UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

"Aprendizado de Máquina Baseado em Separabilidade Linear em Sistema de Classificação Híbrido-Nebuloso Aplicado a Problemas Multiclasse"

Aluno: Carlos Cesar Mansur Tuma

Orientador: Prof. Dr. Maurício Figueiredo

São Carlos

Junho/2009

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

T925am

Tuma, Carlos Cesar Mansur.
Aprendizado de máquina baseado em separabilidade linear em sistema de classificação híbrido-nebuloso aplicado a problemas multiclasse / Carlos Cesar Mansur Tuma. -- São Carlos : UFSCar, 2009.
131 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2009.

1. Inteligência artificial. 2. Aprendizagem de máquina. 3. Classificação. 4. Método geométrico. 5. Sistema classificador nebuloso. I. Título.

CDD: 006.3 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

"Aprendizado de Máquina Baseado em Separabilidade Linear em Sistema de Classificação Híbrido Nebuloso Aplicado a Problemas Multiclasse"

CARLOS **CÉSAR** MANSUR TUMA

Dissertação de **Mestrado** apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação

Membros **da Banca:**

Prof. Dr. Maurício Fernandes Figueiredo
(Orientador – DCNFSCar)

Profa. Dra. Heloisa de Arruda **Camargo**
(DCNFSCar)

Profa. Dra. Myriam Regattieri De Biase da Silva Delgado (DAINF/CPGEI/UTFPR)

São Carlos
Junho12009

Agradecimentos

À minha esposa Célia e às minhas filhas Ana e Marília pelo amor e compreensão.

Ao Criador por propor este desafio.

À Profa. Maria do Carmo Nicoletti pelos conselhos, conhecimentos e compreensão, assim como pelo acesso ao laboratório e seus recursos durante todo o período do mestrado.

Ao meu orientador, Prof. Maurício Fernandes Figueiredo pelas idéias, incentivo, dedicação, amizade e orientação, além das conversas interessantes.

A todos os professores e funcionários do Departamento de Computação, pelas gentilezas e incentivo.

Aos colegas e amigos da pós-graduação pelas trocas de idéias e apoio.

À UFSCar a quem dedico este trabalho.

À CAPES pelo apoio financeiro.

Resumo

Este trabalho de mestrado descreve um sistema classificador inteligente aplicado a problemas multiclasse não-linearmente separáveis chamado Slicer. O sistema adota uma estratégia de aprendizado supervisionado de baixo custo computacional (avaliado em $O(N^2)$) baseado em separabilidade linear. Durante o período de aprendizagem o sistema determina um conjunto de hiperplanos associados a regiões de classe única (subespaços). Nas tarefas de classificação o sistema classificador usa os hiperplanos como um conjunto de regras se-entao-senao para inferir a classe do vetor de atributos dado como entrada (objeto a ser classificado).

Entre outras caracteristicas, o sistema classificador é capaz de: tratar atributos faltantes; eliminar ruídos durante o aprendizado; ajustar os parâmetros dos hiperplanos para obter melhores regiões de classe única; e eliminar regras redundantes.

A teoria nebulosa é considerada para desenvolver uma versão híbrida com características como raciocínio aproximado e simultaneidade no mecanismo de inferência.

Diferentes métodos de classificação e domínios são considerados para avaliação. O sistema classificador Slicer alcança resultados aceitáveis em termos de acurácia, justificando investir em futuras investigações.

Palavras-chave: separabilidade linear, problema de classificação multiclasse não-linear, aprendizagem de máquina, método geométrico de classificação, sistema classificador nebuloso.

Abstract

This master thesis describes an intelligent classifier system applied to multiclass non-linearly separable problems called Slicer. The system adopts a low computacional cost supervised learning strategy (evaluated as $O(N^2)$) based on linear separability. During the learning period the system determines a set of hyperplanes associated to one-class regions (sub-spaces). In classification tasks the classifier system uses the hyperplanes as a set of if-then-else rules to infer the class of the input attribute vector (non classified object).

Among other characteristics, the intelligent classifier system is able to: deal with missing attribute values examples; reject noise examples during learning; adjust hyperplane parameters to improve the definition of the one-class regions; and eliminate redundant rules.

The fuzzy theory is considered to design a hybrid version with features such as approximate reasoning and parallel inference computation.

Different classification methods and benchmarks are considered for evaluation. The classifier system Slicer reaches acceptable results in terms of accuracy, justifying future investigation effort.

Keywords: linear separability, multiclass non-linear problems, machine learning, geometric classification method, fuzzy classifier system.

Sumário

.

CAPÍTULO 3

CAPÍTULO 4

Algoritmos

Tabelas

Figuras

Lista de Acrônimos

AM - Aprendizado de Máquina

BCP - Barycentric Correction Procedure

BD - Base de Dados

BR - Base de Regras

CARVE - Constructive Algorithm for Real-Valued Examples

CLS - Class Linear Separability

k-NN - k Nearest Neighbor

MI - Mecanismo de Inferência

MLP - Multi Layer Perceptron

NLS – Não Linearmente Separável

RDP - Recursive Deterministic Perceptron

RM - Reduced Multivariate Polynomials

SL – Separabilidade Linear

SFBR - Sistema Fuzzy Baseado em Regras

SFCBR - Sistema Fuzzy de Classificação Baseado em Regras

SVM - Support Vector Machines

WM - Wang-Mendel

CAPÍTULO 1

INTRODUÇÃO

Uma das atividades cognitivas mais comuns é a classificação de objetos (exemplos). De uma forma geral a classe de um exemplo é inferida a partir do conjunto de seus valores de atributos. A importância desta capacidade reside no fato de revelar as propriedades do objeto a partir de sua classe (ou seja, propriedades comuns à classe) [Hand et al. 2001].

Na atualidade a capacidade de classificação tem sido necessária em diversos segmentos de aplicação prática, dentre eles: controle de processos, diagnósticos, etc, sendo inclusive o núcleo em sistemas de mineração de dados [Witten & Frank 2005].

Na prática, a tarefa de classificação é realizada por sistemas classificadores automáticos, cujo projeto, em muitos casos, envolve a manipulação de um conjunto de exemplos classificados. O resolução do problema de classificação, tal como é conhecido a manipulação do conjunto de exemplos, não possui uma solução trivial.

Na verdade vários desafios têm sido enfrentados, e devem ser sanados antes de se obter resultados utilizáveis [Briscoe e Caelli 1996]. Entre esses desafios estão:

- Dimensionalidade, ou seja, número de atributos (características) dos exemplos do problema a ser resolvido (alguns problemas ultrapassam a casa de milhar), o que influi direta ou indiretamente no custo computacional, tornando necessário o uso de alguma estratégia para tentar reduzi-la. Esse desafio é conhecido como a maldição da dimensionalidade;
- Cardinalidade, ou seja, número de exemplos a ser utilizado no aprendizado, esse número tem que ser significativo o suficiente para que se possa induzir um mapeamento adequado, porém sua determinação não fácil e varia de problema a problema. Influi diretamente no custo computacional e é comum prejuízos no aprendizado devido a uma cardinalidade baixa ou alta demais;
- Ruídos nos exemplos, ou seja, a classe associada ao exemplo corresponde às características associadas a outra classe. A má qualidade dos exemplos influencia diretamente na qualidade do mapeamento obtido;

- Atributos faltantes, como o número de exemplos é importante no aprendizado, dispensar exemplos com atributos faltantes normalmente não é uma boa prática, e por outro lado pode ocorrer que o exemplo a ser classificado pelo mapeamento não possua todos atributos;
- Tempo necessário para se fazer o aprendizado, leia-se mapeamento dos exemplos, pois em muitas situações o tempo é de suma importância;
- Tempo para classificação, pois em muitas situações se deseja obter respostas rápidas às demandas;
- Espaço de memória para guardar o mapeamento obtido, pois, às vezes, essa restrição é relevante.

Por conta da importância da solução destes problemas e de suas dificuldades, têm-se desenvolvido vários métodos, utilizando-se de diversas estratégias (heurísticas), gerando, pois, uma gama de métodos diferentes [Mitchell 1997].

Algumas das principais propostas utilizadas atualmente são comentadas a seguir:

- Redes neurais: inspiradas biologicamente, as redes neurais são capazes de adquirir conhecimento a partir de um modelo de aprendizagem supervisionado, gerando os pesos de uma estrutura de nós [Mitchell 1997]. Estas estruturas podem ser fixas ou dinâmicas em sua construção, sendo estas últimas conhecidas como redes neurais construtivas [Neto e Nicoletti 2005].

- Programação quadrática: consiste em modelar o problema de classificação como um problema de otimização de programação quadrática, por exemplo, o SVM (“Support Vector Machines”) [Cristiani e Taylor 2000], em que se busca gerar uma representação linearmente separável por meio de várias funções kernels.

- Sistemas Fuzzy Baseados em Regras (SFBR): permite combinar dados numéricos com informação linguística, tal como em [Wang and Mendel 1992]. Nestes casos gera-se uma representação linguística de um problema.

- Métodos geométricos: buscam por meio de separabilidade linear encontrar subconjuntos de classe única ou com mínimo erro, tal como em CLS (Class Linear Separabilty) que busca hiperplanos que separem subconjuntos de exemplos com máxima cardinalidade [Tajine et al. 1997], ou CARVE

(Constructive Algorithm for Real-Valued Examples) que se utiliza de "convex hull" para gerar subconjuntos de classe única [Young & Downs 1998].

- Métodos estatísticos: buscam por meio de entropia e métodos estatisticos localizar de qual classe um vetor de atributos se aproxima mais, tal como o classificador "Naive Bayes" que é um método Bayesiano de aprendizado considerado muito útil em aplicações práticas. Sua simplicidade vem da presunção que os atributos dos exemplos são condicionalmente independentes para a classificação de um novo exemplo [Mitchell 1997].
- Métodos híbridos: buscam unir as características de duas ou mais metodologias gerando um sistema classificador mais robusto, tal como BCP (Barycentric Perceptron) que se utiliza de baricentros e hiperplanos para gerar os nós de redes neurais, [Poulard 1995]; ou RDP (Recursive Deterministic Perceptron) que se utiliza do método CLS para gerar seus nós de rede neural [Tajine & Elizondo 1998]. Quando se diz gerar os nós, significa não só criá-los em número como também seus pesos associados.

Como pode-se perceber, o problema de classificação tem uma fase inerentemente de aprendizagem quando se trata de induzir conhecimento por meio de exemplos. Neste sentido Aprendizado de Máquina (AM), subárea de pesquisa em Inteligência Artificial, tem importância fundamental em problemas de classificação. O modelo de AM caracterizado como indutivo é o modelo mais bem sucedido e o que tem o maior número de algoritmos e sistemas que o implementam [Duda et al.2001] [Mitchell 1997].

Algoritmos e sistemas de aprendizado indutivo de máquina pressupõem a existência de um conjunto de exemplos que representam determinadas categorias (de objetos, situações, problemas, etc.), chamado de conjunto de treinamento, a partir do qual o algoritmo/sistema aprende. Um sistema de aprendizado simbólico induz, a partir do conjunto de treinamento, um conjunto de expressões, que representam as categorias em questão. O sistema aprende por meio da generalização do conjunto de treinamento em um conjunto de expressões, que podem ser usadas, a seguir, para categorizar futuros exemplos. As maneiras como essas expressões são representadas depende do algoritmo/sistema de aprendizado de máquina utilizado.

