Universidade Federal de Pernambuco
Centro de Informática

Pós-graduação em Ciência da Computação

METODOLOGIA PARA AVALIAR TÉCNICAS DE REDUÇÃO DE PROTÓTIPOS: PROTÓTIPOS GERADOS *VERSUS* PROTÓTIPOS SELECIONADOS

Luciano de Santana Pereira

DISSERTAÇÃO DE MESTRADO

Recife
17 de Julho de 2013

Universidade Federal de Pernambuco
Centro de Informática

Luciano de Santana Pereira

METODOLOGIA PARA AVALIAR TÉCNICAS DE REDUÇÃO DE PROTÓTIPOS: PROTÓTIPOS GERADOS *VERSUS* PROTÓTIPOS SELECIONADOS

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientador: *Prof. Dr. George Darmiton da Cunha Cavalcanti*

Recife
17 de Julho de 2013

Catalogação na fonte
Bibliotecária Jane Souto Maior, CRB4-571

Pereira, Luciano de Santana
Metodologia para avaliar técnicas de redução de protótipos: protótipos gerados versus protótipos selecionados / Luciano de Santana Pereira. - Recife: O Autor, 2013.
xi, 75 f. : il., fig., tab.

Orientador: George Darmiton da Cunha Cavalcanti.
Dissertação (mestrado) - Universidade Federal de Pernambuco. CIn, Ciência da Computação, 2013.

Inclui bibliografia.

1. Inteligência artificial. 2. Aprendizagem de máquina. I. Cavalcanti, George Darmiton da Cunha (orientador). II. Título.

006.3 CDD (23. ed.) MEI2013 – 105

Dissertação de Mestrado apresentada por **Luciano de Santana Pereira** à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título "Metodologia para Avaliar Técnicas de Redução de Protótipos: Protótipos Gerados versus Protótipos Selecionados" orientada pelo Prof. George Darmiton da Cunha Cavalcanti e aprovada pela Banca Examinadora formada pelos professores:

_____

Prof. Cleber Zanchettin
Centro de Informática / UFPE

_____

Prof. Renato Fernandes Corrêa
Departamento de Ciência da Informação / UFPE

_____

Prof. George Darmiton da Cunha Cavalcanti
Centro de Informática / UFPE

Visto e permitida a impressão.
Recife, 17 de julho de 2013

_____

Profa. Edna Natividade da Silva Barros
Coordenadora da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.

Para meus pais e minha esposa.

AGRADECIMENTOS

Agradeço aos meus pais, pelos valores passados e pelas lições de vida que sempre me deram.

Agradeço à minha esposa, Betânia, pelo amor, compreensão, incentivo e apoio nas horas mais difíceis.

Agradeço ao meu irmão, Cristiano, pela ajuda, pelo apoio, pelo estímulo.

Agradeço ao meu orientador, Professor Dr. George Darmiton, pela oportunidade, pela paciência e por toda ajuda necessária para realização deste trabalho.

Agradeço ao Professor Dr. Robson Fidalgo, pela compreensão e pela ajuda.

Agradeço ao Professor Dr. Cleber Zanchettin e ao Professor Dr.Renato Fernandes Corrêa pelas contribuições dadas ao trabalho.

Agradeço ao amigo Elias Silva pela ajuda com o Latex.

A persistência é o menor caminho do êxito.

—CHARLES CHAPPLIN

RESUMO

Técnicas de aprendizagem de máquina baseadas em instâncias são utilizadas em várias aplicações, como, por exemplo, reconhecimento de faces, voz e digitais, na medicina para auxiliar médicos na detecção de neoplasias, entre outras. Geralmente, essas técnicas são submetidas a grandes conjuntos de dados, fazendo com que haja necessidade de grande espaço em memória para processamento e 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 da 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. O resultado obtido neste estudo mostra que as técnicas de seleção existentes podem obter taxas equivalentes às técnicas de geração na maioria das bases utilizadas nos experimentos, existindo algumas exceções em que as técnicas de geração obtiveram melhores resultados. Podemos verificar que, na maioria dos casos (83,3%) das 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. Podemos concluir que é possível desenvolver técnicas de seleção, que apresentem taxas de erro estatisticamente iguais às técnicas de geração.

Palavras-chave: Aprendizagem de máquina, aprendizado supervisionado, seleção de protótipos, geração de protótipos, redução de instâncias, vizinho mais próximo.

ABSTRACT

Instance-based Machine Learning Techniques have been applied over several domains as face, voice and handwritten letters recognition, on medicine to predict diseases, and others. In most cases, these techniques are applied to large amounts of data whose demand high storage resource and high computational effort to edit, process and classify data. Aiming to overcome these drawbacks, reduction schemes are used to decrease the amount of data by selecting or generating instances that reach a better representation of the data. As a result of this process, the storage requirements and the computational cost are decreased and, usually, the classification error rate can be decreased. There are two classes of strategies to reduce the size of the data sets: instance selection schemes and prototype generation techniques. In instance selection schemes the result set is entirely composed by some instances chosen from the data set. Prototype generation techniques produces new instances by combining data points using some heuristics. These new instances are named prototypes. The prototypes suffer adjustments during supervised training. Usually, prototype generation techniques have a computational cost higher than the effort of the instance selection schemes. However, in some cases, the prototype generation techniques reach better classification performance than the selective ones. The purpose of this work is to investigate if the selective techniques can obtain results as good as the results reached by the prototype generation techniques. Experimental results show that the selective schemes can obtain classification accuracy rates statistically similar to the ones obtained by the prototype generation techniques, except for some data sets. In 83,3% of the data sets used in the experiments, there were instances that could replace the prototypes generated given the similarity between them. This similarity eliminates the necessity of generation new samples. This work concludes that it is possible to design selection techniques that reach classification error rates statistically equal to the rates obtained by prototype generation techniques.

Keywords: Machine learning, supervised learning, prototype selection, self-generating prototypes, instance reduction, nearest neighbor.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

CAPÍTULO 1

INTRODUÇÃO

1.1 MOTIVAÇÃO

A classificação, que é a tarefa de organizar objetos em uma entre diversas categorias pré-definidas, é um problema universal que engloba muitas aplicações diferentes [34]. Atividades como a classificação de um e-mail em *spam*, baseia-se em informações como remetente, número de destinatários, título e conteúdo da mensagem, que são padrões configurados previamente, e fazem com que o gerenciador e-mails consiga categorizar a mensagem. Como outros exemplos de classificação de padrões podemos citar a detecção de faces, o reconhecimento da fala, categorização de tumores benignos ou malignos, baseados em informações físicas, como tamanho, forma e características bioquímicas, etc. 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. Segundo [34], classificação é a tarefa de aprender uma **função alvo** f que mapeie cada conjunto de atributos x para um dos rótulos y pré-determinados. 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. Esta 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 da pesquisa que buscam a redução de instâncias: a seleção de instâncias e a 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. Essa técnica 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? Sendo assim, o foco deste estudo é verificar se as técnicas de seleção obtêm resultados semelhantes às técnicas de geração.

1.2 OBJETIVO DO ESTUDO

Dado que na literatura os melhores resultados de classificação são obtidos através da utilização de técnicas de geração de protótipos [19], deseja-se verificar se é possível que técnicas de seleção apresentem resultados iguais ou melhores às técnicas de geração. Dentre as técnicas de geração, estudaremos o SGP2, LVQ3, RPOCNN, RSP3, GENN e ICPL2. A escolha das técnicas foi baseada no estudo de Triguero et al. [35]. Utilizaremos também, a técnica de seleção DROP3, para fazer uma comparação com as técnicas de geração citadas anteriormente. 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.

1.3 ESTRUTURA DA DISSERTAÇÃO

Este trabalho apresenta a seguinte estrutura:

- No Capítulo 1 é feita a contextualização do problema e motivação do estudo.

- No Capítulo 2 são apresentadas as técnicas de seleção de prótotipos e de geração de protótipos, descrevendo, detalhadamente, as técnicas que serão utilizadas no estudo.
- No Capítulo 3 é apresentada a metodologia que será usada para avaliar a relação entre algoritmos de geração de protótipos e as instâncias selecionadas.
- No Capítulo 4 são apresentados a metodologia, experimentos e resultados.
- E, finalmente, no Capítulo 5 são apresentados a conclusão e trabalhos futuros.

CAPÍTULO 2

TÉCNICAS DE SELEÇÃO E DE GERAÇÃO DE PROTÓTIPOS

2.1 SELEÇÃO DE PROTÓTIPOS

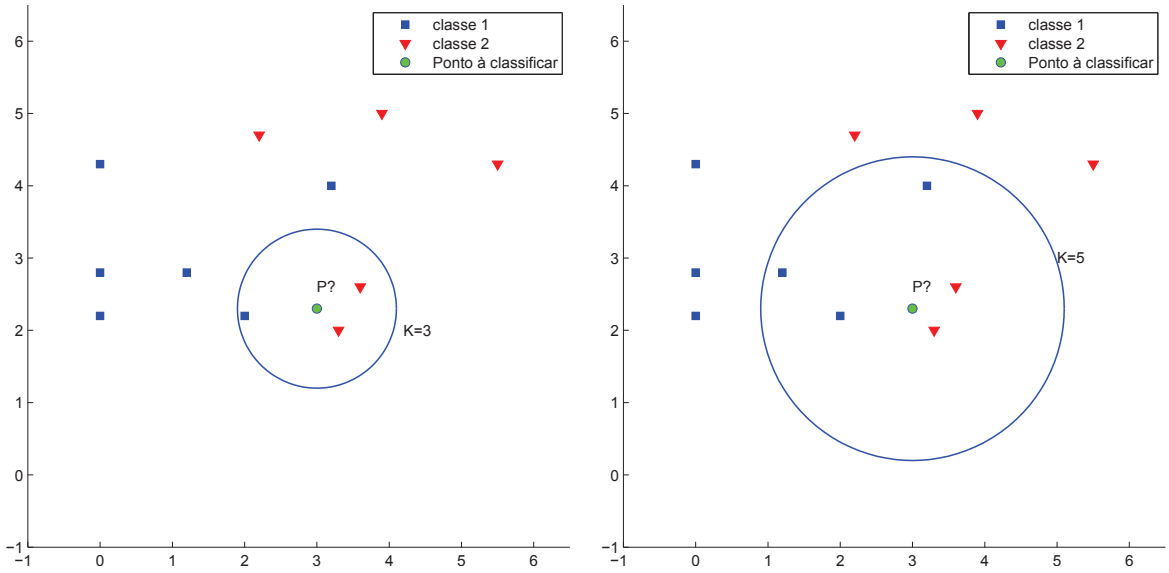
Seleção de Protótipos é uma técnica de aprendizagem de máquina que busca a partir de uma base de dados que representa um problema real, a obtenção de instâncias que representem esta base. Essas instâncias são denominadas protótipos. Devido ao tamanho das bases, a ideia básica da seleção de protótipos é reduzir a quantidade de dados do conjunto de treinamento obtendo como resultado um subconjunto que represente de forma clara as classes contidas no problema, mantendo, assim, o erro de classificação, além da redução do custo computacional necessário para classificar os dados, pois teremos um conjunto reduzido de dados. Uma técnica bastante conhecida de seleção de protótipos baseia-se nos k-vizinhos mais próximos, K-Nearest Neighbor (KNN) [9].

2.2 KNN - *K-NEAREST NEIGHBOR*

É uma técnica simples que apresenta boas taxas de acerto, porém, é sensível a ruídos, isto é, quando os padrões a serem classificados estão próximos de regiões onde temos sobreposição de elementos de classes diferentes. Outra desvantagem é que o KNN é lento em relação a outros classificadores, pois para classificar, é necessário calcular a distância de cada instância i para cada elemento da base de treino.

A Figura 2.1 ilustra uma classificação utilizando a técnica KNN, cujo objetivo é classificar o ponto P, representado por um círculo verde, baseado nos k-vizinhos mais próximos. O ponto de dado é classificado com base na classe dos seus vizinhos mais próximos. Se os vizinhos mais próximos forem de mais de uma classe, o ponto apresentado é classificado de acordo com a classe majoritária de seus vizinhos mais próximos. Quando atribuímos K=3, o círculo verde será classificado como da classe triângulo, pois temos dois vizinhos da classe triângulo, mas quando K=5, o círculo será classificado como quadrado, pois temos um número maior de vizinhos da classe quadrado.

Segundo [34], o exemplo anterior mostra a importância da definição correta do valor de K.



(a) K=3. (b) k=5.

Figura 2.1 Exemplo da aplicação do algoritmo KNN sobre um mesmo conjunto original de dados gerando dois resultados distintos.

Se K for pequeno demais, então o classificador de vizinho mais próximo pode ser susceptível a overfitting por causa do ruído nos dados de treinamento. Na Figura 2.2 é mostrado um exemplo em que o K escolhido é grande demais, fazendo com que o classificador de vizinhos mais próximos classifique erroneamente a instância de teste porque a lista de vizinhos mais próximos pode incluir pontos de dados que estejam localizados longe da sua vizinhança.

Usualmente é utilizada a distância Euclidiana no cálculo dos vizinhos mais próximos. Essa distância é calculada pela equação 2.1.

$$d = \sqrt{\sum_{i=0}^{n} (a_i - b_i)^2} \tag{2.1}$$

Conforme mostrado no algoritmo 1, o KNN é simples de ser implementado, mas como temos que calcular a distância do ponto P em relação a todos os Ti do conjunto T, o processo tem um alto custo computacional.

2.3 ENN - *EDITED NEAREST NEIGHBOR RULE*

Também é uma técnica de seleção de protótipos proposta por Wilson em 1972,[36][7]. Em nosso trabalho, a técnica ENN será utilizada como a camada de filtragem da técnica de geração ICPL2. Esta técnica foi projetada para funcionar como um filtro de ruídos, elimi-

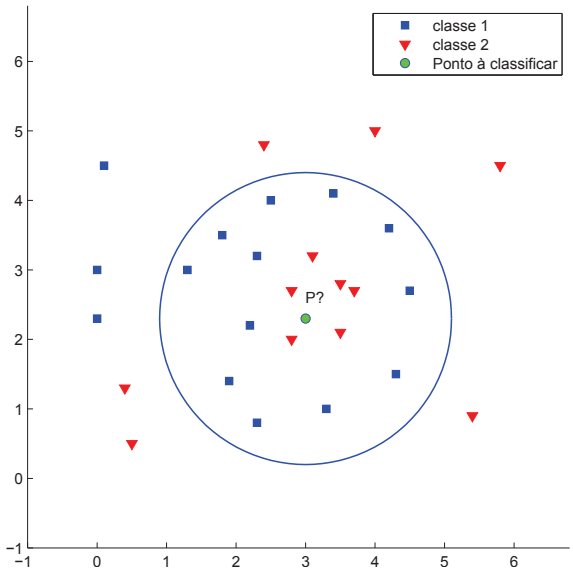


Figura 2.2 Exemplo da aplicação do algoritmo KNN utilizando um k muito grande que levou a uma classificação errônea.

nando instâncias na região de fronteira, região de indecisão com alta possibilidade de erros de classificação. Apresenta baixa taxa de redução, por eliminar somente instâncias de região de fronteira, ou em regiões em que haja a probabilidade de erros.

Por atuar apenas na região de fronteira, esta técnica possui uma baixa capacidade de redução. Seu algoritmo mantém as instâncias que não estão localizadas nesta região, exceto no caso de instâncias com extrema probabilidade de erro.

Algoritmo 1 KNN

Entrada: T: um conjunto de treinamento
Entrada: P: Ponto a ser classificado
Entrada: K: Número de vizinhos mais próximos

1: **Para** $T_i \in T$ **faça**
2: D = DistanciaEuclidiana(P, T_i)
3: $L = L \cup \{D, Classe(T_i)\}$
4: **Fim Para**
5: $L = Ordena(L)$, Ordena L em ordem crescente das distâncias
6: $S = PrimeirosKElementos(K, L)$
7: $C = ClassedeMaiorFreq(S)$
8: **Retorne** C, classe de P

Algoritmo 2 ENN

Entrada: T: Conjunto de treinamento
Entrada: I_e:Contém instâncias que geraram erro na classificação
1: **Para todo** instância I_i em T **faça**
2: Aplique o 3NN sobre I_i utilizando T como treinamento
3: **Se** I_i foi classificado erroneamente **então**
4: salve I_i em I_e
5: **Fim Se**
6: **Fim Para**
7: Remova de T todas as instâncias de I_e
8: **Retorne** T

Podemos variar o k do KNN em função da taxa de redução que queremos obter e em função do tamanho da base de dados, mas o valor usual de k para este filtro é 3. Valores menores de k podem gerar *overffiting*, onde instâncias que não são ruídos também são eliminadas.

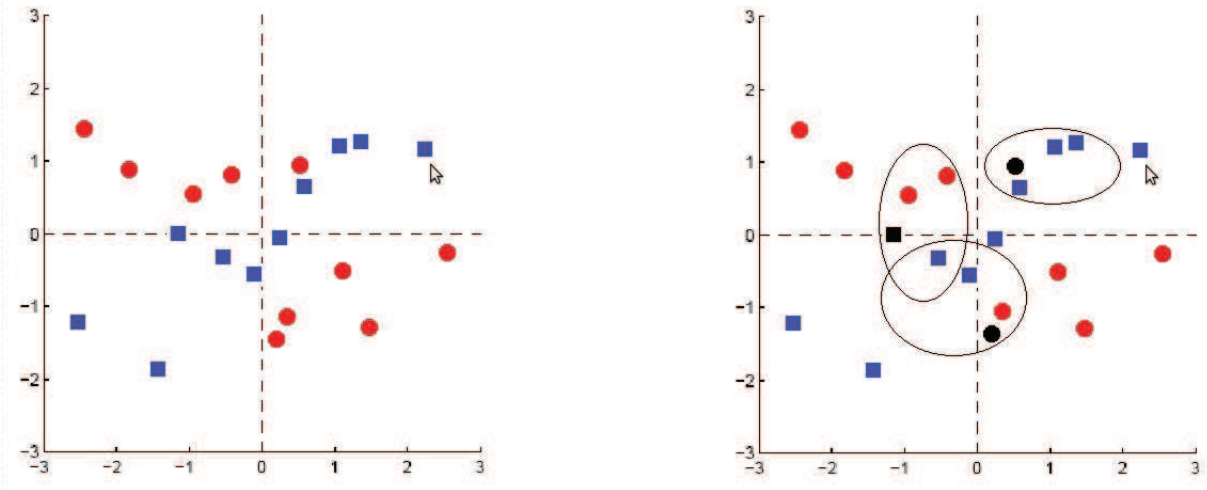


Figura 2.3 ENN aplicado com K=3

Na Figura 2.3 é mostrado um problema de classificação envolvendo uma base com duas classes. Em (a) observamos a base original, isto é, antes da submissão da base ao ENN. Em (b) observamos o resultado da aplicação do ENN, utilizando o parâmetro K=3. As instâncias foram consideradas ruídos, devido ao fato de terem sido classificadas erroneamente pelo KNN, sendo representadas pelos pontos pretos, já as regiões circuladas envolvem os 3 vizinhos mais próximos da instância considerada ruído. Verifica-se também, no exemplo, que após a execução do KNN, temos uma maior separação das classes, mostrando que o ENN consegue de forma eficiente, eliminar as zonas de indecisão. Uma das grandes vantagens do ENN é o fato de ser um método determinístico, isto é, quando se aplica um mesmo k para a base de dados,

os resultados são idênticos, independente da ordem em que os dados são apresentados. Por eliminar só as instâncias que são conseideradas ruídos, o ENN apresenta uma baixa taxa de redução, deixando no conjunto de treinamento muitas instâncias redundantes, isto é, instâncias que representam a mesma área de uma classe.

Devido as características do método de redução de ruídos, o ENN é muito utilizado em conjunto com outro classificador, na fase de tratamento dos dados, eliminando as instâncias que possam vir a gerar erros de classificação.

2.4 DROP1-5 - *DECREMENTAL REDUCTION OPTIMIZATION PROCEDURE*

As técnicas DROP1, DROP2, DROP3, DROP4 e DROP5 foram propostas por Wilson e Martinez [37] com o objetivo de serem tolerantes a ruídos, apresentarem altas taxas de redução e serem determinísticas, isto é, a ordem de apresentação dos dados não influenciam os resultados, além de terem um alto poder de generalização. Essas técnicas apresentam um novo conceito em relação a outras técnicas de seleção. Um novo conceito é introduzido nas técnicas DROP: o conceito de associados de uma instância *i*, que é o conjunto de todas as instâncias que tem *i* como um de seus *k* vizinhos mais próximos. De todas as versões do DROP, o DROP3 apresenta a melhor relação entre o percentual de redução e a acurácia, sendo utilizado como classificador de referência em estudos comparativos encontrados na literatura [37].

