Metodologia para Avaliar Técnicas de Redução de Protótipos: Protótipos Gerados *versus* Protótipos Selecionados

Luciano de Santana Pereira
Centro de Informática (CIn)
Universidade Federal de Permanbuco (UFPE)
Recife - Pernambuco - Brasil
Email:lsp@cin.ufpe.br

George D. C. Cavalcanti
Centro de Informática (CIn)
Universidade Federal de Permanbuco (UFPE)
Recife - Pernambuco - Brasil
Email:gdcc@cin.ufpe.br

***Resumo*—Técnicas de aprendizagem de máquina baseadas em instâncias são utilizadas em várias aplicações, por exemplo: reconhecimento de faces, de voz e de digitais, na medicina para auxiliar médicos na detecção de neoplasias, entre outras. Geralmente, essas técnicas usam grandes conjuntos de dados, exigindo um grande espaço em memória para armazenamento, além do elevado custo computacional para a classificação. Com o objetivo de minimizar esses problemas, as técnicas de redução de instâncias buscam reduzir o tamanho do conjunto de dados, escolhendo ou produzindo elementos que consigam representá-lo, reduzindo a necessidade de memória para o armazenamento do conjunto de dados, o custo computacional e minimizando a taxa de erro. Existem, atualmente, dois ramos de pesquisa que buscam a redução de instâncias: a seleção de instâncias, que faz a redução escolhendo algumas instâncias representantes de todo o conjunto de treinamento e as técnicas de geração de protótipos que buscam a redução de instâncias, produzindo novos protótipos, a partir de várias heurísticas, que irão representar todo o conjunto de treinamento. Esse processo de geração é mais demorado que o processo de seleção. Porém, observa-se na literatura que as técnicas de geração apresentam melhores resultados que as técnicas de seleção. A proposta deste trabalho é investigar se as técnicas de seleção podem obter resultados semelhantes às técnicas de geração. Para tanto, é apresentada uma metodologia para avaliar técnicas de redução de protótipos. O resultado obtido neste estudo mostra que as técnicas de seleção existentes obtiveram taxas equivalentes às técnicas de geração em 83,34% das bases utilizadas nos experimentos. Isto implica que, para as bases testadas, os protótipos gerados tinham instâncias muito próximas, no conjunto de treinamento, que poderiam substituí-los, sem a necessidade de geração de protótipos, que é um processo mais custoso que a seleção de protótipos. Os resultados obtidos sugerem que é possível desenvolver técnicas de seleção, que apresentem taxas de erro equivalentes às técnicas de geração.**

I. Introdução

A classificação de padrões é um ramo da aprendizagem de máquina que tem como objetivo classificar informações com base em informações previamente conhecidas (padrões). Podemos conceituar padrão como um conjunto de características (atributos) que definem um objeto ou um grupo de objetos. Essas características são escolhidas visando à separação das classes de modo a facilitar a tarefa do classificador. Esses padrões são, normalmente, grupos de medidas que definem pontos em um espaço multidimensional. No processo de classificação, os dados de entradas são um conjunto de registros, em que cada registro é denominado de instância ou exemplo, e suas características são denominadas de atributos. Existe também um atributo especial que é denominado rótulo da classe. Classificação é a tarefa de aprender uma **função alvo** f que mapeie cada conjunto de atributos x para um dos rótulos y pré-determinados [1]. No processo de classificação são apresentados ao classificador um **conjunto de treinamento**, que é um conjunto com instâncias cujas classes já são conhecidas, e a partir desses dados, o classificador "aprende" sobre o problema de classificação, essa fase é chamada de treinamento. Após a construção do modelo de classificação, o **conjunto de teste**, que consiste em um conjunto de registros cujas classes são desconhecidas, é submetido ao classificador. A avaliação do desempenho de um modelo de classificação é baseada nas contagens de instâncias classificadas correta e incorretamente pelo modelo.

Devido ao crescimento exponencial do volume de dados armazenados, encontramos conjuntos de dados com tamanhos em gigabyte, terabytes e até petabytes. Esse crescimento no volume de dados gera um aumento do custo computacional e, em muitos casos, a falta de memória principal para armazenar os conjuntos de dados. Para solucionar esses problemas, necessitamos reduzir o número de instâncias do conjunto de treinamento, porém essa redução deve manter ou aumentar a capacidade de generalização dos métodos de aprendizagem, eliminando dados irrelevantes ou redundantes, mas mantendo o desempenho do modelo.