O conjunto de treinamento é de importância fundamental na indução das expressões que descrevem as categorias presentes no conjunto. Se os exemplos não

forem representativos das categorias que representam, tampouco o serão as suas generalizações. Geralmente os exemplos do conjunto de treinamento são descritos por um vetor de pares atributo-valor_de_atributo e uma classe (categoria) associada. O fato da classe do exemplo estar presente na sua descrição e ser utilizada para a indução do conceito caracteriza o chamado aprendizado supervisionado (ou seja, os exemplos que serão usados pelo sistema, para aprender, foram previamente classificados por alguém, ou algum dispositivo).

Há várias perspectivas possíveis consideradas as taxonomias de métodos de classificação. Dentre as classes de métodos indutivos, uma das principais perspectivas se refere à informação disponível. As seguintes classes são observadas neste caso:

- aprendizado supervisionado: cada exemplo de treinamento está associado à classe a que pertence, ou seja, um especialista informa, para cada exemplo, a que classe do conceito a ser aprendido este exemplo pertence.
- aprendizado não supervisionado: os exemplos de treinamento não possuem classes associadas. Geralmente métodos de busca de cluster são usados nestes casos.

Sob o ponto de vista do processo de aprendizagem:

- aprendizado não incremental: o método de aprendizado usa, como entrada, todo o conjunto de treinamento, de uma só vez, induzindo o conceito após isso. Adequado para situações em que se julga ter nos exemplos de treinamento o suficiente para um aprendizado do conceito satisfatório.
- aprendizado incremental: o método de aprendizado usa, como entrada, um único exemplo; e a cada entrada reformula o conceito induzido, caso o conceito atual não se ajuste à entrada dada. Adequado para situações que exigem aprendizado contínuo (on-line).

Aprendizado indutivo pode ser caracterizado, também, através dos vários métodos que o implementam, com destaque a:

- redes neurais;
- árvores de decisão e indução de regras;
- algoritmos genéticos;

- aprendizado baseado em exemplos; e
- aprendizado baseado em separabilidade linear.

Estas categorias de métodos não são, necessariamente, mutuamente exclusivos, podendo um método se encontrar em mais de uma categoria.

Dentre as categorias dos métodos baseados em separabilidade linear, estas se dividem em quatro grupos dependendo de sua estratégia principal [Elizondo 2006]:

- programação linear;
- geometria computacional;
- redes neurais;
- programação quadrática.

Todos estes métodos são influenciados pela cardinalidade do conjunto de exemplos do domínio a ser apreendido. Esta cardinalidade é o parâmetro para o cálculo do custo computacional de um algoritmo (“Big oh”) [Aho & Ullman 1992] [Cormen et al. 2002].

Com o objetivo de criar um método com baixo custo computacional e com a garantia de término do treinamento em tempo determinado, investigam-se várias metodologias utilizadas em aprendizagem em sistemas classificadores.

Entre as metodologias estudadas está a de separabilidade linear a qual garante que, se um conjunto finito de exemplos for consistente, é possível separar pelo menos 1 elemento de cada vez através do uso de um hiperplano e, recursivamente, executar essa operação até esvaziar o conjunto de exemplos e obter um conjunto de hiperplanos separadores.

O sistema classificador (nomeado Slicer) proposto e implementado é voltado a problemas multiclasse não-linearmente separáveis e possui relativamente baixo custo computacional (avaliado em $O(N^2)$).

Em sua fase de treinamento adota uma estratégia de aprendizagem supervisionada cujo mecanismo está baseado em separabilidade linear. Dentre outras características, o sistema classificador nesta fase é capaz de considerar exemplos com atributos faltantes e estimar a presença de ruídos. Além disso, o sistema ainda aprimora o conjunto de hiperplanos, buscando orientações eficientes para corte do espaço e

elimina redundâncias nas regras de classificação geradas. Ao final do treinamento o sistema detém um conjunto de hiperplanos identificadores de regiões de classe única.

Na fase de classificação um mecanismo de inferência fundamentado em regras se-então-senão (cada qual associada a um dos hiperplanos) identifica a classe do vetor de atributos.

Por fim, na sua versão híbrida, o sistema agrega a teoria de sistema nebulosos para então exibir propriedades de raciocínio aproximado e de simultaneidade no mecanismo de inferência (antes seqüencial).

O sistema é avaliado a partir da comparação com propostas bem aceitas na literatura, buscando confirmar suas potencialidades em termos de acurácia e custo computacional.

Uma versão preliminar do Sistema Classificador Slicer, contando apenas com a versão básica, foi usada na identificação de fases de crescimento de cultivo de *S. pneumoniae* obtendo bons resultados e sendo publicado nos Anais do XVII Congresso Brasileiro de Engenharia Química 2008 [Tuma ET AL. 2008].

O restante deste trabalho está organizado da seguinte forma:

O Capítulo 2 apresenta os conceitos, definições e notações relevantes ao entendimento dos capítulos subsequentes.

O Capítulo 3 descreve o algoritmo chamado SLICER, proposta deste trabalho de pesquisa, em cinco versões diferentes. A primeira essencialmente baseada em conceitos geométricos e limitada apenas a domínios sem atributos faltantes e sem tolerância a ruídos. As demais acrescentam novas características à versão básica, passando a admitir ruídos, atributos faltantes, buscar melhor hiperplano separador e reordenando o conjunto de regras com o objetivo de eliminar regras redundantes. Todas as versões permitem aprendizado em espaços com um número finito de dimensões, tendo como conjunto de treinamento exemplos pertencentes a um número finito de classes.

O Capítulo 4 descreve uma versão híbrida Slicer-Nebuloso tendo como variáveis de entrada do modelo nebuloso as regras de saída do método Slicer. Esta versão é desenvolvida com 3 e 7 partições, usando o método Wang-Mendel como gerador de regras nebulosas, embora haja algumas modificações, simplificações e acréscimos, principalmente no tocante aos critérios de classificação.

O capítulo 5 apresenta os resultados obtidos em vários domínios de conhecimento, e para efeito de comparação de desempenho também produzem-se resultados em domínios sem atributos faltantes e domínios com atributos faltantes.

O capítulo 6 apresenta as conclusões retiradas deste trabalho assim como propostas de pesquisas futuras.

Após as referências, apresenta-se o apêndice, no qual se incluem todas as tabelas dos resultados obtidos.

CAPÍTULO 2

FUNDAMENTAÇÃO TEÓRICA

Neste capítulo apresentam-se o cenário em que se encaixa a pesquisa ora desenvolvida e os conceitos e notações utilizados no decorrer do trabalho, com objetivo de familiarizar o leitor com os termos utilizados.

Também são apresentados os métodos utilizados na avaliação e comparação de desempenho.

2.1 - CONCEITUAÇÃO BÁSICA

Nesta subseção e nas próximas apresenta-se um formalismo para todo o desenvolvimento que será utilizado nos próximos capítulos cujo objetivo é descrever o Sistema Classificador Slicer.

Primeiramente serão apresentados conceitos gerais, após isso serão apresentados os conceitos, definições, notações e equações necessários ao Capítulo 3 e, em seguida, conceitos e metodologias de validação e avaliação de desempenho para o Capítulo 5.

2.1.1 - Conceitos gerais

Nesta subseção são apresentados conceitos, definições, notações e equações que são utilizados em todo trabalho baseados nas referências [Mitchell 1997], [Duda et al.2001].

Problema de classificação: é a tarefa de, por meio de exemplos, gerar um mapeamento cujo domínio é o espaço dos atributos e a imagem, as classes, que possibilite a classificação de um novo exemplo não classificado dentro de um conjunto de valores discretos possíveis [Mitchell 1997].

Conjunto de treinamento: é um conjunto de exemplos já classificados necessários à estratégia de aprendizagem supervisionada adotada por um sistema inteligente [Witten & Frank 2005].

Medida de Similaridade: é uma informação que retorna quão similares são dois exemplos. Pode ser baseada em algum atributo específico, ou obtida através de alguma fórmula.

A medida mais utilizada em geometria computacional é a distância Euclidiana, que mede a distância entre dois exemplos (dissimilaridade) e quanto menor for esta distância, maior a similaridade entre elas ($1/dissimilaridade$) [Witten & Frank 2005].

A distância Euclidiana é definida, para os vetores $X, Y \in \mathbb{R}^P$ por:

$$\|X - Y\| = \sqrt{\sum_{i=1}^{P} (x^i - y^i)^2}$$

em que $X = (x^1, x^2, \ldots, x^P)$ e $Y = (y^1, y^2, \ldots, y^P)$.

Separabilidade Linear:

O conceito de separabilidade linear permeia muitas áreas de conhecimento e, com base na definição dada em [Nilsson 1965], pode ser descrito como:

Seja Λ um conjunto finito de N padrões distintos $\Lambda = \{\Lambda_1, \Lambda_2, ..., \Lambda_N\}$, no qual Λ_i $(1 \leq i \leq N)$ é descrito como um conjunto de P pares atributo-valor_de_atributo, sendo P o número total de atributos do exemplo. Considere que os padrões em Λ sejam classificados de tal maneira que cada padrão em Λ pertence a apenas uma das Q classes C_j $(1 \leq j \leq Q)$. Essa classificação divide o conjunto de padrões Λ nos subconjuntos ΛC_1, ΛC_2, … ,ΛC_Q, tal que cada padrão em Λ pertence à uma classe C_i, para i = 1, …, Q.

Se uma máquina linear puder classificar os padrões de Λ em suas respectivas classes, a classificação de Λ é uma classificação linear e os subconjuntos ΛC_1, ΛC_2, … , ΛC_Q são linearmente separáveis. Parafraseando a última sentença, pode se dizer que a classificação de Λ é linear e os subconjuntos ΛC_1, ΛC_2, … ,ΛC_Q, são linearmente separáveis se e somente se existem funções discriminantes lineares g_1, g_2, …, g_n tais que:

$g_i(\Lambda) > g_j(\Lambda)$ para todo $\Lambda \in \Lambda C_i$

j=1,…,Q, j≠i para todo i = 1, …, Q

Uma vez que as regiões de decisão de uma máquina linear são convexas, se os subconjuntos ΛC_1, ΛC_2, …, ΛC_Q são linearmente separáveis, então cada par de subconjuntos ΛC_i, ΛC_j, i, j=1,…,Q, i ≠ j, é também linearmente separável.

Um outro entendimento para o conceito de separabilidade linear vem de [Tajine et al. 1997] em que separabilidade linear é a propriedade associada ao conjunto de exemplos, espacialmente representados, de possibilitar a determinação de dois subconjuntos, um deles de classe única, por meio de um hiperplano.

O conceito visto desta forma permite sua utilização em vários métodos de aprendizado de máquina, pois permite a separação de um conjunto de exemplos em vários subconjuntos, não forçando a necessidade de que cada subconjunto represente apenas um classe e que cada classe seja representada por apenas um subconjunto. Esta visão de separabilidade linear é que orienta toda a metodologia deste trabalho.

O desvio padrão é um número que quantifica a dispersão dos elementos em valores de um conjunto, ou seja, corresponde à média das diferenças entre cada valor e a média central.

Quanto maior o desvio padrão, maior a dispersão e mais afastados do resultado médio estarão os resultados extremos.

Desvio Padrão: σ para um conjunto de valores $x^i, i = 1, \dots, n$ é definido por:

$$\sigma = \sqrt{\sum_{i=1}^{n}(x^i - xm)^2/n}\,,$$

em que *xm* é a média aritmética de todos os valores x^i.