2.4.1 *Decremental Reduction Optimization Procedure 1* - DROP1

Esta técnica é a base para as outras versões do DROP e seu algoritmo é idêntico ao RNN-*Reduced Nearest Neighbor* [16], exceto pela verificação da acurácia que é feita baseada em uma cópia do conjunto de treinamento T, o conjunto S[37]. O processo de redução é feito em S. A remoção da instância é feita baseada no seguinte critério: se a instância não piora a classificação dos associados, pode ser removida. No Algoritmo 3 é mostrado o pseudocódigo do DROP1. Analisando o pseudocódigo, podemos verificar que, inicialmente, temos a inicialização do conjunto S que recebe T e torna-se uma cópia do mesmo. Os próximos laços são necessários para encontrar os $K + 1$ vizinhos mais próximos dos associados de todas as instâncias P. Passando então à etapa de verificação de como a instância influencia a generalização do conjunto. Esta verificação leva em conta o desempenho de classificação dos associados de P, com (*With*) e sem (*Without*) a instância P. Se forem classificados corretamente mais associados sem a necessidade de P, essa instância será eliminada do conjunto S. Toda vez que é feita uma remoção de instância, a lista de vizinhos mais próximos da lista de associados é atualizada. O DROP1 tem a capacidade de remoção de ruídos, porque geralmente uma instância ruidosa P, tem na maioria

das vezes associados de classe diferente, logo a classificação dos associados sem a instância P, implica provavelmente uma classificação correta. O DROP também apresenta a capacidade de remoção de instâncias no centro de aglomerados. Nas regiões próximas à fronteira, a remoção de algumas instâncias pode causar erros de classificação. [37]

Algoritmo 3 DROP1

Entrada: T: Conjunto de treinamento
Entrada: k: Número de vizinhos mais próximos para a seleção dos associados

$S = T$
Para todo $P \in S$ **faça**
  $N(P) = RetornaVizinhosMaisPróximos(k+1, P, S)$
  Para todo $n \in N(P)$ **faça**
    Adiciona P ao conjunto de associados A
    $A(n) = A(n) \cup P$
  Fim Para
Fim Para
Para todo $P \in S$ **faça**
  ComP = Número de associados de P classificados corretamente com P como vizinho
  SemP = Número de associados de P classificados corretamente sem P como vizinho
  Se $SemP - ComP \geq 0$ **então**
    Remove P de S
    Para todo $a \in A(P)$ **faça**
      *Remove a instância **P** de N(a)*
      *Busca o novo vizinho mais próximo de **a** em S*
      *Adiciona em N(a) o novo vizinho mais próximo de **a** em S*
    Fim Para
    Para todo $n \in N(P)$ **faça**
      Remove de A(n) a instância P
    Fim Para
  Fim Se
Fim Para
Retorne S

2.4.2 *Decremental Reduction Optimization Procedure 2* - DROP2

O DROP1 pode apresentar problemas devido a sua capacidade de remoção de ruídos, quando instâncias são consideradas ruídos devido ao fato de seus associados serem de classes diferentes da sua. Se os associados são removidos, a instância ruidosa passa a cobrir um maior espaço. Eventualmente, a própria instância ruidosa será removida. No entanto, podemos ter muitos de seus vizinhos removidos primeiramente. Também podemos ter a inclusão de uma instância de mesma classe, mas do outro lado da fronteira de decisão à lista de associados,

podendo levar a classificações errôneas [37]. Para solucionar, esse problema, foi proposto o DROP2, que avalia o impacto da remoção de uma instância, tomando como base as instâncias do conjunto original T. Além da mudança do conjunto usado para avaliação, o DROP2 utiliza uma ordenação do conjunto a ser processado, tendo como critério a distância aos inimigos mais próximos(ordem decrescente). Com esta ordenação, teremos primeiro a remoção das instâncias distantes das bordas, aumentando a retenção final próxima às fronteiras de decisão [37].

Algoritmo 4 DROP2

Entrada: T: Conjunto de treinamento
Entrada: k: Número de vizinhos mais próximos para a seleção dos associados
1: ** *Ordena as instâncias de T, de acordo com as distâncias aos seus inimigos mais próximos*
2: $S = Ordena(T)$ ** Ordem decrescente
3: **Para todo** $P \in S$ **faça**
4: $N(P) = RetornaVizinhosMaisPróximos(k+1, P, S)$
5: **Para todo** $n \in N(P)$ **faça**
6: *Adiciona a instância P ao conjunto de associados A*
7: $A(n) = A(n) \cup P$
8: **Fim Para**
9: **Fim Para**
10: **Para todo** $P \in S$ **faça**
11: *ComP = Número de associados de P classificados corretamente com P como vizinho*
12: *SemP = Número de associados de P classificados corretamente sem P como vizinho*
13: **Se** $SemP - ComP \geq 0$ **então**
14: Remove P de S
15: **Para todo** $a \in A(P)$ **faça**
16: *Remove a instância P de N(a)*
17: *Busca o novo vizinho mais próximo de P em S*
18: *Adiciona em N(a) o novo vizinho mais próximo de P em S*
19: **Fim Para**
20: **Fim Se**
21: **Fim Para**
22: **Retorne** S

No Algoritmo 4 é mostrado o pseudocódigo do DROP2. Comparando-o ao Algoritmo 3 do DROP1, podemos verificar as diferenças entre as duas versões. Inicialmente, o conjunto S recebe o conjunto T, ordenado em ordem decrescente pelo critério das distâncias aos inimigos mais próximos (linha 1). As linhas de inicialização dos associados continuam sem alterações (linhas 2 a 7). Outra diferença que podemos notar é a retirada do bloco responsável pela atualização na lista de associados de seus vizinhos (linhas 19 a 21 - Algoritmo 3). Este fato implica que podemos ter instâncias P que possuam associados só em T. Como esta técnica faz uma seleção mais criteriosa, os limites de decisão ficam mais bem definidos e as possíveis

remoções de classes inteiras que ocorriam no DROP1 foram eliminados no DROP2 [37].

2.4.3 *Decremental Reduction Optimization Procedure 3* - DROP3

Esta versão do DROP foi proposta com o objetivo de resolver os problemas do DROP2 com relação à não eliminação de pontos de centros das regiões de decisão, que estão próximos a pontos ruidosos de borda. Isso ocorre devido à ordenação utilizada pelo DROP2. Para solucionar este problema o DROP3 faz um pré-processamento do conjunto S, com o objetivo de remoção de ruídos antes da ordenação, fazendo com que os pontos centrais que são mantidos pelo DROP2 sejam eliminados. A filtragem é feita com a utilização da técnica ENN[36], que suaviza as fronteiras de decisão e elimina as instâncias que não são corretamente classificadas com base nos seus k-vizinhos mais próximos.

Algoritmo 5 DROP3

Entrada: T: Conjunto de treinamento
Entrada: k: Número de vizinhos mais próximos para a seleção dos associados
S=T
Para todo $P \in S$ **faça**
  $N(P) = RetornaVizinhosMaisPróximos(k+1, P, S)$
  Para todo $n \in N(P)$ **faça**
    Adiciona a instância P ao conjunto de associados A
    $A(n) = A(n) \cup P$
  Fim Para
Fim Para
S=ENN(S)
*** Ordena as instâncias de T, de acordo com as distâncias aos seus inimigos mais próximos*
$S = Ordena(T)$ ** Ordem decrescente
Para todo $P \in S$ **faça**
  ComP = Número de associados de P classificados corretamente com P como vizinho
  SemP = Número de associados de P classificados corretamente sem P como vizinho
  Se $SemP - ComP \geq 0$ **então**
    Remove P de S
    Para todo $a \in A(P)$ **faça**
      Remove a instância P de N(a)
      Busca o novo vizinho mais próximo de P em S
      Adiciona em N(a) o novo vizinho mais próximo de P em S
    Fim Para
  Fim Se
Fim Para
Retorne S

O pseudocódigo do DROP3 é mostrado no Algoritmo 5. Inicialmente temos a criação da

lista de associados A e logo após é aplicado o filtro ENN, eliminando instâncias ruidosas e próximas às fronteiras de decisão. Só após a filtragem, que é feita a ordenação pelo inimigo mais próximo. A partir deste ponto o DROP3 segue os mesmos passos do DROP2.

2.4.4 *Decremental Reduction Optimization Procedure 4* - DROP4

O DROP4 é outra versão do DROP, que é muito parecida com o DROP3 exceto pelo fato verificação da filtragem, que checa a classificação baseada nos k vizinhos mais próximos e também é checada a performance de classificação com e sem a instância, Algoritmo 6, linha 15. Verificando se a remoção irá piorar a generalização do conjunto. O pseudocódigo do DROP4 é mostrado no Algoritmo 6. Essa técnica visa eliminar possíveis problemas, de demasiada redução de instâncias, gerados pela DROP3, pois o critério de remoção passa a ser duplo, diminuindo a capacidade de redução[37].

Algoritmo 6 DROP4

Entrada: T: Conjunto de treinamento
Entrada: k: Número de vizinhos mais próximos para a seleção dos associados

S=T
Para todo $P \in S$ **faça**
  $N(P) = RetornaVizinhosMaisPróximos(k+1, P, S)$
  Para todo $n \in N(P)$ **faça**
    Adiciona a instância P ao conjunto de associados A
    $A(n) = A(n) \cup P$
  Fim Para
Fim Para
Para todo $P \in S$ **faça**
  ComP = Número de associados de P classificados corretamente com P como vizinho
  SemP = Número de associados de P classificados corretamente sem P como vizinho
  Se $(Classe(P) \neq Classe(KNN(P,T)))$ E $(SemP - ComP \geq 0)$ **então**
    Remove P de S
  Fim Se
Fim Para
*** Ordena as instâncias de T, de acordo com as distâncias aos seus inimigos mais próximos*
$S = Ordena(T)$ ** Ordem decrescente
Para todo $P \in S$ **faça**
  ComP = Número de associados de P classificados corretamente com P como vizinho
  SemP = Número de associados de P classificados corretamente sem P como vizinho
  Se $SemP - ComP \geq 0$ **então**
    Remove P de S
    Para todo $a \in A(P)$ **faça**
      Remove a instância P de N(a)
      Busca o novo vizinho mais próximo de P em S
      Adiciona em N(a) o novo vizinho mais próximo de P em S
    Fim Para
  Fim Se
Fim Para
Retorne S

2.4.5 *Decremental Reduction Optimization Procedure 5* - DROP5

O DROP5 é uma modificação do DROP2, onde a ordenação é modificada passando a ser do inimigo mais próximo ao inimigo mais distante. Logo, as primeiras instâncias a serem removidas serão próximas às fronteiras de decisão, fazendo com que o limite de decisão seja suavizado. O processo é repetido até que a remoção não implique melhoria de classificação [37].

2.5 LVQ - *LEARNING VECTOR QUANTIZATION*

O Learning Vector Quantization (LVQ), proposto por [20] é um dos modelos de classificação baseado nos vizinhos mais próximos, e usa uma estratégia de criar novas instâncias baseadas nas instâncias existentes, sendo assim, uma técnica de geração de protótipos. Nessa técnica, cada classe será representada por um conjunto de vetores, chamados de protótipos, o LVQ faz o ajuste desses, posicionando cada instância em um ponto que seja um representante para as classes. O ajuste dos protótipos, diferentemente do KNN, é feito a partir dos protótipos selecionados. Isto faz com que o custo computacional do classificador seja menor em relação ao KNN. Como nem todas as instâncias são usadas como protótipos, a ordem e qualidade das instâncias alteram o resultado, ou seja, o algoritmo não é determinístico, isto é, se executarmos várias vezes o LVQ, obteremos taxas de acertos diferentes. A ideia é que os protótipos sejam escolhidos de forma aleatória, mas que tenham boa representatividade da base de dados. Outra desvantagem do método é o número de parâmetros, que tem que ser definido empiricamente.

Nas próximas subseções apresentaremos as variações do LVQ, proposto, inicialmente, por Kohonen, mostrando as principais características de ajuste de protótipos.

2.5.1 LVQ1

É a versão inicial do algoritmo proposta por Kohonen. O algoritmo, a partir dos protótipos, que são vetores de características, selecionados inicialmente, começa a ajustá-los, usando as instâncias da base de dados original. O número de protótipos selecionados também é um parâmetro definido livremente. Na equação 2.2 verificamos que se a instância x(t) é classificada corretamente, aproxima-se o protótipo mais próximo, caso a instância seja classificada erroneamente, o protótipo mais próximo será afastado. Os outros protótipos não são ajustados. O ajuste dos protótipos é feito com base nas equações abaixo:

$$m_c(t+1) = \begin{cases} m_c(t) + \alpha(t)[x(t) - m_c(t)], & se \quad x = m_c \\ m_c(t) - \alpha(t)[x(t) - m_c(t)], & se \quad x \neq m_c \end{cases} \qquad (2.2)$$
$$m_i(t+1) = m_i(t), \forall\, i \neq c.$$

Sendo m_c o protótipo mais próximo de $x(t)$, $x(t)$ é um elemento do conjunto de treinamento e $\alpha(t)$ é taxa de aprendizado, variando entre 0 e 1. No processo de convergência, deve-se utilizar, inicialmente, um valor alto de $\alpha(t)$, que deve ser reduzido, objetivando a estabilidade do processo.

Algoritmo 7 LVQ 1

Entrada: T: um conjunto de treinamento
Entrada: P: uma lista para os protótipos
Entrada: $Selection$: um algoritmo para seleção dos protótipos iniciais

1: P = $Selection(T)$
2: **Enquanto** P estiver sub-ajustado **faça**
3: $x(t)$ = $ChooseOne(T)$
4: m_c(t) = 1-NN de x sobre P
5: **Se** $Classe(m_c(t)) = Classe(x)$ **então**
6: $m_c(t) = m_c(t) + \alpha(t) \times [x(t) - m_c(t)]$
7: **Senão**
8: $m_c(t) = m_c(t) - \alpha(t) \times [x(t) - m_c(t)]$
9: **Fim Se**
10: **Fim Enquanto**
11: **Retorne** P, protótipos ajustados

2.5.2 LVQ 2.1

Basicamente o LVQ2.1 é uma das versões do LVQ otimizada, proposta por Kohonen que faz atualização simultânea dos dois protótipos mais próximos de x(t), a cada passo do treinamento. Enquanto o LVQ1 provoca o afastamento dos protótipos nas regiões de indecisão, o LVQ 2.1 atua reduzindo esse afastamento apenas sobre protótipos vizinhos pertencentes a classes diferentes, isto é, protótipos próximos das fronteiras. Dizemos que um elemento x(t) deverá ser ajustado se cair em uma região de indecisão definida pela equação 2.6.

$$min\left(\frac{d_i}{d_j}, \frac{d_j}{d_i}\right) > s, \quad \text{com} \quad s = \frac{1-w}{1+w} \tag{2.3}$$

Onde d_i e d_j são respectivamente as distâncias Euclidiana de x(t) para m_i e m_j. O parâmetro w é a largura relativa da janela e são recomendados valores de w entre 0,2 e 0,3. A região de janela representada pela área sombreada da Figura 2.4, fica em torno das fronteiras de classificação, sendo assim uma região de dúvida.

O ajuste dos protótipos é feito com base nas equações 2.4

$$\begin{cases} m_i(t+1) = m_i(t) + \alpha(t) \times [x(t) - m_i(t)] \\ m_j(t+1) = m_j(t) - \alpha(t) \times [x(t) - m_j(t)] \end{cases} \tag{2.4}$$

O ajuste é feito toda vez que um dos protótipos for da mesma classe de x(t) e o outro for de outra classe. Se os dois protótipos forem da mesma classe que x(t) ou se forem de classes distintas da classe de x(t), não haverá ajuste. Outra restrição que deve ser satisfeita para que

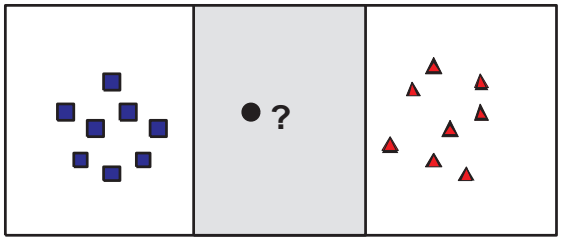


Figura 2.4 Região de Janela é representada pela área cinza

ocorra ajuste é a que x(t) deve cair na região de janela mostrada na Figura 2.4.

O LVQ 2.1 utiliza protótipos previamente ajustados pelo LVQ 1 como o seu conjunto inicial de protótipos.

Algoritmo 8 LVQ 2.1

Entrada: T: um conjunto de treinamento
Entrada: P: uma lista para os protótipos
Entrada: $Selection$: um algoritmo para seleção dos protótipos iniciais

$P = LVQ1(T, Selection)$
Enquanto P estiver sub-ajustado **faça**
  $x(t) = ChooseOne(T)$
  m_i, m_j = 2-NN de x, usando P como treinamento
  Se $CaiuNaJanela(x(t), m_i(t), m_j(t))$ **então**
    Se $Classe(m_i(t)) \neq Classe(m_j(t))$ **então**
      Se $Classe(m_i(t)) = Classe(x(t))$ **então**
        $m_i(t) = m_i(t) + \alpha(t) \times [x - m_i(t)]$
        $m_j(t) = m_j(t) - \alpha(t) \times [x - m_j(t)]$
      Senão
        $m_i(t) = m_i(t) - \alpha(t) \times [x - m_i(t)]$
        $m_j(t) = m_j(t) + \alpha(t) \times [x - m_j(t)]$
      Fim Se
    Fim Se
  Fim Se
Fim Enquanto
Retorne P, protótipos ajustados

Uma das desvantagens do LVQ 2.1 é que ele apresenta um custo computacional maior que o LVQ 1 e a sua aplicação pode gerar sobreajuste dos protótipos na região de decisão. Para reduzir esse sobreajuste foi proposto o LVQ3, detalhado na próxima seção.

2.5.3 LVQ 3

Para reduzir o sobreajuste do LVQ 2.1, Kohonen propôs a terceira versão do LVQ. Assim como no LVQ 2.1, são selecionados dois protótipos, mi(t) e mj(t), mais próximos de x(t), e se apenas um deles for da mesma classe de x(t) e x(t) cair dentro da região de janela, os protótipos serão ajustados,

$$\begin{cases} m_i(t+1) = m_i(t) + \alpha(t)[x(t) - m_c(t)] \\ m_j(t+1) = m_j(t) - \alpha(t)[x(t) - m_c(t)] \end{cases} \tag{2.5}$$

mas se os dois protótipos forem da mesma classe de x(t), também haverá ajuste, porém este será ponderado por um fator de estabilidade ϵ $(0 < \epsilon < 1)$ conforme Equação 2.6.

$$m_k(t+1) = m_k(t) + \epsilon\alpha(t)[x(t) - m_c(t)] \quad k \in \{i, j\} \tag{2.6}$$

Algoritmo 9 LVQ 3

Entrada: T: um conjunto de treinamento
Entrada: P: uma lista para os protótipos
Entrada: $Selection$: um algoritmo para seleção dos protótipos iniciais

$P = LVQ1(T, Selection)$
Enquanto P estiver sub-ajustado **faça**
  $x = ChooseOne(T)$
  $m_i(t), m_j(t) =$ 2-NN de x, usando P como treinamento
  Se $CaiuNaJanela(x(t), m_i(t), m_j(t))$ **então**
    Se $Classe(m_i(t)) \neq Classe(m_j(t))$ **então**
      Se $Classe(m_i(t)) = Classe(x(t))$ **então**
        $m_i(t) = m_i(t) + \alpha(t) \times [x(t) - m_i(t)]$
        $m_j(t) = m_j(t) - \alpha(t) \times [x(t) - m_j(t)]$
      Senão
        $m_i(t) = m_i(t) - \alpha(t) \times [x(t) - m_i(t)]$
        $m_j(t) = m_j(t) + \alpha(t) \times [x(t) - m_j(t)]$
      Fim Se
    Senão Se $Classe(e_i) = Classe(e_j) = Classe(x)$ **então**
      $m_i(t) = m_i(t) + \epsilon \times \alpha(t) \times [x(t) - m_i(t)]$
      $m_j(t) = m_j(t) + \epsilon \times \alpha(t) \times [x(t) - m_j(t)]$
    Fim Se
  Fim Se
Fim Enquanto
Retorne P, os protótipos ajustados

As três versões apresentadas do LVQ, LVQ1, LVQ 2.1 e LVQ 3 apresentam taxas de acerto

semelhantes, embora utilizem filosofias diferentes. LVQ1 e LVQ3 se estabilizam com o treinamento contínuo. Já o LVQ 2.1 é utilizado para melhorar os resultados do LVQ1.[20] De todas as versões o LVQ 3 é o mais robusto, mas um dos principais problemas é a determinação dos parâmetros w, α e ϵ.

2.6 SGP - *SELF-GENERATION PROTOTYPE*

O Self-Generation Prototype (SGP) proposto por [12] é uma técnica de síntese de protótipos bastante eficiente na redução do número de instâncias. Além disso, essa proposta visa solucionar problemas que ocorrem em outras técnicas, como por exemplo o LVQ, que apresenta sensibilidade à posição inicial dos protótipos, a quantidade de protótipos escolhida como representantes de cada classe, além do fato que, muitas vezes, a busca pelos parâmetros certos torna o processo bastante custoso, necessitando de várias interações para combinar os diversos valores dos parâmetros. Outra vantagem do método é que no processo de treinamento, o número e o posicionamento dos protótipos são feitos sem a intervenção humana. A estratégia do SGP é a seguinte: em um primeiro passo são criados grupos com elementos de mesma classe, e o protótipo adotado será a média dos padrões. Durante o treinamento, de acordo com o cumprimento de regras, grupos podem ser divididos ou fundidos, instâncias podem ser movidas de um grupo para outro. As regras abaixo devem ser seguidas até que não haja mais alterações no grupo.

- Se para todos os padrões de um grupo o protótipo mais próximo é o protótipo deste grupo, então nenhuma operação é realizada.
- Se para todos os padrões de um grupo o protótipo mais próximo é de uma outra classe, o grupo será dividido em dois subgrupos (Figura 2.5). Essa divisão é feita separando os padrões pelo hiperplano H que passa pela média do grupo, cujo vetor normal à primeira componente principal gerada pelos padrões do grupo. O hiperplano H é criado usando a Equação 2.7.

$$H : \{x | x - \mu_x . \gamma_1 = 0\} \tag{2.7}$$

  Onde x representa os pontos do grupo original,μ_x é a média do grupo original e γ_1 é a primeira componente principal.

- Se para alguns padrões, o protótipo mais próximo é do outro grupo, mas da mesma classe, os padrões são migrados do grupo original para o grupo do protótipo mais próximo.

- Se para alguns padrões, o protótipo mais próximo é de outra classe, será formado um novo grupo com esses padrões e um novo protótipo será gerado a partir da média destes padrões.

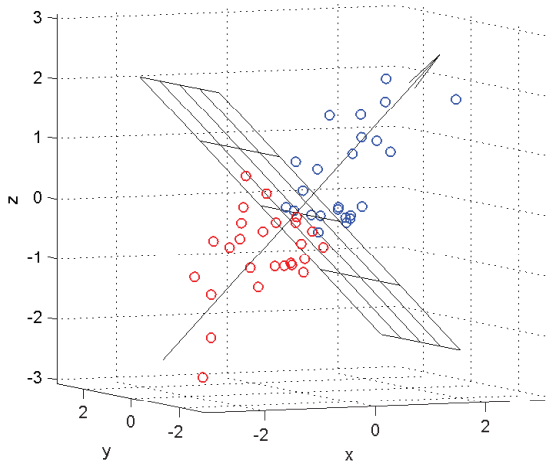


Figura 2.5 Divisão de grupo no SGP. A seta representa a primeira componente principal. [28]

Ao final de cada uma das condicionais acima, os protótipos são ajustados a partir da nova média que é recalculada. O processo é encerrado quando não houver mais mudanças nos grupos.

