PARTICIONAMENTO DINÂMICO PARA ALGORITMO LOCAL DE CLASSIFICAÇÃO

Eric Vinícius de Carvalho Leite

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientadores: Carlos Eduardo Pedreira
Rodrigo Tosta Peres

Rio de Janeiro
Junho de 2016

PARTICIONAMENTO DINÂMICO PARA ALGORITMO LOCAL DE CLASSIFICAÇÃO

Eric Vinícius de Carvalho Leite

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

_____

Prof. Carlos Eduardo Pedreira, Ph.D.

_____

Prof. Rodrigo Tosta Peres, D.Sc.

_____

Prof. Felipe Maia Galvão França, Ph.D.

_____

Prof. Markus Vinícius Santos Lima, D.Sc.

RIO DE JANEIRO, RJ - BRASIL

JUNHO DE 2016

Leite, Eric Vinícius de Carvalho

Particionamento Dinâmico para Algoritmo Local de Classificação / Eric Vinícius de Carvalho Leite. – Rio de Janeiro: UFRJ/COPPE, 2016.

XI, 45 p.: il.; 29,7 cm.

Orientadores: Carlos Eduardo Pedreira

Rodrigo Tosta Peres

Dissertação (mestrado) – UFRJ/COPPE/ Programa de Engenharia de Sistemas e Computação, 2016.

Referências Bibliográficas: p. 42-44.

1. Algoritmo local de classificação. 2. Protótipo. 3. Classificação Supervisionada. I. Pedreira, Carlos Eduardo *et al*. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

PARA MARIANA E MINHA FAMÍLIA

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

PARTICIONAMENTO DINÂMICO PARA ALGORITMO LOCAL DE CLASSIFICAÇÃO

Eric Vinícius de Carvalho Leite

Junho/2016

Orientadores: Carlos Eduardo Pedreira
Rodrigo Tosta Peres

Programa: Engenharia de Sistemas e Computação

Nesta dissertação é proposto um novo algoritmo supervisionado que utiliza uma abordagem local de classificação com uma arquitetura de particionamento dinâmico. Entende-se por métodos locais aqueles que utilizam informações de apenas uma determinada sub-região do espaço para a tomada de decisão, como a vizinhança de uma nova observação, ao invés de tentar encontrar uma única regra de classificação que utiliza todas as observações possíveis do problema. Desse modo, delineou-se uma metodologia de particionamento dinâmico que busca automaticamente por subconjuntos de observações com proporção majoritária de uma única classe. Infere-se através de uma taxa de erro por subconjunto, aqueles que possuem mais chance de pertencerem a uma classe e quais deles apresentam uma região mais complexa de separação de classes, onde observações com atributos similares apresentam classes divergentes. Dez conjuntos de dados reais foram testados. O algoritmo proposto mostrou desempenho bem similar à métodos clássicos da área de Reconhecimento de Padrões. Além disso, em alguns casos, há uma maior interpretabilidade e transparência dos resultados do modelo pois sabe-se em quais subconjuntos são obtidos uma melhor taxa de acerto para novas observações e quais terão taxa menor, diferente de outros métodos que apresentam uma taxa de acerto média para todo o conjunto de dados.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

DYNAMIC PARTITIONING FOR LOCAL CLASSIFICATION ALGORITHM

Eric Vinícius de Carvalho Leite

June/2016

Advisors: Carlos Eduardo Pedreira
Rodrigo Tosta Peres

Department: Systems Engineering and Computer Science

This work presents a new supervised algorithm that uses a local approach to classification with a dynamic partitioning architecture. Local methods can be defined as those that use information from only a particular sub-region of space for decision making, such as the neighborhood of a new observation, rather than trying to find a classification rule that uses all possible observations from the problem. Thus, a dynamic partitioning method that automatically search for subsets of observations with majority proportion of a single class is proposed. It is inferred through an error rate per subset, those more likely to belong to a class and which ones have a more complex area of separation of classes, where observations with similar attributes have different outputs. Ten data sets of real data were tested. The proposed algorithm showed very similar performance to traditional methods of Pattern Recognition. Moreover, in some cases, there is a greater interpretability and transparency of the model results because it is known in which subsets are obtained a better success rate for new observations and which have lower rate, unlike other methods that have an average error rate for the entire data set.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

$\mathbf{b_n}$: número total de subconjuntos homogêneos e consistentes encontrados pelo método até a iteração n

C: conjunto de subconjuntos $\{ C_1, \ldots, C_k \}$.

$\mathbf{D_{in}}$: conjunto de treinamento composto pelas observações $\{ x_{in}^1, x_{in}^2, \ldots, x_{in}^m \}$.

$\mathbf{D_{out}}$: conjunto de validação composto pelas observações $\{ x_{out}^1, x_{out}^2, \ldots, x_{out}^m \}$

Eb: erro de um subconjunto homogêneo e consistente já retirado de $\mathbf{D_{in}}$

$\mathbf{E_n}$: erro total na iteração n

k: quantidade total de subconjuntos e protótipos

LC: limiar de consistência

LE: limiar de homogeneidade

m: quantidade total de observações de $\mathbf{D_{in}}$

n: iteração

P: conjunto de protótipos $\{ P_1, \ldots, P_k \}$ associado com C

Prop1(C_i): proporção da classe 1 em C_i

Prop2(C_i): proporção da classe 2 em C_i

Q: união do conjunto W com protótipos de subconjuntos heterogêneos na última iteração

s: desvio padrão amostral

$\mathbf{S^n}$: conjunto de protótipos que representa os subconjuntos homogêneos e consistentes na iteração n

$\mathbf{T_i}$: número de observações em cada subconjunto C_i

$\mathbf{T_{b_j}}$: número de observações do subconjunto C_j homogêneo e consistente já retirado de D_{in}

W: união de conjuntos de protótipos S^n, $W = \{S^2, S^3, \ldots, S^{last}\}$

$\boldsymbol{\gamma_{C_i}}$**:** proporção da classe minoritária de C_i

LISTA DE ABREVIAÇÕES

Dp: desvio padrão

k-NN: *k-Nearest Neighbors*

LBG: Linde–Buzo–Gray

LVQ: *Learning Vector Quantization*

LVQ1: *Learning Vector Quantization 1*

LVQ2.1: *Learning Vector Quantization 2.1*

MSE: *Mean Squared Error*

OLVQ1: *Optimized Learning Vector Quantization 1*

p.p. : ponto percentual

RN: Redes Neurais

SOM: *Self-Organizing Maps*

SVM: *Support Vector Machine*

VQ: *Vector quantization*

Capítulo 1

INTRODUÇÃO

1.1 Contexto & Motivação

A área de aprendizado de máquina explora o estudo e o desenvolvimento de algoritmos que podem aprender a partir dos dados. Métodos supervisionados de aprendizado de máquina podem ser considerados como métodos estatísticos que têm por objetivo achar uma aproximação útil $\hat{f}$ para uma função f a partir de dados de treinamento. Cada instância desse conjunto é descrita por um vetor de valores de *features* e um rótulo de classe, os quais podem ser tanto discretos quanto contínuos. No aprendizado não supervisionado, uma vez que os dados apresentados não são rotulados, não há erro associado para avaliar $\hat{f}$. Esses métodos e suas aplicações foram extensamente abordados nos útimos anos em livros clássicos na área, como [1-4], e também em livros mais atuais como [5] e [6]. Seu uso é consagrado atualmente, impulsionado por alguns motivos, tais como: quantidade e variedade de dados digitais gerados nos últimos anos, aumento da capacidade de processamento computacional (mais barato e poderoso) e aumento da capacidade de armazenamento de dados de forma acessível.

Citando alguns exemplos de técnicas muito atuais de aprendizado de máquina (que não são objeto de estudo desta dissertação), os algoritmos de *deep learning* têm melhorado consideravelmente o estado-da-arte de áreas como reconhecimento de voz [7,8], detecção de objetos em imagens [9], processamento de linguagem natural [10,11] e bioinformática [12]. Esses algoritmos requerem modelos computacionais compostos de várias camadas de processamento a fim de aprender representações de dados com múltiplos níveis de abstração [13]. A plataforma cognitiva *IBM Watson* [14], por

exemplo, utiliza métodos de inteligência computacional como *deep learning* na área da saúde para apoiar médicos na tomada de decisão de diagnósticos e tratamentos médicos.

A maioria dos exemplos conhecidos de aplicações de aprendizado supervisionado de máquina são sistemas ou algoritmos que realizam a tarefa de classificação. Um algoritmo de classificação infere uma função para mapear novos dados. Idealmente essa função permitirá ao algoritmo determinar corretamente os rótulos discretos de classe para observações fora-da-amostra que não foram usadas no conjunto de treinamento. Isso exige uma capacidade de generalização efetiva na classificação de novos dados. Há vários exemplos de tarefas de classificação supervisionada como a previsão de falhas em equipamento, detecção de fraudes e filtragem de spam. O foco dessa dissertação é o desenvolvimento de um algoritmo supervisionado de classificação.

Uma outra tarefa importante de aprendizado de máquina é a regressão. A diferença entre classificação e regressão é que nesta última a saída do sistema (da função aprendida) é contínua. Por exemplo, prever o preço de uma ação na bolsa de valores, ou prever a pontuação na prova do ENEM dos alunos de um colégio, baseado na pontuação desses mesmos alunos em um simulado escolar. Na regressão linear, por exemplo, a função que deve ser aprendida é uma relação linear entre entrada e saída. Já no contexto de métodos de aprendizagem não supervisionada, o objetivo é segmentar em *clusters* os dados que compartilham tendências e padrões semelhantes. Não há rótulos de classe associado a esses dados. Algoritmos não supervisionados são usados na área de marketing de muitas empresas para entender o perfil de seus clientes, principalmente na área de varejo.

Métodos de classificação podem ser categorizados como esquemas globais ou locais [15-17]. Um algoritmo global usa todo o espaço de atributos, sem privilegiar determinadas regiões, para fornecer informações que especifiquem as fronteiras de decisão. Um exemplo (entre muitos outros possíveis) de um algoritmo global é a rede neural *feedfoward* clássica [1]. Um algoritmo com abordagem local, ao contrário, privilegia determinadas regiões do espaço que são tratadas de forma independente de outras. A ideia desta classe de algoritmos é que, como geralmente é difícil encontrar uma única regra de classificação adequada para todo o conjunto de treinamento, é preferível calcular várias regras locais, que são válidas somente para certas regiões do espaço preditor. Ao prever a classe de uma observação, a previsão é determinada principalmente pela informação local próximo à esta observação. Este tipo de método não faz suposição alguma sobre a distribuição dos dados. O algoritmo de *k-nearest neighbors* (k-NN) [2] é

um exemplo de algoritmo local. Outro exemplo é o algoritmo *support vector machine* (SVM) clássico com *radial basis function* (RBF) [18,19], também considerado um procedimento local. Um estudo de *benchmark* comparando métodos globais e locais pode ser encontrado em [20].

Vector quantization (VQ) [21,22] pode ser visto como uma forma não supervisionada de estimar a distribuição de densidade dos dados usando protótipos. Os algoritmos de *k-means* [4] (que está sendo utilizado nesta dissertação) e Linde–Buzo–Gray (LBG) [23] são exemplos de métodos não supervisionados que utilizam VQ. Embora a natureza desses algoritmos seja sem supervisão, uma família de algoritmos supervisionados foi criada a partir deste. São esquemas baseados em protótipos e conhecidos como *Learning Vector Quantization* (LVQ) [24,25]. LVQ pode ser considerado no contexto local pois uma vez posicionados os protótipos, as observações são associadas a estes gerando o particionamento desejado. Em geral esta associação se dá através da regra do vizinho mais próximo [26].

Em determinados domínios como a medicina ou a economia, pode ser muito interessante o uso de modelos mais transparentes, onde é claro e significativo quais fatores foram usados para fazer uma determinada predição ou classificação. Mais do que isso, alinhado com o contexto desta dissertação, pode ser muito relevante conhecer a exstência de diferentes comportamentos e riscos em diferentes regiões do espaço de atributos. Neste sentido busca-se algoritmos que produzem resultados transparentes e facilmente interpretáveis, em oposição à métodos não lineares como SVM ou redes neurais artificiais, onde uma quantidade muito grande de parâmetros deve ser estimada, dificultando a interpretabilidade do modelo.

Nesta dissertação é proposto um método supervisionado de classificação binária e local que é capaz de gerar resultados interpretáveis. O conceito geral do método parte da ideia intuitiva de que se a região de vizinhança de uma observação tem proporção majoritariamente de uma classe, essa observação tem maior chance de pertencer àquela classe. Este conceito está estritamente relacionado com o Teorema de Bayes e na relação entre a probabilidade *a posteriori* de uma hipótese baseado na verossimilhança e a probabilidade *a priori*. Trazendo a questão para o ponto de vista de classificação de padrões, observa-se que, dado um problema de classificação de duas classes, se tanto a probabilidade à priori dessas classes fossem conhecidas além das respectivas densidades condicionas, a hipótese com maior probabilidade a posteriori definiria qual a classificação

mais provável de uma observação x_0, com o menor erro possível. Para dados reais, o classificador bayesiano ótimo torna-se impossível, visto que as funções de densidade de probabilidade envolvidas no cálculo da regra de Bayes não são conhecidas. Desse modo, são usadas na literatura abordagens que tentam estimar a função de densidade de probabilidade de modo paramétrico e não paramétrico, para então, por Bayes, classificar cada observação para a classe com maior probabilidade *a posteriori* [27]. Essas estimativas podem apresentar limitações de acordo com a dimensão do conjunto de dados e o tamanho das amostras. Visto isso, outros métodos que estimam a probabilidade a posteriori através da estrutura e informação local podem ser relevantes.