Desvio padrão é útil para indicar a regularidade dos resultados quando temos em mãos os resultados das acurácias e/ou regras geradas.

Produto interno (produto escalar): é uma função que a cada par de vetores associa um número real. O produto interno é definido, para os vetores $X, Y \in \mathbb{R}^P$ por:

$$< X , Y >= \sum_{i=1}^{P} x^i \times y^i$$

em que $X = (x^1, x^2, \dots, x^P)$ e $Y = (y^1, y^2, \dots, y^P)$.

Hiperplano: é uma figura geométrica em $\mathbb{R}^n$ formada por pontos que satisfazem à seguinte equação:

$$< V, X > + h = 0$$

em que $X, V \in \mathbb{R}^n; h \in \mathbb{R}$.

Assim, é possível definir um hiperplano a partir de 2 parâmetros: vetor ortogonal V e limitante h.

Ruído: é um exemplo cuja classe destoa das classes associadas aos exemplos de sua vizinhança

Atributo faltante: é a indefinição de um atributo de um exemplo, ou devido a erro de entrada de dados ou devido a não haver valor possível.

A diferença entre ruído e atributo faltante é que no ruído há um valor aceito, dentro do intervalo admitido para os valores do atributo verificado, enquanto no atributo faltante não há valor algum.

2.2 - DEFINIÇÕES, CONCEITOS E NOTAÇÕES PARA O CAPÍTULO 3

Nesta subseção serão apresentados todos os conceitos relevantes ao entendimento do Capítulo 3, assim como as definições utilizadas e as notações aplicadas, tanto como as equações.

Com o objetivo de facilitar a leitura dos nomes variáveis, elementos simples de conjuntos ou componentes de vetores são indicados em letras gregas minúsculas, enquanto nome de constantes, funções, conjuntos e vetores, mesmo que sejam elementos de conjuntos, são representados em letras gregas maiúsculas.

A grande maioria das definições estão ilustradas através de figuras, nas quais os símbolos quadrado, círculo e losango representam as classes 1, 2 e 3 respectivamente; o símbolo X indica a posição do centroide; o símbolo de uma classe envolta por um losango indica o exemplo mais distante do centroide.

Definição 1 – Define-se Λ como sendo o conjunto de exemplos com N elementos. Cada elemento é formado pelos valoresdeatributos e pela classe a qual este elemento está associado. Assim:

$$\begin{cases} \boldsymbol{\Lambda} = \{\lambda^1, \lambda^2, \dots, \lambda^g, \dots, \lambda^N\}; \\ \lambda^g = (\boldsymbol{\Gamma^g}, \chi^g); \\ \boldsymbol{\Gamma^g} = \left(\gamma^{g,1}, \gamma^{g,2}, \dots, \gamma^{g,i}, \dots, \gamma^{g,P}\right); \end{cases} \quad (1)$$

em que: $\chi^g \in \{1,2,\dots,Q\}$ é a classe desse elemento; $\gamma^{g,i} \in \mathbb{R}$, $1 \leq g \leq N; 1 \leq i \leq P$.

Para efeitos de ilustração mostra-se na Tabela 2.1 um possível conjunto Λ tal que N=10, P=2 e Q=3. Este conjunto representa um problema de classificação a ser resolvido, composto de 10 exemplos de 3 classes distintas. A Figura 2.1 representa estes exemplos geometricamente, sendo que os exemplos de classe 1 são representados por quadrados; os de classe 2, por círculos; e os de classe 3, por losangos. O λ^g é igual à composição do vetor Γ^g e a classe χ^g , ou seja, a posição espacial dos exemplos são os Γ^g e a classe desses exemplos são representados pelos padrões mencionados (quadrado, círculo, losango).

Visando facilitar o entendimento de todas as definições posteriores, as figuras desta seção se utilizam do mesmo conjunto de exemplos.

Tabela 2.1 - Um possível conjunto de treinamento.

λ^g {g=1,...,10}	$\gamma^{g,1}$	$\gamma^{g,2}$	χ^g
λ^1	-0.4	0.2	1
λ^2	0.2	-0.4	1
λ^3	0.6	-0.2	1
λ^4	0.8	0.4	1
λ^5	-0.6	-0.4	2
λ^6	0.2	0.2	2
λ^7	0.2	0.6	2
λ^8	-0.4	0.4	3
λ^9	-0.2	0.8	3
λ^{10}	0.4	1	3

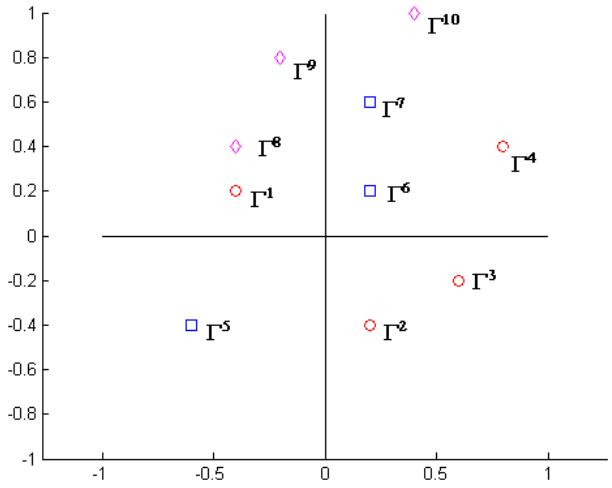


Figura 2.1 - Representação geométrica dos exemplos da Tabela 2.1.

Definição 2 – Define-se Π como sendo o vetor de valores que representa o centroide para o conjunto Λ de exemplos e é denotado como $\Pi = \left(\pi^1, \pi^2, \ldots, \pi^i, \ldots, \pi^P\right)$ $1 \leq i \leq P$; e $\pi^i \in \mathbb{R}$.

Cada π^i é dado por:

$$\pi^i = \sum_{g=1}^{N} \gamma^{g,i} / N \qquad (2)$$

Utilizando os exemplos da Tabela 2.1, mostra-se na Figura 2.2 a representação geométrica do centroide Π, representado como um símbolo X.

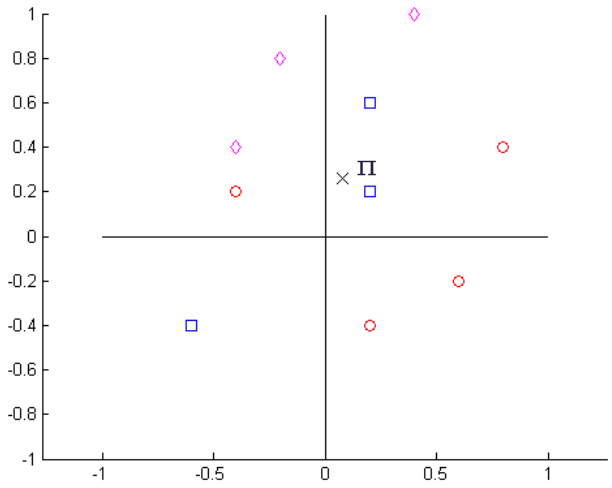


Figura 2.2 – Representação geométrica do centroide Π.

Definição 3 – Define-se Δ como sendo o conjunto das distâncias dos elementos Γ dos exemplos de Λ ao centroide Π, tal como segue:

$$\boldsymbol{\Delta} = \{\delta^g | \delta^g = \|\boldsymbol{\Gamma}^g - \boldsymbol{\Pi}\|, 1 \leq g \leq N, \delta^k \in \mathbb{R}\}. \qquad (3)$$

O δ^g é a distância entre os vetores Π e $\boldsymbol{\Gamma^g}$.

Definição 4 – Define-se μ como sendo o índice do exemplo que possui a maior distância ao centroide Π, como segue:

$$\mu = indice \max_{g=1,N}(\delta^g) \qquad (4)$$

Para verificação do posicionamento dos exemplos no espaço utiliza-se da função hiperplano, a qual quando obtém valor igual a zero indica o próprio hiperplano, valor maior que zero indica exemplos do lado do hiperplano apontado pelo vetor que o gera e valor negativo indica exemplos do lado contrário.

Definição 5 – Define-se a função hiperplano H tal como segue:

$$H(.) = < \Phi, \Gamma^g > - \rho, \qquad (5)$$

em que: Γ^g é o vetor de atributos a ser testado, ρ é o limitante do hiperplano; e $\Phi = (\phi^1, \phi^2, \ldots, \phi^i, \ldots, \phi^P)$ é o vetor que indica a direção ortogonal ao hiperplano, para ϕ^i tal que:

$$\phi^i = \gamma^{\mu,i} - \pi^i,$$

em que: $\gamma^{\mu,i}$ é o componente do vetor Γ^μ e π^i é o componente de Π ambos definidos anteriormente.

Mostra-se na Figura 2.3 o vetor ortogonal Φ e os vetores Γ^μ e Π (centroide). Tem-se na Figura 2.4 uma representação do hiperplano H (H(.)=0) e seus componentes.

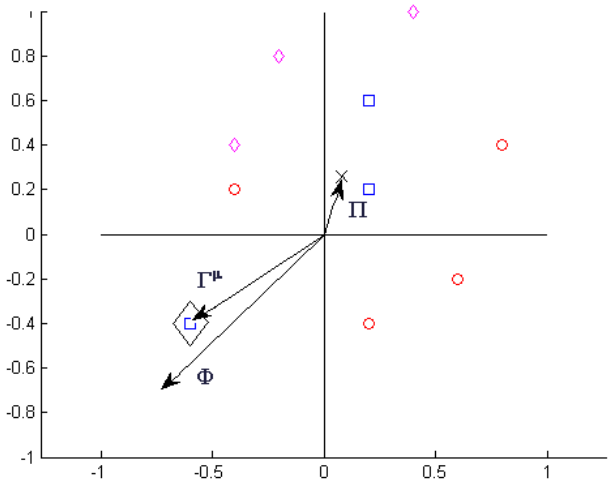


Figura 2.3 - Interpretação geométrica do vetor $\boldsymbol{\Phi}, \boldsymbol{\Phi} = (\boldsymbol{\Gamma^{\mu}} - \boldsymbol{\Pi})$.

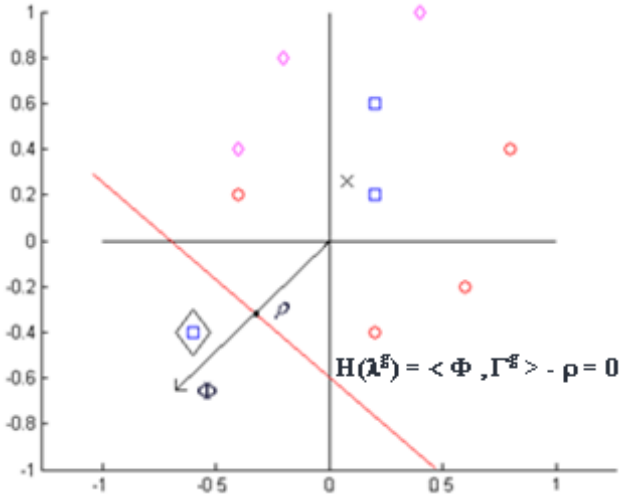


Figura 2.4 – Representação do hiperplano H e seus componentes Φ e ρ.