Com a finalidade de evitar *overfitting*, Fayed et al [12] introduziram dois parâmetros: R_{min} e R_{mis}. O primeiro estabelece o tamanho de um grupo em relação ao maior grupo. Desta forma se escolhermos um $R_{min} = 0,05$ e o maior grupo tiver 1000 exemplos, o tamanho para o menor grupo será de 50, sendo assim todos os grupos com tamanho inferior serão eliminados do processo. O parâmetro Rmis é o limiar para taxa de classificações incorretas de um grupo. Se o número de erros de classificação dividido pelo tamanho do grupo é menor que o Rmis, o grupo permanece inalterado. Segundo [28], a utilização desses parâmetros promove a redução da quantidade de protótipos e melhora a capacidade de generalização do SGP. Os valores sugeridos por Pereira e Darmiton [28] para os parâmetros são os seguintes: $0,01 < R_{min} < 0,2$ e $0,01 < R_{mis} < 0,2$. A versão básica do SGP é chamada de SGP1, mas [12] em seu artigo original, propõe uma versão do SGP otimizada, incluindo estratégias de poda *prunning* e fusão *merge*, esta versão é nomeada de SGP2.

A estratégia do SGP1 para a escolha dos protótipos é mostrada na tabela 2.1

- Inicialmente os exemplos são separados em dois grupos por classes.

Tabela 2.1 Padrões e classes correspondentes[12].

padrão	1	2	3	4	5	6	7	8	9	10	11	12
classe	1	1	1	2	2	2	1	1	2	2	1	1

– Tabela resultante da primeira divisão dos dados em dois grupos e suas respectivas médias.(Tabela 2.2)

Tabela 2.2 Divisão dos grupos iniciais e cálculo das médias

Grupo	Padrões							Classe	Média
G1	1	2	3	7	8	11	12	1	6,29
G2	4	5	6	9	10			2	6,8

Após a divisão inicial da base por classes, serão executados novas divisões baseadas na proximidade do Padrão×Média, isto é, proximidade do valor numérico do padrão com a média dos grupos.

– Os padrões 7,8,11,12 são eliminados do grupo G1, porque estão mais próximos da média 6,8. Os padrões eliminados formam um novo grupo, G3.(Tabela 2.3)

Tabela 2.3 Grupos formados após manipulação de G1.

Grupo	Padrões							Classe	Média
G1	1	2	3					1	6,29
G2	4	5	6	9	10			2	6,8
G3	7	8	11	12				1	9,5

– Os padrões 4,9,10 são eliminados do grupo G2, porque estão mais próximos da média 6,29 e 9,5, que são da classe 1. Os padrões eliminados formam um novo grupo, G4.(Tabela 2.4)

– Os padrões 7,8 são eliminados do grupo G3, porque estão mais próximos da média 6,5. Os padrões eliminados formam um novo grupo G5.(Tabela 2.5)

– O padrão 4 é movido para o grupo G2, pois a média mais próxima é 5,5 (G2) e G2 é da mesma classe do padrão 4. Já o padrão 10 é eliminado de G4, formando um novo padrão G5.(Tabela 2.6)

– Ao final do processo temos os protótipos selecionados são apresentados na Tabela 2.7

Tabela 2.4 Grupos formados após manipulação de G2.

Grupo	Padrões							Classe	Média
G1	1	2	3					1	2
G2	5	6						2	6,5
G3	7	8	11	12				1	9,5
G4	4	9	10					2	7,67

Tabela 2.5 Grupos formados após manipulação de G3.

Grupo	Padrões							Classe	Média
G1	1	2	3					1	2
G2	5	6						2	5,5
G3	11	12						1	11,5
G4	4	9	10					2	7,67
G5	7	8						1	7,5

Tabela 2.6 Grupos formados após manipulação de G4.

Grupo	Padrões							Classe	Média
G1	1	2	3					1	2
G2	5	6	4					2	5
G3	11	12						1	11,5
G4	9							2	9
G5	7	8						1	7,5
G6	10							2	10

Tabela 2.7 Protótipos Gerados

Classe	Protótipos (médias)
1	2
2	5
1	11,5
2	9
1	7,5
2	10

Algoritmo 10 SGP 1

Entrada: T: um conjunto de treinamento
Entrada: PS: uma lista para os representantes
Entrada: GS: uma lista de grupos de instâncias
Entrada: $TUPLAS$: uma lista de duplas de instâncias

Para todo classe C **faça**
  G = $\bigcup$ instâncias da classe C
  $Adicione(G, GS)$
  $Adicione(Centroide(G), PS)$
Fim Para
$count = 1$
Enquanto $count \neq 0$ **faça**
  $count$ = $Quantidade(GS)$
  $Limpe(TUPLAS)$
  Para todo G em GS **faça**
    P = Representante de G
    Para todo e_i em $group$ **faça**
      $NearestP$ = $1NN(e_i, PS)$
      $Adicione((e_i, NearestP), TUPLAS)$
    Fim Para
    Se $\forall\ e_i, NearestP$ em $TUPLAS$, $NearestPS$ = P **então**
      $count$ = $count - 1$
    Senão Se $\forall\ e_i, NearestP$ em $TUPLAS$, a $Classe(NearestP) \neq Classe(P)$ **então**
      $Vector = PrimeiraComponentePrincipal(G)$
      $Hiperplano$ = hiperplano que passa pelo centróide de G e cujo vetor normal é a $Vector$.
      Divida G em 2 grupos, instâncias acima e abaixo de $Hiperplano$
      Atualize GS e PS.
    Senão Se $\exists\ e_i, NearestP$ em T tal que $NearestP \neq P$ e $Classe(NearestP) = Classe(P)$ **então**
      Remova e_i de G e adicione a grupo de $NearestP$.
      Atualize GPS e PS.
    Senão Se $\exists\ e_i, NearestP$ em $TUPLAS$ tal que o $Classe(NearestP) \neq Classe(P)$ **então**
      Remova e_i de G.
      Crie um novo grupo contendo as instâncias removidas.
      Atualize PS e GS
    Fim Se
  Fim Para
Fim Enquanto
Retorne PS

2.7 POC-NN - *PAIRWISE OPPOSITE CLASS-NEAREST NEIGHBOR*

Visando uma redução no número de padrões armazenados em memória durante a fase de treinamento, pelo KNN, Raicharoen e Lursinsap [30], apresentam uma variação do KNN cuja abordagem é a de dividir para conquistar. Este algoritmo é chamado de Pairwise Opposite Class-Nearest Neighbor (POC-NN). Quando olhamos um problema de classificação com duas classes como o apresentado na Figura 2.6, podemos deduzir a qual classe um novo padrão apresentado pertencerá. Se o novo padrão apresentado for posicionado à esquerda, provavelmente, sua classe será azul, caso esteja posicionado à direita, possivelmente, sua classe será a vermelha. A reta traçada separa as regiões de classificação.

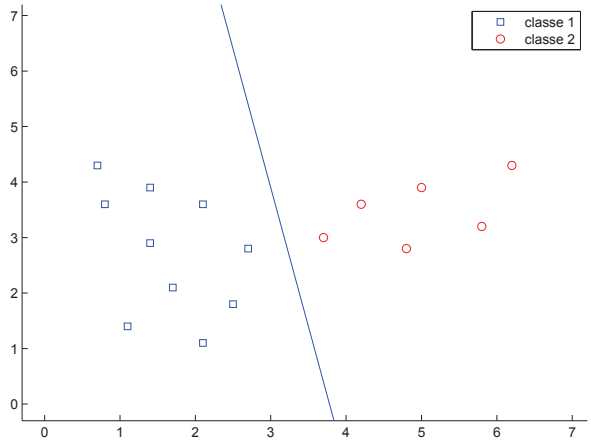


Figura 2.6 Possível solução de um problema de classificação com duas classes distintas.

Como verificado no exemplo dado anteriormente 2.6, o problema de classificação é resolvido, dividindo-se o hiperespaço em regiões menores de classificação que determinam a classe dos seus padrões.

No artigo [30], encontramos duas variações do POC-NN, a primeira é uma técnica de seleção de protótipos denominada de S-POC-NN *(Selection)*, e a segunda uma técnica de geração de protótipos denominada de R-POC-NN *(Replacement)*. As duas técnicas buscam armazenar padrões que possuam uma maior capacidade de generalização, diminuindo assim o número de protótipos e, consequentemente, o espaço necessário em memória de computador, assim como os recursos computacionais. O POC-NN tem sua fase de treinamento dividida em duas etapas: Finding-POC-NN, Selecting-POC-NN ou Replacing-POC-NN, dependendo do método implementado: S-POC-NN ou R-POC-NN.

2.7.1 *Finding-POC-NN* (F-POC-NN)

Para a obtenção do conjunto POC-NN (x_{p1}, x_{p2}) o procedimento adotado é o seguinte: Dado um conjunto de treinamento T de duas classes distintas Classe 1 (C1) e Classe 2 (C2) de modo que $C1 \cap C2 = \phi$ e q_1 é a quantidade de padrões contidas em C1 e q_2 é a quantidade de padrões contidas em C2. A Figura 2.7 ilustra o resultado da execução do algoritmo Finding-POC-NN. Inicialmente o conjunto de treinamento T é dividido nas duas classes C1 e C2, após a divisão é calculada a média dos padrões da classe com maior quantidade de padrões (X_m). Como a classe de maior quantidade de padrões é a C1, x_{p2} será o padrão de C2 mais próximo de X_m e o x_{p1} será o padrão de C1 mais próximo de x_{p2}.

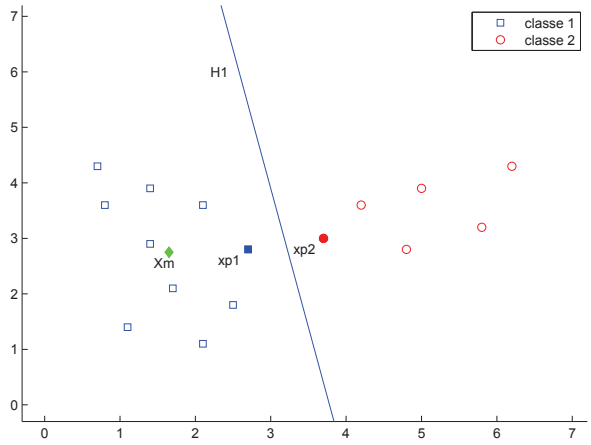


Figura 2.7 Funcionamento da função *Finding-POC-NN*.

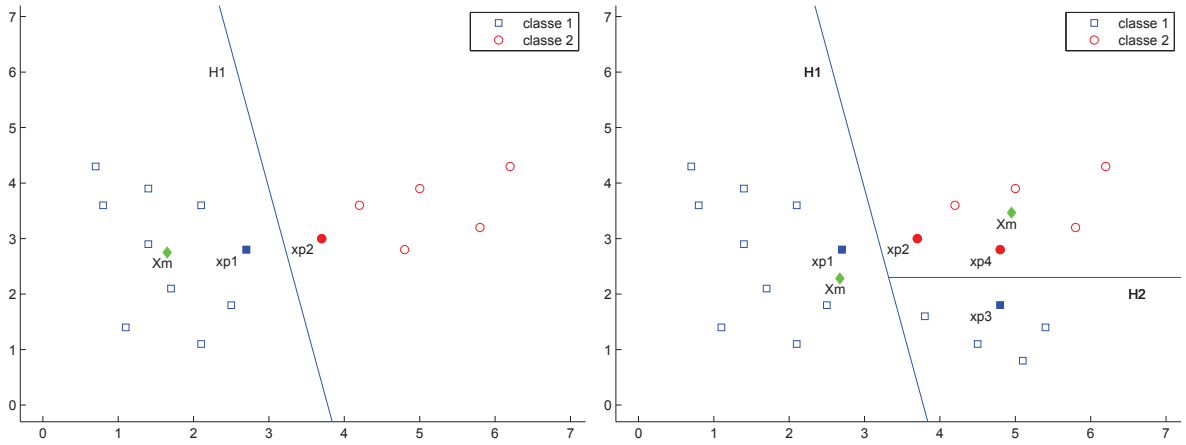
Algoritmo 11 Finding-POC-NN (F-POC-NN)

Entrada: T: um conjunto de treinamento
Entrada: P: Padrão POC-NN(xp1,xp2)
1: $[C1, C2] = Divide(T)$ {Divide em duas classes C1 e C2}
2: $q1 = Tamanhodaclasse C1$
3: $q2 = Tamanhodaclasse C2$
4: **Se** $q1 \geq q2$ **então**
5: X_m = Protótipo médio de C1
6: Xp_2 = padrão de C2 mais próximo de X_m
7: Xp_1 = padrão de C1 mais próximo de Xp_2
8: **Senão**
9: X_m = Protótipo médio de C2
10: Xp_1 = padrão de C1 mais próximo de X_m
11: Xp_2 = padrão de C2 mais próximo de Xp_1
12: **Fim Se**
13: **Retorne** (Xp_1, Xp_2) {protótipos POC-NN}

Na Figura 2.7 verificamos que os pontos x_{p1} e x_{p2} são vizinhos mais próximos de classes opostas(POC-NN). Após os padrões POC-NN terem sidos encontrados, é traçado um hiperplano ortogonal à x_{p1} e x_p2, posicionado exatamente na distância média desses pontos. Esse hiperplano será usado pelo classificador como o limite de decisão. Um detalhe que podemos observar é que independente da ordem de apresentação dos padrões do conjunto de treinamento, é que o padrão POC-NN (x_{p1}, x_{p2}) são sempre os mesmos.

2.7.2 *Selecting-POC-NN*

A função do *Selecting-POC-NN* é classificar padrões de duas classes, mas pode ser usado recursivamente para dividir padrões multiclasses. Essa divisão das regiões de classificação é executada até que todas as regiões estejam corretamente classificadas. A função recebe um conjunto de treinamento T composto por n padrões de duas classes distintas, retornando um POC-NN-SET. Esse conjunto é composto pelos protótipos e respectivos hiperplanos. (Figura 2.8). O funcionamento do Selecting-POC-NN pode ser visto no Algoritmo 12.



(a) Separação inicial pelo hiperplano H1. (b) Segunda separação pelo hiperplano H2.

Figura 2.8 Funcionamento da função *Selecting-POC-NN*.

Algoritmo 12 selecting-POC-NN (S-POC-NN)

Entrada: T: um conjunto de treinamento

1: Encontrar Protótipos POC-NN (Xp_1,Xp_2)=Finding-POC-NN(T)
2: Determinar o ponto central c = $\frac{Xp_1 + Xp_2}{2}$
3: Criar um hiperplano H:$\{x|w.x - b = 0\}$, onde: w = $\frac{Xp_1 - Xp_2}{\|Xp_1 - Xp_2\|}$ e b=w.c
4: Salvar no POC-NN-SET os protótipos (Xp_1, Xp_2) e o hiperplano H
5: Divide T em duas regiões distintas, denominadas R1 e R2, onde $R1 = \{x_i \in T|w.x_i - b \geq 0\}$ e $R1 = \{x_i \in T|w.x_i - b < 0\}$ $\forall$ i variando de 1..n
6: **Se** Algum padrão em R1 é classificado Erroneamente **então**
7: Considerar R1 o novo conjunto de Dados
8: Selecting-POC-NN(R1)
9: **Fim Se**
10: **Se** Algum padrão em R2 é classificado Erroneamente **então**
11: Considerar R2 o novo conjunto de Dados
12: Selecting-POC-NN(R2)
13: **Fim Se**
14: **Se** $\nexists$ padrão classificado erroneamente **então**
15: **Retorne** POC-NN-SET
16: **Fim Se**

2.7.3 Intervalo de Aceitação

Para reduzir a complexidade e a sensibilidade a ruídos do POC-NN, foi introduzido o parâmetro, intervalo de aceitação, transformando o hiperplano H em uma faixa de largura

α definida por $\alpha = \alpha - ratio \times D$ onde D é a distância entre os protótipos *POC-NN*, x_{p1} e x_{p2} e $\alpha - ratio$ é um parâmetro variando entre $0 < \alpha - ratio < 1$. Na Figura 2.9 é mostrado um exemplo onde o $D = 1,79$, e foi atribuído $\alpha - ratio = 0,1$ obtendo assim $\alpha = 0,179$, definindo assim uma faixa de aceitação. A função *Selecting-POC-NN*, encontrou uma instância de dentro deste intervalo, e esta será desprezada por estar dentro do intervalo de aceitação.

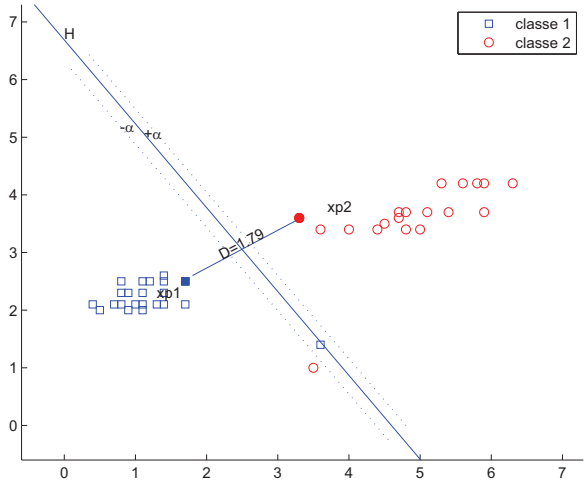


Figura 2.9 Divisão das classes utilizando $\alpha - ratio = 0,1$.

2.7.4 *Replacing-POC-NN* (R-POC-NN)

Uma outra abordagem do POC-NN é a R-POC-NN, essa abordagem utiliza a função Replacing-POC-NN [30] e apresenta vantagens em relação à Selecting-POC-NN, pois em alguns casos, temos melhores resultados e um armazenamento de um menor número de protótipos em memória, visto que enquanto o Selecting-POC-NN ao dividir as regiões seleciona protótipos que as representam, e o Replacing-POC-NN calcula um único protótipo para representar a região, esse protótipo é a média dos protótipos da região e não precisa estar presente no conjunto de treinamento, este é denominado de protótipo de troca. O algoritmo do Replacing-POC-NN é mostrado no Algoritmo 13 que é uma modificação do Selecting-POC-NN, onde o conjunto de retorno deixa de ser o POC-NN-SET e passa a ser o MOR-NN-SET, que é um conjunto de padrões médios que representam cada região. Os protótipos de troca, X_{mor} são mostrados na Figura 2.10, esses protótipos de troca foram gerados por Replacing-POC-NN.

Algoritmo 13 Replacing-POC-NN (R-POC-NN)

Entrada: T: um conjunto de treinamento

1: Encontrar Protótipos POC-NN (Xp_1,Xp_2)=Finding-POC-NN(T)
2: Determinar o ponto central c = $\frac{Xp_1 + Xp_2}{2}$
3: Criar um hiperplano H:$\{x|w.x - b = 0\}$, onde: w = $\frac{Xp_1 - Xp_2}{\|Xp_1 - Xp_2\|}$ e b=w.c
4: Salvar no POC-NN-SET os protótipos (Xp_1, Xp_2) e o hiperplano H
5: Divide T em duas regiões distintas, denominadas R1 e R2, onde $R1 = \{x_i \in T|w.x_i - b \geq 0\}$ e $R1 = \{x_i \in T|w.x_i - b < 0\}$ $\forall$ i variando de 1..n
6: **Se** Algum padrão em R1 é classificado Erroneamente **então**
7: Considerar R1 o novo conjunto de Dados
8: Replacing-POC-NN(R1)
9: **Senão**
10: Salve X_{mor}, média dos padrões de R1 no MOR-NN-SET
11: **Fim Se**
12: **Se** Algum padrão em R2 é classificado Erroneamente **então**
13: Considerar R2 o novo conjunto de Dados
14: Replacing-POC-NN(R2)
15: **Senão**
16: Salve X_{mor}, média dos padrões de R2 no MOR-NN-SET
17: **Fim Se**
18: **Se** $\nexists$ padrão classificado erroneamente **então**
19: Salve X_{mor}, média dos padrões de R1 no MOR-NN-SET
20: Salve X_{mor}, média dos padrões de R2 no MOR-NN-SET
21: **Fim Se**
22: **Retorne** POC-NN-SET

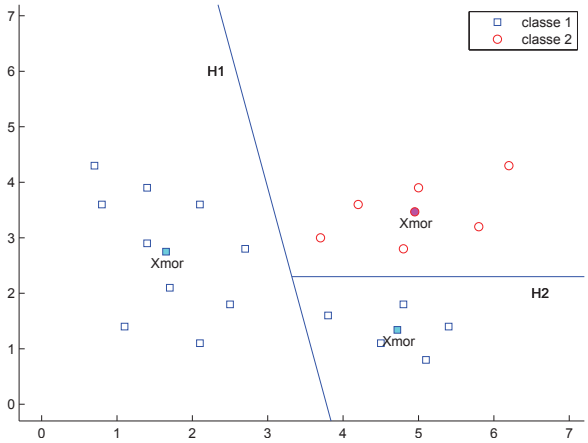


Figura 2.10 Funcionamento da função *Replacing-POC-NN (R-POC-NN)*.

2.7.5 Classificação de Padrões Multiclasses

As funções Selecting-POC-NN e Replacing-POC-NN funcionam fazendo classificações com duas classes. Para resolver o problema de classificação de padrões de diversas classes é utilizada técnica one-against-one (1-n-1), que realiza a combinação de classificadores binários. Na Figura2.11 é mostrado um exemplo da utilização da técnica para classificação envolvendo quatro classes.

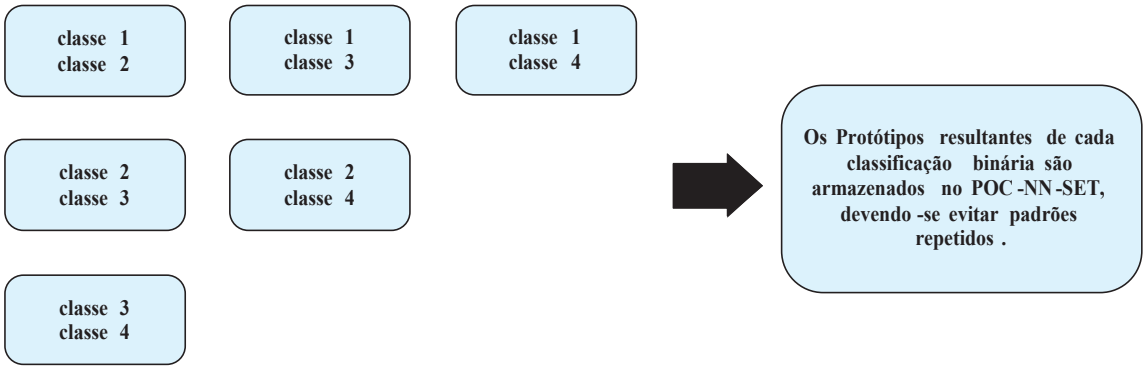


Figura 2.11 Exemplo de utilização da técnica *One-against-one* para problemas de quatro classes.