Uma outra proposta desta dissertação é a delineação de uma arquitetura de particionamento dinâmico. Um ponto crítico para métodos locais que utilizam VQ é a definição da quantidade de protótipos que serão utilizados, o que implica diretamente no poder de generalização para novos dados. Um número superestimado de protótipos em uma amostra pode resultar em *overfitting*, ou seja, no limite, em que se tem um protótipo para cada observação, o modelo gerado é excessivamente complexo, com uma abordagem tão local que memoriza os dados de treinamento e não há aprendizagem de fato. Por outro lado, poucos protótipos em uma amostra complexa e com muitas observações resulta em um modelo muito simples, que não faz relações relevantes entre as variáveis e o *target* de saída, gerando *underfitting*. Dessa forma, uma metodologia que busca dinamicamente por um melhor particionamento do espaço, sem a definição de um número fixo de protótipos, através de regiões homogêneas de classes ao mesmo tempo em que tenta manter boa generalização e baixa taxa de erro fora da amostra torna-se interessante.

1.2 Resumo das contribuições

As principais contribuições desta dissertação são:

- ✓ Proposta e implementação de um novo algoritmo local de classificação visando manter boa generalização e baixa taxa de erro fora da amostra com maior interpretabilidade e transparência dos resultados;
- ✓ Implementação de uma metodologia de particionamento dinâmico ao método de classificação;

- ✓ Validação da metodologia proposta em 10 bancos de dados reais com esquemas de teste fora da amostra e comparação com algoritmos clássicos da literatura;

- ✓ Validação da capacidade do método em descobrir e extrair conhecimento a partir dos dados para tomada de decisão;

- ✓ Discussão dos resultados obtidos e recomendações de melhorias futuras.

1.3 Organização da Dissertação

Esta dissertação está organizada da maneira a seguir. O capítulo 2 revisa, de forma sucinta, os principais algoritmos locais de classificação presentes na literatura. No capítulo 3 é apresentado a metodologia do algoritmo proposto e os bancos de dados reais utilizados nos diversos experimentos realizados para validação do método. O capítulo 4 trata das discussões dos resultados alcançados. Em seguida, para finalizar essa dissertação, encontram-se as conclusões e considerações para trabalhos futuros.

Capítulo 2

ALGORITMOS LOCAIS DE CLASSIFICAÇÃO

Neste capítulo apresenta-se, com algum detalhe, algoritmos que utilizam métodos locais de classificação. Busca-se tornar o texto mais autocontido revisando algumas técnicas clássicas.

Na seção 2.1, é introduzido o conceito de aprendizagem local em métodos que utilizam essa abordagem. A técnica de *Vector quantization* (VQ), muito usada nesses métodos, é descrita na seção 2.2 .

Nas seções 2.3 e 2.4 são descritos, de forma objetiva, algum dos principais algoritmos locais de classificação como o *Learning Vector Quantization* (LVQ) e o *k-nearest neighbors* (k-NN).

Por fim, na seção 2.5, diferentes métodos de amostragem de dados para avaliação de classificadores são exibidos.

2.1 Aprendizagem Local

Métodos supervisionados que utilizam aprendizagem local buscam aprender a função f de classificação (ou probabilidade *a posteriori*) em sub-regiões especificas do espaço. Para isso, dado um espaço de atributos de dimensão *d*, é preciso particionar o conjunto de entrada D_{in} composto por m observações $\{ x_{in}^1, x_{in}^2, \ldots x_{in}^m \}$, $x_{in}^i \in \Re^d$ em *K* regiões R_k onde $R_k \subset \Re^d$ e que, dado duas regiões R_i e R_j, a interseção $R_i \cap R_j = \emptyset$, para todo $i \neq j$. Esses métodos estimam uma função $\hat{f}$ que mais se aproxima da função f

para cada região local [28]. A acurácia do aprendizado pode ser mensurada através do *mean squared error* (MSE), muito útil na comparação de estimadores [29]:

$$EQM(\hat{f}) = E\,[|f - \hat{f}|^2], \tag{2.1}$$

onde E[.] é o operador valor esperado.

Métodos locais apresentam boa performance em espaços de baixa dimensionalidade [30,31]. Entretanto, os mesmos tendem a ser menos efetivos quando a dimensionalidade d aumenta, principalmente para pequenos conjuntos de dados. Este resultado é conhecido por *curse-of-dimensionality*, melhor descrito para métodos locais em [30]. Dado que (2.1) pode ser decomposto em:

$$E\,[|f - \hat{f}|^2] = \underbrace{(|f - E[\hat{f}]|)^2}_{viés^2} + \underbrace{E[(|\hat{f} - E[\hat{f}]|)^2]}_{variância}, \tag{2.2}$$

onde o primeiro termo se refere ao quadradro do viés da função estimada $\hat{f}$ enquanto que o segundo termo se refere à variância da mesma. No intuito de minimizar (2.1), é preciso minimizar tanto a variância, ou seja, minimizar a variação nas estimativas de $\hat{f}$ a partir dos dados, quanto o viés, i.e., o erro ao se tentar aproximar $\hat{f}$ da função f. No entanto, esses são objetivos concorrentes. Enquanto que minimizar o viés demanda regiões locais R_k menores, menores regiões locais criam alta variância. Técnicas de *Bootstrap Aggregating* [32] ou de reamostragem podem ser usadas para reduzir a variância nas predições dos modelos. Nessas técnicas, várias repetições do conjunto de dados original são criadas usando reamostragem aleatória com substituição. Cada conjunto de dados derivado é então utilizado para a construção de um novo modelo e os modelos são reunidos em um único conjunto. Para fazer uma previsão, todos os modelos desse conjunto são levados em consideração e uma média de seus resultados é feita. Um algoritmo supervisionado que faz bom uso desse tipo de técnica é o *Random Forests* [33]. Esse algoritmo funciona através da formação de diversas árvores de decisão cada uma baseada em uma reamostragem diferente dos dados de treinamento inicial. O viés do modelo completo é equivalente ao viés de uma única árvore de decisão (que isoladamente

tem alta variação). Com a criação de muitas destas árvores e realizando a média de seus resultados, a variância do modelo final pode ser altamente reduzida.

2.2 *Vector Quantization* (VQ)

Vector Quantization [21,22] é uma técnica que emula a distribuição de densidade dos dados usando protótipos. Seja um conjunto de dados de entrada formado por *N* vetores de dimensão *d*. A quantização vetorial consiste em mapear esses dados em outro conjunto de $M < N$ protótipos, também de dimensão *d*. No contexto de classificação não supervisionada, o espaço vetorial de entrada de dimensão *d* é dividido em um conjunto de *K* regiões R_k, também de dimensão *d*, chamadas de partições ou células, e um protótipo y_k é associado a cada uma das R_k regiões, totalizando K protótipos. As partições são demilitadas por hiperplanos de dimensões *d* - 1, e todos os vetores de entrada da partição R_k são mapeados para o prototipo y_k, de forma que:

$$M\colon \Re^d \rightarrow \ C\ , \tag{2.3}$$

onde $C = \{\mathrm{y}_1, \dots, \mathrm{y}_K\}, \mathrm{y}_k \in \Re^d$ e $\mathrm{R}_k = \{x_{in}^i \in \Re^d : M(x_{in}^i) = \mathrm{y}_k\}$.

O protótipo y_k escolhido é aquele que conduzir à menor distorção. A medida de distorção mais utilizada em VQ é o MSE. Os algoritmos de *k-means* [4] e LBG [23] são exemplos de métodos de agrupamento que utilizam quantização vetorial e representam protótipos pelos centroides de cada partição.

Embora a natureza original desses algoritmos seja sem supervisão, vários algoritmos supervisionados foram baseados em esquemas de protótipos. Parte-se do princípio que, em primeiro lugar, numa etapa global, todos os dados disponíveis são utilizados para particionar o espaço em subconjuntos. Cada subconjunto tem um protótipo associado a ele. Em seguida, estes subconjuntos podem ser utilizados num procedimento para classificação local. Um exemplo é o caso do algoritmo proposto em [16], onde processo de classificação é definido da seguinte maneira: se o protótipo mais próximo de uma observação fora da amostra representa um subconjunto constituído por apenas observações de uma única classe, essa observação é classificada à mesma classe. Caso contrário, se o subconjunto tem observações de classes diferentes, um procedimento

inspirado na regra de Bayes [3] é utilizado localmente para a classificação. O método é baseado em conceitos intuitivos e mostrou um desempenho bastante competitivo em comparação com os classificadores clássicos e amplamente utilizados. Pode-se citar também o método de Aprendizado por Quantização Vetorial como exemplo de algoritmo supervisionado que utiliza protótipos, detalhado na seção 2.3 a seguir.

2.3 *Learning Vector Quantization* (LVQ)

Aprendizado por Quantização Vetorial (do inglês, *Learning Vector Quantization*, LVQ) se refere a toda uma família de algoritmos como *Learning Vector Quantization* 1 (LVQ1) [24,25], *Learning Vector Quantization* 2.1 (LVQ2.1) [34], *Optimized Learning Vector Quantization* 1 (OLVQ1) [35], entre outros, propostos por Kohonen, criador dos Mapas Auto-Organizáveis [36] (do inglês, *Self-Organizing Maps*, SOM). SOM e algoritmos de LVQ se baseiam em protótipos e seus processos de convergência consistem em fazer os protótipos se distribuirem pelo espaço a fim de representarem melhor as regiões de classe. A diferença principal é que o SOM é um algoritmo sem supervisão e algoritmos de LVQ são supervisionados.

O algoritmo LVQ1 parte do princípio a seguir. Seja D_{in} um conjunto de treinamento composto por m observações $\{ x_{in}^1, x_{in}^2, \ldots x_{in}^m \}, x_{in}^i \in \Re^d$, onde cada observação x_{in}^i pode ser associada a $C_1, \ldots, C_N$ classes possíveis. Seja $P = \{ P_1, \ldots, P_K \}$ um conjunto de protótipos posicionados no espaço para aproximar os dados de D_{in} pelos seus valores quantizados. Assume-se que para cada classe C_i é designado um subconjunto de protótipos, totalizando K protótipos. Determina-se então que a observação x_{in}^i pertence à mesma classe à qual pertence o protótipo P_m mais próximo. Desse modo,

$$c = \arg min_{m=1\ldots K}\, d_E(\, x_{in}^i,\ P_\mathrm{m}) \tag{2.4}$$

define-se o índice c do protótipo vencedor P_m mais próximo a x_{in}^i de acordo com a distância Euclidiana d_E.

Valores para P_m que minimizem o erro de classificação pela regra do vizinho mais próximo por (2.4) podem ser encontrados através do seguinte processo de aprendizagem do método. Seja $P_\mathrm{m}(t)$ os valores seqüenciais de P_m nos instantes t = 0,1,2,... . Inicie a taxa de aprendizagem $\alpha(t)$ com um valor preferencialmente pequeno e

positivo, como 0,1. Em seguida defina, para o protótipo vencedor P_c , aplique a seguinte regra:

$$\begin{cases} \text{Se classe}(x_{in}^{i}) = \text{classe}(P_c), \text{faça } P_c(t+1) = P_c(t) + \alpha(t)\,[x_{in}^{i} - P_c(t)] \\ \text{Se classe}(x_{in}^{i}) \neq \text{classe}(P_c), \text{faça } P_c(t+1) = P_c(t) - \alpha(t)\,[x_{in}^{i} - P_c(t)] \end{cases}$$

A taxa de aprendizagem deve decrescer linearmente a cada iteração [24].

O algoritmo OLVQ1 segue o mesmo princípio do LVQ1, com excessão que cada protótipo tem sua própria taxa de aprendizagem específica. Já o algoritmo LVQ2.1 é uma variação de LVQ1 no sentido em que dois protótipos, que são os vizinhos mais próximos, segundo a distância Euclidiana, de uma observação x_{in}^{i}, são atualizados simultaneamente. Um dos protótipos deve pertencer à classe correta e o outro à classe errada. Assim, x_{in}^{i} deve cair numa zona chamada de "janela" definida pelo ponto médio desses protótipos.

Uma questão-chave nos algoritmos da família LVQ é a escolha de uma medida adequada de distância ou semelhança no treinamento e classificação dos dados. Um estudo demonstrando técnicas que adaptam uma medida de distância parametrizada no treinamento do modelo pode ser encontrado em [37].

2.4 *k- Nearest Neighbor* (k-NN)

O algoritmo de classificação baseado no *k* vizinho mais próximo (do inglês, *k-nearest neighbors*, k-NN) é uma técnica amplamente usada na área de Reconhecimento de Padrões e de fácil interpretação. Proposto inicialmente por Cover e Hart em 1967 [26], esse método faz parte da categoria de algoritmos locais não-paramétricos, ou seja, que não dependem de suposições a respeito da distribuição de probabilidade dos dados. Conforme dito no capítulo 1, muitos métodos utilizam classificação baseada em estimativas das densidades condicionais a partir dos dados. O algoritmo de vizinhos mais próximos é um exemplo desses métodos. Dado um inteiro positivo *k*, uma observação x_0 e um problema de classificação binária de classes C_1 e C_2, o classificador *k-NN* primeiramente identifica os *k* vizinhos mais próximos de x_0, no conjunto de treinamento.