Definição 6 – Define-se $\Psi, \boldsymbol{\Psi} = \{\psi^1, \psi^2, \ldots, \psi^g, \ldots, \psi^N\}$, como sendo o conjunto de projeções dos exemplos de Λ, tal que:

$$\psi^g =< \Phi, \Gamma^g > \qquad (6)$$

em que: $\psi^g \in \mathbb{R}$, $1 \leq g \leq N$ e ϕ é o vetor ortogonal descrito na definição anterior.

Na Figura 2.5 tem-se a representação do conjunto de projeções Ψ.

Definição 7 – Seja ψ^g de acordo com a definição 6, definem-se:

$\theta = \max_{g=1,N}(\psi^g)$ e $\boldsymbol{\Psi}^*$ como o conjunto de projeções dos exemplos de Λ, tal que:

$$\boldsymbol{\Psi}^* = \{\, \psi^{*g} \,|\, \psi^{*g} =< \Phi, \Gamma^g > e\ \psi^{*g} < \theta, \psi^{*g} \in \mathbb{R} \,\} \qquad (7)$$

em que: $1 \leq g \leq J$, $J = |\Psi^*|$.

Definição 8 – Considerando a definição 7, define-se θ^* tal como:

$$\theta^* = \max_{g=1,J}(\psi^{*g}); \text{ em que: } J=|\Psi^*|. \quad (8)$$

Definição 9 – Define-se $\bar{\Psi}$, , como sendo o conjunto de projeções dos exemplos de Λ, tal que:

$$\bar{\Psi} = \{\, \bar{\psi}^g \,|\, \bar{\psi}^g =< \Phi, \Gamma^g > e\ \bar{\psi}^g = \theta^*, \bar{\psi}^g \in \mathbb{R} \,\} \quad (9)$$

em que: $\chi^g = \chi^\mu$ e $1 \leq g \leq L$, $L = |\bar{\Psi}|$.

Definição 10 – Define-se $\bar{\bar{\Psi}}$ como o conjunto de projeções dos exemplos de Λ, tal que:

$$\bar{\bar{\Psi}} = \left\{\, \bar{\bar{\psi}}^g \,|\, \bar{\bar{\psi}}^g =< \Phi, \Gamma^g > e\ \bar{\bar{\psi}}^g = \theta^*, \bar{\bar{\psi}}^g \in \mathbb{R} \,\right\} \quad (10)$$

em que: $\chi^g \neq \chi^\mu$ e $1 \leq g \leq M$, $M = \left|\bar{\bar{\Psi}}\right|$.

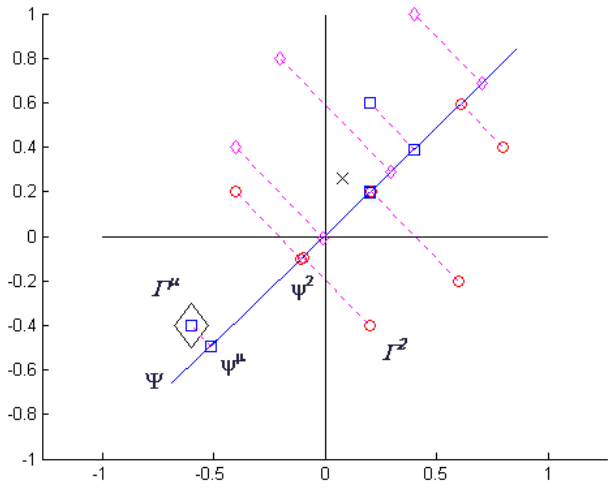


Figura 2.5 – Representação geométrica do conjunto de projeções Ψ.

Definição 11 – Considere o conjunto de projeções Ψ, e considere θ e κ^*, ambos pertencentes ao Ψ, e as seguintes propriedades:

- θ é a projeção imediatamente maior ao limitante ρ (limite superior)

$$\theta \in \{\psi^g \mid \psi^g > \rho\ e\ \nexists \psi^k, \psi^g > \psi^k > \rho, k = 1, \ldots, N\}.$$

- κ^* é a menor projeção do conjunto Ψ (projeção minima);

- κ é a projeção imediatamente menor ao limitante ρ (limite inferior), ou valor inferior se tal projeção não existir, ou seja:

$$\kappa = \begin{cases} \theta - \varepsilon, & se\ \theta = \kappa^* \\ \in \{\psi^g \mid \psi^g < \rho\ e\ \nexists \psi^k, \psi^g < \psi^k < \rho, k = 1, \ldots, N\}, & caso\ contrário \end{cases}$$

em que: $\varepsilon > 0$ é a metade do menor intervalo entre duas projeções consecutivas, caso existam, senão seu valor é igual a 0.1.

Para estas condições define-se ρ como a seguir:

$$\rho = (\kappa + \theta)/2 \qquad (11)$$

O motivo de ε ser a metade do menor intervalo é com o objetivo de garantir um número pequeno e não maior que o intervalo entre duas projeções.

A primeira condição é satisfeita quando o conjunto de projeções só contém uma projeção ou então θ é a menor projeção do conjunto.

Os valores de θ, ρ, κ^* e κ são calculados no algoritmo do método.

Observe que se utiliza a Figura 2.5 como base para a Figura 2.6, nota-se que é inserido o limitante ρ, limite superior θ e limite inferior κ, no lugar das projeções indicadas na Figura 2.5.

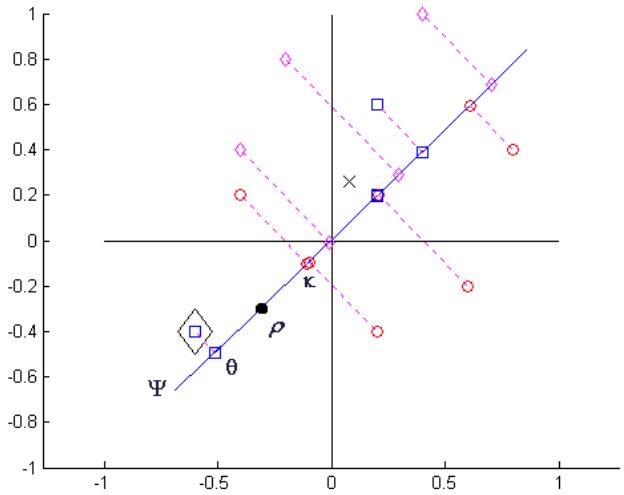


Figura 2.6 – Representação do limitante ρ, limite superior θ e limite inferior κ.

Definição 12 – Define-se Ω, $\Omega=\{\omega^1, \omega^2, ..., \omega^j, ..., \omega^R\}$, como sendo o conjunto de regras com R elementos, tal que:

$$\omega^j=\{T^j, \alpha^j, \beta^j\} \qquad (12)$$

em que: $T^j = (\tau^{j,1}, \tau^{j,2}, ..., \tau^{j,i}, ..., \tau^{j,P})$ é o vetor ortogonal, $\tau^{j,i} \in \mathbb{R}$;

$\alpha^j \in \mathbb{R}$ é o limitante do hiperplano; e

$\beta^j \in \mathbb{N}$ é a classe associada à regra ω^j; $1 \leq j \leq R$.

Embora Ω possa ser interpretado como um conjunto de hiperplanos associados às classes que separam, no contexto deste trabalho será notado como sendo um conjunto de regras, porém nada impede que venha a ser utilizado como entrada em outros métodos classificatórios.

Na Figura 2.7 representam-se uma regra e seus componentes. Assume-se que $T^j = \Phi$ e $\alpha^j = \rho$, $\beta^j = \chi^\mu$ dentro do contexto da criação de regra.

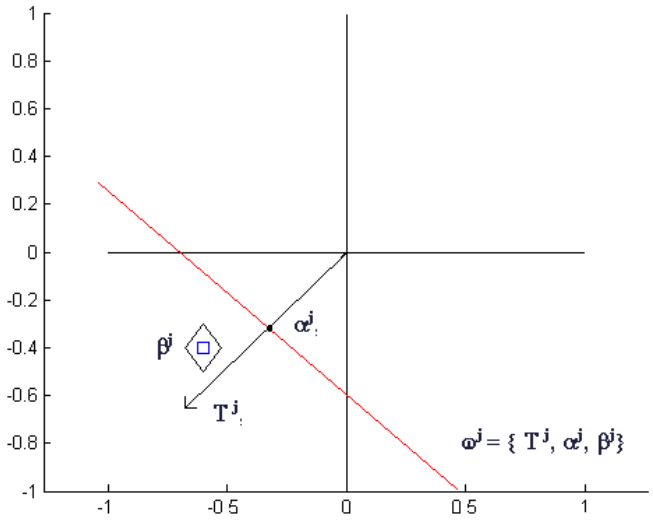


Figura 2.7 – Representação de uma regra ω^j e seus componentes.

Considere a Figura 2.5 e note-se um conjunto ordenado de projeções, representados sobre o segmento de reta inclinado passando pela origem. As projeções estão em ordem decrescente no sentido de $\psi^\mu(\psi^5)$ para ψ^2. A classe associada a cada uma das projeções está indicada pelo símbolo utilizado na representação, ou seja, quadrado indica classe 1, círculo indica classe 2 e losango indica classe 3. Nota-se ainda que o vetor Γ^μ é o ponto mais distante do centroide e que o símbolo quadrado representa sua classe.

Na Figura 2.8 este segmento de reta está reproduzido de forma que as projeções estão em ordem decrescente da direita para a esquerda.

Neste cenário, "gap" pode ser entendido como o subconjunto de projeções, de classes diferentes da representada pelo símbolo quadrado, delimitados em suas margens inferior e superior por projeções de classe representada pelo quadrado.

Durante a pesquisa pelo hiperplano correspondente ao vetor Φ, o exemplo mais distante determina a "classe de busca". Na pesquisa pelo hiperplano, busca-se pela projeção mais distante em uma sequência de projeções de classe igual à "classe de busca".

Se em meio a uma sequência de projeções delimitadas por projeções de classe igual a "classe de busca" surgir uma projeção de classe distinta esta projeção pode ser associada a um ruído.

Definição 13 – Define-se "gap" como sendo o conjunto de projeções, em sequência, de classe distinta da classe de busca, se for delimitado em ambos os extremos por projeções de classe igual à classe de busca.

Na Figura 2.8 é possível se identificar que a classe de busca é a classe 1 (quadrado), iniciando por ψ^5 encontra-se um gap de tamanho (comprimento) 3 entre ψ^6 *e* ψ^5, composto pelas projeções ψ^1, ψ^2 *e* ψ^8; e outro gap de tamanho 2 entre ψ^7 *e* ψ^6, composto pelas projeções ψ^3 *e* ψ^9. As projeções ψ^4 *e* ψ^{10} não estão dentro de nenhum gap, pois não são limitadas em seu extremo inferior por projeção da classe 1 (quadrado).

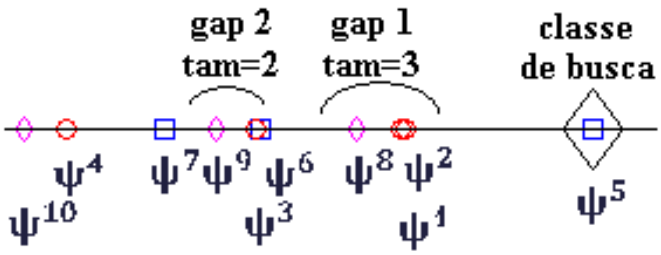


Figura 2.8 - Conjunto ordenado das projeções Ψ.