Existem, atualmente, dois ramos de pesquisa que buscam a redução de instâncias: seleção de instâncias e geração de protótipos. As técnicas de seleção de instâncias utilizam como protótipos, elementos ou instâncias do próprio conjunto de treinamento, elegendo elementos que possam representar da melhor forma possível esse conjunto de treinamento. Já nas técnicas de geração, também chamadas de síntese de protótipos, os protótipos são gerados artificialmente e sua posição é calculada de forma a representar bem a classe geradora. A geração de protótipos necessita de maior tempo para processamento, devido à fase adicional de ajuste do posicionamento de cada protótipo gerado. Dessa forma, seriam pertinentes as seguintes perguntas: Em que situações devemos utilizar uma ou outra técnica? Será que, na maioria dos casos, as técnicas de seleção apresentam resultados satisfatórios na solução dos

problemas de classificação?

Dado que na literatura os melhores resultados de classificação são obtidos através da utilização de técnicas de geração de protótipos [4], deseja-se verificar se é possível que técnicas de seleção apresentem resultados iguais ou melhores às técnicas de geração. Esse trabalho propõe uma metodologia para avaliar técnicas de redução de protótipos. Dentre as técnicas de geração, utilizaremos em nosso estudo as seguintes: SGP2 [5], LVQ3 [6], RPOCNN [7], RSP3 [8], GENN [9] e ICPL2 [10]. Comprovando-se essa hipótese, poderemos desenvolver técnicas de seleção que busquem a instância ótima no conjunto de treinamento, obtendo as mesmas taxas de acerto com maior velocidade que as técnicas de geração.

Este trabalho apresenta a seguinte estrutura: na Seção II é apresentada a metodologia que será usada para avaliar a relação entre algoritmos de geração de protótipos e as instâncias selecionadas, na Seção III são apresentados os experimentos e os resultados, e na Seção IV, a conclusão.

II. Metodologia para Avaliar a Relação entre Algoritmos de Geração e Seleção de Instâncias

Dado um conjunto de treinamento Γ, os algoritmos de seleção de instâncias buscam por um subconjunto $\Phi \subset \Gamma$, reduzindo, assim, os dados para o estabelecimento de uma regra de classificação. Esse subconjunto terá instâncias relevantes que consigam representar Γ, trazendo as vantagens de redução do número de instâncias que participarão do processo de classificação, redução da quantidade de memória necessária para armazenamento e maior velocidade no processamento dos dados. Por outro lado, algoritmos de geração de protótipos, buscam reduzir o conjunto de treinamento Γ em um subconjunto Φ, gerando protótipos, artificialmente, através de cálculos, visando aumentar a precisão da classificação. Nesta seção será apresentada a metodologia para verificar se os protótipos gerados possuem instâncias no conjunto de treinamento Γ que possam substituí-los, mantendo a mesma taxa de erro de classificação, não sendo necessária, nesses casos, a geração de protótipos.

A. Metodologia proposta

Para esta seção adotaremos a seguinte notação:

- p - Protótipo gerado;
- x_I - Instância mais próxima a p, de forma que $classe(p) = classe(x_I)$;
- C_P - Conjunto de protótipos obtidos a partir de uma técnica de geração;
- C_I - Conjunto de instâncias x_I selecionadas;
- Γ - Conjunto de treinamento;
- Ψ - Conjunto de teste;
- d - Distância entre o p e x_I;
- e_G - Taxas de erro obtidas na geração de protótipos;
- s_G - Desvios padrões obtidos na geração de protótipos;
- e_I - Taxas de erro obtidas na seleção por instâncias mais próximas de p;
- s_I - Desvios padrões obtidos na seleção por instâncias mais próximas de p;
- α - Nível de significância para o teste *T-Student*;
- GL - Grau de liberdade para o teste *T-Student*.

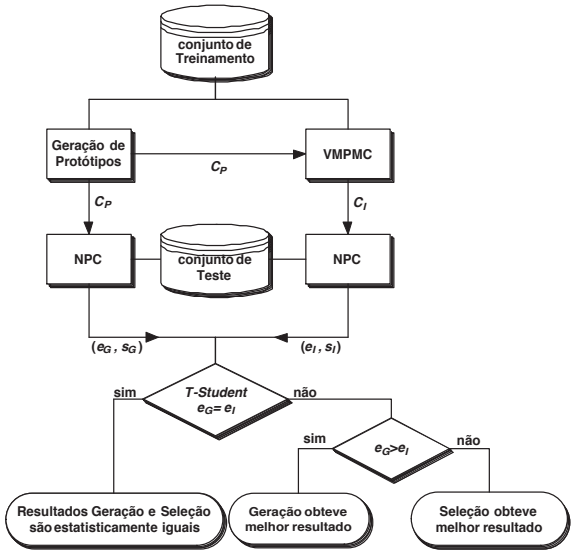

Figura 1. Fluxograma da Metodologia Proposta.