A quantidade de classificadores binários ou combinações é dada pela Equação 2.8, onde C é a quantidade de classificadores binários e K é o número de classes envolvidas no problema de classificação.

$$C = \frac{K(K-1)}{2} \tag{2.8}$$

Conforme mostrado na Figura 2.11 para a classificação envolvendo 4 classes será necessária a utilização de 6 classificadores binários.

$$C = \frac{4(4-1)}{2} = \frac{4 \times 4}{2} = \frac{12}{2} = 6 \tag{2.9}$$

Em relação ao classificador K-NN, o POC-NN apresenta vantagem de reduzir a necessidade de memória, pois armazena menos padrões, além da redução do custo computacional, mas as taxas de erro apresentadas pelo POC-NN são maiores que as apresentadas pelo K-NN, devendo-se analisar a relação custo x benefício da utilização de um método ou outro.

2.8 RSP - *REDUCTION BY SPACE PARTITIONING*

O método proposto por J.S.Sanchez [31] também visa a redução do custo computacional e do espaço em memória necessário para o armazenamento de todas as instâncias necessárias ao funcionamento do classificador NN. A geração nos algoritmos propostos é baseada em diferentes heurísticas de particionamento de espaço de características, e várias técnicas para a definição de protótipos. A técnica foi inspirada na técnica proposta por Chen e Józwik [8], mas tentando evitar mudanças na forma da fronteira de decisão associada ao conjunto de treinamento T, o que constitui uma das principais desvantagens ligadas ao algoritmo. Os esquemas de redução foram denominados de RSP1-RSP3.

2.8.1 RSP1 - *Diameter of a Set*

Um dos problemas associados à heurística introduzida por Chen e Józwik [8] refere-se ao fato de que, em alguns casos práticos, todas as instâncias de uma classe podem ser descartadas do conjunto de treinamento T. E, consequentemente, o classificador falhará em todas as amostras que pertencem à classe eliminada[31]. Para tentar solucionar esse problema o RSP1 foi proposto. O seu funcionamento pode ser observado no Algoritmo 14, que inicia o processo de divisão calculando o diâmetro de T. Esse diâmetro é a distância entre os dois pontos de T mais afastados, cujo exemplo é mostrado na Figura 2.12.

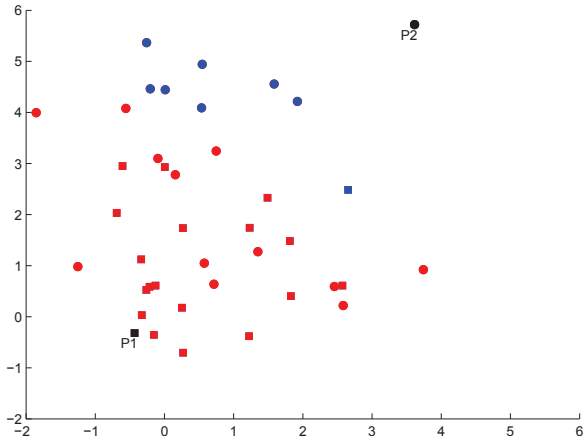


Figura 2.12 Pontos mais distantes do conjunto de treinamento T.

Após encontrar os dois pontos (P1 e P2), T é dividido em dois blocos, cujo critério de separação é a proximidade da instância com relação a um dos pontos. O bloco 1 conterá todas as instâncias mais próximas de P1 e o bloco 2 conterá a diferença. O processo de divisão é repetido até a obtenção do número de blocos especificado pelo parâmetro ”b”.

Esse parâmetro é definido de forma empírica para cada base a ser classificada. Após o processo de divisão, o RSP1 calcula os centróides dos pontos de cada classe contidos no bloco, adicionando-os ao conjunto de protótipos por blocos c.

Esta técnica de redução utiliza uma heurística para garantir que nenhuma das classes estará vazia após a aplicação do algoritmo.

Algoritmo 14 RSP1

Entrada: T: um conjunto de treinamento
Entrada: b: número de blocos a serem gerados
1: Faça $bc = 1$, $i = 1$, $B = T$
2: Encontre os pontos de maior distância entre si, P1 e P2, em B
3: Divida B em dois subconjuntos B1 e B2 onde:
4: $B1 = \{P \in B : d(P, P1) \geq d(P, P2)\}$
5: $B2 = \{P \in B : d(P, P2) < d(P, P1)\}$
6: Faça $bc = bc + 1$, $C(i) = B1$, $C(bc) = B2$
7: Faça $I1 = \{i : C(i)$ contendo instâncias de pelo menos duas classes diferentes $\}$
8: Faça $I2 = \{i : i \leq bc\} - I1$
9: **Se** $I1 \neq \emptyset$ **então**
10: Faça $I = I1$
11: **Senão**
12: Faça $I = I2$
13: **Fim Se**
14: Encontre os pontos de maior distância entre si, q1e q2 para cada C(i), $i \in I$.
15: Encontre o maior diâmetro de C(j)
16: Faça $B = C(j)$, $P1 = q1(i)$ e $P2 = q2(j)$
17: **Se** $bc < b$ **então**
18: Volta para o passo 6
19: **Fim Se**
20: Encontre os centróides de c(I,i) para cada classe I no subconjunto C(i)
21: S = Todos os centróides c(I,i)

2.8.2 RSP2 - *Overlapping Degree*

Na versão anterior do algoritmo RSP, o critério de divisão do grupo (passos 12 e 13) é o seguinte: dividir o maior diâmetro. A ideia é que o maior subconjunto, provavelmente, deve conter mais exemplos de que qualquer outro subconjunto e, portanto, apresentando maior taxa de redução. No entanto, a teoria dita que instâncias de uma mesma classe devem estar mais próximas o possível, enquanto instâncias de classes diferentes, devem estar mais distantes, tanto quanto possível.

Por conseguinte, de um ponto de vista teórico, isto pode ser mais apropriado para dividir o subconjunto com a mais alta sobreposição de grau entre instâncias de classes distintas. Nesta versão do algoritmo RSP, o critério de divisão dos grupos passa a ser o *overlapping degree* (grau de sobreposição) e é definido como a razão entre as distância média entre instâncias de classes diferentes, $D^{\neq}$, e a distância média entre as intâncias de mesma classe, $D^{=}$.

2.8.3 RSP3 - *Class Homogeneity*

Esta é a terceira heurística de particionamento do RSP e consiste em dividir os subconjuntos até que todos os elementos sejam da mesma classe. O RSP3 eliminou o parâmetro b, que define o número de subgrupos para a divisão, presente nas versões RSP1-RSP2. No RSP3 o critério de divisão dos subgrupos pode ser qualquer um dos citados anteriomente: Método do maior diâmetro ou Método do grau de sobreposição, pois como todos os subgrupos com mais de uma classe serão divididos até ficarem homogêneos. O número de protótipos gerados por esta abordagem geralmente é maior que nas outras abordagens do RSP. Essa abordagem é sensível a ruídos, pois como a rotina divide os subgrupos até os mesmos tornarem-se homogêneos, podemos ter subgrupos com apenas uma instância, podendo esta instância ser um ruído. Devido à sensibilidade a ruídos da técnica, Sanchez [31] sugere a utilização de um filtro antes da execução do RSP. O algoritmo RSP3 mostra como é feita a divisão dos subgrupos e geração dos protótipos. (Algoritmos 15 - 16).

Algoritmo 15 RSP3

Entrada: T: um conjunto de treinamento

1: Faça $B = T$
2: $C = RSPDivideGrupos(C, B)$
3: $S = []$
4: **Para** Subgrupo C(i) **faça**
5: Acrescente a S o centróide de cada subgrupo C(i)
6: **Fim Para**
7: **Retorne** S

Algoritmo 16 RSPDivideGrupos

Entrada: C: Vetor de subgrupos resultantes das divisões.
Entrada: B: Subgrupo a ser dividido.
1: Faça $B = T$
2: Faça $CR = C$
3: Encontre os pontos mais distantes P1 e P2, em B
4: Divida B em dois subconjuntos B1 e B2 onde:
5: $B1 = \{P \in B : d(P, P1) \geq d(P, P2)\}$
6: $B2 = \{P \in B : d(P, P2) < d(P, P1)\}$
7: **Se** B1 é homogêneo **então**
8: Adiciona B1 a CR
9: **Senão**
10: CR = RSPDivideGrupos(CR,B1)
11: **Fim Se**
12: **Se** B2 é homogêneo **então**
13: Adiciona B2 a CR
14: **Senão**
15: CR = RSPDivideGrupos(CR,B2)
16: **Fim Se**
17: **Retorne** CR

2.9 ICPL - *INTEGRATED CONCEPT PROTOTYPE LEARNER*

Visando reduzir os problemas inerentes dos classificadores NN como o alto custo computacional, a exigência do armazenamento de todas as instâncias do conjunto de treinamento e sensibilidade a ruídos, Wai Lam (et.al.) [22] propuseram o framework ICPL (integrated Concept Prototype Learner) que utiliza duas abordagens de aprendizagem de máquina: filtragem e abstração. A filtragem é uma técnica de redução de dados, retendo do conjunto de treinamento os exemplos mais representativos. E a abstração é uma abordagem que reduz o conjunto de treinamento, gerando protótipos artificiais que resumam características representativas das classes. Com relação à técnica de abstração utilizada no Framework ICPL, os métodos existentes como *clustering* possuem um alto custo computacional e são incapazes de manipularem grandes conjuntos de dados [10]. Para minimizar esse problema foi proposto um método de abstração, baseado em tipicidade, com custo relativamente baixo. [22]

2.9.1 Filtragem

Métodos de filtragem de instâncias fazem uso de regras de edição para selecionar instâncias representativas. Algoritmos de filtragem de instâncias diferem da direção da pesquisa e da localização da instância selecionada. Direção de pesquisa incluem: inclusão e remoção de instância e a combinação da inclusão e remoção. Em termos de localização de instâncias selecionadas, Wai Lam (et.al.)[22] identificaram três categorias de retenção:

- Instâncias de Centro
- Instâncias de fronteira
- Instâncias não-borda (*nonborder*)

Algumas técnicas de filtragem que podem ser utilizadas no ICPL são o *Condensed Nearest Neighbor* (CNN), provavelmente, um dos mais simples métodos de seleção de instâncias representativas [18]. O CNN busca a eliminação dos elementos mais afastados das fronteiras de classificação, mantendo os elementos nas regiões de dúvida. Essa abordagem torna a técnica sensível a ruídos. Uma variante do CNN, o *Reduced Nearest Neighbor*-RNN proposta por Gates [16], é uma técnica que remove instâncias, se esta remoção não causa erro de classificação de outras instâncias. A Edited Nearest Neighbor-ENN proposta por Wilson[36], é uma técnica de filtro que elimina instâncias classificadas erroneamente pelo K-NN, eliminando ruídos, possui baixa taxa de redução. A técnica Tipical Instance-Based Learning introduzida por Zhang[39], armazena instâncias das regiões centrais, em detrimento das zonas de bordas de decisão. Podemos também utilizar na etapa de filtragem os filtro IB2 e IB3 propostos por Aha et al., que são tolerantes a ruído, mas sensíveis a ordem de apresentação[1]. Wilson e Martinez introduzem o RT3 que inicialmente utiliza o filtro ENN para eliminar ruídos, e remove um exemplo, se esta remoção não afetar a classificação de seus associados. Essa técnica deu origem DROP1-DROP5[37]. Métodos de filtragem são geralmente simples e rápidos e devem ter seus comportamentos investigados, com relação à precisão, sensibilidade a ruídos, overfitting, sensibilidade à ordem de apresentação de dados, taxa de redução, etc.

2.9.2 Abstração

A abstração é outra abordagem utilizada para encontrar exemplos representativos dos dados originais. Essa abordagem gera protótipos a partir das instância originais.

Wai Lam et at. [22] desenvolveram uma técnica de abstração ou geração de protótipos, denominada *Tipical Prorotype Abstration*-(TPA), que se mostrou ser eficiente na generalização, além de ser tolerante a ruídos. Este filtro utiliza a medida de tipicidade proposta por Zhang [39]. Para calcularmos a tipicidade, utilizamos os conceitos de similaridade de duas instâncias x e y, $Sim(x, y)$ definida na Equação2.10.

$$Sim = 1 - Dist(c, y), \tag{2.10}$$

onde $Dist(x, y)$ é a distância Euclidiana, para dados contínuos. A Equação 2.11 define a tipicidade.

$$Tip(i) = \frac{MSMC(i)}{MSCD(i)} \tag{2.11}$$

onde $MSMC(i)$ é a média das similaridades de *i* para instâncias de mesma classe e $MSCD(i)$ é a média das similaridades de "i"para instâncias de classes diferentes.

Propriedade da tipicidade:

- $Tipicidade > 1$, instâncias típicas.
- $Tipicidade = 1$, instâncias próximas a regiões de borda.
- $Tipicidade < 1$, instâncias ruidosas.

A partir da tipicidade, Wai Lam et al. desenvolveram uma função *IdentifyBorder* para identificar instâncias de bordas por classe (Algoritmo 17).

Algoritmo 17 IdentifyBorder

Entrada: C: Vetor de subgrupos resultantes das divisões.
Entrada: BP: Vetor contendo a informação se a instância *i* é ou não borda.

Para cada instância I **faça**
  Faça $Typ(I) = Tipicidade(I)$
Fim Para
Para cada Classe C **faça**
  Tmean = A Tipicidade médias das instâncias da classe C
  Tsd = O desvio padrão das Tipicidade médias das instâncias da classe C
  Ordene C em ordem descrescente de tipicidade
Fim Para
Para cada Classe C **faça**
  Para cada I da classe C em T **faça**
    Se $Typ(I) < (Tmean(C) - Tsd(C))$ **então**
      Faça $Bp(I) = True$, indica que a instância é borda
    Senão
      Faça $Bp = False$
    Fim Se
  Fim Para
Fim Para
Retorne Bp

O TPA é mostrado no algoritmo 18. Inicialmente, a função IdentifyBorder é executada para identificar as regiões de bordas. Posteriormente, as instâncias de maior tipicidade por classe são processadas, mesclando as instâncias não-bordas. Como resultado dessa mesclagem temos os protótipos, resultante da média das instâncias. O algoritmo termina quando todas as instâncias não-borda são processadas. A função *Merge* mostrada no Algoritmo 19 recebe como parâmetros, o conjunto de treinamento T, o conjunto de protótipos S, um vetor lógico contendo a informação de quais instâncias de T são bordas, e uma instância T. Essa função funde continuamente a instância I com seus vizinhos mais próximos até que seja encontrada uma instância de outra classe ou de borda.

Algoritmo 18 TPA

Entrada: T: Conjunto de treinamento.

```
1: Faça S = []
2: Faça Bp = IdentifyBorder(T)
3: Para cada Classe C faça
4:    Faça I =primeira instância da classe C em T
5:    Para cada instância I não-borda faça
6:       Se I não processado então
7:          Faça P=Merge(T, S, Bp, I)
8:          Se Número de Instâncias em P¿1 então
9:             Adicione P em S
10:         Fim Se
11:      Fim Se
12:      Faça I= próxima instância da classe C em T
13:   Fim Para
14: Fim Para
15: Retorne S
```

Algoritmo 19 Merge

Entrada: T: Conjunto de treinamento.
Entrada: S: Conjunto de protótipos.
Entrada: Bp: Vetor contendo valores que indicam se a instância é borda.
Entrada: I: Instância.

```
1: Faça P = I
2: Faça N=Vizinho mais próximo de I em T
3: Faça C=classe de I
4: Enquanto (Classe de N<> C) ou (Não Bp(N)) faça
5:    Se (Classe de N <> C) então
6:       Faça N = Próximo viznho mais próximo de I em T
7:    Senão
8:       Encontre M, protótipo contendo N de S
9:       Merge P e M
10:      Remove M de S
11:      Retorne P
12:   Fim Se
13: Fim Enquanto
14: Retorne P
```

Os estudos realizados por Wai Lam (et.al.)[22] indicaram que o ICPL apresenta bons resultados com relação à acurácia, pois a precisão do classificador é melhorada bastante quando integramos a abstração à filtragem.

2.10 GENN - GENERALIZED EDITING NEAREST NEIGHBOR

A regra do vizinho mais próximo (NN - Nearest Neighbor) é uma técnica das mais antigas e conhecidas para a resolução de problemas de classificação, mas apresenta problemas com relação a ser bastante influenciada por ruídos e por ter que armazenar grande quantidade de dados. A proposta de Koplowitz, Jack e Brow, Thomas [21] é de criar um classificador mais robusto que possa corrigir erros ou omissões no conjunto de treinamento (T), sendo capaz de adaptar-se às alterações do ambiente. A regra NN é empregada como classificador central, devido a sua flexibilidade, pois o processo de aprendizagem deve continuar automático. Mas esta modificação do NN, produz não só a eliminação de instância mas também a mudança de *label* da classe. Nesta técnica temos dois parâmetros k e k' de forma que $\frac{(k+1)}{2} \leq k' \leq k$. Para cada protótipo x_i de T seus k-vizinhos mais próximos são procurados em T. Se determinada classe tem representantes de pelo menos k' entre os k vizinhos mais próximos, este protótipo será reidentificado de acordo com a classe dos seus vizinhos. Caso contrário x_i será eliminado. Resumindo, a técnica busca modificações da estrutura da amostra do treinamento através da mudança de rótulos de alguns padrões e remoção de outros. Verificamos que essa técnica apresenta boas taxas de classificação, mas isso ocorre devido a baixíssima taxa de redução, fazendo com que quase todas as instâncias do conjunto de treinamento sejam utilizadas como conjunto de protótipos. O pseudocódigo é mostrado no Algoritmo 20.

Algoritmo 20 GENN

Entrada: T: Conjunto de treinamento.
Entrada: K: k-vizinhos mais próximos.
Entrada: $Klinha$: k'-vizinhos mais próximos.

```
P=[]
Para todo x_i ∈ T faça
    Dados = T − x_i
    Pontos=KvizinhosMaisPróximos(k,T,Dados)
    Verifica a classe majoritária em Pontos
    Se Número de instâncias da Classe majoritária > Klinha então
        P = [P,Dados[i] classeMajoritária]
    Fim Se
Fim Para
Retorne P
```

CAPÍTULO 3

METODOLOGIA PARA AVALIAR RELAÇÃO ENTRE ALGORITMOS DE GERAÇÃO E DE 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. Neste capítulo 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.

3.1 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;
- Υ - 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*.

Para validar a nossa hipótese de que, muitas vezes, o protótipo gerado possui instância muito próxima, que poderia substituí-lo, tornando desnecessária a sua geração, utilizamos a seguinte estratégia: a partir do conjunto de protótipos gerados C_P por uma técnica de geração de protótipos, selecionamos para cada protótipo $p \in C_P$ no conjunto de treinamento Γ a instância mais próxima ao protótipo p de modo que a classe do protótipo p seja igual a classe da instância x_I. Ao encontrarmos essa instância, adicionamos esta ao conjunto de instâncias selecionadas C_I. Na Figura 3.1 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 o p, sendo escolhido como x_I, aquela de menor d ,ou seja, a mais próxima de p.

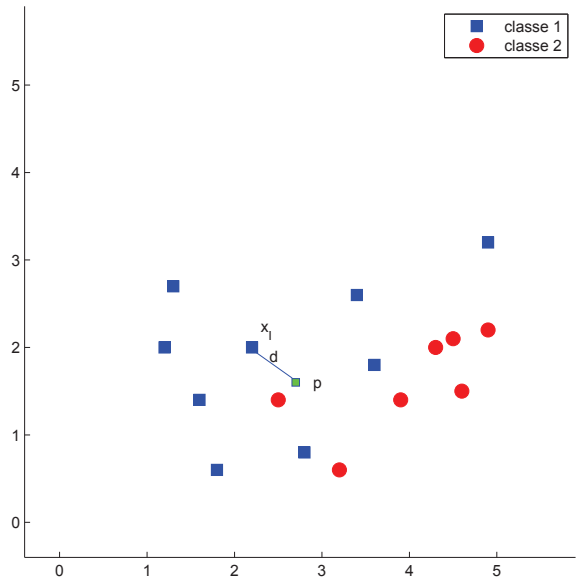


Figura 3.1 Seleção da instância x_I.

O conjunto de teste Υ é avaliado usando o classificador vizinho mais próximo NPC [33]. Cada um dos conjuntos C_P e C_I é usado como conjunto de referência. 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. Usamos como hipótese nula $H_0 : \mu_0 = \mu_1$ e como hipótese alternativa $H_a : \mu_0 \neq \mu_1$ e o 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, o resultado obtido pela técnica de geração é estatisticamente igual ao obtido por seleção de instâncias. Caso contrário, as médias das taxas de erro são diferentes, e neste caso, a melhor técnica será a que apresentar a menor taxa de erro. O pseudocódigo é mostrado no Algoritmo 21.

Algoritmo 21 Geração $\times$ Seleção

Entrada: Γ: um conjunto de treinamento
Entrada: Υ: um conjunto de teste
1: */ Execução de uma técnica de geração /*
2: $C_P = Geracao(\Gamma)$
3: $C_I = []$
4: **Para** $p \in C_P$ **faça**
5: $x_I = VMPMC(p, \Gamma)$
6: $C_I = C_I \cup \{x_I\}$
7: **Fim Para**
8: $[e_G, s_G] = NPC(C_P, \Upsilon)$
9: $[e_I, s_I] = NPC(I_P, \Upsilon)$
10: */ Verifica a hipótese nula das Taxas de Erros serem iguais /*
11: **Se** $TStudent(e_I, s_I, e_G, s_G, \alpha, GL) == True$ **então**
12: */ Como as Taxas de Erro são iguais, e a seleção pode substituir a geração /*
13: **Senão**
14: **Se** $E_G < E_I$ **então**
15: */ Neste caso a geração obteve melhores resultados /*
16: **Senão**
17: */ Neste caso a seleção obteve melhores resultados /*
18: **Fim Se**
19: **Fim Se**

O algoritmo começa com a inicialização do conjunto de treinamento Γ, conjunto de teste Υ. O próximo passo é a obtenção do conjunto de protótipos a partir de uma técnica de geração de protótipos (linha 3), então, de posse do conjunto de protótipos gerados C_P, utilizamos a função VMPMC (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 Υ ao classificador NPC - *Nearest Prototype Classifier*, 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). Nas linhas 11 a 19, 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 22. 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 22 VMPMC

Entrada: Γ: um conjunto de treinamento
Entrada: p: protótipo $p \in C_P$
1: $T = \Gamma$
2: **Para** $t \in T$ **faça**
3: d = Distancia-euclidiana(p, t)
4: $D = D \cup \{d\}$
5: **Fim Para**
6: */ Ordena T na ordem crescente das distâncias de D /*
7: Ordena(T,D)
8: **Para** $t \in T$ **faça**
9: **Se** Classe(p)==Classe(t) **então**
10: **Retorne** t
11: **Fim Se**
12: **Fim Para**

3.2 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, utilizaremos a base artificial Banana [3]. A Figura 3.2 mostra as instâncias do conjunto de treinamento (80% da base).