Definição 14 – O número de gaps aceitável na busca do limitante (do hiperplano separador) é denominado $Nrogap$. Se for igual a zero indica que não se admite gaps na versão. O tamanho (comprimento) máximo aceitável para cada gap é denominado *Tamgap*.

2.3 - DEFINIÇÕES, CONCEITOS E NOTAÇÕES PARA O CAPÍTULO 4

Nesta subseção são apresentados todos os conceitos relevantes ao entendimento do Capítulo 4, assim como as definições utilizadas e as notações aplicadas, tanto como as equações e algoritmos.

2.3.1 - DEFINIÇÕES, CONCEITOS E NOTAÇÕES

A arquitetura básica de um Sistemas Fuzzy de Classificação Baseado em Regras (SFCBR) é constituida de: uma base de conhecimento (BC) e um Motor (ou mecanismo) de inferência (MI), além das interfaces de entrada (IF) e saída (ID). A BC é composta pela base de regras (BR) e pela base de dados (BD). Na Figura 2.9, x, entrada, é um vetor de atributos e y, saída, é a classe associada à entrada x.

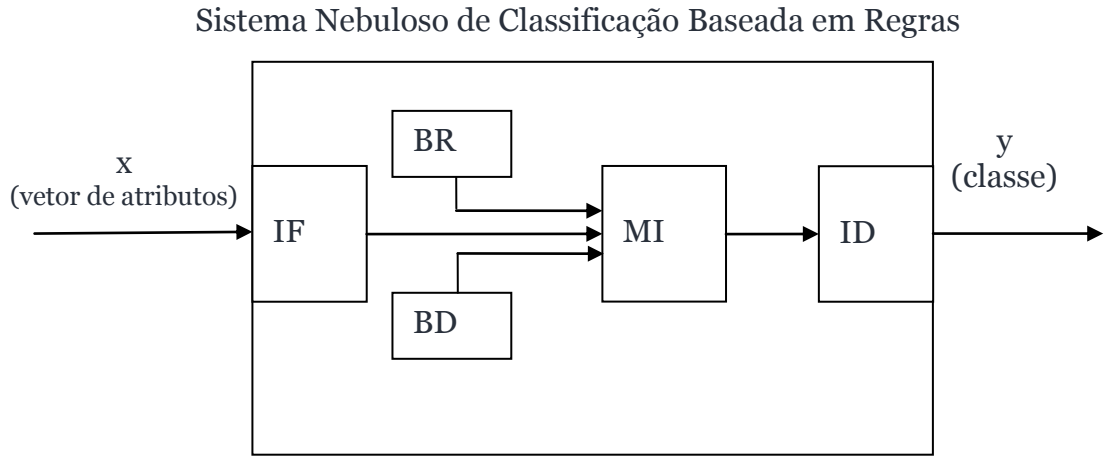


Figura 2.9 – Arquitetura de um SFCBR.

A IF faz o mapeamento de $x \in \mathbb{R}^P \longrightarrow x \in \mathbb{U}^P$, transformando os valores do vetor de atributos em pares de pertinências nos conjuntos de partições, ou seja, "fuzifica" as entradas. O conjunto $\mathbb{U}$ é o universo de discurso sobre o qual os conjuntos nebulosos estão definidos.

A ID faz o mapeamento de $y \in \mathbb{U} \longrightarrow y \in \mathbb{R}$, transformando a pertinência da partição da classe em um valor numérico, caso a classe seja "fuzificada" também, o que ocorre quando a classe é um número real.

A BR é o conjunto de regras de inferência que representa a modelagem de conhecimento nebuloso. Neste trabalho é usado o método Wang-Mendel como gerador da BR.

A BD é o conjunto de informações que definem as partições nas quais se baseiam as regras de inferência.

A MI se utiliza da BC para inferir a classe de um vetor de atributos. Calcula os

graus de relevância de todas as regras na BR, utilizando as partições de BD para este vetor de atributos. Uma das estratégias de inferência adota a regra vencedora para a classificação, ou seja, a regra que obtém maior grau de relevância fornece a classe ao vetor de atributos. Caso todas as regras obtenham grau de relevância igual a zero, normalmente se classifica como uma classe padrão ou se considera como sendo um erro.

Notação para partições:

São utilizadas duas notações para as partições, sendo a primeira utilizada em figuras que representam estas partições; e a segunda é utilizada em algoritmos, tabelas e explicações de procedimentos, pois a manipulação da primeira notação em algoritmos se torna difícil.

Primeira notação:

Dado uma certa constante W, calcula-se o número de conjuntos de partições, se W=1 conjuntos de partições=3, se W=3 conjunto de partições =7.

De outra forma $W = ((número\_de\_conj\_de\_partições - 1)/2)$.

Denotam-se as partições em função da constante W anteriormente mencionada. As partições são nomeadas em S(small), CE(center) e B(big) e as partições S e B são numeradas. As partições S são numeradas em ordem decrescente, indo do W ao 1, e as partições B são numeradas em ordem crescente, indo do 1 ao W. Assim:

Para um particionamento em 3, W=1, S1, CE, B1;

Para um particionamento em 7, W=3, S3, S2, S1, CE, B1, B2, B3.

Representam-se os conjuntos de partições triangulares da seguinte forma: partição (inf, int, sup), em que inf indica o limite inferior da partição, int indica o valor intermediário e sup indica o limite superior.

Segunda notação:

Seja DESCPART, DESCPART={$part^1$, $part^2$, ..., $part^k$, ..., $part^Z$} o conjunto de descrições de partições das regras Slicer, em que: Z é o número de variáveis a ser particionadas.

Como as funções de pertinência neste trabalho são somente do tipo triangular, então notam-se sempre $part^{k,w}$=(inf ^{k,w}, int ^{k,w}, sup ^{k,w}), que indicam os valores extremo inferior, intermediário e extremo superior de cada partição.

No método Slicer-Nebuloso aqui descrito, o número de partições é 3 ou 7, e as funções de pertinência estão na forma triangular.

Notação das regras de inferência:

Seja RF, RF={rf^1,rf^2,...,rf^h,..rf^S}, o conjunto de S regras de inferência nebulosa, em que: rf^h=(ANT^h,con^h);

ANT^h=($ant^{h,1}$,$ant^{h,2}$,...,$ant^{h,v}$,..$ant^{h,Z}$);

$con^h \in \{0,1,...Q\}$;

em que: ANT^h é o vetor de antecedentes da h-ésima regra nebulosa, con^h é a classe da h-ésima regra nebulosa (consequente), $ant^{h,v} \in \mathbb{N}^*$ é a partição da h-ésima regra Slicer a qual este antecedente está associado e Z é a cardinalidade do conjunto Ω de regras do Slicer.

No contexto do capítulo 4 chama-se as regras do Slicer de regras crisp, o que significa regras que geram valores absolutos como resposta; ao contrário de regras do Slicer-Nebuloso que fornecem graus de pertinência de um valor pertencer a um determinado conjunto.

Deve-se lembrar também que as chamadas regras Slicer são na verdade tuplas formadas por: hiperplano definido por vetor ortogonal e limitante; e a classe a qual este hiperplano está associado.

2.3.2 - Método Wang-Mendel

O método Wang-Mendel [Wang & Mendel 1992] é um método de geração de regras nebulosas reconhecido como tendo boa acurácia em suas classificações, além de produzir um conjunto de regras que cobre todo o conjunto de exemplos e livre de inconsistências ou redundâncias.

O Algoritmo usado nesta dissertação se encontra no Capitulo 4, Quadro 12. A seguir uma descrição breve do método:

Passo 1 - para cada elemento de conjunto de exemplos criar uma regra nebulosa. Na criação dos antecedentes, atribui a cada atributo o conjunto nebuloso em que possuir a maior pertinência. Em caso de empate, escolhe o conjunto mais à esquerda. O consequente é a classe associada ao exemplo;
Passo 2 – criar para todas as regras nebulosas um grau de relevância, que é a t-norma das pertinências de seus antecedentes;
Passo 3 - verificar todas as regras que possuem os mesmos antecendentes e manter apenas a que possuir o maior grau de relevância. Em caso de empate, manter a regra que vem primeiro;
Passo 4 – finalizar, dando saída em um conjunto de regras nebulosas RF.

Note-se que o número máximo de regras nebulosas NWM, já eliminadas as redundantes e inconsistentes, criadas pelo método Wang-Mendel está ligado diretamente ao número de exemplos no conjunto de treino e ao máximo de combinações possíveis entre as partições. Este número é calculado como:

$NWM = \text{númerodepartições}^{\text{número de variáveis}}$.

Considerando que todas as variáveis possuam o mesmo número de conjunto de partições e que a classe seja não "fuzificada", pode-se concluir que uma otimização do Slicer, leia-se diminuição de número de regras, é sempre bem vinda para diminuir o número de regras final.

Por exemplo:

Considere uma base com 400 exemplos, com 4 atributos (variáveis), o número de regras geradas pelo WM está limitado por 2 máximos:

Máximo 1 - criação de 400 regras pelo método WM, sendo que esse número provavelmente é diminuído devido às redundâncias e inconsistências, ou seja, das regras com mesmos antecedentes só permanece as que possuírem maior grau de relevância pelo uso da t-norma;

Máximo 2 - número máximo de combinações possíveis, NWM.

Assim duas situações se apresentam, dentro do contexto deste trabalho, que admite apenas 3 ou 7 partições:

Caso 1: 3 partições: existe a possibilidade de 3^4= 81 regras diferentes, sem inconsistências, ou seja, de 400 do máximo 1 ao final apenas 81 regras diferentes são possíveis..

Caso 2: 7 partições: 7^4=2401 regras são possíveis, mas o número máximo de regras é limitado pelo número de exemplos.

2.4 – METODOLOGIA DE AVALIAÇÃO DE SISTEMAS CLASSIFICADORES

Tendo em vista que um método de AM pode obter resultados variáveis dependendo do domínio em que é aplicado, torna-se muito importante verificar sua acurácia em cada domínio em que se faça atuar. Esta medida servirá de parâmetro de comparação entre métodos indicando inclusive qual é mais eficaz para aplicação em um domínio específico.

Geralmente os métodos de avaliação costumam dispor do conjunto de treinamento, o qual é dividido em dois subconjuntos. Um deles serve para treinamento (aprendizado) e o outro, para fazer testes de acurácia. Esta acurácia equivale à razão entre o número de exemplos de testes corretamente classificados e o número total de exemplos de teste.

Pode-se classificar os métodos de avaliação da seguinte forma [Mitchell 1997]:

- "Holdout": Os exemplos disponíveis são divididos em dois subconjuntos, um para treinamento (aprendizado) e outro para testes (avaliação); neste caso o conjunto de treinamento e o conjunto de teste são disjuntos. Para divisões diferentes obtêm-se resultados de acurácia diferentes, tornando o valor da acurácia fortemente influenciada pela partição feita. Normalmente se usa a divisão 70% para treinamento e 30% para avaliação, ou 80-20.

- Leave-one-out: O conjunto de treinamento utiliza (N – 1) exemplos, com o teste sendo realizado no exemplo de fora; o processo é repetido N vezes para cada exemplo de teste que fica fora do conjunto de treinamento. Apresenta muita variação entre os experimentos e alto custo computacional.

- f-validação cruzada: Divide o total de exemplos em f subconjuntos, com quantidades aproximadas de exemplos; utiliza (f – 1) subconjuntos para treinamento e o subconjunto restante para testes; o processo é repetido f vezes para cada subconjunto de teste que fica fora do conjunto de treinamento.