Então é estimado a probabilidade condicional para as classe C_1 e C_2 como uma proporção numérica desses *k* vizinhos os quais pertencem a classe C_1 e C_2, respectivamente. *k-NN* então aplica regra de Bayes e classifica a observação x_0 para a classe de maior probabilidade *a posteriori* [27].

A vizinhança de cada observação de teste é definida por uma função de distância, ou por uma função de similaridade, aonde quanto maior a similaridade entre dois elementos, menor a distância entre eles. Diversas métricas no cálculo da distância podem ser usadas como a distância de Manhattan, Minkowski, ou, sendo a mais comum, a distância Euclidiana.

A escolha do parâmetro *k* influencia no resultado final de classificação. Valores muito baixos de *k* podem aumentar a contribuição de exemplos ruidosos enquanto que valores muito altos podem aumentar a contribuição de exemplos pouco similares, e assim, menos relevantes para o problema. Trazendo para o contexto de aprendizado estatístico, para valores de *k* muito baixos dependendo da amostra, o algoritmo produz um modelo que tem baixo viés e alta variância. Para valores muito altos de *k*, pode-se produzir modelos flexíveis com baixa variância, mas alto viés.

2.5 Avaliação de Classificadores

Um ponto importante ao avaliar diferentes algoritmos de classificação supervisionada é o método de amostragem dos dados que serão utilizados para a indução do classificador. Parte-se do princípio que esses métodos formem conjuntos disjuntos para treinamento e teste, ou seja, uma parte das observações é usada para o aprendizado do algoritmo e outra para avaliação do desempenho do classificador para novos dados. Assim, os métodos de amostragem auxiliam na obtenção de uma estimativa mais fiel de erro do classificador [4]. A seguir são descritos três métodos de amostragem para avaliação de classificadores:

- ***Holdout:*** Divide-se as observações do conjunto de dados em uma porcentagem fixa *p* para treino e (1 – *p*) para teste. Um valor muito usado é *p* = 2/3

- ***Cross-Validation:*** Neste método os exemplos são divididos em partições disjuntas, chamadas de *folds*. O número de *folds* utilizados pode variar conforme a quantidade de exemplos e sua proporção em cada classe. Para *k-folds cross-validation*, o conjunto de dados é dividido em *k folds*, sendo que
  (k – 1) são usados para treinamento e os *folds* restantes para teste. Este processo se repete *k* vezes até que todos os *folds* tenham sido utilizados para teste. No caso da validação cruzada estratificada, a proporção de dados em cada uma das classes é mantida para cada *fold* gerado.
- ***Leave-One-Out:*** Este é um caso específico do *k-folds* em que *k* se refere ao número de observações do conjunto de dados. Desse modo, (k – 1) exemplos são utilizados para treinar o algoritmo de classificação e apenas um exemplo é utilizado para teste. Este método é utilizado em bancos com poucas observações devido ao seu alto custo computacional, já que esse processo se repete *k* vezes, para que todos os exemplos sejam usados uma vez para teste.

Todos esses três métodos foram implementados e usados nos experimentos da seção 4.1 do capítulo 4.

Capítulo 3

METODOLOGIA E DADOS

Neste capítulo descreve-se a metodologia do algoritmo proposto e o conjunto de dados utilizados na validação e *benchmark* com outros algoritmos de classificação já consagrados na literatura.

Na seção 3.1, é feito uma breve introdução ao método. Em seguida, na seção 3.2 é detalhado o procedimento de treinamento do algoritmo. Ainda nesta seção, um exemplo ilustrativo com dados artificiais também é mostrado.

Na seção 3.3 é descrito o procedimento de teste fora da amostra. Neste esquema é detalhado como uma nova observação, fora do conjunto de treinamento, é classificada. Para finalizar, a seção 3.4 trata dos conjuntos de dados reais e públicos usados nos experimentos do Capítulo 4.

3.1 Introdução ao Método

Este trabalho propõe um novo algoritmo supervisionado que busca uma maior interpretabilidade e transparência dos resultados do modelo gerado e boa generalização para dados fora da amostra.

Com o intuito de atingir esses objetivos, dois novos conceitos são introduzidos nesta dissertação: homogeneidade e consistência. Delineou-se uma arquitetura de particionamento dinâmico que busca automaticamente por regiões de observações com proporções majoritárias de uma única classe, subdividindo regiões mais complexas do espaço em subconjuntos menores, sem necessidade de definir um número fixo de protótipos. Isto pode ser feito através de métodos que verificam, dentro de cada

subconjunto, um limiar de homogeneidade de classes, ou seja, um percentual de tolerância para definir se aquele subconjunto é majoritariamente de uma única classe, além de um limiar de consistência, em que uma técnica de *bootstrap* com repetição é aplicada a cada subconjunto, visando minimização da variância e melhor generalização do método para novas observações. Ao final do algoritmo pode-se inferir através de uma taxa de erro por subconjunto, aqueles que possuem mais chance de pertencerem a uma classe, o que permite uma análise sobre a importância dos atributos que definem determinado subconjunto; e quais deles apresentam uma região mais complexa de separação de classes, onde observações com atributos similares apresentam classes divergentes. A Figura 3.1 ilustra, de forma geral, a ideia do método. Mais detalhes podem ser encontrados nas próximas seções.

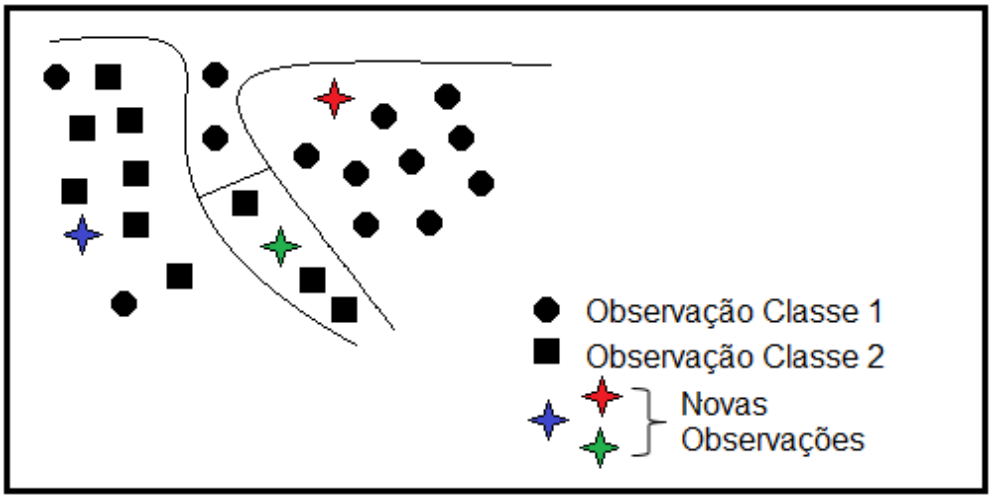

Figura 3.1: Este é um espaço preditivo imaginário em $\Re^2$. Círculos e quadrados pretos são exemplos de observações de duas classes distintas. Estrelas são observações fora da amostra de treinamento que devem ser classificadas. As linhas representam as fronteiras de decisão do método. Pelo exposto, a estrela vermelha deve ser classificada como sendo da classe 1 com alta probabilidade de acerto pois pertence a uma região cuja proporção de membros de classe 1 é majoritária. De maneira similar, a estrela verde deve ser classificada como da classe 2 com alta probabilidade de acerto. Já a estrela azul deve ser classificada como da classe 2, porém com probabilidade de erro associado maior.

3.2 Método de Treinamento

Seja D_{in} um conjunto de treinamento composto por m observações { x_{in}^1, x_{in}^2, ... x_{in}^m }, $x_{in}^i \in \Re^d$, para uma classificação binária, onde cada observação x_{in}^i pode ser associada a uma das duas classes possíveis. Algoritmos que utilizam protótipos segmentam D_{in} em partições compostas por k subconjuntos $C_1, \ldots, C_k$. Seja P = { $P_1,\ldots, P_k$} sendo $P_\text{i} \in \Re^d$, um conjunto de protótipos associados com C = { $C_1, \ldots, C_k$}. Observe que P_i é o centróide do subconjunto C_i que pode conter observações de ambas as classes. Denota-se por Prop1(C_i) e Prop2(C_i) as proporções de observações da classe 1 e da classe 2 em C_i, respectivamente. Dito isso, dois conceitos importantes são introduzidos neste trabalho: Homogeneidade e Consistência.

Nota-se que, após particionar D_{in}, alguns subconjuntos C_i podem conter observações que pertencem, a sua maioria, à mesma classe. Estes subconjuntos são chamados de homogêneos e é utilizado um limiar LE para estabelecer uma porcentagem limite da proporção de observações de outra classe neste subconjunto. Subconjuntos que não são homogêneos são chamados heterogêneos.

Em paralelo, com o intuito de minimizar a variância do método, é verificado se as proporções Prop1(C_i) e Prop2(C_i) são boas estimativas da real proporção deste subconjunto ao aplicar um procedimento chamado Teste de Consistência. Este procedimento se baseia na reamostragem aleatória com repetição das observações de C_i 50 vezes e, para cada subconjunto reamostrado {$\text{C}_{\text{i}1}$,...,$\text{C}_{\text{i}50}$}, é calculada a sua proporção minoritária $\gamma_{C_{ij}}$ = min (Prop1(C_i), Prop2(C_i)). Em seguida é calculado o desvio padrão amostral s_i de $\left(\gamma_{C_{i1}}, \ldots, \gamma_{C_{ij}}, \ldots, \gamma_{C_{i50}}\right)$. Se $s_i > LC$, C_i é definido como inconsistente. Caso contrário, é definido como consistente. LC é o limiar de consistência e seu valor é atualizado dinâmicamente, iteração a iteração. Isto significa que, se existe uma alta dispersão entre as proporções minoritárias de classe dos vários subconjuntos reamostrados de um mesmo C_i, não há robustez necessária para afirmar que essa proporção pode representar o percentual de erro de um subconjunto. Nota-se que há 50 proporções minoritárias possíveis, sempre entre 0 e 0.5. O desvio padrão amostral desses valores é, no máximo, por volta de 0.5, conforme comprovado matematicamente no Anexo A. É sugerido que LC_1 seja maior que 0.5 no primeiro passo do algoritmo.

Este procedimento é detalhado na Figura 3.2.

Teste de Homogeneidade

Entrada*: LE – limiar de homogeneidade*
Entrada*: C^n– conjunto de subconjuntos na iteração n*
1: **para** *cada subconjunto* C_i *em* C^n, **faça**
2: *Calcule as proporções Prop1(C_i) e Prop1(C_i)*
3: *Calcule a proporção minoritária* γ_{C_i} *= min (*$Prop1(C_i), Prop1(C_i)$)
4: ***se*** $\gamma_{C_i} \leq LE$ **então**
5: *Define-se que* C_i *é homogêneo*
6: **senão**
7: *Define-se que* C_i *é heterogêneo.*
8: **fim se**
9: **fim para**
10: **se** *houver ao menos um subconjunto homogêneo em* C^n, **então**
11: *Teste de Homogeneidade é verdadeiro*
12: **senão**
13: *Teste de Homogeneidade é falso*
14: **fim se**

Teste de Consistência

Entrada*: LC_n – limiar de consistência na iteração n*
Entrada*: C^n– conjunto de subconjuntos na iteração n*
1: **para** *cada subconjunto* C_i *em* C^n **faça**
2: *Aplique técnica de reamostragem com repetição localmente em* C_i
3: **para** *cada subconjunto reamostrado* $\{C_{i1}, \ldots, C_{i50}\}$ **faça**
4: *Calcule a proporção minoritária* $\gamma_{C_{ij}}$ *= min (*$Prop1(C_i), Prop1(C_i)$)
5: **fim para**
6: *Calcule o desvio padrão* $s_i\ (\gamma_{C_{i1}} \ldots \gamma_{C_{i50}})$
7: ***se*** $s_i > LC_n$ **então**
8: *Define-se que* C_i *é inconsistênte*
9: **senão**
10: *Define-se que* C_i *é consistênte*
11: **fim se**
12: **fim para**
13: **se** *todos os subconjunto* C_i *em* C^n *forem consistentes* **então**
14: *Teste de Consistência é verdadeiro*
15: **senão**
16: *Teste de Consistência é falso*
17: **fim se**

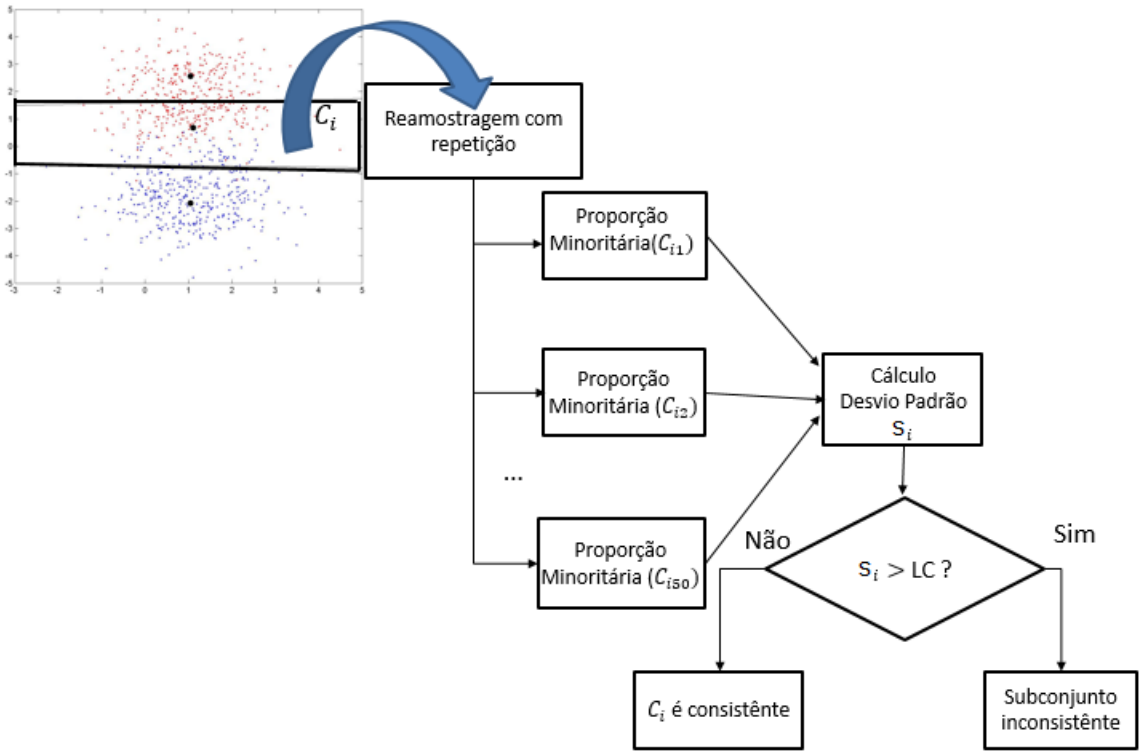

Figura 3.2: Procedimento de reamostragem com repetição para Teste de Consistência.