A partir do conjunto de treinamento, utilizamos a técnica de geração de protótipos SGP2,

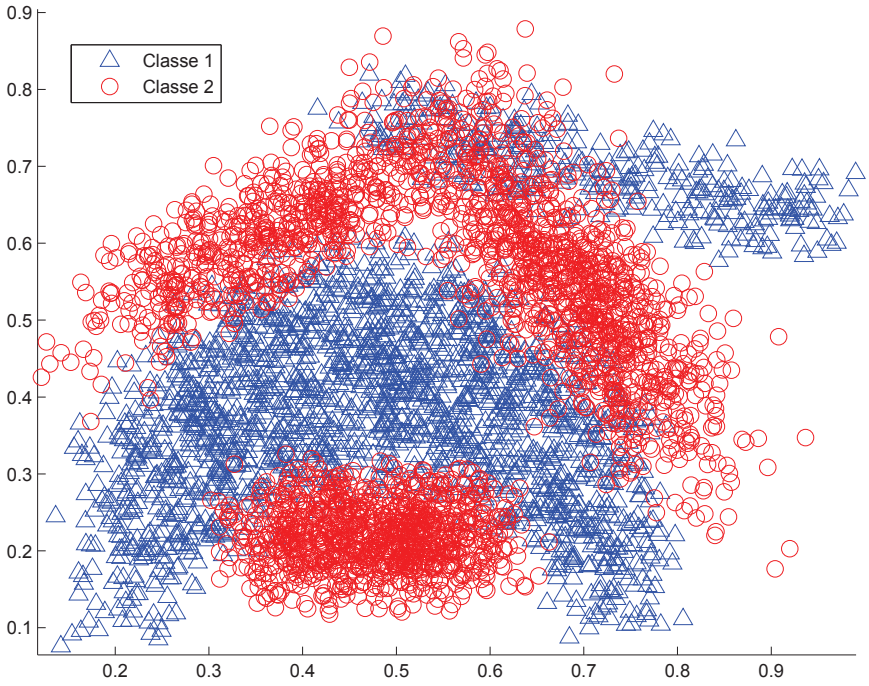


Figura 3.2 Distribuição das instâncias do conjunto de treinamento da base banana.

que gerou 44 protótipos, que foram plotados na Figura 3.3.

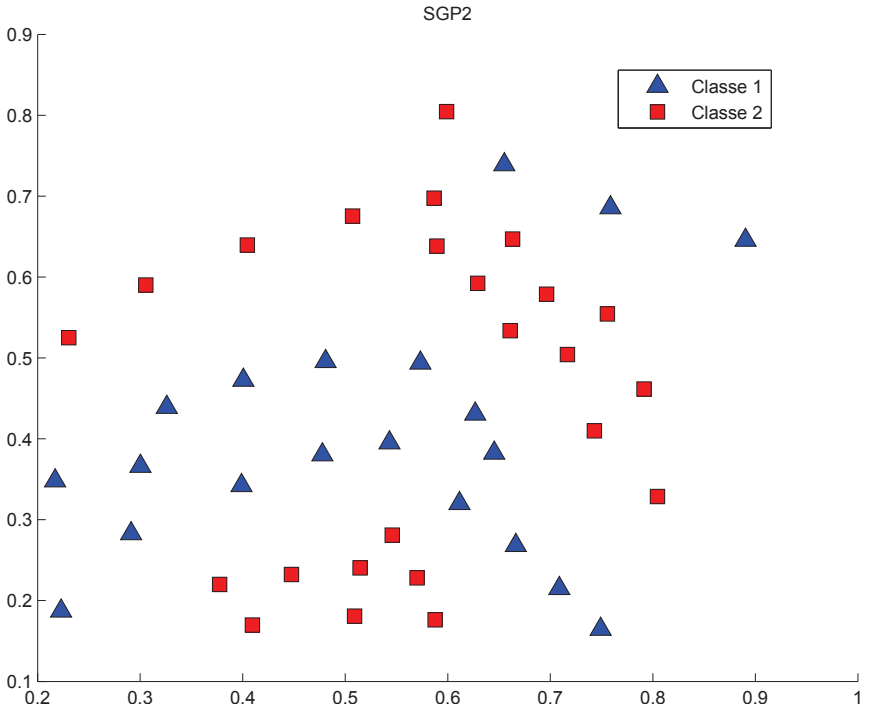


Figura 3.3 Protótipos gerados pela técnica SGP2.

Para cada protótipo $p \in C_P$ foram encontradas as instâncias x_I. Verifique na Figura 4.4(b), que as instâncias x_I estão muito próximas dos protótipos p gerados.

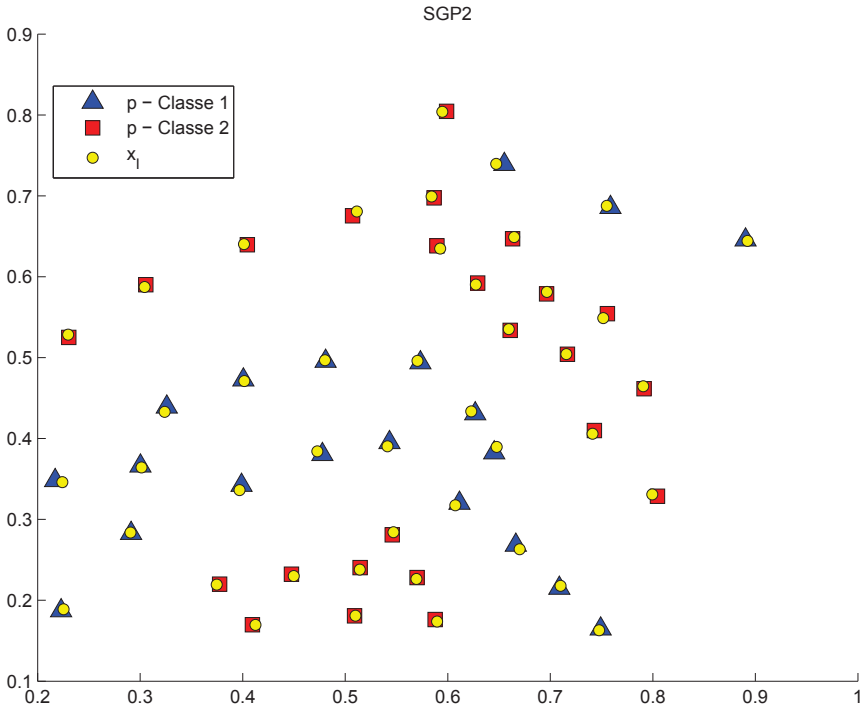


Figura 3.4 Instâncias x_I selecionadas a partir dos protótipos gerados.

Na Tabela 3.1 são apresentados os protótipos gerados, com seus atributos, e as instâncias selecionadas x_I com seus atributos, a classe do protótipo e a distância "d" entre os protótipos p e suas instâncias x_I. A partir dessa tabela, podemos verificar a proximidade das instâncias aos protótipos. Finalmente, submetendo o conjunto de teste (20% da base) ao classificador NPC, 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 e 11,64% para a seleção. Podemos observar na Figura 3.5, que as taxas são estatisticamente iguais apesar de uma leve tendência de erro menor na seleção.

Verificamos que, após esse estudo experimental com a base artificial Banana, uma seleção de instâncias adequada obteve taxas de acerto tão boas quanto a técnica de geração SGP2.

Tabela 3.1 Comparativo entre protótipos gerados e instâncias selecionadas.

Protótipo p		Instância x_I		classe	d
x	y	x	y		
0,300	0,366	0,301	0,364	1	0,002
0,306	0,590	0,304	0,587	2	0,003
0,890	0,646	0,892	0,644	1	0,002
0,805	0,329	0,800	0,331	2	0,005
0,481	0,496	0,480	0,497	1	0,001
0,509	0,181	0,510	0,181	2	0,001
0,645	0,382	0,648	0,390	1	0,008
0,696	0,579	0,697	0,581	2	0,002
0,543	0,395	0,541	0,390	1	0,005
0,409	0,170	0,412	0,170	2	0,003
0,478	0,380	0,473	0,384	1	0,006
0,405	0,640	0,401	0,640	2	0,003
0,573	0,494	0,570	0,496	1	0,004
0,791	0,462	0,791	0,465	2	0,003
0,514	0,240	0,514	0,238	2	0,003
0,629	0,592	0,627	0,590	2	0,003
0,448	0,232	0,450	0,230	2	0,003
0,401	0,473	0,402	0,471	1	0,002
0,666	0,268	0,670	0,263	1	0,006
0,507	0,675	0,511	0,681	2	0,007
0,743	0,410	0,741	0,406	2	0,004
0,589	0,638	0,593	0,635	2	0,005
0,599	0,804	0,595	0,804	2	0,004
0,717	0,504	0,716	0,504	2	0,001
0,223	0,187	0,225	0,189	1	0,003
0,749	0,165	0,748	0,163	1	0,002
0,399	0,342	0,397	0,336	1	0,006
0,230	0,525	0,230	0,528	2	0,003
0,655	0,739	0,647	0,739	1	0,008
0,588	0,176	0,590	0,174	2	0,003
0,570	0,228	0,569	0,226	2	0,002
0,627	0,431	0,623	0,434	1	0,005
0,663	0,647	0,665	0,649	2	0,003
0,611	0,320	0,607	0,318	1	0,005
0,546	0,281	0,547	0,284	2	0,004
0,326	0,439	0,324	0,433	1	0,006
0,587	0,697	0,584	0,699	2	0,003
0,661	0,534	0,660	0,535	2	0,002
0,291	0,283	0,291	0,284	1	0,001
0,709	0,215	0,710	0,218	1	0,003
0,217	0,348	0,224	0,346	1	0,007
0,759	0,686	0,755	0,688	1	0,004
0,377	0,220	0,375	0,219	2	0,003
0,756	0,554	0,752	0,549	2	0,007
Média					0,004
Desvio padrão					0,002

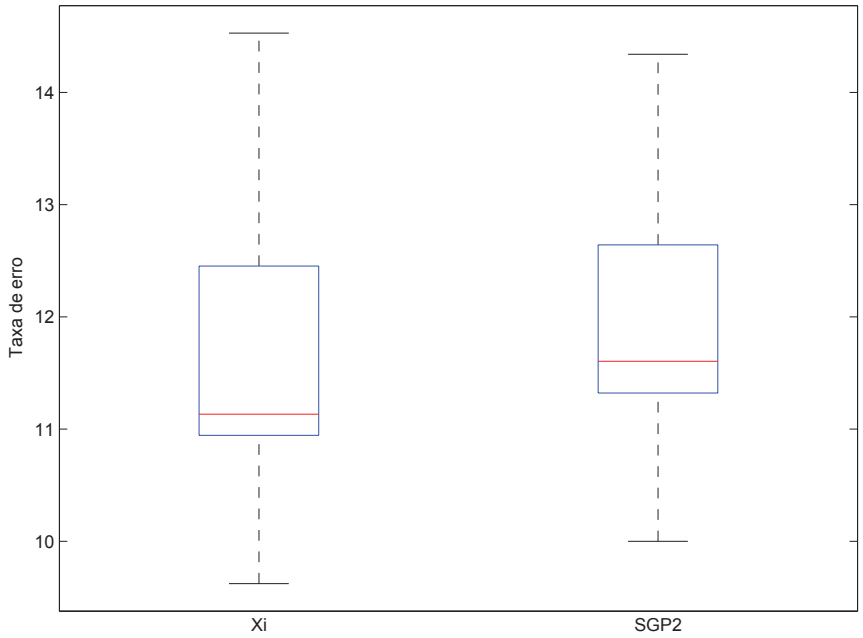


Figura 3.5 Boxplot - Taxa de erro SGP2 $\times$ seleção x_I.

CAPÍTULO 4

EXPERIMENTOS

4.1 INTRODUÇÃO

Este capítulo é dedicado aos experimentos. Descreveremos as bases de dados utilizadas na próxima seção, informando o tipo de problema envolvido, a distribuição de instâncias por classe, número de classes e o repositório onde a mesma está disponível. Na Seção 4.3 é descrita a metodologia utilizada nos experimentos. E, finalmente, na Seção 4.4 são apresentados os resultados.

4.2 BASES

A seguir uma breve descrição das bases de dados utilizadas nos experimentos.

Appendicitis - Esta base de dados real contém informações médicas sobre apendicites aguda. O objetivo desta base é o diagnóstico de apendicites agudas. A tabela possui 106 exemplos, com 7 atributos e duas classes 1 ou 0. A classe 1 contém 88 exemplos (83,0%) e identifica pacientes com apendicite aguda e a classe 0 contém 18 exemplos (17,0%) e indica que o paciente não é portador de apendicite aguda. Essa base está disponível em *KEEL Dataset Repository* [2]. (KEEL)[1]

Australian - (*Australian Credit Approval*) - Esta base contém dados do mundo real, de aplicações de cartão de crédito. Os nomes dos atributos foram modificados para nomes sem sentido, com a finalidade de manter a confidencialidade dos dados. A base possui 690 exemplos, 14 atributos e duas classes 1 ou 0. A classe 1 contém 307 exemplos (44,5%) e a classe 0 contém 383 exemplos (55,5%). Essa base está disponível em *UCI Machine Learning Repository* [3]. (UCI)[2]

Balance - Esta base de dados foi criada para modelar resultados de experimentos de psicologia reportados por Siegler, R. S. (1976). A base possui 625 exemplos, 4 atributos e 3 classes. A classe 1 contém 49 exemplos(7,8%) e indica que a balança está em equilíbrio,

[1] http://sci2s.ugr.es/keel/datasets.php
[2] http://archive.ics.uci.edu/ml/datasets.html

a classe 2 contém 288 exemplos (46,1%) e indica que a balança está em desequilíbrio para a direita e a classe 3 contém 288 exemplos(46,1%) e indica que a balança está em desequilíbrio para a esquerda. Essa base está disponível em *UCI Machine Learning Repository* [3]. (UCI)

Banana - Esta base é uma base artificial e apresenta *clutters* em formato de banana, contém 5300 exemplos, 2 atributos e duas classes 1 e 2. A classe 1 contém 2376 exemplos (44,8%) e a classe 2 contém 2924 exemplos (53,2%). Essa base está disponível em *UCI Machine Learning Repository* [3]. (UCI)

Bupa (*Liver Disorders*) - Esta base contém dados do mundo real sobre distúrbios hepáticos que podem surgir devido ao consumo excessivo de álcool. Seu propósito é classificar indivíduos que sofrem de alcoolismo. A base foi construída agrupando dados de exames de sangue que estão ligados a doenças do fígado e um atributo ligado ao número de doses de bebida. A base contém 345 exemplos, 6 atributos e duas classes: Portador de doenças hepáticas 200 exemplos (58%) e Não portador 145 exemplos(42%). Essa base está disponível em *UCI Machine Learning Repository* [3]. (UCI)

Cancer (*Breast Cancer Wisconsin(Original)*) - Esta base contém informações para o diagnóstico de câncer de mama, cujo objetivo é classificar se um tumor de mama é maligno ou benigno. Os atributos das células foram obtidos através de exames microscópicos. Também foram considerados atributos morfológicos como tamanho e forma do nódulo, grau de aderência, entre outros. A tabela contém 699 exemplos, 9 atributos e duas classes: Tumor maligno com 241 exemplos (34,5%) e tumor benigno com 458 exemplos (65,5%). Essa base está disponível em *UCI Machine Learning Repository* [3]. (UCI)

Card - Esta base contém dados para a predição de aprovação cadastral para a concessão de cartão de crédito. A tabela contém 690 exemplos, 51 atributos e duas classes: Aprovados 307 exemplos (44,5%) e Reprovados 383 exemplos (55,5%). Essa base está disponível em *UCI Machine Learning Repository* [3]. (UCI)

Ecoli - Esta base contém dados para a predição da localização de proteínas. A tabela contém 336 exemplos, 7 atributos e 8 classes de localização. A Distribuição de instâncias por classes é mostrada na Tabela 4.1. Essa base está disponível em *UCI Machine Learning Repository* [3]. (UCI)

Gene - Esta base contém dados para a detecção de junções íntron/éxon em sequências de nucleotídeos. A base possui 3175 exemplos, 120 atributos e 3 classes: Se é uma junção íntron/éxon(doadora), 765 exemplos(24%), se é uma junção éxon/íntron (receptora), 762 exemplos(24%) ou nenhuma das junções, 1648 (52%). Essa base está disponível em

Tabela 4.1 Ecoli: Distribuição de instâncias por classes

Classe	Descrição	# Exemplos	%
1	Citoplasma	143	42,8
2	Membrana Int. s/ sinal	77	23,1
3	Periplasma	52	15,6
4	Membrana Int. c/ sinal	35	10,5
5	Membrana ext.	20	6,0
6	Lipoproteína	5	1,5
7	Membrana Int. c/ sinal clivável	2	0,6

Proben1 [29].[3]

German Credit Data (Statlog) - Esta base contém dados reais para a avaliação de risco de crédito de uma pessoa. A base possui 1000 exemplos, 20 atributos e duas classes: 1-Bom 300 exemplos(30%) e 2-Ruim 700 exemplos (70%). Essa base está disponível em *UCI Machine Learning Repository* [3]. (UCI)

Glass - Esta base contém dados para a classificação de tipos de vidros. O conjunto de atributos selecionados são índice de refração, quantidade de elementos químicos como: sódio, magnésio, alumínio, silício, potássio, cálcio, bário e ferro presentes em estilhaços de vidro. A base possui 214 exemplos, 9 atributos e 6 classes, cuja distribuição é mostrada na Tabela 4.2.Essa base está disponível em Proben1 [29].

Tabela 4.2 Glass: Distribuição de instâncias por classes

Classe	Descrição	# Exemplos	%
1	Vidro para Edifícios (float processed)	70	32,7
2	Vidro para Edifícios (non float processed)	76	35,5
3	Vidro para veículos (float processed)	17	7,9
4	Vidro para recipientes	13	6,1
5	Vidro para mesas	9	4,2
6	Vidro para lâmpadas	29	13,6

Haderman - Esta base contém dados resultante de um estudo realizado entre 1958 e 1970, na Universidade de Billings Hospital de Chicago, sobre a sobrevivência de pacientes submetidos à cirurgia de câncer de mama. Esta base possui 306 registros, 3 atributos: idade, ano da cirurgia e número de nódulos detectados. As duas classes da base informam se o paciente sobreviveu mais de 5 anos após a cirurgia, 225 exemplos (73,5%) e os que morreram antes de completar 5 anos de cirurgia, 81 exemplos (26,5%). Essa base está disponível em *KEEL Dataset Repository* [2]. (KEEL)

[3] ftp://ftp.ira.uka.de/pub/neuron/proben1.tar.gz

Heart (*StatLog*) - Esta base contém dados para a predição de doenças cardíacas. O conjunto de características selecionadas são: idade, sexo, dor no peito, pressão arterial em repouso, taxa de colesterol (mg/dl), taxa de açúcar no sangue $>$ 120mg/dl, resultado do eletrocardiograma em repouso, máxima frequência cardíaca, exercício para indução de angina, heart depressão ST induzida por exercícios, entre outros. A base contém 270 exemplos, 13 atributos e 2 classes: Ausência de doença cardíaca 150 exemplos (55,6%) e doença cardíaca 120 exemplos(44,4%). Essa base está disponível em *UCI Machine Learning Repository* [3]. (UCI)

Ionosphere - Esta base contém dados capturados por 16 antenas de alta frequência à procura de informações de elétrons livres na ionosfera. A coleta de dados foi realizada utilizando o sistema Goose Bay, Labrador, pelo Grupo de Física Espacial da Universidade de John Hopkins. A base possui 351 registros, sendo composta por 34 atributos contínuos e 2 classes: Bons, 126 exemplos(35,9%) e Ruins, 225 exemplos(64,1%). Essa base está disponível em *UCI Machine Learning Repository* [3]. (UCI)

Iris - Esta base contém dados sobre a classificação de plantas da espécie Iris. A base contém 150 exemplos, 4 atributos reais: comprimento e largura de pétalas e sépala e 3 classes: *Iris Setosa*, 50 exemplos(33,33%), *Iris Vesicolour*, 50 exemplos(33,33%) e *Iris Virginica*, 50 exemplos (33,33%). Essa base está disponível em *UCI Machine Learning Repository* [3]. (UCI)

Lymphography - Esta base contém dados de Linfografias obtidos a partir do Centro Médico da Universidade, *Institute of Oncology, Ljubljana*, Iugoslávia, para a predição de câncer no sistema linfático. A base possui 148 exemplos, 19 atributos e 4 classes c, como por exemplo: número de nódulos, formato do nódulo, mudança na estrutura, entre outros dados citológicos. A base apresenta 4 classes cuja distribuição é mostrada na Tabela 4.3.