Para validar a nossa hipótese de que, muitas vezes, o protótipo gerado p possui instância x_I muito próxima, de mesma classe, que poderia substituí-lo, tornando desnecessária a sua geração, utilizamos a estratégia apresentada no fluxograma (Figura 1). Inicialmente, utilizamos uma técnica de geração de protótipos, tendo como entrada o conjunto de treinamento Γ. Como resultado, a técnica de geração retorna um conjunto de protótipos gerados C_P. Esse conjunto de protótipos, junto com o conjunto de treinamento Γ, são submetidos à função VMPMC - Vizinho mais próximo do protótipo p de mesma classe. Essa função procura no conjunto de treinamento Γ, a instância mais próxima de cada protótipo gerado $p \in C_P$ de modo que a classe da instância x_I seja igual à classe do protótipo gerado p.

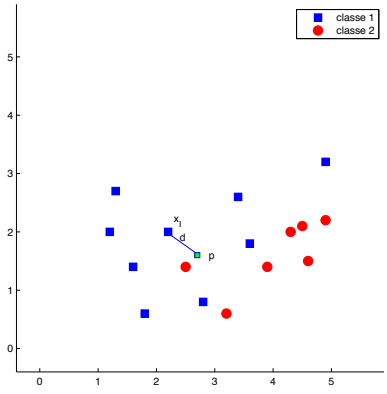

Figura 2. Seleção da instância x_I.

Na Figura 2 é mostrado um exemplo da seleção da instância x_I. Nela é apresentado o protótipo gerado p para a classe quadrado e a instância da classe quadrado mais próxima ao protótipo gerado p. A escolha dessa instância é baseada no cálculo das distâncias d entre as instâncias da classe quadrado e p, sendo escolhido como x_I, aquela de menor d, ou seja, a mais próxima de p.

Cada instância x_I encontrada pela função VMPMC, é adicionada ao conjunto de instâncias selecionadas C_I. De posse dos dois conjuntos de Protótipos: Gerados (C_P) e Selecionados (C_I), passamos para a etapa de teste.

Na etapa de teste, o conjunto de teste Ψ é avaliado usando o classificador vizinho mais próximo NPC - *Nearest Prototype Classifier* [15]. Cada um dos conjuntos C_P e C_I é usado como conjunto de referência separadamente. Como resultado da classificação, obtivemos para cada um dos conjuntos de referência as respectivas taxas de erros e desvios padrões. Para compararmos as taxas de erros obtidas na técnica de geração e as obtidas quando submetemos o mesmo conjunto de teste ao classificador, mas utilizando como referência o conjunto de protótipos C_I, utilizamos o teste de hipótese *T-Student*. O objetivo é o de verificar se há diferenças significativas em relação às médias das taxas de erros encontradas. Sendo μ_0 a média das taxas de erros do método de geração e μ_1 a média das taxas de erros obtida com o conjunto de instâncias selecionadas, usamos como hipótese nula $H_0 : \mu_0 = \mu_1$, hipótese alternativa $H_a : \mu_0 \neq \mu_1$ e nível de significância $\alpha = 5\%$. Logo, se a hipótese nula H_0 for rejeitada no teste, podemos afirmar que as médias das taxas de erro são estatisticamente diferentes e, neste caso, a melhor técnica será a que apresentar a menor taxa de erro. Caso contrário, as médias das taxas de erro são iguais, e neste caso, o resultado obtido pela técnica de geração é estatisticamente igual ao obtido por seleção de instâncias. O pseudocódigo da metodologia proposta é mostrado no Algoritmo 1.

Esse algoritmo possui dois parâmetros de entrada: o conjunto de treinamento Γ e o conjunto de teste Ψ. Após o recebimento dos parâmetros de entrada, é feita a obtenção do conjunto de protótipos a partir de uma técnica de geração de protótipos (linha 2), então, de posse do conjunto de protótipos gerados C_P, utilizamos a função VMPMC-Vizinho mais próximo de p de mesma classe - (linha 5) que retorna para cada $p \in C_P$ a instância x_I, obtendo assim o conjunto C_I. De posse dos dois conjuntos C_P e C_I, submetemos o conjunto de teste Ψ ao classificador NPC [15], obtendo como resultado as taxas de erros e desvios padrões para geração e seleção, respectivamente e_I, e_G, s_I, s_G (linhas 8 e 9). Das linhas 11 a 22, temos a execução do teste de hipótese *T-Student* e as verificações das taxas de erros, que indicam se a melhor técnica é geração ou seleção.