A Figura 3.3 ilustra a ideia geral do algoritmo proposto. Primeiramente, definem-se as condições iniciais na primeira iteração do método, tendo LC_1 = 0.7 como limiar de consistência e LE = 0.05 (5%) como limiar de homogeneidade. Em seguida há o procedimento de segmentação, sendo k, número de subconjuntos, igual a um. Utiliza-se o algoritmo *k-means* de agrupamento com distância Euclidiana como medida de similaridade para segmentar D_{in} e o algoritmo *k-means++* [38] para inicialização dos protótipos. Como é a primeira iteração, o subconjunto é consistente. Se este não for homogêneo, incrementa-se k em 1, k ← k+1, e D_{in} é re-segmentado. Caso contrário, a solução é trivial e sugere-se que o algoritmo seja reinicializado com um valor menor para LE.

Da segunda iteração em diante, o método de particionamento dinâmico proposto atualiza k em busca de subconjuntos de observações com proporções majoritárias de uma única classe, retirando conjuntos homogêneos e consistentes da base de dados e subdividindo subconjuntos mais complexos em subconjuntos homogêneos menores.

Tanto LC_2 quanto o erro total E_n são atualizados dinâmicamente em cada iteração. E_n na iteração n é calculado pela equação 3.1:

$$E_n = \frac{\sum_{i=1}^{k_n} \min(Prop1(C_i), Prop2(C_i)) T_i + \sum_{j=1}^{b_n} Eb_j . Tb_j}{m}, \quad (3.1)$$

onde m é o número total de observações no conjunto de treinamento original, $m = \#D_{in}$, T_i é o número de observações do subconjunto C_i na iteração atual, $T_i = \#C_i$, k_n é o número de subconjuntos na iteração n, Eb_j é o erro do subconjunto C_j homogêneo e consistente já retirado de D_{in}, Tb_j é o número de observações do subconjunto C_j homogêneo e consistente já retirado de D_{in} e b_n é o número total de subconjuntos homogêneos e consistentes encontrados pelo método até a iteração n.

Nota-se que o erro total E_n representa a média ponderada de cada proporção minoritária (ou erro associado, no contexto desse método) de um subconjunto na iteração n em relação a todo conjunto de treinamento D_{in}, somado à parcela de erro que já foi calculado para subconjuntos homogêneos e consistentes e retirados de D_{in}. Isto é utilizado para atualizar o Limiar de Consistência LC_n pela equação 3.2:

$$LC_n = LC_{n-1} * (E_n / E_{n-1}) \quad . \quad (3.2)$$

Observa-se que o Limiar de Consistência acompanha proporcionalmente a variação do erro total E_n. Quanto mais E_n for diminuindo à medida em que mais subconjuntos homogêneos são encontrados, o Limiar de Consistência (LC) também fica menor e, consequentemente, mais rigoroso em relação à consistência.

Assim, se todos os subconjuntos são consistentes, há duas possibilidades:

(i) Se todos são também heterogêneos, k é incrementado em 1, k ← k+1 e a base de dados é re-segmentada;

(ii) Se há algum subconjunto homogêneo, remove-se as observações desse subconjunto e segmenta o novo D_{in} com k ← 1.

No caso de existir ao menos um subconjunto inconsistente, procura-se pelos conjuntos homogêneos e consistentes, removendo suas observações e fazendo k ← 1 para segmentar o novo D_{in} outra vez. Caso não haja subconjuntos homogêneos e todos forem

heterogêneos, D_{in} é segmentado com $k_n \leftarrow k_n - 1$ protótipos e pode-se existir uma entre três possibilidades na iteração seguinte (n+1):

(i) Se algum subconjunto for homogêneo e consistente, o processo é reinicializado novamente, com $k_{n+1} \leftarrow 1$. Caso não haja mais observações no conjunto D_{in} , o algoritmo para.

(ii) Se todos os subconjuntos permanecerem heterogêneos havendo ainda alguma inconsistência, o processo continuará e D_{in} será segmentado com $k_{n+1} \leftarrow k_{n+1} - 1$. Caso $k_{n+1} < 1$, o algoritmo para;

(iii) Se todos os subconjuntos forem consistentes e heterogêneos, o algoritmo para.

Em resumo, a cada iteração o método busca por subconjuntos que satisfaçam a condição de homogeneidade e consistência. Caso haja total consistência e heterogeneidade entre os subconjuntos, incrementa-se o número de protótipos a fim de encontrar novas e menores subregiões homogêneas. Caso alguma inconsistência seja encontrada e não haja homogeneidade entre os subconjuntos, decrementa-se o número de protótipos até que a condição de consistência entre todos os subconjuntos seja encontrada. No limite, quando não houver mais dados no conjunto de treinamento ou $k_n < 1$, o algoritmo para na configuração de protótipos da iteração atual.

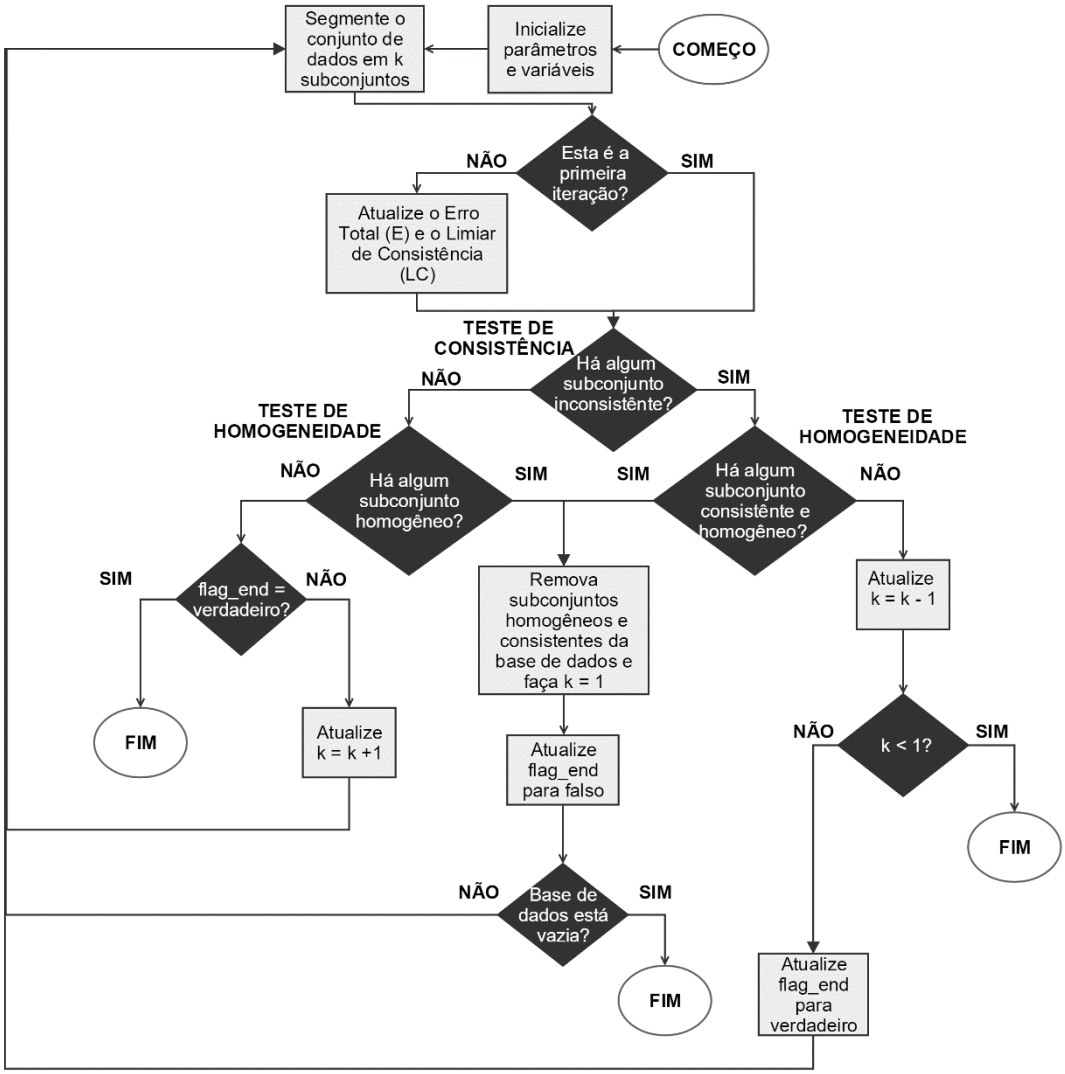

Figura 3.3: Diagrama em blocos do método de treinamento do algoritmo proposto.

Para estabelecer o método fora da amostra é necessário apresentar conceitos adicionais ainda no procedimento de treinamento. Seja P^n como o conjunto de todos os protótipos na iteração n e $W = \{S^2, S^3,\ldots,S^{last}\}$, a união de conjuntos de protótipos S^n, onde S^2 é o conjunto de protótipos que representa os subconjuntos homogêneos e consistentes na segunda iteração, …, S^{last} é o último conjunto de protótipos que representa os subconjuntos homogêneos e consistentes na última iteração do algoritmo. Não é considerado o conjunto S^1, porque se na primeira iteração os únicos subconjuntos fossem consistentes e homogêneos, o algoritmo pararia. Se, em alguma iteração n, não houver nenhum subconjunto homogêneo e consistente, o correspondente S^n é um conjunto vazio.

Caso contrário, W = W U S^n. Na última iteração, se não houver conjuntos homogêneos, define-se Q = $W_n \cup \{P_n\}$ como a união de W com os protótipos de subconjuntos heterogêneos restantes.

O pseudo-código completo do algoritmo proposto para treinamento é apresentado abaixo.

Método de Treinamento

Entrada: D_{in} *– conjunto de treinamento*
Saída: Q *– solução final de protótipos*
Inicializar: LC_n*, LE, flag_end e n para primeira iteração*
1: **enquanto** $D_{in} \neq \{\emptyset\}$ **faça**
2: *Particione* D_{in} *em* k_n *subconjuntos* $C^n = \{C_1,\ldots,C_k\}$ *representados pelos protótipos* $P^n = \{P_1,\ldots,P_k\}$ *usando um algoritmo de agrupamento baseado em centróides.*
3: *Calcule o erro total* $E_n = \frac{\sum_{i=1}^{k_n} min(Prop1(C_i),Prop2(C_i))T_i + \sum_{j=1}^{b_n} Eb_j.Tb_j}{m}$
4: **se** $n \neq 1$ **então**
5: $LC_n \leftarrow LC_{n-1} * (E_n/E_{n-1})$
6: **fim se**
7: *Aplique o Teste de Homogeneidade para todos os* k_n *subconjuntos em* C^n
8: *Aplique o Teste de Consistência para todos os* k_n *subconjuntos em* C^n
9: **se** *Teste de Consistência é falso* **então**
10: **se** *houver ao menos um subconjunto homogêneo e consistente em* C^n **então**
11: $k_{n+1} \leftarrow 1$
12: *flag_end* $\leftarrow$ *false*
13: **para** *cada subconjunto homogêneo e consistente em* C^n **faça**
14: $S^n = S^n \cup \{P_i\}$
15: $D_{in} = D_{in}$ *- observações* $\{C_i\}$
16: **fim para**
17: $W_n = W_n \cup \{S^n\}$
18: **senão**
19: $k_{n+1} \leftarrow k_n - 1$
20: **se** $k_{n+1} < 1$ **então**
21: $Q = W_n \cup \{P_n\}$
22: **Fim algoritmo**
23: **senão**
24: *flag_end* $\leftarrow$ *true.*
25: **fim se**
26: **fim se**
27: **senão**
28: **se** *Teste de Homogeneidade é verdadeiro* **então**
29: $k_{n+1} \leftarrow 1$
30: *flag_end* $\leftarrow$ *false*
31: **para** *cada subconjunto homogêneo e consistente em* C^n **faça**

32: $S^n = S^n \cup \{P_i\}$

33: $D_{in} = D_{in}$ - *observações* $\{C_i\}$
34: **fim para**
35: $W_n = W_n \cup \{S^n\}$
36: **senão**
37: **se** *flag_end is true* **então**
38: $Q = W_n \cup \{P_n\}$
39: **Fim algoritmo**
40: **senão**
41: $k_{n+1} \leftarrow k_n + 1$
42: **fim se**
43: **fim se**
44: **fim se**
45: *Incremente iteração* $n \leftarrow n+1$
46: **fim enquanto**
47: $Q = W_n$
48: **Fim algoritmo**

3.2.1 Caso ilustrativo

O método proposto é aplicado em um conjunto de dados artificias gerado a partir da mistura de duas distribuições Gaussianas em $\Re^2$, possuindo 400 observações da classe vermelha e 400 da classe azul. Neste exemplo, há grandes regiões homogênas para as ambas as classes e uma zona de fronteira aonde as classes se misturam, tornando o problema mais complexo naquela região. As Figuras 3.4, 3.5, 3.6 e 3.7 ilustram o funcionamento do método passo a passo. Em cada sunconjunto é mostrado informações na ordem crescente de: identificador numérico do subconjunto, quantidade de observações, proporção classe minoritária (usado para comparar com LE) e desvio padrão amostral do subconjunto reamostrado 50 vezes (usado para comparar com LC).