Tabela 4.3 Lymphography: Distribuição de instâncias por classes

Classe	Descrição	# Exemplos	%
1	resultado normal	2	1,4
2	metastase	81	54,7
3	linfoma maligno	61	41,2
4	fibroses	4	2,7

Magic - Esta Base contém dados gerados pelo programa Monte Carlo, Corsika, para simular o registro de partículas Gamma de alta energia em um telescópio de Cherenkov, utilizando técnica de imagem, *(Imaging Atmospheric Cherenkov Technique)* para classificar raios Gamma e chuveiros hadrônicos (hadrôn). A Base possui 19020 exemplos, 10

atributos e duas classes: g, partículas Gamma, 12332 exemplos (64,8%) e h, hadrôns, 6688 exemplos (35,2%). Essa base está disponível em *UCI Machine Learning Repository* [3]. (UCI)

Pageblock - A base contém dados necessários à classificação de blocos da disposição de página de um documento que foi detectada por um processo de segmentação. Este método é utilizado para separar áreas de texto a partir de áreas de gráficos. A base apresenta as seguintes características: 5472 exemplos, 10 atributos: altura do bloco, largura do bloco, área do bloco, excentricidade (lagura/altura), entre outros e 5 classes cuja distribuição é mostrada na Tabela 4.4. Essa base está disponível em *UCI Machine Learning Repository* [3]. (UCI)

Tabela 4.4 Pageblock: Distribuição de instâncias por classes

Classe	Descrição	# Exemplos	%
1	Texto	4913	89,8
2	Linha horizontal	329	6,0
3	Imagem	28	0,5
4	Linha vertical	87	1,6
5	Gráfico	115	2,1

Penbased - Esta base contém dados para a resolução do problema de classificação de números manuscritos.Os dígitos foram obtidos a partir da escrita de dígitos de 44 pessoas, cada um delas forneceu 250 exemplos. A base possui 10992 exemplos, 16 atributos e 10 classes (0 a 9), cuja distribuição é mostrada na Tabela 4.5. Essa base está disponível em *UCI Machine Learning Repository* [3]. (UCI)

Tabela 4.5 Pendbased: Distribuição de instâncias por classes

Classe	Descrição	# Exemplos	%
0	Dígito 0	1143	10,4
1	Dígito 1	1143	10,4
2	Dígito 2	1144	10,4
3	Dígito 3	1055	9,6
4	Dígito 4	1144	10,4
5	Dígito 5	1055	9,6
6	Dígito 6	1056	9,6
7	Dígito 7	1142	10,4
8	Dígito 8	1055	9,6
9	Dígito 9	1055	9,6

Pima - Esta base contém dados para a predição de diabetes em índios Pima, tendo como

base atributos como: idade, IMC - índice de massa corpórea, pressão arterial, intolerância a glicose. A tabela contém 768 exemplos, 8 atributos e duas classes: Diabético, 268 exemplos (34,9%) ou Não diabéticos, 500 exemplos(65,1%). Essa base está disponível em *UCI Machine Learning Repository* [3]. (UCI)

Phoneme - Esta base foi utilizada no projeto ESPRIT 5516: ROARS(*Robust Analytical Speech Recognition System*), cujo objetivo era a recognição de fala em tempo real para os idiomas Francês e espanhol. O objetivo da base é classificar vogais nasais e orais. A base contém dados de vogais nasais e orais. Tendo um total de 5404 exemplos, 5 atributos que são respectivamente as amplitudes dos 5 primeiros harmônicos AHi e duas classes: Classe 0: Nasal, 3818 exemplos(70,7%) e Classe1:Oral, com 1586 exemplos (29,3%). Essa base está disponível em *KEEL Dataset Repository* [2]. (KEEL)

South African Hearth (Saheart) - Esta base contém dados de indivíduos do sexo masculino com doenças cardíacas na região de alto risco, Western Cape, na África do Sul. A Base possui 462 exemplos, 9 atributos: pressão arterial, quantidade de tabaco consumida, colesterol LDL, adiposidade, histórico familiar, obesidade, consumo de álcool, idade, entre outros e duas classes: Classe 0: Não tem doença coronariana, com 302 exemplos (65,4%) e Classe 1: Doença coronariana, com 160 exemplos (34,6%). Essa base está disponível em *UCI Machine Learning Repository* [3]. (UCI)

Satimage - Esta base contém dados para classificação de imagens de satélites. A base consiste em valores multi-espectrais de pixels de uma vizinhança 3x3 de imagens de satélite. Sendo a classificação definida pelo pixel central de cada vizinhança. Esta base contém 6435 exemplos, 36 atributos e 6 classes de imagens cuja distribuição é mostrada na Tabela 4.6. Essa base está disponível em *UCI Machine Learning Repository* [3]. (UCI)

Tabela 4.6 Satimage: Distribuição de instâncias por classes

Classe	**Descrição**	**# Exemplos**	**%**
0	Solo vermelho	1533	23,8
1	Plantação de algodão	703	10,9
2	Solo cinza	1358	21,1
3	Solo cinza úmido	626	9,7
4	Solo com vegetação rasteira	707	11,0
5	Solo cinza muito úmido	1508	23,4

Segmentation - Esta base contém dados de imagens segmentadas. Os dados foram obtidos através da segmentação manual de 7 imagens de ambientes externos. Cada instância

é uma região de 3x3. A base possui 2310 exemplos, 19 atributos (valores reais) e 7 classes estão balanceadas, cada uma com 330 exemplos (14,29%). Essa base está disponível em *UCI Machine Learning Repository* [3]. (UCI)

Connectionist Bench (*Sonar, Mines vs. Rocks*) - Esta base possui 111 exemplos (53,4%) dados obtidos pela reflexão de sinais de sonar fora de um cilindro de metal em ângulos diferentes e sob condições diferentes (Sonar Mines) e 97 exemplos (46,6%) obtidos a partir de rochas em condições semelhantes (Sonar Rocks). A Base possui um total de 208 exemplos, 60 atributos e duas classes: Classe 1: Sonar Rocks e Classe 2: Sonar Mines. Essa base está disponível em *UCI Machine Learning Repository* [3]. (UCI)

Splice - Esta base contém dados de reconhecimento de fronteiras de exons (partes da sequência de DNA retirada após splicing) e itrons (partes de sequencias de Junctions em uma sequência de DNA que estão fora do spliced). Essa base possui 3190 exemplos, 60 atributos e 3 classes: Classe EI: (exons) com 767 exemplos(25%), Classe IE(itrons) com 768 exemplos (25%) e Classe N (nenhum) com 1655 exemplos (50%). Essa base está disponível em *UCI Machine Learning Repository* [3]. (UCI)

Thyroid Disease - A base contém dados para predição de doenças da tireoide. Os dados são do mundo real e a base contém 7200 exemplos, 21 características como sexo, idade, além de taxas de hormônios específicos da tireoide, e 3 classes: Classe 1: Normal, possui 6666 exemplos (92,6%), Classe 2: Hipertiroidismo, possui 368 exemplos (5,5%) e Classe 3: Hipotiroidismo com 166 exemplos (1,9%). Essa base está disponível em *UCI Machine Learning Repository* [3]. (UCI)

Twonorm - A base contém dados artificiais para classificação de classes. Cada classe é traçada a partir de uma distribuição normal multivariada com variância 1. Possui 7400 exemplos, 20 atributos e 2 classes: Classe 0: com 3703 exemplos (50%) e Classe 1: 3697 exemplos (50%). Essa base está disponível em *KEEL Dataset Repository* [2]. (KEEL)

Vehicle - Esta base contém dados para a classificação de veículos a partir de sua silhueta. Possui 846 exemplos e 18 atributos, que são as medidas geométricas extraídas da silhueta de cada veículo. Os dados contidos na base contemplam 4 classes: Opel, Saab, ônibus e van. A distribuição das classe é mostrada na Tabela4.7. Essa base está disponível em *UCI Machine Learning Repository* [3]. (UCI)

Vowel - Esta base contém informações sobre reconhecimento de voz independente das 11 vogais estáveis do Inglês Britânico. Possui 990 exemplos, 13 atributos e 11 classes com 90 instâncias (9,09%) cada. Essa base está disponível em *KEEL Dataset Repository* [2]. (KEEL).

Tabela 4.7 Vehicle: Distribuição de instâncias por classes

Classe	Descrição	# Exemplos	%
1	Opel	199	23,5
2	Saab	217	25,7
3	Ônibus	218	25,8
4	Van	212	25,1

Waveform - Esta base contém dados artificiais de formas de ondas geradas em um programa escrito em Linguagem C. Possui 5000 exemplos, 21 atributos com valores contínuos (0-6) e 3 classes de ondas cuja distribuição é de 33,33% para cada classe. Essa base está disponível em *UCI Machine Learning Repository* [3]. (UCI)

Wine - Esta base contém dados para a classificação de vinhos. Seus dados são obtidos a partir de análises químicas para determinar a origem dos vinhos. Possui 178 exemplos, 13 características como por exemplo: teor alcoólico, ácido málico, alcalinidade, teor de fenóis, teor de magnésio, entre outros, e 3 classes cuja distribuição é a seguinte: classe 1, 59 exemplos (33,1%), classe 2, 71 exemplos (39,9%) e classe com 48 exemplos (27%). Essa base está disponível em *UCI Machine Learning Repository* [3]. (UCI)

Yeast - Esta base contém informações para a predição da localização de proteínas. Possui 1484 exemplos, 8 atributos e 10 classes cuja distribuição é mostrada na Tabela 4.8.Essa base está disponível em *UCI Machine Learning Repository* [3]. (UCI)

Tabela 4.8 Yeast: Distribuição de instâncias por classes

Classe	Descrição	# Exemplos	%
CYT	Citoesqueleto	463	31,2
NUC	Nuclear	429	28,9
MIT	Mitocondrial	244	16,4
ME3	Proteína de membrana - 3	163	11,0
ME2	Proteína de membrana - 2	51	3,4
ME1	Proteína de membrana -1	44	3,0
EXC	Extracelular	37	2,5
VAC	Vacuolar	30	2,0
POX	Peroxisomal	20	1,3
ERL	Lúmen do retículo endoplasmático	5	0,3

Zoo - Esta base contém dados para a predição das classes de animais. Possui 101 exemplos, 16 atributos lógicos como por exemplo se a classe de animais possui cabelos(pelos), se põe ovos, se é mamífero, se é aquático, aéreo, se é predador, se venenoso entre outros. Possui também um atributo numérico que diz respeito ao número de pernas do animal e

7 classes de animais cuja distribuição é mostrada na Tabela 4.9.Essa base está disponível em *UCI Machine Learning Repository* [3]. (UCI)

Tabela 4.9 Zoo: Distribuição de instâncias por classes

Classe	**Descrição**	**# Exemplos**	**%**
1	Porco, antílope, urso, girafa...	41	40,6
2	Galinha, pato, flamingo, piguin...	20	19,8
3	Cobra,tartaruga	5	5,0
4	Peixes	13	12,9
5	Sapos	4	4,0
6	Pulga, mosquitos...	8	7,9
7	Molusco, carangueijo, polvo...	10	9,9

Na Tabela 4.10 é apresentado um resumo das bases utilizadas nos experimentos e suas principais características: número de exemplos, número de atributos e número de classes. Podemos verificar também nas duas últimas linhas da tabela os valores mínimos e máximos para cada característica.

Tabela 4.10 Resumo das bases utilizadas nos experimentos

Base	**# Exemplos**	**# Atributos**	**# Classes**
Appendicitis	108	7	2
Australian	690	14	2
Balance	625	4	3
Banana	5300	2	2
Bupa	345	6	2
Cancer	699	9	2
Card	690	51	2
Ecoli	336	7	8
Gene	3175	120	3
German	1000	20	2
Glass	214	9	6
Haderman	306	3	2
Heart	270	13	2
Ionosphere	351	34	2
Iris	150	4	3
Lymphography	148	19	4
Magic	19020	10	2
Pageblock	5472	10	5
Penbased	10992	16	10
Pima	768	8	2
Phoneme	5404	5	2
South African Hearth (saHeart)	462	9	2
Satimage	6435	36	6
Segmentation	2310	19	7
Sonar	208	60	2
Splice	3190	60	3
Thyroid	7200	21	3
Twonorm	7400	20	2
Vehicle	846	18	4
Vowel	990	13	11
Waveform	5000	21	3
Wine	178	13	3
Yeast	1484	8	10
Zoo	101	16	7
Mínimo	**101**	**2**	**2**
Máximo	**19020**	**120**	**11**

4.3 METODOLOGIA

Para validar a hipótese do trabalho foram realizados experimentos nas 35 bases de dados da *UCI Learning Machine* [3], PROBEN1 [29] e *KEEL Dataset Repository* [2], descritas na Seção 4.2. 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. [35], selecionamos 6 técnicas de geração de protótipos para nossos experimentos: SGP2, LVQ3, POC-NN, GENN, ICPL2, RSP3, 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 4.1, as famílias das técnicas de geração e as técnicas utilizadas nos experimentos. As técnicas de geração de acordo com suas características são divididas em quatro famílias. A primeira família é a *Positioning Adjustment*, nessa técnica, são gerados protótipos iniciais, cujas localizações são ajustadas a cada iteração, com o objetivo de se obter a melhor taxa de classificação. Alguns representantes dessa família são o LVQ [20], LVQTC [27], VQ [38], MSE [11], HYB [19], LVQPRU [23], PSO [26], PSCSA [14], ENPC [13], AMPSO [5]. Já a família *Class Re-labeling*, utiliza a heurística de eliminação ou reidentificação de instância, isto é, troca do *label* da classe, caso esta troca implique melhores taxas de classificação. O GENN [21] e o Depur [32] são os únicos representantes desta família. A terceira família dos métodos de geração de protótipos é a *Centroids*, essa família utiliza a heurística do cálculo de centróides de grupos de instâncias, temos como representantes dessa família o SGP [12], ICPL2 [22], PNN [6], MCA [4], GMCA [25], BTS3 [17], MixtGauss [24]. E, finalmente, a última família das técnicas de geração é a *Space Splitting* que utiliza a heurística de divisão sucessiva do espaço amostral até que o mesmo possua instâncias de mesma classe e então seja calculado o protótipo que irá representar este espaço. Como representantes dessa família temos: Chen [8], RSP3 [31] e o POC [30].

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 4.1, onde $\overline{x}$ 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.

$$Atributo(i) = \frac{(Atributo(i) - \overline{x}_{Atributo})}{s_{Atributo}} \tag{4.1}$$

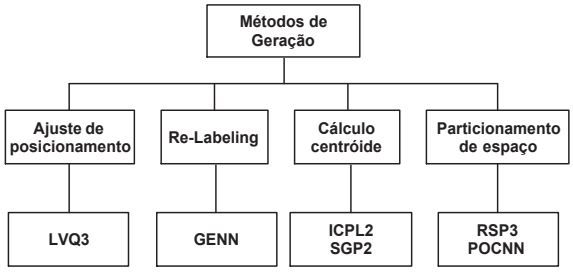


Figura 4.1 Famílias das técnicas de geração de protótipos selecionadas para o estudo

Utilizando a estratégia descrita no Capítulo 3 para cada uma das bases, obtivemos como resultado a média das taxas de erro de classificação e a média dos desvios padrões. Os experimentos foram realizados aos pares, isto é, os resultados da técnica de geração SGP 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 SGP. Essa comparação é utilizada para todas as técnicas. Os resultados estão registrados na Tabela 4.16. Essa tabela é estruturada da seguinte forma: Para cada base temos o resultado da taxa de erro para a técnica de geração e classificação baseada no C_I, o desvio padrão é informado , entre parênteses, ao lado da taxa de erro. Baseado na rejeição da hipótese nula do *T-Sudent*, para taxas estatisticamente diferentes, verificamos qual a melhor taxa de erro (menor taxa) e inserimos o sinal de "+" ao lado da taxa de erro na Tabela 4.16. Na penúltima linha da Tabela 4.16 uma contagem é apresentada no formato $+|+|=$, onde o primeiro sinal "+"representa o número de bases, cuja menor taxa de erro foi obtida pela técnica de geração de protótipos, o segundo "+", representa o número de bases onde a menor taxa de erro é obtida pela utilização do conjunto de instâncias C_I e o sinal "="representa o número de bases onde as taxas de erro obtidas pela técnica de geração de protótipos são estatisticamente iguais às obtidas pela utilização do conjunto de instâncias C_I. Na última linha da tabela, temos o percentual de bases cujas taxas de erro obtidas com o C_I são iguais ou melhores que as obtidas pela técnica de geração.

Observamos que para as bases Gene, Ringnorm, Splice, Waveform e Twonorm, as técnicas de geração apresentavam melhores resultados, e a partir dessa observação, fizemos

um estudo das distâncias e vizinhanças dos protótipos gerados. Esse estudo busca identificar as razões que levaram as técnicas de geração terem melhor taxa de acerto. Para isso, foram utilizadas algumas estratégias que tentam caracterizar o espaço em torno dos protótipos gerados, instâncias x_I e seus vizinhos mais próximos. Esse estudo buscou, por exemplo, identificar se o protótipo gerado estava ocupando uma área vazia. Exemplo na Figura 4.2. Nesse caso, podemos verificar que nenhuma instância selecionada pode substituir satisfatoriamente o protótipo gerado p, uma vez que o centro do agrupamento não está representado por nenhuma instância. Sempre que os agrupamentos apresentarem essa forma, as técnicas de geração apresentarão melhor desempenho, pois nenhuma instância consegue ter o nível de representatividade de PG em relação à propriedade de generalização. A partir do conjunto de protótipos gerados, buscamos, inicialmente, a instância de mesma classe mais próxima, calculamos essa distância, depois, buscamos o inimigo I mais próximo de PG, isto é, a instância de classe diferente da classe de PG que estivesse mais próxima ao PG. Calculamos a distância entre a x_I e sua amiga mais próxima (γ), e, finalmente, contamos quantos protótipos estão mais próximos aos amigos e os mais próximos aos inimigos. Das 35 bases, analisamos 34 com relação à distância e à vizinhaça, pois a base *Lymphography* devido ao fato de ser uma base desbalanceada, não pôde ser analisada, pois em alguns *folds* não tínhamos representantes de todas as classes. Como essa base apresentou taxas de erro estatisticamente iguais para geração e seleção, essa análise não se fez necessária.

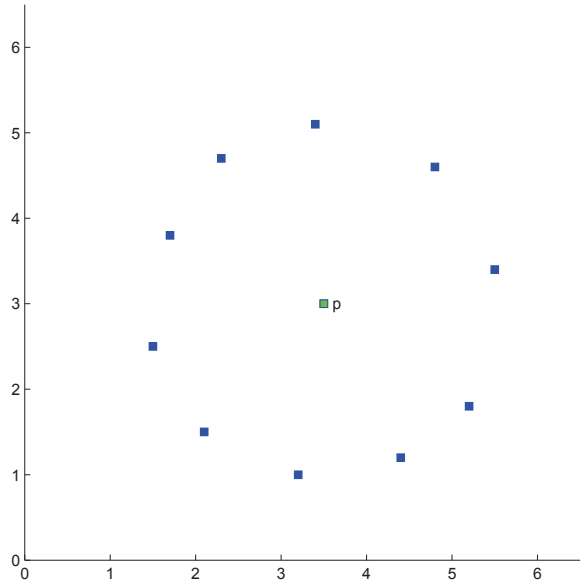


Figura 4.2 Protótipo gerado em área vazia.

A Figura 4.3 mostra as distâncias e elementos analisados, sendo (p) o protótipo gerado,

(x_I) a instância de mesma classe mais próxima ao p, (I) o inimigo mais próximo de p, (θ) a Distância do p ao seu amigo, (β) a Distância de PG ao inimigo mais próximo, (γ) a Distância entre a x_I e a instância de mesma classe mais próxima (amigos).

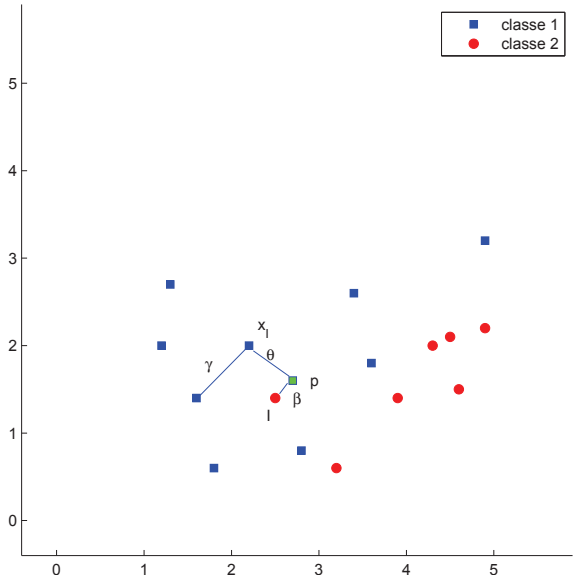


Figura 4.3 Estudo das distâncias e vizinhanças.

4.3.1 Parâmetros

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 4.11.

Tabela 4.11 Parâmetros utilizados nos experimentos

Método	**Parâmetros**
SGP	*Rmis= 0,1; Rmim=0,08*
LVQ	*Iterações = 50; alpha= 0,065; WindowWidth=0,2; epsilon=0,1*
POC-NN	*Alpharatio=0,05*
GENN	*k=7; $k^{'}$=4*
RSP3	*Não paramétrico*
ICPL2	*Método de filtragem = ENN*
ENN	*k=3*

4.4 RESULTADOS

Nesta seção serão analisados os resultados dos testes realizados com as bases descritas na Seção 4.2, aplicando-se as técnicas de geração descritas na Seção 4.3 e a técnica de seleção DROP3. A escolha da técnica de seleção DROP3 foi baseada no estudo de Garcia et al. [15], que mostra que essa técnica é uma das técnicas mais usadas em comparativos de classificadores, além de apresentar boa capacidade de redução, imunidade a ruídos e boa taxa de acerto.

Podemos verificar, na última linha da Tabela 4.16, que na maioria dos casos, a seleção de instâncias obtiveram 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. Na Figura 4.4, podemos verificar a proximidade dos protótipos gerados a partir das técnicas SGP2, LVQ3, R-POC-NN, RSP3, ICPL2 e as instâncias x_I próximas aos protótipos gerados, para a base banana. Utilizamos para o comparativo a base banana, devido ao fato dessa base possuir dois atributos, podendo ser representada graficamente. Omitimos a figura da técnica GENN, pelo fato desta técnica, para a base banana, apresentar uma taxa de redução igual a zero (Tabela 4.12).

Tabela 4.12 Taxas de Redução para a base banana

Técnica	SGP2	LVQ3	R-POC-NN	RSP3	ICPL2	GENN
Taxa de Redução	99,16%	99,43%	99,71%	75,42%	94,17%	0,00%

A Tabela 4.16 mostra as taxas de erro de classificação de cada método de geração em comparação com as taxas de erro obtidas a partir das instâncias selecionadas x_I. Os desvios padrões são apresentados entre parênteses. A análise é feita em pares, isto é, a taxa de erro do método de geração e a taxa de erro do método de seleção. Para verificar se as taxas apresentadas são estatisticamente diferentes, foi utilizado o teste de hipótese *T-Student*. Taxas de erro seguidas do sinal " $+$ " indicam as menores taxas de erro para o par analisado, isto é, técnica de geração $\times$ técnica de seleção.

Na comparação do SGP2 com o método de seleção de instâncias próximas de mesma classe, verificamos que das 35 bases utilizadas no estudo, em 27 bases (77,14%), as taxas de erro obtidas pela técnica de seleção são estatisticamente iguais ou melhores que o SGP2 e só em 8 bases (22,86%) o SGP2 apresentou melhores resultados.

No método LVQ3 das 35 bases testadas, verificamos que em 25 bases (74,29%) a técnica de seleção apresentou taxas estatisticamente idênticas ao LVQ3 e em 9 bases (25,71%) o

LVQ3 apresentou melhores resultados. Na comparação seleção $\times$ LVQ3, na base Magic, obtivemos menor taxa de erro na seleção.