O Pseudocódigo da função VMPMC é mostrado em Algoritmo 2. Esta função busca no conjunto de treinamento Γ o vizinho mais próximo de um protótipo p, de modo que a classe dessa instância seja igual à classe do protótipo p. Recebe como parâmetro o conjunto de treinamento Γ e o protótipo p. Inicialmente, T recebe uma cópia do conjunto de treinamento original Γ. Então, calculamos a distância Euclidiana do protótipo p para cada instância de C_P, adicionando-a ao vetor D (linhas 3 e 4). Após o cálculo de todas as distâncias, ordenamos em ordem crescente o conjunto de treinamento T de acordo com o vetor de distâncias D. Finalmente, buscamos em T a primeira instância que satisfaça a condição $classe(p) = classe(t)$ (linhas 9 a 11).

Algoritmo 1 Metodologia Proposta

Require: Γ: um conjunto de treinamento
Require: Ψ: um conjunto de teste
1: */ Execução de uma técnica de geração /*
2: $C_P = Geracao(\Gamma)$
3: $C_I = []$
4: **for** $p \in C_P$ **do**
5: $x_I = VMPMC(p, \Gamma)$
6: $C_I = C_I \cup \{x_I\}$
7: **end for**
8: $[e_G, s_G] = NPC(C_P, \Psi)$
9: $[e_I, s_I] = NPC(I_P, \Psi)$
10: */ Verifica a hipótese nula das Taxas de Erros serem iguais /*
11: **if** $TStudent(e_I, s_I, e_G, s_G, \alpha, GL) == True$ **then**
12: */ Taxas de Erro são iguais, Seleção=Geração /*
13: **return** 0
14: **else**
15: **if** $e_G < e_I$ **then**
16: */Geração obteve melhores resultados /*
17: **return** 1
18: **else**
19: */ Seleção obteve melhores resultados /*
20: **return** 2
21: **end if**
22: **end if**

Algoritmo 2 VMPMC

Require: Γ: um conjunto de treinamento
Require: p: protótipo $p \in C_P$
1: $T = \Gamma$
2: **for** $t \in T$ **do**
3: d = Distancia-euclidiana(p, t)
4: $D = D \cup \{d\}$
5: **end for**
6: */ Ordena T na ordem crescente das distâncias de D /*
7: Ordena(T,D)
8: **for** $t \in T$ **do**
9: **if** Classe(p)==Classe(t) **then**
10: **return** t
11: **end if**
12: **end for**

B. Aplicação da metodologia a uma base de dados artificial

Com o objetivo de ilustrar a metodologia de busca de instâncias que possam substituir os protótipos gerados, utilizamos a base artificial Banana [12]. A Figura 3 mostra as instâncias do conjunto de treinamento (80% da base).

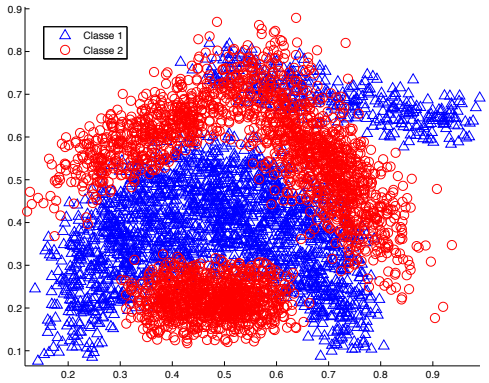

Figura 3. Distribuição das instâncias do conjunto de treinamento da base banana.

A partir do conjunto de treinamento, utilizamos a técnica de geração de protótipos SGP2 [5], que gerou 44 protótipos (Figura 4). A classe 1 representada por triângulos, a classe 2 representada por quadrados. Para cada protótipo $p \in C_P$ foram encontradas as instâncias x_I, instâncias mais próximas aos protótipos p gerados cuja classe é a mesma classe de p, representadas por círculos. Podemos observar que as instâncias x_I estão muito próximas dos protótipos p gerados.