O algoritmo começa com um único protótipo, k = 1, para todo o conjunto de dados, e o problema, não surpreendentemente, não pode ser resolvido. Na iteração seguinte, o número de protótipos é aumentado em 1, mas ambos subconjuntos não atingiram o limiar LE de 1% para serem considerados homogêneos. Para k = 3, dois dos três subconjuntos tem as proporções minoritárias de uma classe menor ou igual a LE e, como também são consistentes, as suas observações são removidas do conjunto de dados inicial. O algoritmo então recomeça com um único protótipo na região de fronteira de classes, e continua iterando e aumentando o número de protótipos em busca de subconjuntos homogêneos. Subconjuntos homogêneos foram removidos do conjunto de dados da iteração 3 para 4,

7 para 8 e 11 para 12. Na iteração 15 o teste de Consistência é falso e, para aquela área mais complexa, o método pode estar se tornando local demais, com risco de *overfitting*. O algoritmo então decrementa o número de protótipos para alcançar uma configuração de subconjuntos aonde todos são consistentes, parando na iteração 16.

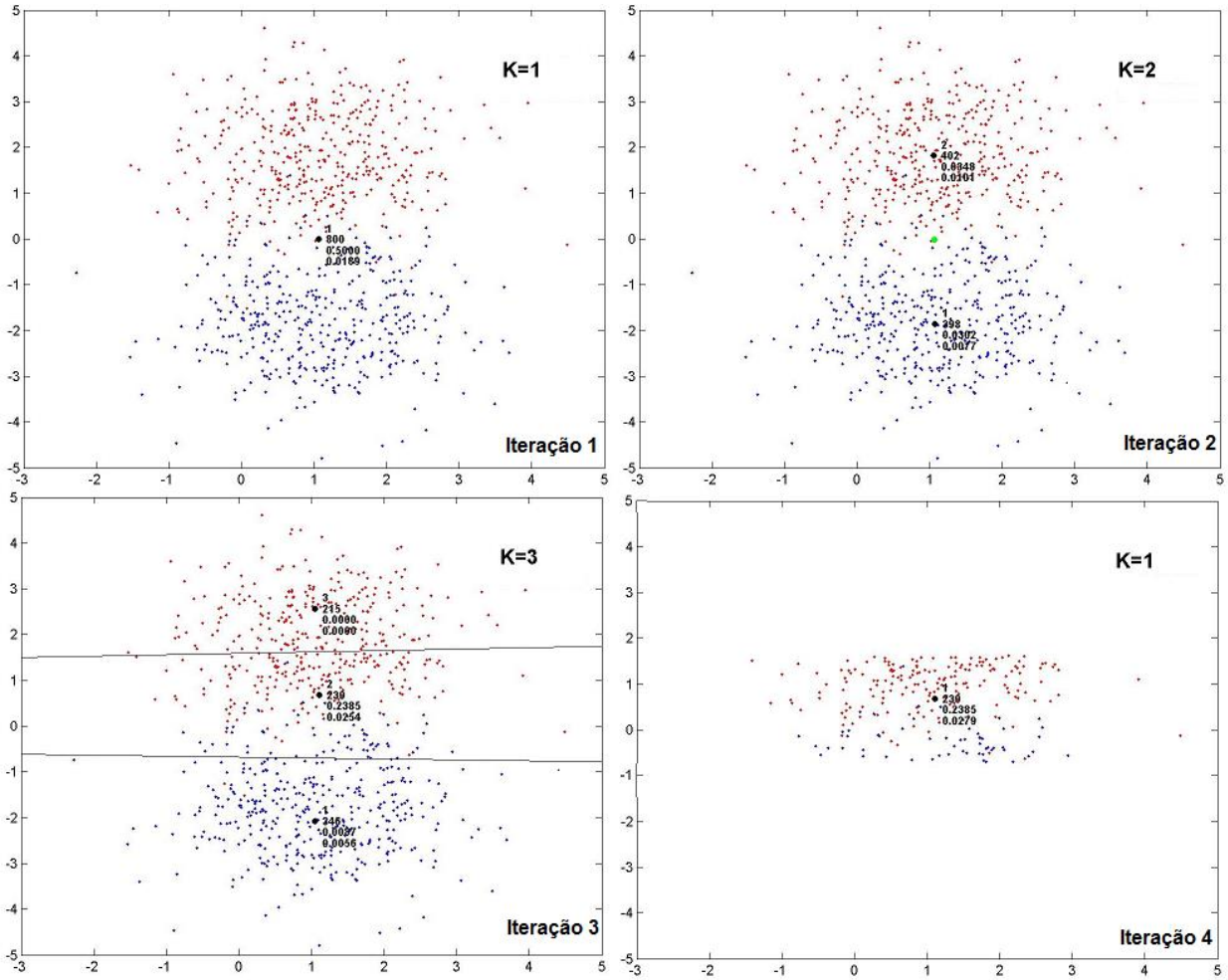

Figura 3.4: Sequência ilustrativa do método, da iteração 1 a 4. Para K=3 na iteração 3, dois subconjuntos homogêneos são encontrados.

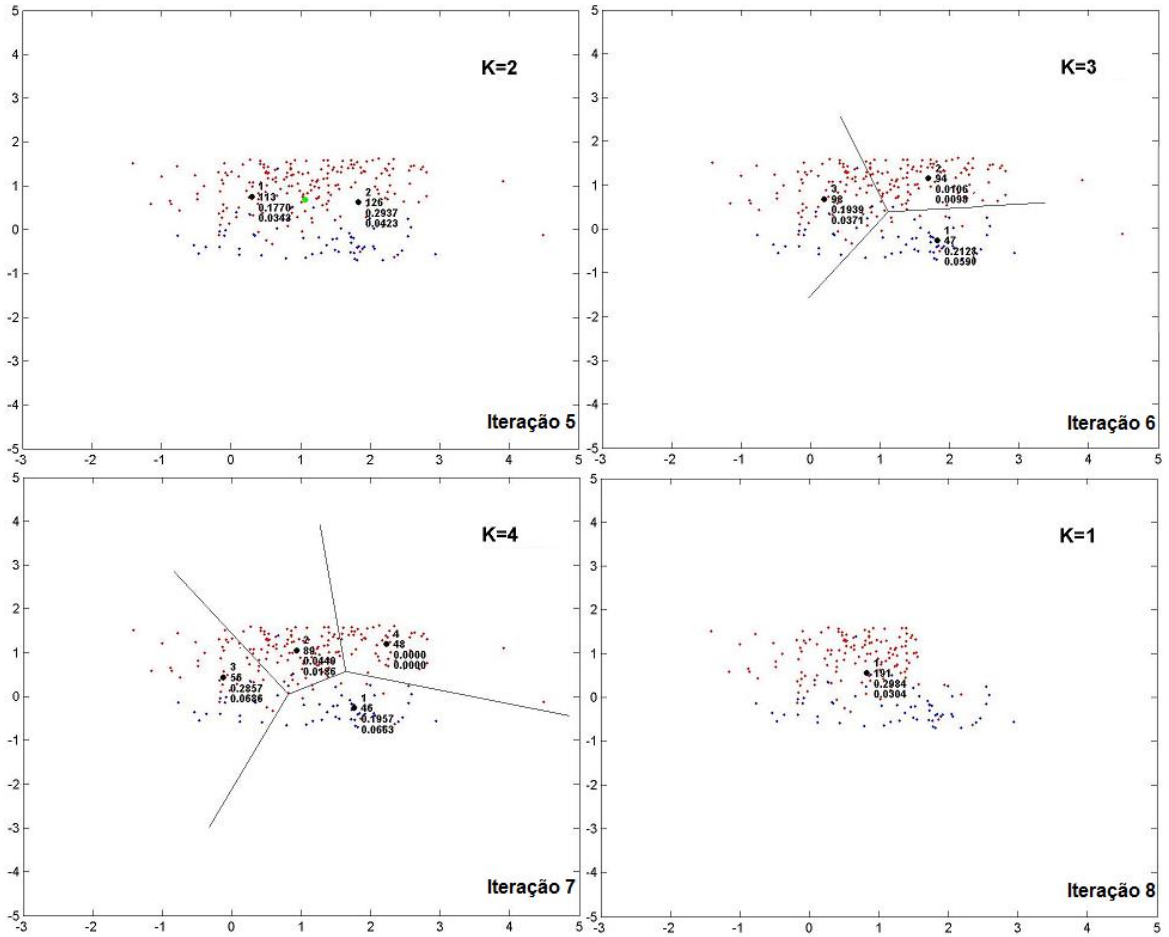

Figura 3.5: Sequência ilustrativa do método, da iteração 5 a 8. Para K=4 na iteração 7, um subconjunto homogêneo é encontrado.

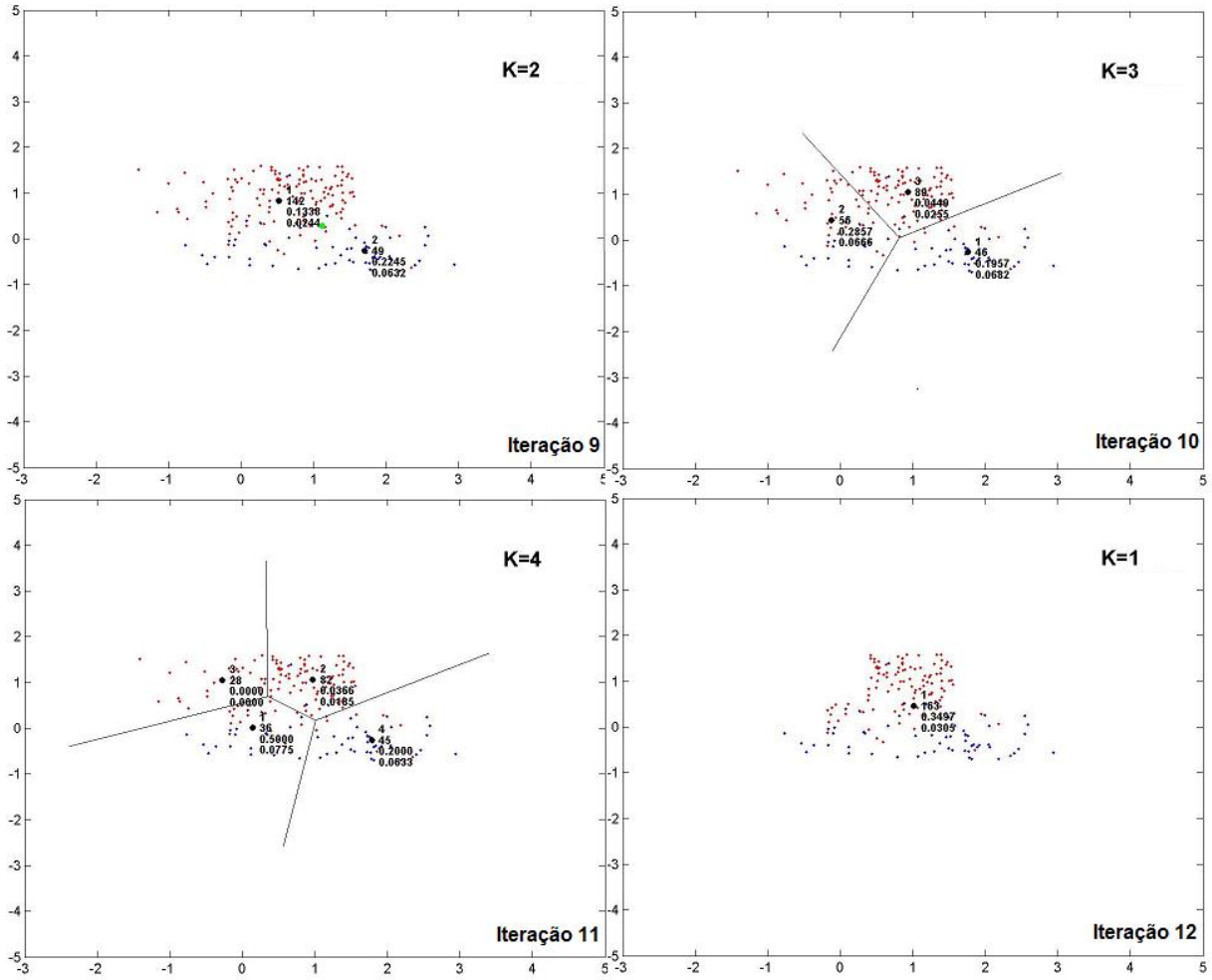

Figura 3.6: Sequência ilustrativa do método, da iteração 9 a 12. Para K=4 na iteração 11, um subconjunto homogêneo é encontrado.