Em relação ao método POC-NN, verificamos que em 28 bases (82,86%) apresentaram estatisticamente as mesmas taxas de erro. Em 6 bases (17,14%), no método de geração POC-NN, obtivemos melhores taxas de erro. Nesta comparação, na base Vowel, obtivemos melhores taxas de erro quando utilizamos o conjunto de instâncias selecionadas, C_I.

No caso do método GENN, para todas as bases testadas 35 (100%), obtivemos as mesmas taxas de erro, isso ocorre devido à baixíssima redução da técnica GENN. Quase todas as instâncias da base de treinamento são utilizadas como protótipos.

Comparando o método RSP3 com o método de seleção, obtivemos taxas estatisticamente iguais em 28 bases(80,00%) e em 7 bases (20%), o método de geração RSP3 obteve melhores resultados.

Quando comparamos o método ICPL2 com o método de seleção, obtivemos taxas estatisticamente iguais em 32 bases(91,43%) e em apenas 3 bases (8,57%), o ICPL2 apresentou melhores resultados.

Na Tabela 4.13 são apresentadas médias das taxas de erro e desvios padrões das técnicas de geração de protótipos, apresentadas na Tabela 4.16. A partir de teste de hipóteses executados, verificamos que para as bases testadas as técnicas SGP2, GENN, RSP3 e ICPL2 apresentaram taxas de erro equivalentes. As técnicas LVQ3 e POC-NN obtiveram pior desempenho em relação às técnicas citadas anteriormente.

Tabela 4.13 Média das taxas de erro das técnicas de geração.

Técnica	**%Erro**
SGP2	19,7 (12,8)
LVQ3	23,9 (13,8)
POC-NN	27,5 (14,5)
GENN	18,5 (12,0)
RSP3	19,0 (12,1)
ICPL2	17,1 (10,9)

Também notamos que as bases Gene, Ringnorm, Splice, Twonorm e Waveform, apresentaram melhores taxas de erro quando submetidas a técnicas de geração.

Na tabela 4.14, apresentamos os percentuais das bases com menores taxas de erro por técnica.

Tabela 4.14 Percentuais das bases com menores taxas de erro ($Geração \times x_I$).

	% Bases com menores taxas de erros		
	Geração (Cp)	Seleção (Ci)	(Cp) = (Ci)
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	**83,33**

Com base na Tabela 4.14, verificamos para a maioria das bases $(83,33\%)$, as técnicas de seleção que utilizaram o conjunto de instâncias C_I apresentaram 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 performace no processo de classificação, visto que as técnicas de geração, geralmente, são mais lentas em relação às técnicas de seleção de instâncias.

São apresentadas na Tabela 4.17 as taxas de erro de classificação de cada método de geração em comparação com as taxas de erro obtidas no DROP3. Os desvios-padrão são apresentados entre parênteses. Assim como na tabela anterior, as comparações são feitas aos pares, isto é, a técnica de geração *versus* a técnica de seleção DROP3. Também foi executado o teste de hipótese *T-Student* para comparar estatisticamente a média de erro das duas técnicas.

Na comparação do SGP2 com o DROP3, verificamos que das 35 bases utilizadas no estudo, o DROP3 obteve em 30 bases (85,71%) 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$\times$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$\times$DROP3. E, finalmente, na comparação da técnica ICPL2$\times$DROP3, verificamos que em 27 bases (77.14%) o DROP3 obteve as taxas melhores ou taxas es-

tatisticamente iguais ao ICPL2. E em 8 bases (22,86%) o ICPL2 apresentou melhores resultados.

Na Tabela 4.15 apresentamos os percentuais das bases com menores taxas de erro por técnica.

Tabela 4.15 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	71,43
LVQ3	2,86	22,86	74,29
POC-NN	0	22,86	77,14
GENN	20	11,43	68,57
RSP3	20	11,43	68,57
ICPL2	22,86	2,86	74,29
Média	13,34	14,29	**72,38**

Com base na Tabela 4.15, verificamos que para a maioria das bases ($72,38\%$), 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 performace no processo de classificação. Além disso, temos um percentual de 14,29% em que o DROP3 apresentou melhores resultados. Nesse comparativo, observamos que as técnicas de geração obtiveram melhores resultados para bases específicas.

Para o estudo das distâncias entre protótipos×instâncias e vizinhanças, utilizamos como base a técnica de geração SGP2, por apresentar os melhores resultados em relação a outras técnicas de geração. A Tabela 4.18 mostra os dados relativos às distâncias médias entre instâncias, distâncias $p \times$ instâncias, vizinhos mais próximos e inimigos mais próximos. Na base Gene, notamos inicialmente que a diferença das taxas de erro entre a técnica SGP2($Erro = 15,69\%$) e a seleção ($Erro = 47,59\%$) é considerável para o SGP2, verificamos que a distância média entre instâncias é menor que a distância entre os protótipos gerados e as instâncias amigas, isto mostra que o protótipo gerado está cobrindo uma área que não estava sendo coberta por uma instância. As bases Card, Penbased e Vowel apresentam comportamento semelhante à base Gene com relação às distâncias entre instâncias amigas e as distâncias entre protótipo gerado e a instância amiga, neste caso, também verificamos que a distância média entre instâncias é menor que a distância média entre instância $\times$ protótipo, logo o protótipo gerado está representando melhor a área da outra instância presente no conjunto de treinamento.

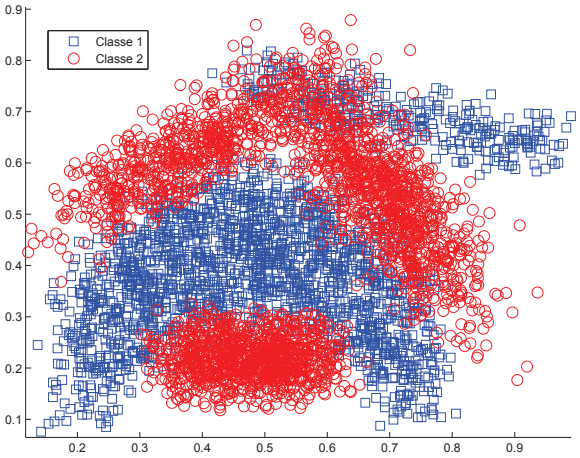
Como exemplo de bases que apresentaram taxas de erro bem próximas, temos a Banana, cujas distâncias entre amigos está bem próxima da distância protótipo $\times$ instância, logo os protótipos gerados não são necessários, pois estão sendo gerados em áreas que podem ser representadas por instâncias existentes no conjunto de treinamento. Podemos verificar na Figura 4.4, que na maioria dos casos, as instâncias selecionadas estão bem próximas ou estão na mesma localização do protótipo gerado, não havendo necessidade de se gerar protótipos para esta base.

Tabela 4.16 Comparação das técnicas de geração × seleção

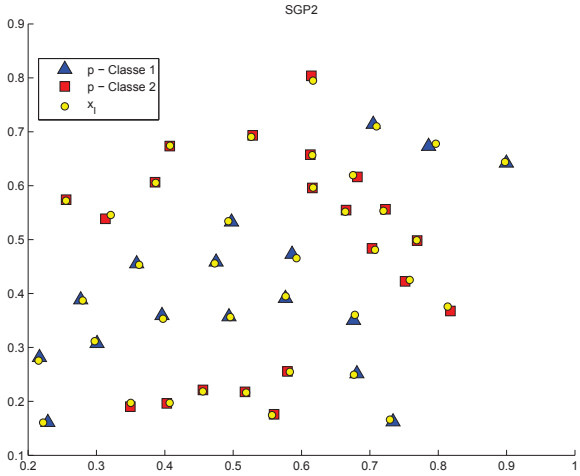
Base	SGP2	Seleção	LVQ3	Seleção	R-POC-NN	Seleção	GENN	Seleção	RSP3	Seleção	ICPL2	Seleção
Appendicitis	12,273 (7,422)	15,182 (10,357)	24,727 (9,957)	30,182 (11,411)	31,364 (16,980)	30,273 (14,154)	19,909 (12,009)	19,909 (12,009)	25,545 (8,610)	22,727 (13,293)	15,091 (8,403)	14,182 (9,432)
Australian	13,913 (4,592)	15,797 (3,164)	38,696 (4,895)	39,275 (3,333)	33,478 (5,667)	26,812 (7,676)	19,420 (4,772)	19,420 (4,772)	24,058 (3,953)	25,362 (4,213)	15,507 (4,350)	15,217 (5,585)
Balance	19,846 (6,268)	21,769 (4,573)	20,622 (4,900)	22,552 (4,086)	29,427 (6,189)	28,461 (5,338)	36,953 (6,192)	36,953 (6,192)	33,269 (7,607)	36,802 (5,094)	13,441 (1,927)	13,922 (2,594)
Banana	11,830 (1,113)	11,642 (1,367)	12,132 (1,110)	12,245 (1,250)	19,208 (3,639)	19,491 (3,872)	13,245 (0,851)	13,245 (0,851)	15,528 (1,151)	15,566 (1,020)	11,415 (1,602)	11,547 (1,685)
Bupa	38,277 (6,570)	41,504 (8,646)	34,513 (7,328)	36,261 (7,714)	46,941 (6,037)	48,672 (4,320)	35,647 (3,841)	35,647 (3,841)	37,950 (4,915)	41,714 (4,620)	36,471 (8,744)	36,462 (8,537)
Cancer	3,718 (2,039)	4,433 (3,043)	3,580 (1,476)	4,151 (1,198)	8,441 (2,255)	8,584 (3,833)	4,435 (2,254)	4,435 (2,254)	6,863 (3,872)	7,865 (3,623)	3,865 (1,581)	3,580 (1,608)
Card	$16,087^{+}$ (2,287)	19,710 (3,497)	16,812 (5,112)	19,420 (3,845)	34,203 (7,194)	36,377 (7,870)	17,536 (2,858)	17,536 (2,629)	22,029 (5,054)	25,072 (5,267)	17,101 (5,177)	17,971 (3,502)
Ecoli	17,611 (7,024)	18,191 (6,732)	16,070 (4,981)	16,684 (4,549)	27,389 (3,757)	29,144 (7,108)	19,679 (5,280)	19,679 (5,280)	22,656 (4,985)	22,674 (4,720)	15,820 (5,293)	15,517 (4,905)
Gene	$15,686^{+}$ (2,013)	47,558 (3,063)	$26,143^{+}$ (3,566)	48,156 (2,914)	$33,702^{+}$ (1,806)	44,504 (3,270)	28,001 (2,776)	26,332 (2,751)	$21,482^{+}$ (2,577)	31,498 (2,540)	$24,725^{+}$ (1,464)	32,254 (2,400)
German	28,300 (5,587)	28,800 (5,594)	28,200 (3,682)	31,100 (4,784)	35,400 (3,200)	37,500 (4,153)	28,700 (4,496)	28,700 (4,496)	30,400 (5,024)	32,600 (3,929)	28,900 (4,323)	28,900 (5,856)
Glass	38,312 (8,526)	38,355 (5,932)	37,792 (7,652)	36,861 (4,392)	39,740 (6,399)	42,229 (10,020)	29,978 (9,850)	29,978 (9,850)	30,455 (12,083)	31,797 (11,479)	33,680 (10,413)	31,320 (10,896)
Haberman	30,366 (5,362)	28,409 (4,190)	28,183 (6,390)	27,527 (7,788)	48,602 (13,289)	48,968 (14,254)	32,333 (7,182)	32,333 (7,182)	34,312 (6,507)	37,247 (5,754)	29,419 (5,645)	29,097 (5,749)
Heart	20,326 (3,672)	21,413 (4,943)	19,022 (5,264)	21,957 (4,683)	29,891 (4,074)	30,326 (5,086)	19,674 (3,418)	19,565 (3,403)	24,130 (2,908)	25,543 (2,629)	20,326 (3,038)	19,891 (3,507)
Ionosphere	11,952 (5,192)	14,238 (5,690)	16,516 (4,880)	17,056 (6,579)	14,794 (4,642)	13,944 (5,430)	14,508 (5,544)	14,222 (5,905)	13,103 (4,807)	12,548 (5,001)	11,111 (6,815)	12,817 (6,413)
Iris	4,667 (5,207)	4,667 (7,062)	2,667 (3,266)	2,667 (4,422)	6,000 (6,289)	6,667 (5,963)	4,000 (3,266)	4,000 (3,266)	5,333 (4,989)	4,667 (4,269)	5,333 (4,989)	5,333 (4,989)
Lymphography	22,905 (11,249)	24,952 (13,473)	20,333 (11,008)	30,619 (14,084)	24,095 (15,771)	26,810 (14,238)	25,619 (10,952)	24,286 (10,319)	22,238 (10,253)	25,619 (8,242)	20,857 (8,589)	23,667 (8,076)
Magic	22,103 (1,433)	22,687 (1,395)	62,492 (1,424)	$32,334^{+}$ (1,816)	33,738 (3,594)	34,595 (3,867)	18,948 (0,597)	18,948 (0,597)	$21,935^{+}$ (0,773)	23,596 (0,505)	17,829 (0,978)	17,960 (0,855)
Pageblocks	10,216 (0,054)	10,216 (0,057)	$16,100^{+}$ (6,160)	43,938 (10,143)	46,252 (7,898)	40,751 (6,403)	4,185 (0,709)	4,185 (0,709)	4,971 (0,989)	5,135 (1,021)	4,952 (0,971)	6,616 (2,573)
Penbased	$10,871^{+}$ (1,212)	12,236 (1,215)	5,013 (0,942)	5,913 (1,088)	4,530 (0,985)	4,694 (0,962)	0,664 (0,261)	0,664 (0,261)	0,955 (0,212)	1,137 (0,285)	1,628 (0,441)	1,719 (0,461)
Phoneme	24,187 (2,762)	24,095 (3,280)	22,206 (1,204)	23,649 (1,861)	26,943 (3,563)	26,758 (3,589)	9,697 (1,230)	9,697 (1,230)	12,213 (0,834)	13,009 (1,643)	13,546 (1,011)	13,805 (1,223)
Pima	29,036 (2,467)	28,385 (3,557)	26,946 (6,231)	28,373 (4,231)	41,152 (6,511)	43,621 (5,049)	29,303 (2,804)	29,303 (2,804)	33,228 (6,716)	33,872 (5,458)	28,377 (3,459)	27,464 (3,183)
Ringnorm	$13,743^{+}$ (1,319)	21,716 (2,475)	$17,959^{+}$ (2,153)	29,014 (2,966)	$15,716^{+}$ (1,177)	20,378 (1,466)	24,865 (1,150)	24,865 (1,150)	$18,081^{+}$ (1,265)	21,770 (1,084)	$6,014^{+}$ (1,145)	10,432 (2,299)
Saheart	45,241 (6,311)	41,549 (8,714)	36,133 (6,891)	36,152 (7,667)	40,254 (8,438)	37,655 (7,234)	33,131 (7,869)	33,131 (7,869)	34,422 (8,417)	35,731 (10,324)	34,635 (7,658)	33,779 (8,747)
Satimage	17,700 (0,757)	18,135 (0,695)	$13,053^{+}$ (1,207)	14,639 (1,526)	23,403 (1,966)	22,890 (2,588)	9,712 (0,729)	9,712 (0,729)	$9,775^{+}$ (0,917)	11,904 (0,965)	11,111 (0,966)	11,499 (0,846)
Segmentation	13,117 (1,984)	14,242 (2,277)	18,961 (3,558)	20,216 (2,316)	13,074 (2,702)	13,896 (2,483)	2,900 (0,751)	2,900 (0,751)	3,680 (0,756)	4,026 (0,929)	6,234 (0,952)	6,277 (0,892)
Sonar	16,881 (10,810)	24,524 (11,123)	$26,476^{+}$ (11,378)	38,524 (11,661)	$19,714^{+}$ (5,014)	27,881 (6,289)	13,024 (3,915)	13,024 (3,915)	14,952 (6,002)	14,929 (7,632)	21,619 (4,356)	24,976 (7,428)
Splice	$15,546^{+}$ (1,419)	39,619 (2,964)	$23,740^{+}$ (3,514)	37,112 (2,180)	$23,235^{+}$ (2,402)	34,804 (2,637)	26,813 (1,176)	26,512 (1,337)	$20,227^{+}$ (1,990)	27,481 (2,413)	$22,769^{+}$ (1,996)	27,684 (2,882)
Thyroid	7,417 (0,068)	7,417 (0,072)	46,583 (7,414)	47,847 (5,915)	45,250 (7,467)	40,222 (8,567)	7,500 (0,683)	7,500 (0,683)	11,597 (0,883)	12,139 (1,319)	8,111 (2,034)	8,486 (2,172)
Twonorm	$2,311^{+}$ (0,450)	5,405 (0,650)	$3,405^{+}$ (0,843)	4,649 (1,313)	$6,378^{+}$ (0,822)	9,000 (1,500)	5,351 (0,736)	5,351 (0,736)	$6,865^{+}$ (0,957)	9,162 (0,707)	5,932 (0,819)	5,905 (0,829)
Vehicle	38,420 (4,072)	39,007 (2,825)	40,528 (5,100)	45,148 (4,402)	36,531 (3,682)	38,780 (6,951)	30,018 (2,845)	30,018 (2,845)	30,370 (3,130)	31,786 (3,743)	32,263 (4,138)	31,074 (5,326)
Vowel	$50,303^{+}$ (4,882)	57,071 (3,846)	$33,131^{+}$ (6,091)	44,040 (6,344)	34,949 (4,764)	$24,343^{+}$ (2,978)	0,909 (1,233)	0,909 (1,233)	1,515 (1,297)	3,030 (2,020)	3,939 (2,043)	4,242 (2,109)
Waveform	$12,960^{+}$ (2,241)	16,420 (1,648)	$12,240^{+}$ (1,464)	14,500 (1,330)	$15,780^{+}$ (1,606)	18,760 (1,720)	14,280 (1,857)	14,280 (1,857)	$13,540^{+}$ (1,937)	16,820 (1,985)	13,040 (1,649)	13,300 (1,570)
Wine	2,810 (3,752)	3,399 (4,033)	27,516 (6,724)	28,105 (5,594)	6,797 (5,622)	7,843 (5,641)	5,065 (3,995)	5,065 (3,995)	3,399 (5,758)	6,209 (6,479)	10,654 (10,440)	10,654 (10,440)
Yeast	45,148 (4,051)	45,888 (3,490)	51,209 (3,461)	54,850 (5,461)	62,606 (4,550)	60,989 (5,486)	48,113 (3,288)	48,113 (3,288)	50,137 (2,995)	50,674 (2,781)	44,006 (3,771)	45,419 (3,087)
Zoo	6,000 (6,633)	6,000 (6,992)	6,818 (7,521)	5,909 (6,584)	5,000 (5,000)	9,000 (7,000)	11,909 (9,820)	11,909 (9,820)	5,000 (5,000)	5,000 (6,708)	7,909 (7,464)	8,909 (8,300)
	+|+|=		+|+|=		+|+|=		+|+|=		+|+|=		+|+|=	
	8 |0|27		9 |1|25		6 |1|28		0 |0|35		7 |0|28		3 |0|32	
Seleção/Geração	77,14%		74,29%		82,86%		100,00%		80,00%		91,43%	