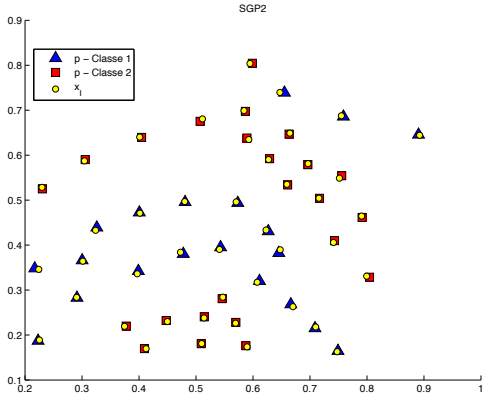

Figura 4. Protótipos gerados pela técnica SGP2 e Instâncias x_I selecionadas a partir dos protótipos gerados.

Finalmente, submetendo o conjunto de teste (20% da base) ao classificador NPC [15], utilizando como conjunto de referência o C_P e posteriormente o C_I, obtivemos as seguintes taxas de erro de classificação: 11,83% para o SGP2 [5] e 11,64% para a seleção. Podemos observar na Figura 5, que as taxas são estatisticamente iguais apesar de uma leve tendência de erro menor na seleção. Assim, para a base artificial Banana, uma seleção de instâncias adequada obteve taxas de acerto tão boas quanto a técnica de geração SGP2 [5].

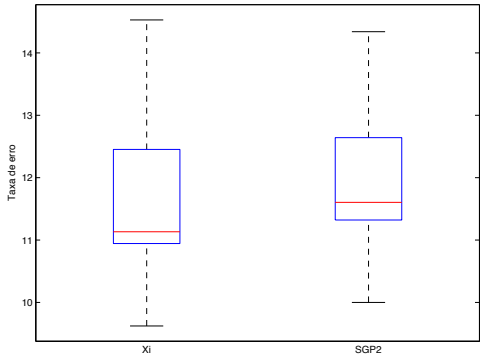

Figura 5. Boxplot - Taxa de erro Seleção x_I × SGP2.

III. Experimentos

Realizamos experimentos em 35 bases de dados: *Appendicitis*, *Australian*, *Balance*, *Banana*, *Bupa*, *Cancer*, *Card*, *Ecoli*, *Gene*, *German*, *Glass*, *Haderman*, *Heart*, *Ionosphere*, *Iris*, *Lymphography*, *Magic*, *Pageblock*, *Penbased*, *Phoneme*, *Ringnorm*, *Phoneme*, *SAHeart*, *Satimage*, *Segmentation*, *Sonar*, *Splice*, *Thyroid*, *Twonorm*, *Vehicle*, *Vowel*, *Waveform*, *Wine*, *Yeast*, *Zoo*, disponíveis nos repositórios da *UCI Learning Machine* [12], PROBEN1 [13] e *KEEL Dataset Repository* [14].

Com o objetivo de obter a melhor performance de classificação em cada método de geração de protótipos, uma etapa preliminar de experimentos foi realizada para detectar o conjunto de parâmetros para cada método. Mais uma vez foi utilizada a validação cruzada na estimação dos parâmetros. Após esses experimentos preliminares, escolhemos os parâmetros que são apresentados na Tabela I.

Tabela I. Parâmetros utilizados nos experimentos

Método	**Parâmetros**
SGP	R_{mis}=0,1; R_{min}=0,08
LVQ	*Iterações=50;*$\alpha = 0,065$*;width=0,2;*$\epsilon = 0,1$
POC-NN	*Alpharatio=0,05*
GENN	*k=7;* $k^{'}$*=4*
RSP3	*Não paramétrico*
ICPL2	*Método de filtragem = ENN [17] [18]*
ENN	*k=3*

O objetivo dos experimentos foi fazer uma análise comparativa dos resultados obtidos em técnicas de geração de protótipos, com os resultados obtidos, quando utilizamos como conjunto de protótipos, o conjunto de instâncias mais próximas aos protótipos gerados, de forma que a $classe(x_I) = classe(p)$. Baseados no estudo de Triguero et al. [2], selecionamos 6 técnicas de geração de protótipos para nossos experimentos: SGP2 [5], LVQ3 [6], RPOCNN [7], RSP3 [8], GENN [9] e ICPL2 [10]. O critério utilizado para a escolha das técnicas de geração foi o de selecionar as técnicas que apresentaram melhores resultados por família. São mostradas, na Figura 6, as famílias das técnicas de geração e as técnicas utilizadas nos experimentos. Utilizamos um método de estimação não paramétrico, validação cruzada de 10 pastas (*10-Fold Cross-validation*). Dividimos os dados de forma estratificada, e após a divisão, os dados foram normalizados, usando a Equação 1, na qual $\overline{x}$ é a média aritmética dos valores do atributo de todas as instâncias do conjunto de treinamento, $s_{Atributo}$ é o

desvio padrão do atributo de todas as instâncias do conjunto de treinamento. O objetivo da normalização dos dados é o de garantir que a magnitude dos valores dos atributos não interfira negativamente no cálculo das distâncias evitando, assim, que essas distâncias sejam influenciadas mais fortemente pelas variáveis de maior magnitude. Com o objetivo de garantir que todos os métodos fossem submetidos às mesmas condições, utilizamos as mesmas combinações de pastas.