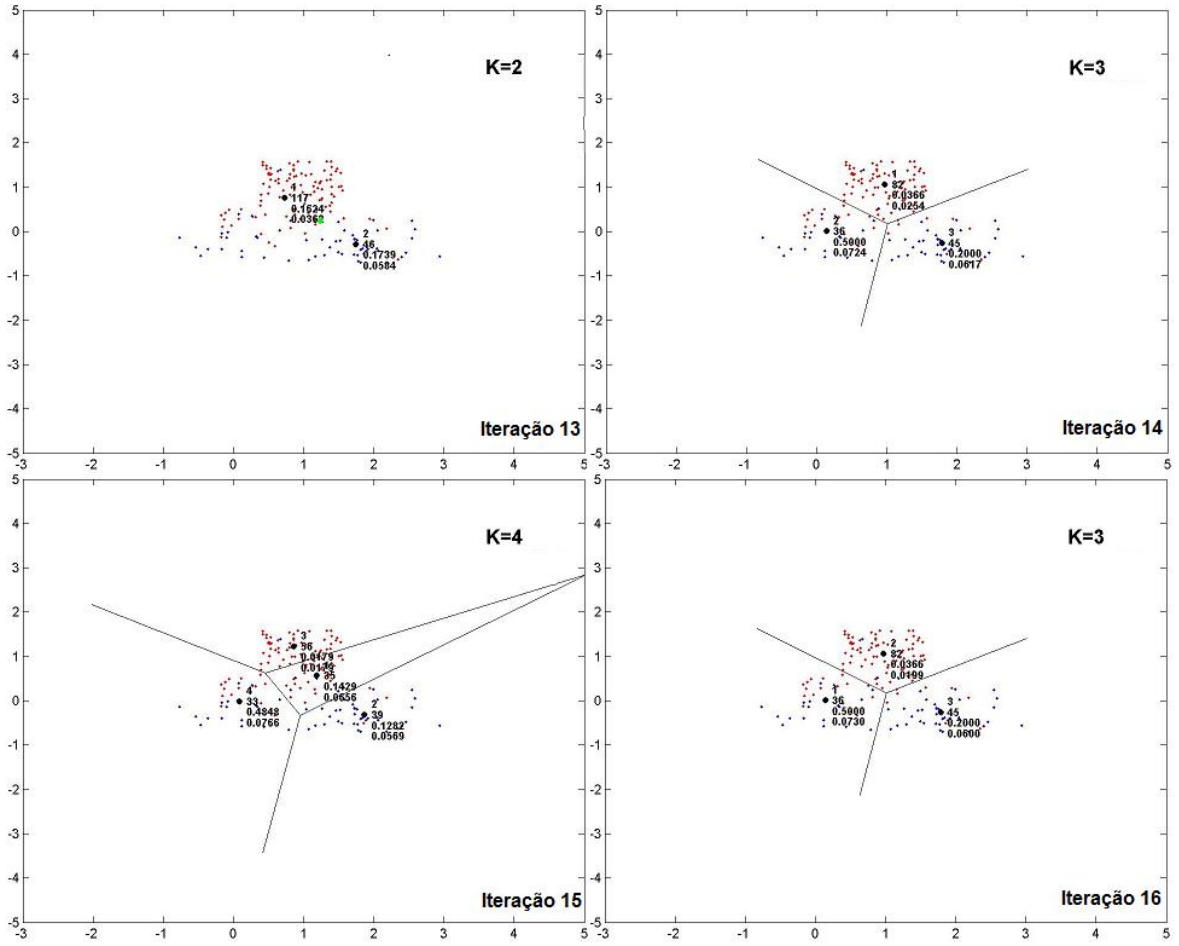

Figura 3.7: Sequência ilustrativa do método, da iteração 13 a 16. O método termina com 3 subconjuntos heterogêneos na iteração 16.

Ao final, conforme descrito na Tabela 3.1, o espaço de treinamento estará particionado em 7 subconjuntos: 4 homogêneos (A, B, C e D), sendo 3 da classe Vermelha e 1 da Azul; e 3 heterogêneas (E, F e G), com Erro da Classe Minoritária de 50%, 20% e 3,66% respectivamente. O erro total dentro da amostra foi de 4,125%, ou seja, 33 das 800 observações do conjunto de treinamento foram classificadas em subconjuntos onde a proporção majoritária era de uma classe diferente.

Tabela 3.1: Relação de subconjuntos de saída do algoritmo proposto.

Subconjunto	Tipo	Observações	Classe Majoritária	Erro Classe Minoritária
A	Homogêneo	215	Vermelha	0,00%
B	Homogêneo	346	Azul	0,87%
C	Homogêneo	48	Vermelha	0,00%
D	Homogêneo	28	Vermelha	0,00%
E	Heterogêneo	36	-	50,00%
F	Heterogêneo	45	Azul	20,00%
G	Heterogêneo	82	Vermelha	3,66%

3.3 Método de Teste

Para o procedimento fora-da-amostra, considere uma nova observação x_{out}. Se o protótipo mais próximo de x_{out} em P^2 pertencer a S^2, x_{out} será capturado pelo protótipo que representa um subconjunto homogêneo e consistente. Assim, x_{out} é classificado de acordo com a classe majoritária desse subconjunto. Caso contrário, o algoritmo verifica o protótipo mais próximo em P^3 e assim por diante, até o ultimo. O procedimento termina quando o protótipo mais próximo pertence à um conjunto em W, sempre classificando novas observações de acordo com a classe majoritária do subconjunto homogêneo correspondente. Se a busca é na última iteração e o protótipo mais próximo não pertence a S^{last} , ou S^{last} é vazio, isso significa que esse protótipo representa um subconjunto inconsistente ou consistente e heterogêneo, e x_{out} é classificado de acordo com a proporção da classe majoritária desde subconjunto.

É importante notar que no caso de o protótipo mais próximo representar um subconjunto inconsistente ou consistente e heterogêneo, o algoritmo fará uma estimativa do erro de classificação. Por exemplo, se o subconjunto heterogêneo associado conter 65% das observações de treino como classe 1 e 35% da classe 2, pode-se afirmar que há uma taxa de acerto de 65% da classificação ser correta para classe 1 majoritária. Isto significa que informação local sobre a estrutura dos dados está sendo levada em conta e o proposto algoritmo tem a vantagem de reconhecer se uma observação está numa região em que a classificação é difícil, proporcionando alguma informação sobre a incerteza local.

Procedimento fora da amostra

Entrada*:* $D_{out}-$ *conjunto de dados de validação*
Entrada: $Q-$ *solução de protótipos a partir do treinamento do método*
Saída*:* $Label\,\{x_{out}\}$
1: **para** *cada observação* x_{out}^{i} *no conjunto de validação* D_{out} **faça**
2: **para** *cada* S^j *em* Q **faça**
3: *Calcule as distancias entre* x_{out}^{i} *e cada protótipo em* $P^j = \{P_1,\ldots,P_k\}$
4: ***se*** *o protótipo mais próximo de* x_{out}^{i} *na iteração j representar* S^j **então**
5: $Label\,\{x_{out}\} \leftarrow 1,$ *se* $Prop_1(C_h) \geq Prop_2(C_h)$.
6: $Label\,\{x_{out}\} \leftarrow 2,$ *caso contrário.*
7: *onde* C_h *é o subconjunto homogêneo e consistente associado ao protótipo que representa Sj*
8: **senão**
9: ***se*** $S^j = S^{last}$ *ou* $S^{las} = \{\emptyset\}$ **então**
10: x_{out}^{i} *é classificado de acordo com a classe majoritária do subconjunto heterogêneo associado.*
11: **fim se**
12: **fim se**
13: **fim para**
14: **fim para**

3.4 Base de Dados

Nesta seção, são descritos os bancos de dados usados nos experimentos feitos nessa dissertação. Para o experimento referente à classificação, descreve-se brevemente dez bancos de dados públicos do repositório UCI [39] utilizados para aferir o desempenho do método proposto: *Glass*, *Lung Canc*er, *Sonar*, *Cleveland*, *Ionosphere*, *Pima Indians Diabetes*, *Breast Cancer Wisconsin*, *Letter Recognition*, *Statlog Project* e *Waveform Database Generator*. Para o experimento orientado à descoberta de conhecimento, o banco de dados *Student Performance* é detalhado.

O conjunto de dados *Glass* é composto de 9 atributos de entrada e 214 observações e é usado em ciência forense para a identificação de dois tipos de vidro: *window* e *non-window*. No conjunto de dados original, são fornecidos quatro tipos diferentes de vidro do tipo *window* e três tipos diferentes de vidro do tipo *non-window*. Aqui, usamos tipo *window* contra tipo *non-window*.

Três tipos de patologia de câncer de pulmão com 56 atributos de entrada constitui o banco *Lung Cancer*. Foi utilizado patologia 1 contra as outras. Este conjunto de dados tem apenas 27 observações após a remoção de valores faltantes, 8 da classe C1 (patologia 1) e 19 da classe 2 (outras patologias).

O conjunto de dados *Sonar* contém 208 observações obtidas pela aplicação de sinais de sonar em cilindros de metal e rochas em vários ângulos e sob várias condições. Cada observação é um conjunto de 60 atributos de entrada que representa a energia dentro de uma banda particular de frequência, integrada ao longo de um determinado período de tempo. O objetivo principal é o de distinguir rochas de cilindros.

O conjunto de dados de *Cleveland* ou *Heart Disease* é fornecido pela *Cleveland Clinic Foundation* visando prever a presença de doença cardíaca em pacientes. Foi utilizado o *Processed Cleveland Data* com 296 observações e 13 atributos de entrada.

O conjunto de dados *Ionosphere* refere-se a dados recolhidos por um sistema de radar em Goose Bay, Labrador, Canadá. Este sistema consiste de um agrupamento por fase de 16 de antenas de alta frequência, com uma potência de transmissão total da ordem de 6,4 kilowatts. Retornos considerados “bons” do radar são aqueles que mostram evidência de algum tipo de estrutura na Ionosfera. Retornos "ruins" são aqueles que não o fazem; seus sinais passam através da Ionosfera. Houve 17 pulsos numéricos descritos por dois atributos por pulsos, resultando em 34 variáveis em 351 observações.

O conjunto *Pima Indians Diabetes* diz respeito a 768 observações com 8 atributos de entrada, 500 sendo negativos e 268 positivos para diabetes de pacientes fêmeas da tribo indígena Pima.

Breast Cancer Wisconsin é um conjunto de dados que está relacionado com o diagnóstico citológico de massa de mama. Nove atributos são calculados a partir de uma imagem digitalizada de um aspirado de agulha fina (FNA) de uma massa de mama. Eles descrevem as características dos núcleos celulares presentes na imagem. As duas classes de saída são tumores classificados em "benigno" e "maligno". Foi realizada uma técnica de pré-processamento para remover os valores faltantes, resultando em 683 observações.

Outro conjunto de dados utilizado a partir de UCI é o *Letter Recognition*. O objetivo é identificar 26 letras maiúsculas do alfabeto Inglês. Para a classificação binária do método, definiu-se que a letra B corretamente identificada seria uma classe contra as outras 25 letras. As imagens pretas e brancas contêm letras que foram baseadas em 20 fontes diferentes onde cada letra foi aleatoriamente distorcida para produzir um banco de 20.000 observações únicas com 16 atributos (contagem de borda e medidas estatísticas), onde 10.600 observações foram utilizadas para treino e 9400 para testes fora de amostra.

O conjunto de dados *Statlog Project* é um subconjunto da base de dados utilizada no *European Statlog Project* e usa valores multi-espectrais de pixels em formato padrão de 3x3 na imagem de satélite. O objetivo é classificar as imagens dados os valores multi-espectrais. Foi testada a classe 1 contra as outras. Há 6.435 observações com 36 atributos de entradas, onde 4.435 foram utilizados em testes dentro da amostra, deixando 2.000 para a testes fora da amostra.

A base de dados *Waveform Database Generator* diz respeito a três classes de ondas. Cada classe é gerada a partir de uma combinação de 2 de 3 ondas de "base". Neste trabalho, foi testado classe 1 contra as outras duas. O conjunto de dados é composto por 5.000 observações com 21 atributos de entrada, dos quais 3.000 foram utilizadas em testes dentro da amostra e 2.000 para testes fora de amostra.

A Tabela 3.2 mostra um resumo das características desses 10 conjuntos de dados. Para cada conjunto são apresentados:

- # Instâncias: quantidade de instâncias ou observações
- # Atributos: quantidade atributos, contínuos ou discretos
- Erro da Classe Majoritária: erro cometido no caso de novas instâncias serem classificadas como sendo pertencentes à classe majoritária.