- f-validação cruzada estratificada: Similar ao f-validação cruzada, porém neste método busca-se manter a proporção de exemplos de cada classe no conjunto de teste.

Na prática encontram-se alguns problemas no uso do f-validação cruzada estratificada, pois o conjunto pode possuir exemplos de cada classe em números não exatamente divisíveis por f, gerando-se conjuntos de teste com tamanhos irregulares. Neste trabalho a diferença da divisão é distribuída nas primeiras divisões, obtendo-se divisões iniciais maiores que as finais.

Por exemplo: o conjunto de exemplos de 9 exemplos, com 1 atributo e 2 classes abaixo seria assim dividido:

{1 1, 2 1, 3 1, 4 1, 5 1 , 6 2, 7 2, 8 2, 9 2}

3-validação cruzada

{1 1,4 1, 7 2}, { 2 1, 5 1, 8 2}, {3 1, 6 2, 9 2}

3-validação cruzada estratificada

{1 1, 4 1, 6 2, 9 2} { 2 1, 5 1, 7 2} {3 1, 8 2}

Pode-se também fazer repetições da avaliação, como em [Tran et. al. 2005], em que usam-se 10 repetições de 10-validação cruzada estratificada, buscando-se obter resultados o mais próximo da realidade.

Em algumas situações o conjunto de dados costuma ser dividido em três partes, uma delas sendo o conjunto de validação, além dos conjuntos de treinamento e teste. Este conjunto de validação auxilia na avaliação da qualidade final das regras obtidas.

Neste trabalho utiliza-se em todas as avaliações 10 x 10 validação cruzada estratificada.

2.5 – METODOLOGIA PARA COMPARAÇÃO ENTRE SISTEMAS CLASSIFICADORES

A principal vantagem de se conhecer o desempenho de um método de AM em vários domínios de conhecimento é que passa-se a conhecer em que tipo de domínios este se comporta melhor, permitindo a escolha mais acertada quando se busca um método para um domínio que se sabe ser assemelhado a um dos testados.

Para que se possa fazer comparações entre métodos de AM, e mesmo fazer a avaliação de um método isoladamente, torna-se necessária a execução e medida de

desempenho sobre os mesmos conjuntos de treinamento.

Com esse objetivo costuma-se usar os domínios de conhecimento, garantindo igualdade de condições entre os métodos avaliados.

Domínios de conhecimento representam áreas de conhecimento, artificiais ou não, e podem ser representados por conjunto de exemplos, produzidos artificialmente ou vindos de alguma fonte de dados, como a [Asuncion & Newman 2007], ou mesmo obtidos automaticamente por algum sistema computacional.

Entre as características que podem influir na escolha de um método de AM estão:

- presença de exemplos com valores de atributos não especificados;

- presença de exemplos inconsistentes, ou seja, dois ou mais exemplos com todos seus atributos iguais, mas com classe diferente;

- presença de exemplos redundantes, ou seja, dois ou mais exemplos exatamente iguais em todos seus valores;

- presença de exemplos com atributos e classe diferentes de numéricos;

- número de atributos e classes admitidos;

- valores mínimos e máximos admitidos em seus atributos e classe;

- desempenho em domínios conhecidos.

Geralmente costuma-se usar vários domínios de conhecimento no momento de se fazer comparação entre o desempenho de métodos de AM, possibilitando uma base mais sólida para as comparações.

Através de testes de desempenho em domínios de conhecimento, é possível detectar com mais facilidade em que tipos de domínios um método específico tem melhor resultado e consequentemente é mais indicado.

CAPÍTULO 3

DESCRIÇÃO DO SLICER BÁSICO E VERSÕES ESTENDIDAS

O Slicer é um sistema classificador que se utiliza de um método indutivo supervisionado de aprendizado de máquina. O Slicer tem seu mecanismo de aprendizado baseado no conceito de *separabilidade linear*, bem como no conceito geométrico de *centroide*. Devido aos conceitos empregados, o método utilizado pelo Slicer pode ser caracterizado como um método fundamentado em conceitos da geometria computacional.

Nesta dissertação os termos sistema classificador, algoritmo e método são utilizados em referência ao Slicer.

Ressalta-se que o termo método refere-se às heurísticas utilizadas pelo Slicer, ou seja, criação de hiperplanos separadores usando centroide e exemplo mais distante; já o termo algoritmo é utilizado indicando o passo a passo para a execução do método; enquanto o termo sistema é uma referência ao sistema computacional composto pela implementação de todos os algoritmos necessários para a execução dos métodos descritos, incluindo entrada de dados, validação e tabulação dos resultados obtidos.

O nome “slicer”, significando fatiador, separador, já informa o princípio básico do sistema classificador Slicer, que por meio de hiperplanos separa conjuntos de exemplos.

De uma forma simplista, o método utilizado pelo Slicer pode ser entendido como um procedimento que particiona o espaço no qual estão representados os exemplos de treinamento, por meio da construção de hiperplanos usando, para tal, o centroide do conjunto de exemplos. Este método cria tantos hiperplanos quantos forem necessários para separar subconjuntos de exemplos linearmente separáveis. O resultado da execução do aprendizado por este método é, pois, um conjunto de hiperplanos associados às classes que separam, no espaço dimensional definido pelo conjunto de treinamento. No

processo de classificação cada hiperplano é associado a uma regra do tipo se-então-senão de aplicação simples e direta.

As pré-condições para o algoritmo básico ser operacional são:

- 1 - exemplos de treinamento devem ser numéricos;

- 2 - classes devem ser representadas por valores inteiros;

- 3 - o conjunto de treinamento não deve ter inconsistências.

Na verdade, o algoritmo Slicer se apresenta em diversos níveis de sofisticação, a partir de uma versão básica, cada nível correspondendo a uma versão:

1 – Slicer-G: admite ruídos na entrada;

2 – Slicer-F: admite atributos faltantes na entrada;

3 – Slicer-R: busca melhorar o vetor ortogonal ao hiperplano;

4 – Slicer-O: reordena as regras geradas (hiperplano e classe associada), visando reduzir o número de hiperplanos.

Uma vez desenvolvida a versão Slicer-F, esta substitui a versão Slicer Básico em todas as execuções da fase de aprendizado, pois os resultados são idênticos para bases sem atributos faltantes, e a versão Slicer Básico não tem a capacidade de tratar bases com atributos faltantes.

O sistema Slicer pode ser configurado para operar segundo qualquer combinação de versões, ou seja, é possível configurar o sistema para durante a execução da fase aprendizado aplicar as alterações correspondentes às versões Slicer-R e/ou Slicer-G, e após este aprendizado, aplicar ou não a versão Slicer-O. Deste modo pode-se acrescentar ao aprendizado benefícios (vantagens) associados a estas versões. A fase de classificação é sempre a mesma, independente de quais versões foram utilizadas durante o aprendizado.

Algumas das principais características do Slicer:

1 – todas as versões funcionam em conjunto, admitindo-se ruídos em conjuntos com atributos faltantes, com rotação de seus vetores e otimização de suas regras;

2 - admite domínios com um número finito de dimensões e classes discretas (multiclasse);

3 - gera regras de classificação através da definição de hiperplanos separadores de subconjuntos de uma só classe, linearmente separáveis do conjunto de exemplos usado no treino.

O fato do método central do sistema classificador Slicer usar de conceitos tais como: centroide, distância Euclidiana, vetor, produto interno e hiperplano; permite

considerá-lo um método baseado em geometria computacional. Além disso, por usar um processo de geração de hiperplanos para delimitação de conjuntos de classe única, também é possível dizer que é baseado em separabilidade linear.

3.1 - Descrição do Slicer Básico

O sistema classificador Slicer busca por meio do cálculo do centroide dos exemplos, do exemplo mais distante do centroide, e da diferença entre estes vetores, criar hiperplanos separadores de subconjuntos de classe única, gerando, uma a uma, as regras que compõem sua base de conhecimento.

Nomeou-se a primeira versão do Slicer de Slicer Básico, pois esta versão tem as características do método utilizado pelo Slicer e gera um conjunto de regras que conseguem resolver o problema de classificação.

A operação do sistema classificador Slicer é dividida em 2 fases: primeira, a fase de aprendizado, na qual se produzem as regras de classificação; segunda, a fase de classificação, na qual se usam as regras geradas para classificar novos exemplos.

Nos algoritmos apresentados, as variáveis, constantes e equações usadas foram definidas no Capítulo 2.

3.1.1 – Fase Aprendizado

O Slicer Básico gera regras que são utilizadas no processo de classificação por meio da determinação de hiperplanos separadores de subconjuntos de classe única, linearmente separáveis. Estes hiperplanos são determinados por meio dos seguintes parâmetros:

- vetor ortogonal (Φ): obtido pela diferença entre o vetor centroide (Π) e o exemplo mais distante do centroide (Γ^{μ});

- limitante (ρ): calculado a partir do conjunto de projeções (Ψ) de todos os exemplos.

O Quadro 1 corresponde ao algoritmo de aprendizado do Slicer Básico. As variáveis e definições utilizadas estão descritas no Capítulo 2.

Slicer Básico - Algoritmo de Aprendizado

Entrada: conjunto de exemplos Λ.

Saída: conjunto de regras Ω.

Passo 1 – Faça j = 0 .

Passo 2 - Calcula-se o centroide Π do conjunto de exemplos Λ.

Passo 3 – Calcula-se o conjunto de distâncias Euclidianas Δ.

Passo 4 – Localiza-se o índice μ do exemplo mais distante do centroide Π.

Passo 5 – Calcula-se vetor ortogonal Φ.

Passo 6 – Calculam-se o conjunto de projeções Ψ e ε.

Passo 7 – Busca do limitante:

Passo 7.1- Faça: classe de busca *E* igual a classe de λ^{μ}, $E = \chi^{\mu}$;

limite superior θ igual a Max(Ψ), $\theta = max(\Psi)$;

limite inferior κ igual a θ menos ε, $\kappa = \theta - \varepsilon$.

Passo 7.2 - Busca os conjuntos de projeções $\overline{\Psi}$ e $\overline{\overline{\Psi}}$.

Passo 7.3 - Se $|\overline{\Psi}| = 0$ e $\left|\overline{\overline{\Psi}}\right| = 0$, ou seja, se não há mais projeções a considerar, vá para o Passo 7.6.

Passo 7.4 – Se $\left|\overline{\overline{\Psi}}\right| = 0$, ou seja, todas as projeções são da mesma classe que a classe da busca, então:

Faça: limite superior θ igual ao primeiro elemento de $\overline{\Psi}$, $\theta = \bar{\psi}^1$; e

limite inferior κ igual a θ menos ε, $\kappa = \theta - \varepsilon$. Vá para o Passo 7.2.

Passo 7.5 - Faça limite inferior $\kappa = \bar{\bar{\psi}}^1$ e vá para o Passo 7.6.

Passo 7.6 - Faça limitante $\rho = (\theta + \kappa)/2$.

Passo 8 – Incrementa-se j e cria-se a regra ω^j, baseado no vetor ortogonal Φ, limitante ρ e classe χ^{μ}.

Passo 9 - Exclui-se do conjunto de exemplos Λ todos os exemplos que possuem projeção maior que o limitante, ou seja, exclui-se todos os exemplos que pertençam à classe χ^{μ}: $\Lambda \leftarrow \Lambda - \{\lambda^{g} | H(\lambda^{g}) \geq 0; 1 \leq g \leq N\}$.