$$Atributo(i) = \frac{(Atributo(i) - \overline{x}_{Atributo})}{s_{Atributo}} \quad (1)$$

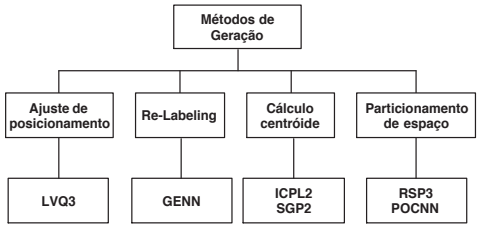

Figura 6. Famílias das técnicas de geração de protótipos selecionadas para o estudo.

Utilizando a estratégia descrita na Seção II para cada uma das 35 bases, obtivemos como resultado a média das taxas de erro de classificação e os desvios padrões. Os experimentos foram realizados aos pares, isto é, os resultados da técnica de geração SGP2 são comparados com os resultados obtidos quando utilizamos o conjunto de instâncias selecionadas C_I que é obtido a partir do conjunto de protótipos gerados C_P, do próprio SGP2. A Figura 7 apresenta um gráfico de barras comparativo, contendo o número de bases em que cada técnica obteve o melhor resultado ou se o resultado obtido é estatisticamente igual, isto é, Geração = Seleção a partir do conjunto de instâncias selecionadas C_I.

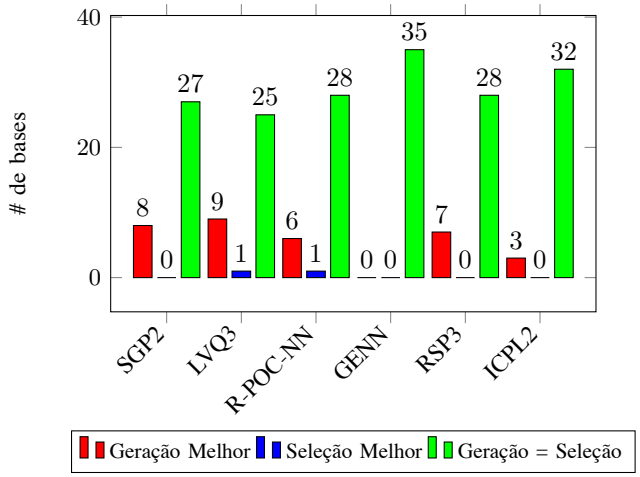

Figura 7. Comparação dos resultados obtidos pela técnica de Geração × Instâncias Selecionadas (C_I).

A. Resultados

Nesta seção serão analisados os resultados dos testes realizados com as 35 bases, aplicando-se a metodologia proposta na Seção II-B, além do estudo comparativo das técnicas de Geração com a técnica de seleção DROP3 - *Decremental Reduction Optimization Procedure* [11]. A escolha da técnica de seleção DROP3 foi baseada no estudo de Garcia et al. [16], que mostra que essa técnica é uma das técnicas mais usadas em comparativos de classificadores, além de apresentar boa capacidade de redução e boa taxa de acerto.

Podemos verificar, na Figura 7, que na maioria dos casos, a classificação feita com o conjunto de instâncias selecionadas (C_I) obteve resultados iguais ou melhores que as técnicas de geração de protótipos, sendo assim, não haveria necessidade de geração, pois o ponto que pode representar a classe já está presente no conjunto de treinamento.

A Tabela II apresenta os percentuais de bases em que cada técnica obteve melhor resultado. Verificamos para a maioria das bases ($83,34\%$), que a classificação com o conjunto de instâncias C_I apresentou taxas de erro estatisticamente iguais às técnicas de geração, implicando que os protótipos gerados possuem instâncias no conjunto de treinamento, que poderiam substituí-los sem a necessidade de geração, com o ganho de desempenho no processo de classificação, visto que as técnicas de geração, geralmente, são mais lentas em relação às técnicas de seleção de instâncias.