Tabela 3.2: Resumo dos Conjuntos de Dados usados para Classificação

Conjunto de Dados	# Instâncias	# Atributos	Erro da Classe Majoritária
Glass	214	9	23,8%
Lung Cancer	27	56	29,6%
Sonar	208	60	46,6%
Cleveland	296	13	46,1%
Ionosphere	351	34	35,9%
Pima Indians Diabetes	768	8	34,9%
Breast Cancer Wisconsin	683	9	35,0%
Letter Recognition	10600	16	3,8%
Statlog Project	6435	36	24,2%
Waveform Database Generator	5000	21	32,9%

Com ênfase no processo de descoberta de conhecimento (mais detalhes na seção 4.2 do Capítulo 4), dados de duas escolas secundárias de Portugal também foram analisados. O conjunto de dados *Student Performance* [40] compreende diversos atributos como notas de provas de Matemática em diferentes semestres (isto é, G1 e G2 para o primeiro e segundo semestre, respectivamente), número de faltas, idade, o consumo de álcool, escolaridade da mãe e outras variáveis demográficas e sociais. O objetivo é prever o desempenho dos alunos e identificar as principais variáveis que afetam o sucesso / fracasso educacional. A nota final de Matemática (ou seja, G3) é prevista seguindo um esquema de classificação binária: passa de ano se $G3 \geq 10$ e reprova em caso contrário. Devido ao atributo alvo G3 ter uma forte correlação com os atributos G2 e G1, foi utilizada apenas G1 na base de treinamento, resultando em um conjunto de dados com 395 observações e 33 atributos no total.

Capítulo 4

RESULTADOS E DISCUSSÃO

Neste capítulo são apresentados os resultados de dois tipos de experimentos. No primeiro, da seção 4.1, compara-se a performance em um esquema fora da amostra de diversos algoritmos já consagrados na literatura com o método proposto. O objetivo é validar o poder de discriminação de classes do método em problemas de classificação de padrões em bancos de dados reais.

Já o segundo experimento é orientado à Descoberta de Conhecimento e tem como objetivo mostrar a capacidade do algoritmo em obter conhecimento e *insights* a partir dos dados, e dessa forma, ser um diferencial aos outros algoritmos.

4.1 Experimentos orientados à Classificação

O algoritmo proposto é avaliado em um cenário de classificação supervisionada onde os conjuntos de dados foram divididos em três grupos em relação ao número de observações, com diferentes estratégias para validação fora da amostra em cada um desses grupos. O grupo A formado por base de dados com menos de 300 observações compreende os conjuntos *Glass*, *Lung*, *Sonar* e *Cleveland*. O grupo B compõe base de dados com 300 a 1000 observações como *Ionosphere*, *Pima Indians Diabetes* e *Breast Cancer Wisconsin*. Por último, os conjuntos *Letter Recognition*, *Statlog Project e Waveform Database Generator* compõem o grupo C com mais de 1000 observações. Para teste fora da amostra, o método *hold-out validation* foi aplicado para os conjuntos do grupo C. Nos grupos B e A, métodos de *10-fold cross-validation* e *leave-one-out cross-validation* foram aplicados, respectivamente.

Três outros métodos de classificação foram implementados para comparação com o método proposto: k-NN, SVM e Redes Neurais (RN). Para o k-NN, 3, 5 e 7 vizinhos foram testados; Para SVM, foi testado radial basis kernel com constante C e γ (gamma) iguais a 1; As Redes Neurais são treinadas pelo algoritmo de *backpropagation* resiliente, com os neurônios de camada escondida e de saída com função tangente hiperbólica de ativação. A função de correção de pesos é por gradiente descendente. Como não é possível determinar a priori qual a melhor topologia da rede, são realizados vários treinos com variação do número de neurônios na camada escondida, mais especificamente de 1 a 20. Além disso, para evitar a possibilidade de uma inicialização da rede ruim, são realizadas 10 inicializações, escolhendo-se portanto o melhor resultado dentre elas. O algoritmo proposto nesta dissertação, incluindo todos os procedimento de treinamento e validação, foi programado diretamente no *software* MATLAB, assim como os métodos de validação *hold-out validation*, *leave-one-out cross-validation* e *10-fold cross-validation*. As *toolboxes* de *Statistics and Machine Learning* e *Neural Network* foram usadas para testar os algoritmos de k-NN, SVM e RN.

Para o método proposto, nós usamos um limiar de 5% de erro para LE, com exceção para o conjunto de dados *Letter Recognition* onde foi usado 1% devido ao desbalanceamento na proporção de classes. Os resultados dos experimentos para todos os conjuntos de dados foram consolidados na Tabela 4.1.

Algumas observações pondem ser feitas a partir dos resultados obtidos. Em geral, o método proposto mostrou resultados competitivos em todos os dez conjuntos de dados reais quando comparado com algoritmos clássicos como k-NN, SVM e RN. A taxa de acerto média fora da amostra para o algoritmo proposto terminou apenas 0,91 pontos percentuais (p.p.) atrás do k-NN e superou o SVM em mais de 4p.p. . É importante ponderar que existem diversas técnicas e heurísticas de otimizações de parâmetros que podem melhorar o desempenho dos algoritmos citados, porém nesta dissertação visa-se apenas uma comparação de algoritmos com parâmetros ajustados por padrão.

Para conjuntos de dados que possuem entre 300-1000 observações (grupo B), o método mostrou o melhor desempenho, alcançando 85,98% na média de acerto fora da amostra, permanecendo 1,27p.p. do *k*-NN (84,71%), 1,61p.p. do RN (84,38%) e 3,93p.p. do SVM (82,05%). Em grandes conjuntos de dados (grupo C), os resultados também foram bastante competitivos: 94,36% contra 94,91% para RN, uma diferença de 0,56p.p. .Por outro lado, conjuntos de dados com poucas observações e alta dimensionalidade

como *Sonar Dataset* (208 observações e 60 atributos de entrada) destaca um problema intrínseco de algoritmos que usam quantização vetorial, medidas de similaridade baseado em funções euclidianas de distância ou de busca por vizinhos mais próximos: baixa generalização em altas dimensões [29]. Isso levou a um desempenho pior comparado ao SVM e RN por exemplo.

Tabela 4.1**:** Acerto fora da amostra (parênteses indicam acerto dentro da amostra) para conjunto de dados do grupo A (menos de 300 observações), grupo B (entre 300-1000 observações) e grupo C (mais de 1000 observações). Dp indica desvio padrão.

Base de Dados	Método proposto		KNN		SVM		RN	
	Acerto (%)	Dp	Acerto (%)	Dp	Acerto (%)	Dp	Acerto (%)	Dp
Glass	94,46 (95,33)	-	92,52 (95,77)	-	95,33 (99,53)	-	91,11 (96,42)	-
Lung	85,19 (89,03)	-	85,19 (92,88)	-	70,37 (100,0)	-	85,20 (100,0)	-
Sonar	73,08 (80,63)	-	81,73 (88,94)	-	87,02 (99,45)	-	86,23 (98,02)	-
Cleveland	61,95 (74,05)	-	63,97 (76,14)	-	53,87 (100,0)	-	64,21 (77,12)	-
Média Grupo A	**78,67 (84,76)**		**80,85 (88,43)**		**76,65 (99,74)**		**81,69 (92,89)**	
Ionosphere	88,29 (93,54)	4,98	84,61 (90,82)	3,10	92,32 (99,43)	5,04	88,10 (99,12)	4,20
Diabetes	73,17 (80,11)	3,67	71,87 (80,08)	5,66	65,11 (100,0)	0,36	70,12 (70,61)	6,60
B. Cancer	96,48 (97,25)	2,42	97,66 (97,90)	1,57	88,72 (100,0)	3,60	94,91 (99,81)	3,50
Média Grupo B	**85,98 (90,3)**	**3,69**	**84,71 (89,6)**	**3,44**	**82,05 (99,81)**	**3,0**	**84,38 (89,85)**	**4,77**
L. Recognition	98,11 (98,82)	-	99,44 (99,75)	-	96,47 (100,0)	-	98,62 (99,81)	-
Statlog	97,50 (97,70)	-	99,55 (99,57)	-	96,21 (100,0)	-	97,39 (99,90)	-
Waveform	87,46 (89,28)	-	88,32 (94,39)	-	67,06 (100,0)	-	88,72 (98,63)	-
Média Grupo C	**94,36 (95,26)**		**95,77 (97,90)**		**86,58 (100)**		**94,91 (99,45)**	
Média Total	**85,57 (89,57)**		**86,48 (91,62)**		**81,25 (99,84)**		**86,46 (93,94)**	

4.2 Experimento orientado à Descoberta de Conhecimento

Aqui o objetivo não é avaliar as capacidades de classificação de padrões do algoritmo proposto, conforme visto na secção anterior, mas mostrar a capacidade desse procedimento em extrair conhecimento a partir dos dados. Ao aplicar o método proposto no conjunto de dados *Student Performance* [40], podemos facilmente aferir os principais fatores que influenciam no desempenho dos alunos, agrupados por subconjuntos. Tanto subconjuntos homogêneos, quanto subconjuntos heterogêneos têm características que diferenciam um do outro como mostrado na Tabela 4.2. Além disso, a modelagem do desempenho de estudantes é uma importante ferramenta para educadores e alunos, uma vez que ajuda a melhor compreender os aspectos sociais desse contexto e encontrar soluções adequadas para cada caso.

O resultado desse experimento pode demonstrar diversos perfis de estudantes. Há, por exemplo, aqueles que são muito dedicados e bem-sucedidos como o subconjunto homogêneo A na Tabela 4.2. Estudantes desse subconjunto são comprometidos com os estudos, possuem pouca ou quase nenhuma falta escolar, praticamente nenhuma reprovação e notas altas em provas passadas (i.e, G1), conforme visto na Figura 4.1. Ao mesmo tempo, também há os alunos que são bem-sucedidos na prova final, apesar de notas menores na G1, mas que são diferentes do subconjunto homogêneo A. Eles consomem álcool durante os fins de semana, com uma vida social bem agitada e estudam menos de 2 horas por semana. Educadores poderiam por aplicar ações específicas para aumentar ainda mais as notas desse perfil de aluno. Em contraste, alunos que estudam menos de 2 horas por semana, que possuem pais com baixa educação formal e não querem cursar o ensino superior são estudantes de alto risco que precisam de ações específicas e especial apoio de professores e educadores, conforme visto no subgrupo C. Alunos que vivem em área rural e estudam menos de 2 horas por semana também podem apresentar problemas, como mostrado no subconjunto D. Já os conjuntos heterogêneos representam regiões nebulosas onde estudantes possuem comportamentos similares mas diferentes resultados na nota final G3. Subconjunto E por exemplo, demonstra 6 estudantes que

passaram (classe 1) e 6 estudantes que reprovaram (classe 2). Tendo entre 13 e 19 faltas escolares e baixas notas passadas (i.e.,G1) entre outros atributos, estudantes desse subconjunto tem 50% de chance de serem aprovados ou reprovar. Esses são exemplos mais relevantes e ilustrativos do total de 20 subconjuntos homogêneos e heterogêneos encontrados pelo método.

Tabela 4.2: Relação dos 6 subconjuntos mais relevantes encontrados pelo método

Subconjunto A - Homogêneo	Subconjunto B - Homogêneo
101 alunos aprovados (classe 1)	7 alunos aprovados (classe 1)
Baixo consumo de álcool	Alto consumo de álcool
Índice de reprovação muito baixo	Índice de reprovação muito baixo
2-5 horas de estudo por semana	Menos de 2 horas de estudo por semana
Baixo número de faltas	Sai com amigos muitas vezes

Subconjunto C - Homogêneo	Subconjunto D - Homogêneo
7 alunos reprovados (classe 2)	6 alunos reprovados (classe 2)
Alto consumo de álcool	Moderado consumo de álcool
1-3 reprovações	0-3 reprovações
Menos de 2 horas de estudo por semana	Menos de 2 horas de estudo por semana
Não quer cursar ensino superior	Quer cursar ensino superior
Pais com baixa escolaridade	Pais com baixa escolaridade
Sai com amigos muitas vezes	Vive em área rural

Subconjunto E - Heterogêneo	Subconjunto F - Heterogêneo
6 alunos aprovados (classe 1)	4 alunos aprovados (classe 1)
6 reprovados (classe 2)	10 reprovados (classe 2)
Baixo índice de reprovações	0-3 reprovações
Maioria estuda de 2 a 5 horas por semana	Baixo consumo de álcool
Quer cursar ensino superior	Quer cursar ensino superior
13-19 faltas escolares	Pais com baixa escolaridade