Passo 10 - Caso o conjunto de exemplos Λ não esteja vazio, volta-se ao passo 2; senão finaliza-se o algoritmo.

Quadro 1 – Algoritmo da fase de aprendizado do Slicer Básico.

A seguir apresenta-se uma execução ilustrativa passo-a-passo do algoritmo, acompanhado de ilustrações e resultados. Todos os cálculos da execução estão indicados passo-a-passo na Tabela 3.2.

Execução demonstrativa do Slicer Básico:

A Figura 3.1 mostra a legenda utilizada nas figuras ilustrativas da execução do sistema classificador Slicer, em que: círculo, quadrado e losango são símbolos para identificação de classe, o símbolo X identifica o centroide; e um losango "preenchido" representa o exemplo mais distante do centroide.

Legenda	
	exemplo da classe 1
	exemplo da classe 2
	exemplo da classe 3
	centroide Π
	exemplo λ^{μ}

Figura 3.1 – Legendas utilizadas neste capítulo.

Nas ilustrações demonstrativas usa-se um conjunto de exemplos Λ com apenas dois atributos, ou seja, duas dimensões. Denotam-se os atributos $\gamma^{g,1}$como x^g e $\gamma^{g,2}$ como y^g, tal como definido no Capítulo 2, Definição 1.

A Tabela 3.1 apresenta o conjunto de treinamento a ser utilizado na execução demonstrativa do Slicer Básico. A Figura 3.2 corresponde à representação geométrica deste conjunto de exemplos.

Tabela 3.1 - Conjunto de treinamento Λ.

λ^g {g=1,...,4}	x	y	χ^g
λ^1	0.23	0.34	1
λ^2	0.51	0.30	1
λ^3	0.62	-0.21	2
λ^4	0.29	0.80	2

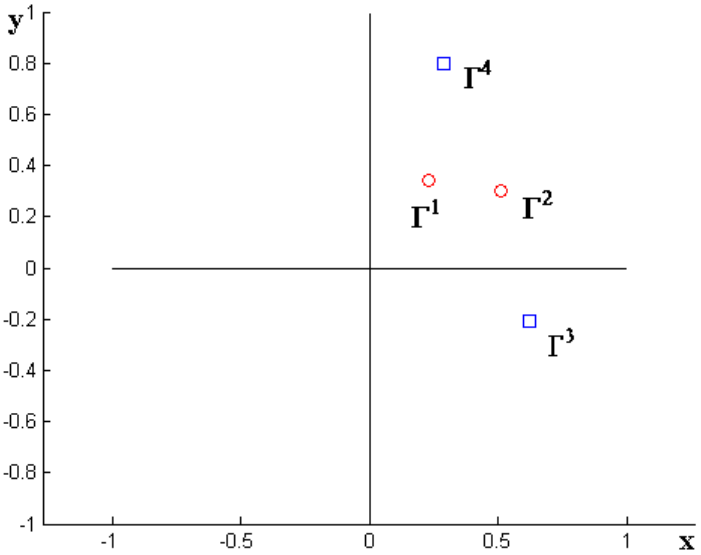


Figura 3.2 - Representação geométrica do conjunto de exemplos Λ.

As linhas da Tabela 3.2 correspondem ao rastreamento das variáveis durante a execução demonstrativa do algoritmo na primeira iteração do Slicer Básico (passo 2 ao passo 10). Note-se que este processo deve ocorrer mais duas vezes, além da mostrada, para que ao final tenha-se um total de 3 regras definidas.

Tabela 3.2 – Resultados de cálculos para cada um dos passos da primeira iteração do algoritmo do Slicer Básico.

Passos	**Situação das variáveis**
Passo 1	j = 0; Λ = {((0.23,0.34),1),((0.51,0.30),1),((0.62,–0.21),2),((0.29,0.80),2)}
Passo 2	Π = (0.4125, 0.3075)
Passo 3	Δ = { 0.1854, 0.0978, 0.5576, 0.5075}
Passo 4	μ = 3
Passo 5	Φ = (0.2075, -0.5175)
Passo 6	Ψ = { -0.1282, -0.0494, 0.2373, -0.3538} ε = 0.0394
Passo 7	Passo 7.1 : E=2; θ = 0.2373; κ = 0.1979. Passo 7.2 : $\bar{\Psi} = \{\}$; $\bar{\bar{\Psi}} = \{-0.0494\}$. Passo 7.3: ($\lvert\bar{\Psi}\rvert = 0$ e $\lvert\bar{\bar{\Psi}}\rvert > 0$) então Passo 7.4. Passo 7.4: ($\lvert\bar{\bar{\Psi}}\rvert > 0$) então Passo 7.5. Passo 7.5: Limite inferior $\kappa = \overline{\overline{\psi^1}} = -0{,}0494$. Passo 7.6: Limitante $\rho = (\theta + \kappa)/2 = (0.2373 - 0.0494)/2 = 0{,}0940$.
Passo 8	j=j+1; Ω = { Φ, ρ, χ^{μ}} = {(0.2075, -0.5175), 0.0940, 2.0000}.
Passo 9	Λ={((0.23,0.34),1),((0.51,0.30),1),((0.29,0.80),2)}.
Passo 10	Λ não é vazio então Passo 2.

Acompanhamento passo-a-passo da execução para a primeira iteração:

Passo 1 - Faz-se j = 0.

Passo 2 - Calcula-se o centroide Π do conjunto de exemplos Λ (Definição 2), mostrado na Figura 3.3. De acordo com o Passo 2 da Tabela 3.2, Π = (0.4125, 0.3075).

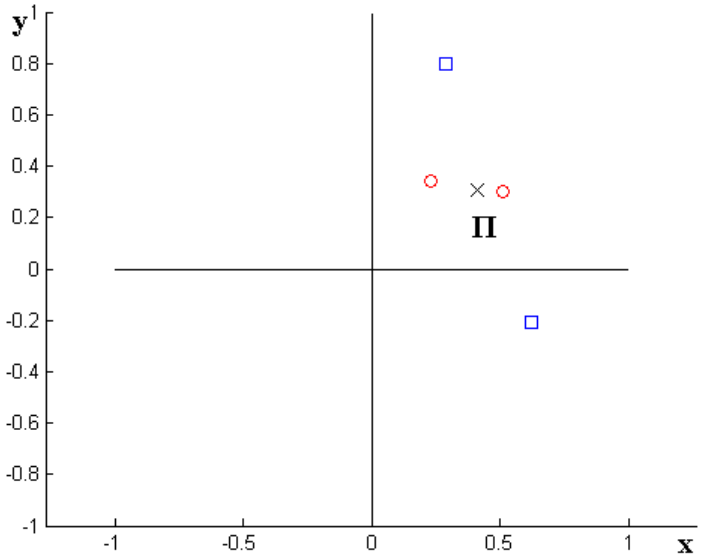


Figura 3.3 - Representação do centroide Π.

Passo 3 - Calcula-se o conjunto Λ das distâncias de todos os exemplos para este centroide Π (Definição 3). De acordo com o Passo 3 da Tabela 3.2, Δ = {0.1854, 0.0978, 0.5576, 0.5075}.

Passo 4 – Localiza-se o índice μ do exemplo mais distante (γ^{μ}) do centroide Π (Definição 4). Caso haja mais de um exemplo mais distante, assume-se o exemplo de menor índice. De acordo com o Passo 4 da Tabela 3.2, μ = 3.

A Figura 3.4, de acordo com os Passos 3 e 4, representa o centroide Π e o exemplo de índice μ do exemplo com maior distância ao centroide Π. A circunferência de centro em Π (centroide) e passando pelo exemplo Γ^{μ} (exemplo mais distante) envolve os demais exemplos, deixando claro que este exemplo efetivamente é o mais distante.

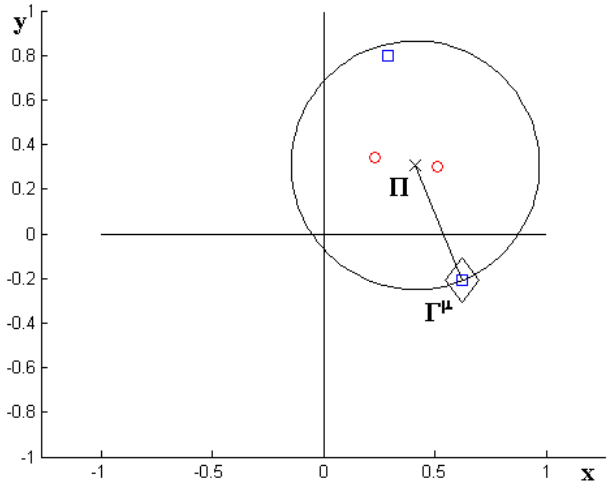


Figura 3.4 - Representação do centroide Π e do exemplo Γ^{μ}, circunferência centrada em Π envolvendo exemplos diferentes de Γ^{μ}.

Passo 5 - Calcula-se o vetor Φ, pela diferença dos vetores Γ^{μ} e o centroide Π, de acordo com a Definição 5, conforme mostrado na Figura 3.5. De acordo com o Passo 5 da Tabela 3.2, Φ = (0.2075, -0.5175).

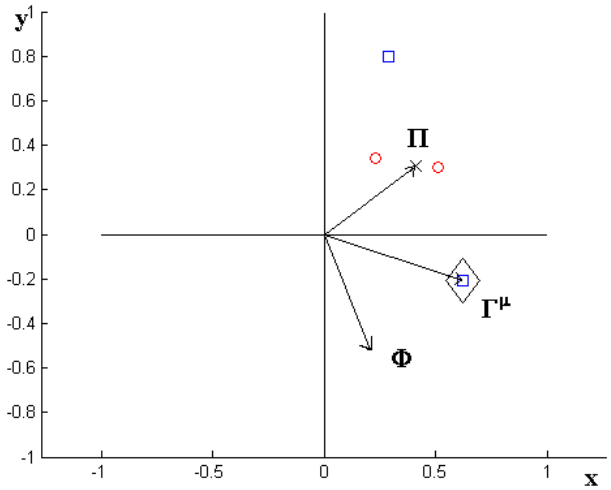


Figura 3.5 – Vetor Φ ($\boldsymbol{\Phi = \Gamma^{\mu} - \Pi}$).

Passo 6 – Gera-se o conjunto Ψ de projeções dos exemplos de Λ. Estas projeções são calculadas pelo produto interno entre Φ e o vetor de atributos Γ do exemplo, de acordo com a Definição 6 (Figura 3.6). Observe que as projeções estão em uma reta paralela ao vetor Φ e seus valores crescem no sentido deste vetor. No caso Ψ={ -0.1282, -0.0494, 0.2373, -0.3538} e ε = 0.0394.

Calcula-se a variável $\varepsilon = intproj/2$, em que *intproj* corresponde ao valor do menor intervalo entre duas projeções consecutivas.