Tabela II. PERCENTUAIS DE BASES COM MENORES TAXAS DE ERRO (GERAÇÃO×C$_I$).

	% Bases com menores taxas de erros		
	Geração (Cp)	Seleção (C_I)	(Cp) = (C_I)
SGP2	22,86	0,00	77,14
LVQ3	25,71	2,86	71,43
POC-NN	17,14	2,86	80,00
GENN	0,00	0,00	100,00
RSP3	20,00	0,00	80,00
ICPL2	8,57	0,00	91,43
Média	15,71	0,95	**83,34**

Na Figura 8, é apresentado um gráfico de barras comparativo, contendo o número de bases em que cada técnica obteve melhor resultado ou se o resultado obtido é estatisticamente igual, isto é, Geração = DROP3.

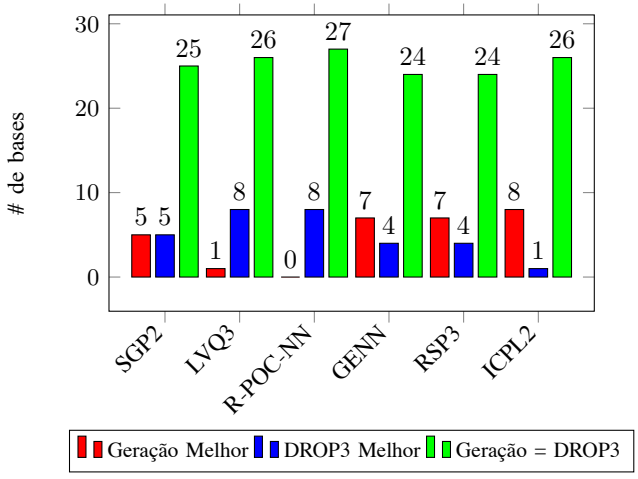

Figura 8. Comparação dos resultados obtidos pela técnica de Geração × DROP3.

Na comparação do SGP2 com o DROP3, verificamos que das 35 bases utilizadas no estudo, o DROP3 obteve em 30 bases (85,73%) as melhores taxas ou taxas estatisticamente iguais ao SGP2 e em 5 bases (14,29%), o SGP2 apresentou melhores resultados. Já no LVQ3, das 35 bases testadas, verificamos que em 34 bases (97,14%), o DROP3 obteve taxas melhores ou taxas estatisticamente iguais ao LVQ3 e em 1 base (2,86%), o LVQ3 apresentou melhores resultados. No caso do R-POC-NN×DROP3, para todas as bases testadas 35 (100%), o DROP3 obteve as taxas melhores ou taxas estatisticamente iguais ao R-POC-NN.

Na comparação do método GENN, verificamos que em 28 bases (80,00%), o DROP3 obteve as taxas melhores ou taxas estatisticamente iguais ao GENN. E em 7 bases (20,00%), o GENN apresentou melhores resultados. O mesmo comportamento foi observado na comparação RSP3×DROP3. E, finalmente, na comparação da técnica ICPL2×DROP3, verificamos que em 27 bases (77.14%), o DROP3 obteve as taxas melhores ou taxas estatisticamente iguais ao ICPL2. E em 8 bases (22,86%), o ICPL2 apresentou melhores resultados.

Tabela III. PERCENTUAIS DAS BASES COM MENORES TAXAS DE ERRO (GERAÇÃO×DROP3)

	% Bases com menores taxas de erros		
	Geração (Cp)	DROP3	Geração = DROP3
SGP2	14,29	14,29	71,42
LVQ3	2,86	22,86	74,28
POC-NN	0	22,86	77,14
GENN	20	11,43	68,57
RSP3	20	11,43	68,57
ICPL2	22,86	2,86	74,28
Média	13,34	14,29	**72,37**

Com base na Tabela III, verificamos que para a maioria das bases (72, 37%), a técnica de seleção DROP3 apresentou taxas de erro estatisticamente iguais às técnicas de geração, implicando que o protótipo gerado possui instância no conjunto de treinamento, que poderia substituí-lo sem a necessidade de geração, com o ganho de desempenho no processo de classificação. Além disso, temos um percentual de 14,29% em que o DROP3 apresentou melhores resultados.