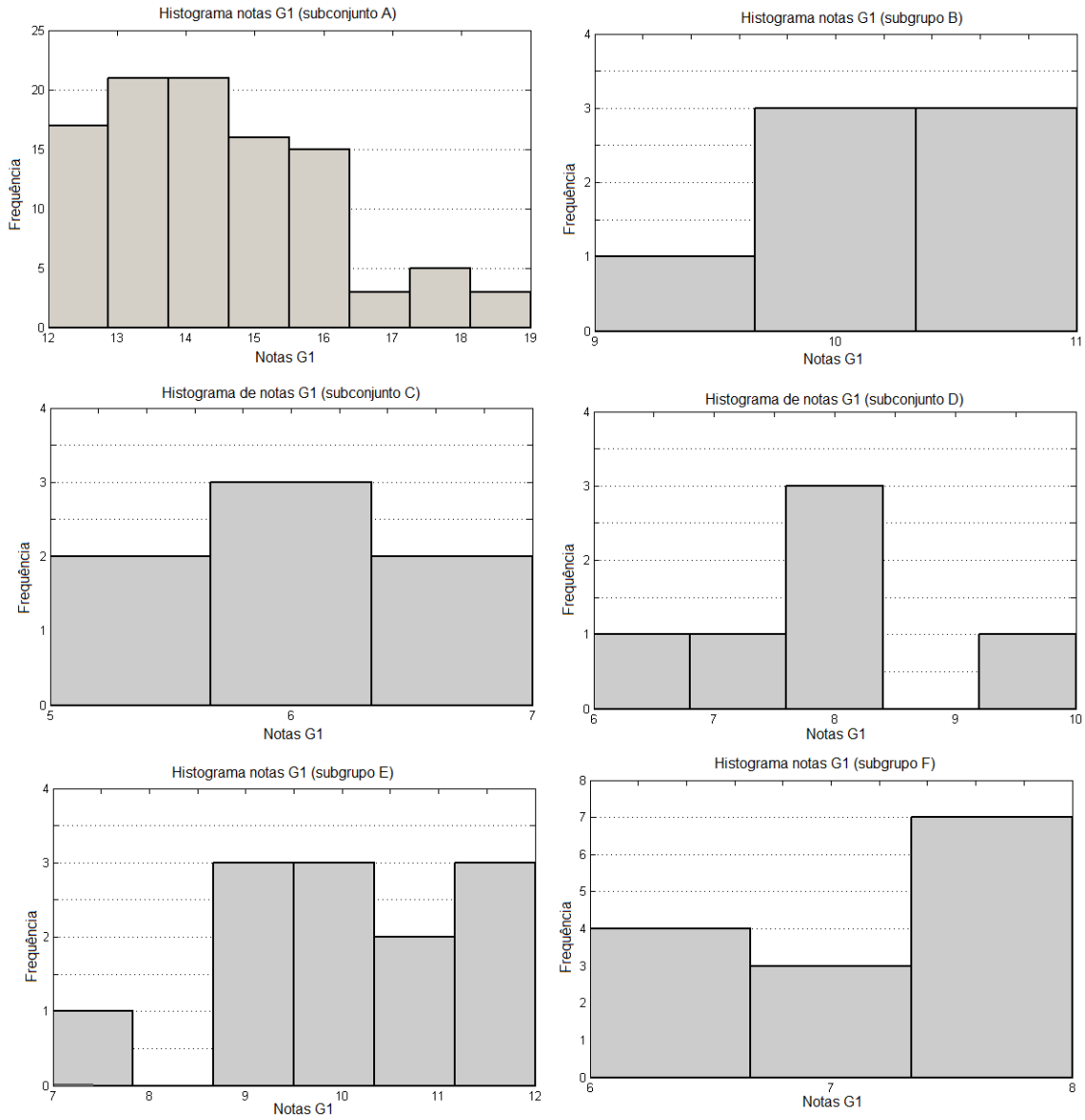

Figura 4.1: Histograma das notas G1 dos 6 subconjuntos mais relevantes encontrados pelo método.

Desse modo, demonstra-se a capacidade do algoritmo proposto em usar informação local dos dados de treinamento no auxílio à tomada de decisão.

CONCLUSÃO

Nesta dissertação foi proposto um novo algoritmo de classificação que utiliza abordagem local e particionamento dinâmico para classificação de novas observações. Busca-se automaticamente, através da estrutura e informação local, por subconjuntos de observações com proporções majoritárias de uma única classe, subdividindo regiões mais complexas do espaço em subconjuntos menores. Para isso, delineou-se uma arquitetura de particionamento dinâmico que garanta tanto a homogeneidade dos subconjuntos quanto à sua consistência. Ao final do algoritmo, pode-se inferir através de uma taxa de erro por subconjunto, aqueles que possuem mais propensão a ser de uma classe e as suas características que os definem; e quais subconjuntos apresentam uma região mais complexa de separação de classes, onde observações com atributos similares apresentam diferentes *outputs* de saída.

Haja vista os experimentos realizados, o algoritmo proposto se mostrou efetivo ao manter uma boa generalização fora da amostra e ter um desempenho bem similar a métodos clássicos da área de *Machine Learning* e Reconhecimento de Padrões como SVM, k-NN e RN. Seus *outputs* em forma de subconjuntos podem fornecer fácil interpretação dos resultados, o que pode ser útil em alguns domínios onde modelos não lineares desenvolvidos por técnicas como SVM ou Redes Neurais, que possuem uma quantidade muito grande de parâmetros deve ser estimada, dificulta a interpretabilidade do método. Além disso, é conhecido quais subconjuntos apresentarão uma melhor taxa de acerto para novas observações e quais terão taxa menor, diferente de outros métodos que apresentam uma taxa de acerto média para todo o conjunto de dados. Pode-se então afirmar que o algoritmo apresentado cumpriu os requisitos delineados nessa dissertação.

Limitações do método incluem sensibilidade a conjunto com poucos dados e de alta dimensionalidade, pois esses afetam a aprendizagem local ligada à protótipos. Além disso, para conjunto de dados muito grandes e complexos, com classes muito misturadas, pode-se ter um custo computacional alto na busca por regiões homogêneas.

Como trabalho futuro, é proposto generalizar o método para multiclasses. Além disso, há outros diversos métodos estatísticos que podem ser usados para testar a consistência e estabilidade de um subconjunto ou cluster, como por exemplo, o índice de Jaccard [41]. Outra proposta é automatizar o processo de descoberta de conhecimento do

algoritmo ao propor um método de identificação de atributos mais relevantes para cada subconjunto. Neste caso, ficaria mais evidente quais características são mais importantes e que melhor definem determinado subconjunto ou grupo de subconjuntos apresentado como *output* do método.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] BISHOP, C. M., *Pattern Recognition and Machine Learning*, 1st ed., New York, Springer-Verlag, 2006.

[2] DUDA, R. O., HART, P.E., STORK, G., *Pattern recognition*, 2nd ed., USA, Wiley, 2001.

[3] MITCHELL, T., *Machine Learning*, 1st ed., New York, McGraw-Hill, 1997.

[4] HAN, J., KAMBER, M., PEI, J., *Data Mining: Concepts and Techniques*, 3nd ed., Morgan Kaufmann, 2011.

[5] JAMES, G., *et al*., *An Introduction to Statistical Learning with Applications in R*, Springer Science+Business Media New York, 2013.

[6] ABU-MOSTAFA, Y.S., MAGDON-ISMAIL, M., LIN, H-T., *Learning From Data*, AMLBook, 2012.

[7] MESNIL, G., *et al*., 2015, "Using recurrent neural networks for slot filling in spoken language understanding". *IEEE Transactions on Audio, Speech, and Language Processing*, v.23, n.3, pp.530-539.

[8] HANNUN, A., *et al*., "Deep Speech: Scaling up end-to-end speech recognition", *ArXiv e-prints*, 1412-5567, December 2014.

[9] CIRESAN, D., MEIER, U., SCHMIDHUBER, J., *Multi-column Deep Neural Networks for Image Classification*. In: Technical Report No. IDSIA-04-12, 2012.

[10] SOCHER, R., *et al*., 2013, "Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank", In: *Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642, Stroudsburg, PA, Outubro.

[11] GAO, J., HE, X., YIH, W., DENGE, L., 2014, "Learning Continuous Phrase Representations for Translation Modeling", In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pp. 699–709, Baltimore, Maryland, Junho.

[12] CHICCO, D., SADOWSKI, P., BALDI, P., 2014, "Deep autoencoder neural networks for gene ontology annotation predictions". In: *Proceedings of the 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics*. pp. 533–540, Newport Beach, CA, USA, Setembro.

[13] LECUN, Y., BENGIO, Y., HINTON, G., 2015, "Deep Learning", *Nature*, v.521, pp. 436–444.

[14] IBM Watson http://www.ibm.com/watson/ Acessado em 01-08-2016

[15] PERES, R. T., 2008, *Novas Técnicas de Classificação de Padrões Baseadas em Métodos Local-Global*. Tese de D.Sc., PUC, Rio de Janeiro, RJ, Brasil.

[16] PERES, R. T., PEDREIRA, C. E., 2010, "A new local-global approach for classification", *Neural Networks* v.23, pp. 887-891.

[17] PERES, R. T., PEDREIRA, C. E., 2009, "The generalized risk zone: Observations selection for classification", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 31, n. 7, pp. 1331-1337.

[18] VAPNIK, V., *Estimation of Dependences Based on Empirical Data*, 1st ed., New York, Springer-Verlag, 1982.

[19] CORTES, C., VAPNIK, V., 1995, "Support-vector networks". *Machine Learning*, v.20, n. 3, pp.273–297.

[20] SCHIFFNER, J., WEIHS, C., BISCHL, B., 2013, "Benchmarking local classification methods", *Comput Stat*, n. 28, pp. 2599–2619.

[21] GERSHO, A., GRAY, R. M., *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, 1999.

[22] GRAY, R.M., NEUHOFF, D.L., 1998, "Quantization", *IEEE Trans. Information Theory*, v.44, n.6, pp. 2325-2383.

[23] LINDE, Y., BUZO, A., GRAY, R., 1980, "An Algorithm for Vector Quantizer Design". *IEEE Transactions on Communications* v.28, n.1, pp 84-95.

[24] KOHONEN, T., Self-organization and associative memory, 1st ed., Berlin, Springer -Verlag, 1989.
GOODFELLOW, I., BENGIO, Y., COURVILLE, A., *Deep Learning*, MIT Press, 2016.

[25] KOHONEN, T., 1995, "Learning vector quantization", In: *M.A. Arbib, The Handbook of Brain Theory and Neural Networks*, Cambridge, MA, MIT Press, pp. 537–540.

[26] COVER, T., HART, P., 1967, "Nearest neighbor pattern classification", *IEEE Transactions on Information Theory*, v.13, n.1, pp.21-27.

[27] HASTIE, T., TIBSHIRANI, R., FRIEDMAN, J. H., *The elements of statistical learning: data mining, inference, and prediction*, 2nd ed., Springer New York, 2009.

[28] PENG, J., *et al.*, 2001, "Local Discriminative Learning for Pattern Recognition", *Pattern Recognition*, v. 34, p. 139-150.

[29] LEHMANN, E. L., CASELLA, G., Theory of Point Estimation, 2nd ed., New York, Springer-Verlag, 1998.

[30] FRIEDMAN, J.H., *Flexible metric nearest neighbor classification*. Tech. Report, Dept.of Statistics, Stanford University, Stanford, CA, 1994

[31] SALZBERG, S., 1991, "A nearest hyperrectangle learning method", *Mach. Learning* v.6, pp. 251-276.

[32] BREIMAN, L., 1996, "Bagging predictors", *Machine Learning*, v.24, n.2, pp. 123–140.

[33] HO, T. K., 1995, "Random Decision Forests". In: *Proceedings of the 3rd International Conference on Document Analysis and Recognition*, pp. 278–282, Montreal, QC, Agosto.

[34] KOHONEN, T., 1990, "Improved versions of Learning Vector Quantization". In: *Proceeding of the National Joint Conference of Neural Networks*, pp. 545-550, San Diego, CA, USA.

[35] KOHONEN, T., 1992, "New Developments of Learning Vector Quantization and the Self-Organazing Map". In: *Symposium on Neural Networks: Alliances and Perspectives*, Senri, Osaka, Japão.

[36] KOHONEN, T., 1990, "The self-organizing maps". In: *Proceedings of the IEEE*, v.78, pp. 1464-1480.

[37] SCHNEIDER, P., HAMMER, B., BIEHL, M., 2009, "Adaptive Relevance Matrices in Learning Vector Quantization", *Neural Computation*, v.21, pp. 3532–3561.

[38] VASSILVITSKII, A., D., S., 2007, "K-means++: The Advantages of Careful Seeding." In: *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1027–1035, New Orleans, Janeiro.

[39] "UCI Repository". Disponível em: http://archive.ics.uci.edu/ml/ Acesso em 23-03-2016.

[40] CORTEZ, P., SILVA, A., 2008, "Using Data Mining to Predict Secondary School Student Performance". In: *Proceedings of 5th Future Business Technology Conference*, pp. 5-12, Porto, Portugal, Abril.

[41] HENNING, C., 2007, "Cluster-wise assessment of cluster stability". *Comp Stat Data Anal 2007*, v. 52, pp. 258–271.

ANEXO A

Neste anexo visa-se demonstrar matematicamente como o desvio padrão amostral *s* usado na comparação com LC no Teste de Consistência, descrito com mais detalhes na seção 3.2 do capítulo 3, pode ser no máximo igual a 0.5.

Dada 50 proporções minoritárias $\gamma_1 \ldots \gamma_{50}$ de uma variável $\boldsymbol{\gamma}$ com valores dentro do intervalo [0,0.5] por definição. O valor esperado E($\boldsymbol{\gamma}$), por dedução, também estará dentro deste intervalo. Dado que o desvio padrão amostral de $\boldsymbol{\gamma}$, por definição, é igual a:

$$s = \sqrt{E((\boldsymbol{\gamma} - \mathrm{E}(\boldsymbol{\gamma}))^2)}$$

A partir disso, prova-se que s também será menor ou igual a 0.5 pela dedução abaixo:

$$|\gamma_i - \mathrm{E}(\boldsymbol{\gamma})| \leq 0.5$$

$$(\gamma_i - \mathrm{E}(\boldsymbol{\gamma}))^2 \leq 0.5^2$$

$$(\gamma_i - \mathrm{E}(\boldsymbol{\gamma}))^2 \leq {}^1/_4$$

$$\frac{\sum_{i=1}^{50}(\gamma_i - \mathrm{E}(\boldsymbol{\gamma}))^2}{50} \leq \frac{50.{}^1/_4}{50}$$

$$\sqrt{\frac{\sum_{i=1}^{50}(\gamma_i - \mathrm{E}(\boldsymbol{\gamma}))^2}{50}} \leq 0.5$$

$$s \leq 0.5$$