - a clustering toolkit. Dept. of Computer Science, University of Minnesota.

Morrison, D. F. (1990). *Multivariate Statistical Methods*. McGraw–Hill, third edition.

Moussiades, L. and Vakali, A. (2005). Pdetect: A clustering approach for detecting plagiarism in source code datasets. *The computer journal*, 48(6):651–661.

Oliveira, M., Monroy, N., Zandonade, E., and Oliveira, E. (2014). Análise de componentes latentes da aprendizagem de programação para mapeamento e classificação de perfis. In *Anais do Simpósio Brasileiro de Informática na Educação (SBIE 2014)*, volume 25, pages 134–143.

Oliveira, M., Nogueira, M., and Oliveira, E. (2015). Sistema de apoio à prática assistida de programação por execução em massa e análise de programas. In *CSBC 2015 - Workshop de Educação em Informática (WEI)*, Recife-PE.

Oliveira, M. and Oliveira, E. (2015). Abordagens, práticas e desafios da avaliação automática de exercćios de programação. In *CSBC 2015 - DesafIE 2015*, Recife-PE.

Pieterse, V. (2013). Automated assessment of programming assignments. In *Proceedings of the 3rd Computer Science Education Research Conference on Computer Science Education Research*, CSERC '13, pages 4:45–4:56, Open Univ., Heerlen, The Netherlands, The Netherlands. Open Universiteit, Heerlen.

Schaeffer, S. E. (2007). Survey: Graph clustering. *Comput. Sci. Rev.*, 1(1):27–64.