IV. CONCLUSÃO

Nesse trabalho foi proposta uma metoddologia para avaliar técnicas de redução de protótipos, e a partir dos experimentos com 35 conjuntos de dados e da análise estatística, verificamos que as taxas de erros obtidas a partir de um conjunto de instâncias de mesma classe mais próximas dos protótipos gerados (C_I) são na maioria dos casos iguais ou melhores às técnicas de geração. Sendo assim, o protótipo gerado possui uma instância dentro do conjunto de treinamento que pode substituí-lo, sem a necessidade de geração de protótipo. Em algumas bases como a Gene, Ringnorm, Splice, Twonorm e Waveform, em que as técnicas de geração obtiveram melhores resultados, observamos que para esses casos existiam áreas não cobertas por instâncias já existentes no conjunto de treinamento, sendo necessária a geração de protótipos. Como *benchmark*, também executamos todas as técnicas de geração × DROP3, que é uma técnica de seleção, e os resultados também mostraram a mesma tendência, a técnica de seleção obteve resultados iguais ou melhores que as obtidas pelas técnicas de geração para a maioria das bases. Para algumas bases, técnicas de geração se mostraram mais eficientes, quando comparadas ao DROP3, mas em mais de 80% dos casos, a DROP3 apresentou melhores resultados. Os resultados sugerem que, na maioria dos casos, as técnicas de geração geram protótipos muito próximos a instâncias do conjunto de treinamento, sendo assim, a geração de protótipo não é necessária. Notamos que o processo de geração encontra menos protótipos, implicando maiores taxas de redução e que as técnicas de seleção necessitam encontrar o protótipo ótimo, que já está na base de treinamento, mas não foi selecionado.

REFERÊNCIAS

[1] P. Tan, M. Steinbak, and V. Kumar. Introdução ao datamining: Mineração de dados. Ciência Moderna, Rio de Janeiro, 2009,

[2] I. Triguero, J. Derrac, S. Garcí, and F. Herrera. A taxonomy and experimental study on prototype generation for nearest neighbor classification. IEEE Transactions on Systems, Man, and Cybernetics, 42(1):86–100, 2012.

[3] T. Cover and P.Hart. Nearest Neighbor Pattern Classification. IEEE Transactions on Systems, Man, and Cybernetics, 13(1):21-27, 1967.

[4] S.-W. Kim and B. J. Oommen. A Brief Taxonomy and Ranking of Creative Prototype Reduction Schemes. Pattern Analysis and Applications, 6(3):232–244, 2003.

[5] H. A. Fayed, S. R. Hashem, and A. F. Atiya. Self-generating prototypes for pattern classification. Pattern Recognition, 40(5):1498-1509, 2007.

[6] T. Kohonen. Learning Vector Quantization for Pattern Recognition. Technical Report TKK-F-A601, Helsinki University of Technology, Espoo, Finland, 1986.

[7] T. Raicharoen and C. Lursinsap. A divide-and-conquer approach to the pairwise opposite class-nearest neighbor (poc-nn) algorithm. Pattern Recognition, 26(10):1554-1567, 2005.

[8] J.S. Sanchez. High training set size reduction by space partitioning and prototype abstraction. Pattern Recognition, 37:1561–1564, 2004.

[9] J. Koplowitz and T.A. Brown. On the relation of performance to editing in nearest neighbor rules. Pattern Recognition, 13: 251–255, 1981.

[10] W. Lam and C.K. Keung and D. Liu. Discovering useful concept prototypes for classification based on filtering and abstraction. IEEE Transactions on Pattern Analysis and Machine Intelligence, 14(8): 1075-1090, 2002.

[11] D.R. Wilson and T.R. Martinez. Reduction Tecniques For Instance-Based Learning Algorithms. Machine Learning, 38(3):257-286, 2000.

[12] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[13] L. Prechelt. Proben1 a set of neural-network benchmark problems. University of Karlsruhe, Germany, 1994.

[14] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, and S. García. Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. Multiple-Valued Logic and Soft Computing, 17(2-3):255–287, 2011.

[15] S. Seo, M. Bode, and K. Obermeyer. Soft nearest prototype classification. IEEE Transactions on Neural Networks, 14(2):390–398, 2003.

[16] S. Garcia, J. Derrac, J. Cano, and F. Herrera. Prototype selection for nearest neighbor classification: Taxonomy and empirical study. IEEE Transactions on Pattern Analysis and Machine Intelligence, 34(3):417–435, 2012.

[17] D. Wilson. Asymptotic properties of nearest neighbor rules using edited data. IEEE Transactions on Systems, Man, and Cybernetics, 2(3):408–421, 1972.

[18] R. Chang, Z. Pei, and C. Zhang. A modified editing k-nearest neighbor rule. JCP, 6(7):1493–1500, 2011.