Tabela 4.17 Comparação das técnicas de geração × DROP3

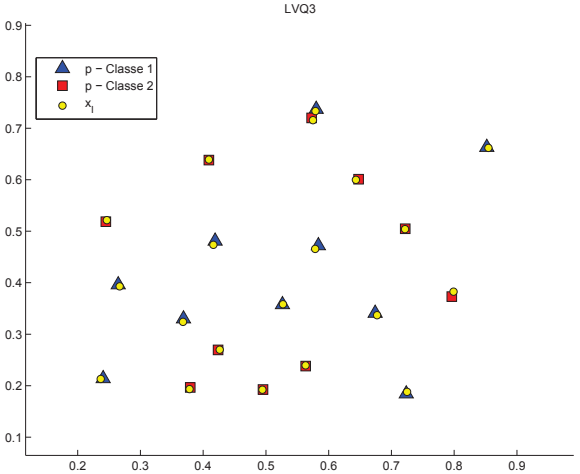
Base	SGP2	DROP3	LVQ3	DROP3	R-POC-NN	DROP3	GENN	DROP3	RSP3	DROP3	ICPL2	DROP3
Appendicitis	12,273 (7,422)	14,091 (8,354)	24,727 (9,957)	14,091 (8,354)	31,364 (16,980)	14,091 (8,354)	19,909 (12,009)	14,091 (8,354)	25,545 (8,610)	14,091 (8,354)	15,091 (8,403)	14,091 (8,354)
Australian	13,913 (4,592)	18,986 (5,555)	38,696 (4,895)	$18,986^{+}$ (5,555)	33,478 (5,667)	$18,986^{+}$ (5,555)	19,420 (4,772)	18,986 (5,555)	24,058 (3,953)	18,986 (5,555)	15,507 (4,350)	18,986 (5,555)
Balance	19,846 (6,268)	14,560 (2,066)	20,622 (4,900)	14,560 (2,066)	29,427 (6,189)	14,560 (2,066)	36,953 (6,192)	14,560+ (2,066)	33,269 (7,607)	$14,560^{+}$ (2,066)	13,441 (1,927)	14,560 (2,066)
Banana	11,830 (1,113)	12,057 (1,319)	12,132 (1,110)	12,057 (1,319)	19,208 (3,639)	12,057 (1,319)	13,245 (0,851)	12,057 (1,319)	15,528 (1,151)	12,057 (1,319)	11,415 (1,602)	12,057 (1,319)
Bupa	38,277 (6,570)	37,017 (8,897)	34,513 (7,328)	37,017 (8,897)	46,941 (6,037)	37,017 (8,897)	35,647 (3,841)	37,017 (8,897)	37,950 (4,915)	37,017 (8,897)	36,471 (8,744)	37,017 (8,897)
Cancer	3,718 (2,039)	4,437 (2,595)	3,580 (1,476)	4,437 (2,595)	8,441 (2,255)	4,437 (2,595)	4,435 (2,254)	4,437 (2,595)	6,863 (3,872)	4,437 (2,595)	3,865 (1,581)	4,437 (2,595)
Card	16,087 (2,287)	18,696 (4,122)	16,812 (5,112)	18,696 (4,122)	34,203 (7,194)	18,696 (4,122)	17,536 (2,858)	18,696 (4,122)	22,029 (5,054)	18,696 (4,122)	17,101 (5,177)	18,696 (4,122)
Ecoli	17,611 (7,024)	16,417 (5,942)	16,070 (4,981)	16,417 (5,942)	27,389 (3,757)	16,417 (5,942)	19,679 (5,280)	16,417 (5,942)	22,656 (4,985)	16,417 (5,942)	15,820 (5,293)	16,417 (5,942)
Gene	$15,686^{+}$ (2,013)	32,065 (2,141)	26,143 (3,566)	32,065 (2,141)	33,702 (1,806)	32,065 (2,141)	28,001 (2,776)	32,065 (2,141)	$21,482^{+}$ (2,577)	32,065 (2,141)	$24,725^{+}$ (1,464)	32,065 (2,141)
German	28,300 (5,587)	30,600 (5,389)	28,200 (3,682)	30,600 (5,389)	35,400 (3,200)	30,600 (5,389)	28,700 (4,496)	30,600 (5,389)	30,400 (5,024)	30,600 (5,389)	28,900 (4,323)	30,600 (5,389)
Glass	38,312 (8,526)	34,589 (9,278)	37,792 (7,652)	34,589 (9,278)	39,740 (6,399)	34,589 (9,278)	29,978 (9,850)	34,589 (9,278)	30,455 (12,083)	34,589 (9,278)	33,680 (10,413)	34,589 (9,278)
Haberman	30,366 (5,362)	28,731 (4,615)	28,183 (6,390)	28,731 (4,615)	48,602 (13,289)	28,731 (4,615)	32,333 (7,182)	28,731 (4,615)	34,312 (6,507)	28,731 (4,615)	29,419 (5,645)	28,731 (4,615)
Heart	20,326 (3,672)	19,348 (2,949)	19,022 (5,264)	19,348 (2,949)	29,891 (4,074)	19,348 (2,949)	19,674 (3,418)	19,348 (2,949)	24,130 (2,908)	19,348 (2,949)	20,326 (3,038)	19,348 (2,949)
Ionosphere	11,952 (5,192)	15,952 (6,279)	16,516 (4,880)	15,952 (6,279)	14,794 (4,642)	15,952 (6,279)	14,508 (5,544)	15,952 (6,279)	13,103 (4,807)	15,952 (6,279)	11,111 (6,815)	15,952 (6,279)
Iris	4,667 (5,207)	7,333 (5,538)	2,667 (3,266)	7,333 (5,538)	6,000 (6,289)	7,333 (5,538)	4,000 (3,266)	7,333 (5,538)	5,333 (4,989)	7,333 (5,538)	5,333 (4,989)	7,333 (5,538)
Lymphography	30,366 (5,362)	27,000 (9,351)	20,333 (11,008)	27,000 (9,351)	24,095 (15,771)	27,000 (9,351)	25,619 (10,952)	27,000 (9,351)	22,238 (10,253)	27,000 (9,351)	20,857 (8,589)	27,000 (9,351)
Magic	22,103 (1,433)	18,675 (0,769)	62,492 (1,424)	$18,675^{+}$ (0,769)	33,738 (3,594)	$18,675^{+}$ (0,769)	18,948 (0,597)	18,675 (0,769)	21,935 (0,773)	18,675 (0,769)	17,829 (0,978)	18,675 (0,769)
Pageblocks	$10,216^{+}$ (0,054)	14,346 (3,346)	16,100 (6,160)	14,346 (3,346)	46,252 (7,898)	14,346 (3,346)	$4,185^{+}$ (0,709)	14,346 (3,346)	$4,971^{+}$ (0,989)	14,346 (3,346)	$4,952^{+}$ (0,971)	14,346 (3,346)
Penbased	10,871 (1,212)	$4,230^{+}$ (0,796)	5,013 (0,942)	4,230 (0,796)	4,530 (0,985)	4,230 (0,796)	$0,664^{+}$ (0,261)	4,230 (0,796)	$0,955^{+}$ (0,212)	4,230 (0,796)	$1,628^{+}$ (0,441)	4,230 (0,796)
Phoneme	24,187 (2,762)	$15,156^{+}$ (1,289)	22,206 (1,204)	$15,156^{+}$ (1,289)	26,943 (3,563)	$15,156^{+}$ (1,289)	9,697+ (1,230)	15,156 (1,289)	12,213+ (0,834)	15,156 (1,289)	$13,546^{+}$ (1,011)	15,156 (1,289)
Pima	29,036 (2,467)	$27,211^{+}$ (3,811)	26,946 (6,231)	27,211 (3,811)	41,152 (6,511)	27,211 (3,811)	29,303 (2,804)	$27,211^{+}$ (3,811)	33,228 (6,716)	$27,211^{+}$ (3,811)	28,377 (3,459)	$27,211^{+}$ (3,811)
Ringnorm	13,743 (1,319)	14,973 (5,364)	17,959 (2,153)	14,973 (5,364)	15,716 (1,177)	14,973 (5,364)	24,865 (1,150)	$14,973^{+}$ (5,364)	18,081 (1,265)	$14,973^{+}$ (5,364)	$6,014^{+}$ (1,145)	14,973 (5,364)
Saheart	45,241 (6,311)	32,031 (6,552)	36,133 (6,891)	32,031 (6,552)	40,254 (8,438)	32,031 (6,552)	33,131 (7,869)	32,031 (6,552)	34,422 (8,417)	32,031 (6,552)	34,635 (7,658)	32,031 (6,552)
Satimage	17,700 (0,757)	$15,758^{+}$ (2,840)	13,053 (1,207)	15,758 (2,840)	23,403 (1,966)	15,758 (2,840)	$9,712^{+}$ (0,729)	15,758 (2,840)	$9,775^{+}$ (0,917)	15,758 (2,840)	$11,111^{+}$ (0,966)	15,758 (2,840)
Segmentation	13,117 (1,984)	7,359 (0,928)	18,961 (3,558)	$7,359^{+}$ (0,928)	13,074 (2,702)	$7,359^{+}$ (0,928)	$2,900^{+}$ (0,751)	7,359 (0,928)	$3,680^{+}$ (0,756)	7,359 (0,928)	$6,234^{+}$ (0,952)	7,359 (0,928)
Sonar	16,881 (10,810)	29,810 (6,789)	26,476 (11,378)	29,810 (6,789)	19,714 (5,014)	29,810 (6,789)	$13,024^{+}$ (3,915)	29,810 (6,789)	14,952 (6,002)	29,810 (6,789)	21,619 (4,356)	29,810 (6,789)
Splice	$15,546^{+}$ (1,419)	28,285 (2,319)	23,740 (3,514)	28,285 (2,319)	23,235 (2,402)	28,285 (2,319)	26,813 (1,176)	28,285 (2,319)	$20,227^{+}$ (1,990)	28,285 (2,319)	$22,769^{+}$ (1,996)	28,285 (2,319)
Thyroid	$7,417^{+}$ (0,068)	17,333 (4,106)	46,583 (7,414)	$17,333^{+}$ (4,106)	45,250 (7,467)	$17,333^{+}$ (4,106)	$7,500^{+}$ (0,683)	17,333 (4,106)	11,597 (0,883)	17,333 (4,106)	8,111 (2,034)	17,333 (4,106)
Twonorm	$2,311^{+}$ (0,450)	6,392 (0,870)	3,405+ (0,843)	6,392 (0,870)	6,378 (0,822)	6,392 (0,870)	5,351 (0,736)	6,392 (0,870)	6,865 (0,957)	6,392 (0,870)	5,932 (0,819)	6,392 (0,870)
Vehicle	38,420 (4,072)	34,151 (3,377)	40,528 (5,100)	34,151 (3,377)	36,531 (3,682)	34,151 (3,377)	30,018 (2,845)	34,151 (3,377)	30,370 (3,130)	34,151 (3,377)	32,263 (4,138)	34,151 (3,377)
Vowel	50,303 (4,882)	$8,081^{+}$ (4,140)	33,131 (6,091)	8,081+ (4,140)	34,949 (4,764)	$8,081^{+}$ (4,140)	0,909 (1,233)	8,081 (4,140)	1,515 (1,297)	8,081 (4,140)	3,939 (2,043)	8,081 (4,140)
Waveform	12,960 (2,241)	13,940 (1,473)	12,240 (1,464)	13,940 (1,473)	15,780 (1,606)	13,940 (1,473)	14,280 (1,857)	13,940 (1,473)	13,540 (1,937)	13,940 (1,473)	13,040 (1,649)	13,940 (1,473)
Wine	2,810 (3,752)	10,654 (10,440)	27,516 (6,724)	$10,654^{+}$ (10,440)	6,797 (5,622)	$10,654^{+}$ (10,440)	5,065 (3,995)	10,654 (10,440)	3,399 (5,758)	10,654 (10,440)	10,654 (10,440)	10,654 (10,440)
Yeast	45,148 (4,051)	44,886 (4,096)	51,209 (3,461)	$44,886^{+}$ (4,096)	62,606 (4,550)	44,886+ (4,096)	48,113 (3,288)	$44,886^{+}$ (4,096)	50,137 (2,995)	$44,886^{+}$ (4,096)	44,006 (3,771)	44,886 (4,096)
Zoo	6,000 (6,633)	10,909 (8,322)	6,818 (7,521)	10,909 (8,322)	5,000 (5,000)	10,909 (8,322)	11,909 (9,820)	10,909 (8,322)	5,000 (5,000)	10,909 (8,322)	7,909 (7,464)	10,909 (8,322)
	+|+|=		+|+|=		+|+|=		+|+|=		+|+|=		+|+|=	
	5 |5|25		1 |8|26		0 |8|27		7 |4|24		7 |4|24		8 |1|26	
DROP3/Geração	85,71%		97,14%		100,00%		80,00%		80,00%		77,14%	



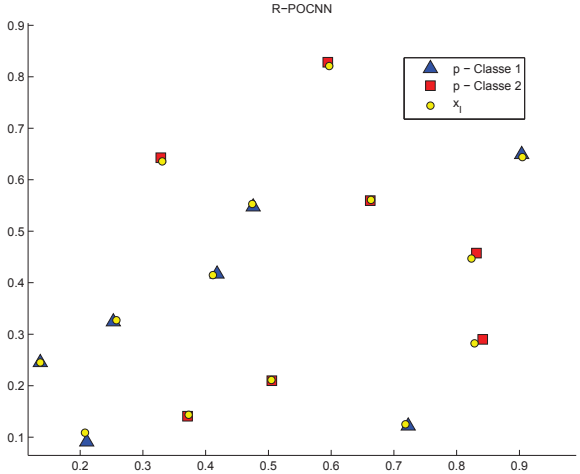
(a) Distribuição de instâncias por classes da base banana



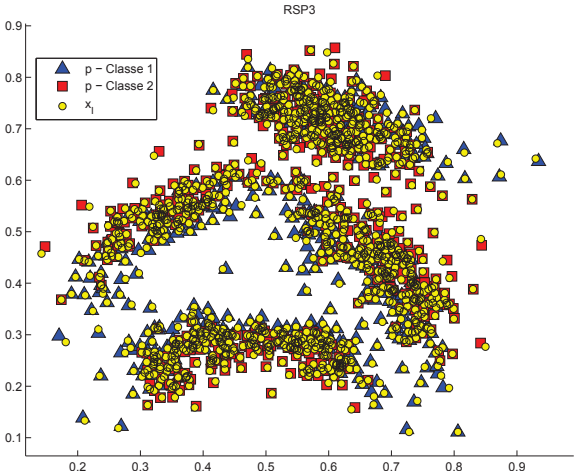
(b) Instâncias x_I × protótipos p gerados pelo SGP2



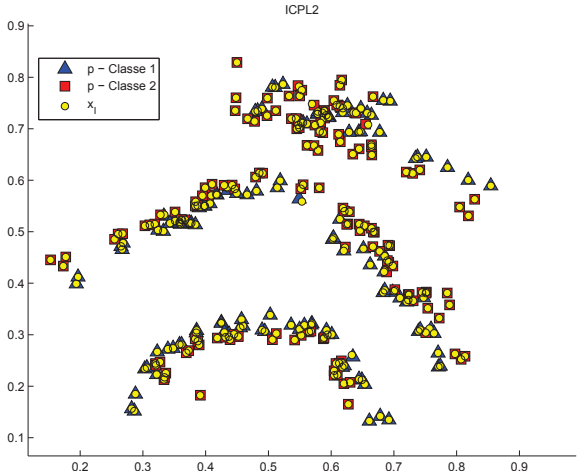
(c) Instâncias x_I × protótipos p gerados pelo pelo LVQ3



(d) Instâncias x_I × protótipos p gerados pelo R-POCNN



(e) Instâncias x_I × protótipos p gerados pelo RSP3



(f) Instâncias x_I × protótipos p gerados pelo ICPL2

Figura 4.4 Instâncias de mesma classe mais próximas dos protótipos gerados.

Tabela 4.18 Estudo das distâncias e vizinhanças dos protótipos

Base	Tx. Erro Ger.	Tx. Erro Seleção	Dist. Entre Amigos	Dist. Prot. × Amigo	Dist. Prot. × Inimigo	Cont. Inst. × Inst.	Cont. Prot. × Inst.	Cont. Prot. Mais Prox.Amigos	Cont. Prot. Mais Prox. Inimigos
appendicitis	12,273%	15,182%	0,172	0,119	0,187	0,4	1,6	1,7	0,3
australian	13,913%	15,797%	0,123	0,668	0,690	2	0	2	0
balance	19,846%	21,769%	0,256	0,163	0,359	0,1	43,4	41,4	2,1
banana1	11,830%	11,642%	0,005	0,004	0,039	13,9	26,1	38,3	1,7
bupa	38,277%	41,504%	0,133	0,053	0,137	5,6	93,4	95,6	3,4
cancer1	3,718%	4,433%	0,199	0,238	0,504	1,1	0,9	2	0
card1	16,087%	19,710%	0,376	0,942	1,204	15,9	1,3	15,7	1,5
ecoli	17,611%	18,191%	0,091	0,075	0,142	1,7	3,8	5	0,5
gene1	15,686%	47,558%	4,274	4,939	5,112	3,4	6,9	9,6	0,7
german1	28,300%	28,800%	0,620	0,554	0,750	9,7	25,8	31,9	3,6
glass1	38,312%	38,355%	0,135	0,090	0,201	6,5	27,7	33	1,2
haberman	30,366%	28,409%	0,056	0,044	0,090	11,4	18,7	26,3	3,8
heart1	20,326%	21,413%	0,312	0,687	0,920	17,6	2,1	19,6	0,1
ionosphere	11,952%	14,238%	0,327	0,379	0,869	3,9	4,5	7,4	1
iris	4,667%	4,667%	0,056	0,052	0,287	1,9	3,1	5	0
lymphography	NA	NA	NA	NA	NA	NA	NA	NA	NA
magic	22,103%	22,687%	0,067	0,071	0,107	11,3	9,5	19	1,8
pageblocks	10,216%	10,216%	0,020	0,025	0,176	1,4	0,6	2	0
penbased	10,871%	12,236%	0,164	0,199	0,530	15,5	6,4	21,9	0
phoneme	24,187%	24,095%	0,034	0,056	0,092	2	0,2	2	0,2
pima	29,036%	28,385%	0,151	0,141	0,194	7,5	13,1	18,5	2,1
ringnorm1	13,743%	21,716%	0,317	0,281	0,302	9,3	55,4	26,3	38,4
saheart	45,241%	41,549%	0,232	0,249	0,245	1,2	1,8	1,7	1,3
satimage1	17,700%	18,135%	0,181	0,157	0,309	2,5	12,3	14,3	0,5
segmentation	13,117%	14,242%	0,059	0,090	0,301	13,2	4	17,1	0,1
sonar	16,881%	24,524%	0,816	0,721	1,148	5	11,3	15,8	0,5
splice1	15,546%	39,619%	2,124	1,758	1,889	0,6	15,1	12	3,7
thyroid1	7,417%	7,417%	0,013	0,163	0,170	2	0	2	0
twonorm	2,311%	5,405%	0,308	0,254	0,322	0	2	2	0
vehicle1	38,420%	39,007%	0,211	0,163	0,270	4,2	32,1	34,7	1,6
vowel	50,303%	57,071%	0,104	0,189	0,310	35	3,8	37,5	1,3
waveform1	12,960%	16,420%	0,333	0,294	0,403	0,8	6,1	5,7	1,2
wine1	2,810%	3,399%	0,305	0,250	0,555	0,1	2,9	3	0
yeast	45,148%	45,888%	0,074	0,063	0,102	26,5	69	85,3	10,2
zoo	6,000%	6,000%	0,313	0,550	1,485	5,4	1,6	7	0

CAPÍTULO 5

CONCLUSÕES

5.1 INTRODUÇÃO

O objetivo deste trabalho foi verificar se existia necessidade de utilizar as técnicas de geração de protótipos ou de síntese, tendo em vista que técnicas de seleção de protótipos apresentam boas taxas de classificação e geralmente são mais rápidas. Analisando seis técnicas de geração de diferentes famílias e utilizando 35 bases de dados, fizemos um estudo para verificar a necessidade de se gerar protótipos ou utilizar técnicas de seleção de protótipos. Fizemos também um *benchmark* das técnicas de geração e uma técnica de seleção de referência, DROP3. Além do estudo das distâncias e vizinhanças dos protótipos gerados.

5.2 CONTRIBUIÇÕES

A partir dos extensos 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 representar a área, sem a necessidade de geração de protótipo. Então por que gerá-lo, se podemos selecioná-lo? Em algumas bases como a Gene, Ringnorm, Splice, Twonorm e Waveform, onde 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 $\times$ 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. Podemos concluir 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.

5.3 TRABALHOS FUTUROS

A partir da pesquisa realizada, alguns direcionamentos futuros podem ser apontados:

- pesquisar as técnicas de seleção de instâncias na literatura;
- desenvolver um modelo de seleção que busque instância ótima;
- pesquisar e desenvolver um modelo de classificador híbrido que utilize as técnicas de geração de protótipos e/ou seleção de instâncias de acordo com as características das classes envolvidas no problema a ser classificado;
- pesquisar as técnicas de seleção de vetores para SVM.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] D. W. Aha and D. Kibler. Noise-tolerant instance-based learning algorithms. In *International Joint Conference on Artificial Intelligence - Volume 1*, pages 794–799, 1989.

[2] 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.

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

[4] J. Bezdek, T. Reichherzer, G. Lim, and Y. Attikiouzel. Multiple prototype classifier design. *IEEE Transactions on Systems and Man and Cybernetics - Part C*, 28(1):67–69, 1998.

[5] A. Cervantes, I. Galván, and P. Isasi. An adaptive michigan approach pso for nearest prototype classification. In *2nd International Work-Conference on the Interplay Between Natural and Artificial Computation*, volume 4528, pages 287–296. Springer, 2007.

[6] C.-L. Chang. Finding prototypes for nearest neighbor classifiers. *IEEE Transactions on Computers*, 23(11):1179–1184, 1974.

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

[8] C. Chen and A. Józwik. A sample set condensation algorithm for the class sensitive artificial neural network. *Pattern Recognition Letters*, 17:819–823, 1996.

[9] T. Cover and P. Hart. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21–27, 1967.

[10] P. Datta and D. Kibler. Learning symbolic prototypes. In *International Joint Conference on Artificial Intelligence*, pages 158–166, 1997.

[11] C. Decaestecker. Finding prototypes for nearest neighbour classification by means of gradient descent and deterministic annealing. *Pattern Recognition*, 30(2):281–288, 1997.

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

[13] F. Fernandez and P. Isasi. Evolutionary design of nearest prototype classifiers. *Journal of Heuristics*, 10(4):431–454, 2004.

[14] U. Garain. Prototype reduction using an artificial immune model. *Pattern Analysis and Applications*, 11(3-4):353–363, 2008.

[15] 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.

[16] G. Gates. The reduced nearest neighbor rule (corresp.). *Information Theory, IEEE Transactions on*, 18(3):431–433, 1972.

[17] Y. Hamamoto, S. Uchimura, and S. Tomita. A bootstrap technique for nearest neighbor classifier design. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(1):73–79, 1997.

[18] P. E. Hart. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, IT-14:515–516, 1968.

[19] 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.

[20] T. Kohonen. Learning vector quantization for pattern recognition. Technical Report TKK-F-A601, Helsinki University of Technology, Espoo, Finland, 1986.

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

[22] W. Lam, C. 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.

[23] J. Li, M. Manry, C. Yu, and D. Wilson. Prototype classifier design with pruning. *International Journal on Artificial Intelligence Tools*, 14(1-2):261–280, 2005.

[24] M. Lozano, J. Sotoca, J. Sanchez, F. Pla, E. Pekalska, and R. Duin. Experimental study on prototype optimisation algorithms for prototype-based classification in vector spaces. *Pattern Recognition*, 39(10):1827–1838, 2006.

[25] R. Mollineda, F. Ferri, and E. Vidal. A merge-based condensing strategy for multiple prototype classifiers. *IEEE Transactions on Systems and Man and Cybernetics B*, 32(5):662–668, 2002.

[26] L. Nanni and A. Lumini. Particle swarm optimization for prototype reduction. *Neurocomputing*, 72(4-6):1092–1097, 2009.

[27] R. Odorico. Learning vector quantization with training count (lvqtc). *Neural Networks*, 10(6):1083–1088, 1997.

[28] C. S. Pereira and G. D. C. Cavalcanti. Prototype Selection: Combining Sef-Generating and Gaussian Mixtures for Pattern Classification. In *International Joint Conference on Neural Networks. IEEE World Congress on Computacional Intelligence*, pages 3505–3510, 2008.

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

[30] 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, July 2005.

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

[32] J. Sanchez, R. Barandela, A. Marques, R. Alejo, and J. Badenas. Analysis of new techniques to obtain quality training sets. *Pattern Recognition Letters*, 24:1015–1022, 2003.

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

[34] P. TAN, M. STEINBACK, and V. KUMAR. *Introdução ao datamining: mineração de dados*. Ciência Moderna, Rio de Janeiro, 2009.

[35] 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 and Man and Cybernetics*, 42(1):86–100, 2012.

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

[37] D. Wilson and T. Martinez. Reduction tecniques for instance-based learning algorithms. *Machine Learning*, 38(3):257–286, 2000.

[38] Q. Xie and C. Laszlo. Vector quantization technique for nonparametric classifier design. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(12):1326–1330, 1993.

[39] J. Zhang. Selecting typical instances in instance-based learning. In *International Workshop on Machine Learning*, pages 470–479, 1992.

Este volume foi tipografado em LaTeX na classe UFPEThesis (www.cin.ufpe.br/~paguso/ufpethesis).