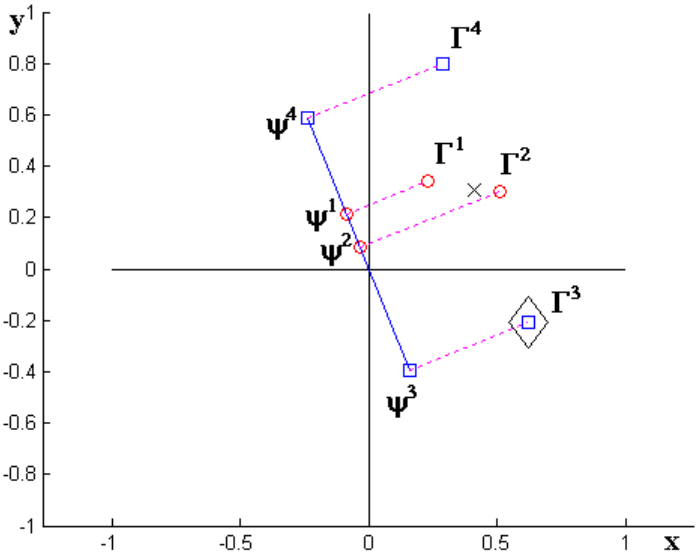


Figura 3.6 - Conjunto ordenado das projeções Ψ.

Passo 7 – Busca do limitante do hiperplano sendo criado:

Passo 7.1 – Atribui-se: a *E* a classe que está sendo separada, ou seja, $E = \chi^{\mu} = 2$; ao limitante θ o valor da maior projeção em Ψ e ao limitante κ o valor de θ menos ε, ou seja, $\theta = max(\boldsymbol{\Psi})$ e $\kappa = \theta - \varepsilon$. No caso *E*=2, $\theta = 0.2373$ e $\kappa = 0.1979$.

Passo 7.2 – Criam-se os conjuntos de projeções $\bar{\boldsymbol{\Psi}}$ e $\bar{\bar{\boldsymbol{\Psi}}}$. Uma sequência de procedimentos é necessária para determinação destes conjuntos.

- Inicialmente cria-se $\boldsymbol{\Psi}^*$ com os elementos de $\boldsymbol{\Psi}$ menores que θ, ou seja, $\Psi^* = \Psi - \{\theta\}$;
- Em seguida localiza-se θ^* como sendo a maior projeção em Ψ^*, ou seja, $\theta^* = max(\boldsymbol{\Psi}^*)$; e
- Finalmente cria-se $\bar{\Psi}$ com as projeções iguais a θ^* e de classe igual a χ^{μ}; e $\bar{\bar{\Psi}}$ com as projeções iguais a θ^* e de classe diferente de χ^{μ}. No caso, $\Psi^* = \{-0.1282, -0.0494, -0.3538\}$, $\theta^* = -0.0494$, $\bar{\boldsymbol{\Psi}} = \{\ \}$; $\bar{\bar{\boldsymbol{\Psi}}} = \{-0.0494\}$.

Passo 7.3 – Se ambos os conjuntos estivessem vazios então passaria-se para o passo 7.6, porém como o conjunto $\bar{\bar{\boldsymbol{\Psi}}}$ possui elementos, passa-se para o passo 7.4.

Passo 7.4 – Se o conjunto $\bar{\boldsymbol{\Psi}}$ tivesse elementos, então significaria que todas as projeções são da mesma classe da classe de busca, e se continuaria a busca, atualizando θ e κ e

passando para o passo 7.2, porém como $\overline{\boldsymbol{\Psi}}$ não possui elementos, passa-se para o passo 7.5.

Passo 7.5 – Quando este passo é executado significa que encontrou-se uma projeção de classe diferente da classe de busca *E*. Então atualiza-se o limitante inferior *k* e passa-se para o Passo 7.6. No caso limite inferior $\kappa = \overline{\overline{\overline{\psi^1}}} = -0{,}0494$;

Passo 7.6 Calcula-se o limitante do hiperplano, ou seja, $\rho = (\theta + \kappa)/2 = (0.2373 - 0.0494)/2 = 0{,}0940$.

Os valores assumidos pelas variáveis κ, θ e ρ são representados na Figura 3.7.

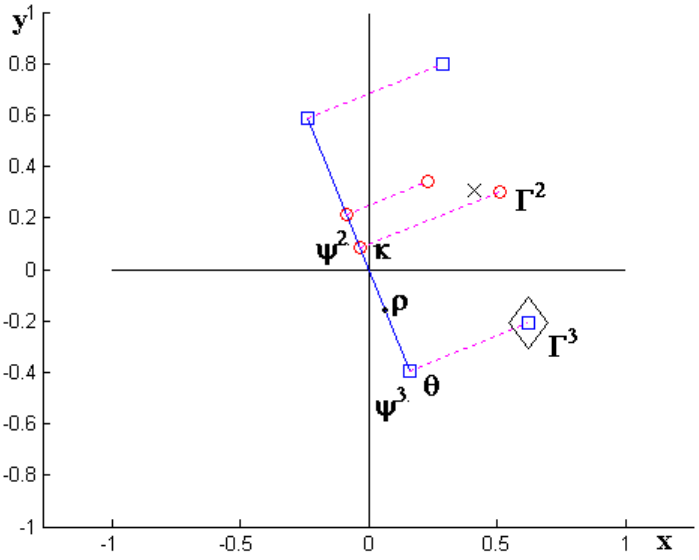


Figura 3.7 - Representação de κ, θ e ρ.

Sempre ao final do Passo 7 é determinado o hiperplano H (Definição 5) a partir do vetor $\boldsymbol{\Phi}$ (Passo 5) e do limitante ρ (Passo 7). Este hiperplano é o separador do subconjunto de classe única igual à classe de busca E. Na Figura 3.8 pode-se ver a representação deste hiperplano e do conjunto de exemplos separados.

Passo 8 – Incrementa-se j e cria-se a regra ω^j utilizando-se o vetor $\boldsymbol{\Phi}$, o limitante ρ e a classe χ^{μ}, conforme Definição 12.

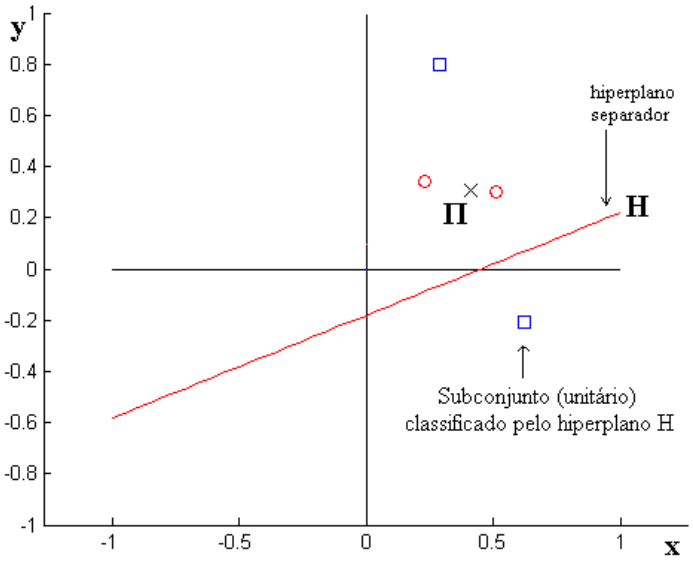


Figura 3.8 - Representação do hiperplano separador H e do subconjunto separado.

Passo 9 - Retira-se do conjunto Λ os exemplos que são separados por H como ilustrado na Figura 3.8.

Passo 10 – Como o conjunto Λ ainda não é vazio, reinicia-se o processo retornando ao Passo 2.

Este processo é repetido mais duas vezes gerando mais duas regras, ou seja, o número de regras é igual ao número de execuções dos passos 2 ao 10.

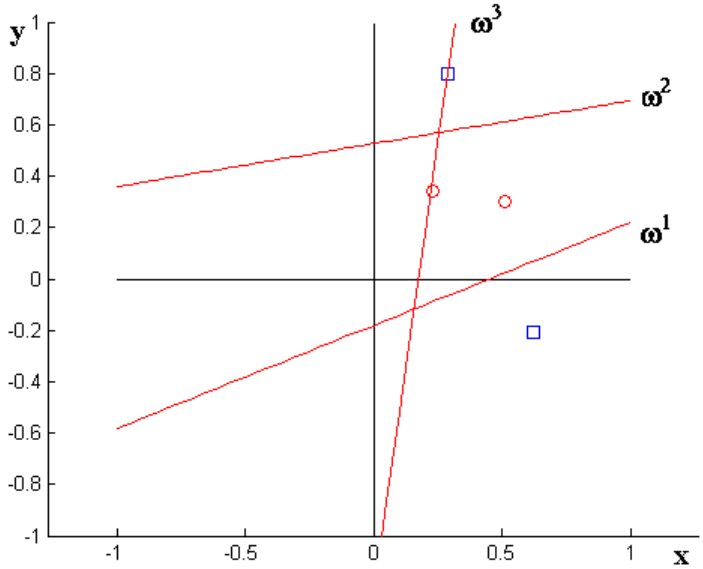


Figura 3.9 - Representação dos hiperplanos gerados pelo Slicer Básico.

Neste trabalho denota-se por regra o par (hiperplano, classe associada).

Ao final do processo tem-se um conjunto de regras Ω, construído no passo 8 e de acordo com a Definição 12 de formação da regra. Estas regras estão na Tabela 3.3 e são representados como mostra a Figura 3.9, onde se pode ver os hiperplanos separadores na forma de regras.

Tabela 3.3 – Regras obtidas ao final do aprendizado do Slicer Básico.

Regras Ω	Vetor ortogonal T	Limitante α	Classe β
ω^1	$\mathrm{T}^1 = (0.2075, -0.5175)$	$\alpha^1 = 0.0940$	$\beta^1 = 2$
ω^2	$\mathrm{T}^2 = (-0.0533, 0.3200)$	$\alpha^2 = 0.1685$	$\beta^2 = 2$
ω^3	$\mathrm{T}^3 = (0.1400, -0.0200)$	$\alpha^3 = 0.0244$	$\beta^3 = 1$

Estas regras podem ser utilizadas como base para diversas implementações de sistemas de classificação assumindo os seguintes papéis, entre outros:

- argumentos (junto com o vetor de atributos) para uma função cuja imagem corresponde ao conjunto de classes;

- regras se-então-senão;

- pesos na camada oculta de uma rede neural construtiva tal como nos métodos BCP [Poulard 1995] e RDP [Tajine & Elizondo 1998];

- argumentos de sistemas híbridos, tal como aquele a ser apresentado no Capítulo 4, um sistema classificador Slicer-Nebuloso.

Sejam *vat*, o vetor de atributos que se deseja classificar, e o conjunto de regras Ω obtidos na execução ilustrativa anterior.

A interpretação na forma de regras se-então-senão deste conjunto de regras é a seguinte:

Se $(< \mathrm{T}^1, vat >) \geq \alpha^1$

Então classe de *vat* é igual a β^1

Senão Se $(< \mathrm{T}^2, vat >) \geq \alpha^2$

Então classe de *vat* é igual a β^2

Senão Se $(< \mathrm{T}^3, vat >) \geq \alpha^3$

Então classe de *vat* é igual a β^3

Senão classe de *vat* é igual a β^3 (por padrão)

A interpretação em linguagem natural deste conjunto de regras é a seguinte:

Se *vat* se encontra no semi-espaço definido pelo hiperplano (T^1,α^1) e do lado indicado pelo T^1, então associa-se a classe β^1 ao *vat*, senão

Se *vat* se encontra no semi-espaço definido pelo hiperplano (T^2,α^2) e do lado indicado pelo T^2, então associa-se a classe β^2 ao *vat*, senão

Se *vat* se encontra no semi-espaço definido pelo hiperplano (T^3,α^3) e do lado indicado pelo T^3, então associa-se a classe β^3 ao *vat*, senão

Associa-se a classe β^3 ao *vat* por padrão.

A seguir apresenta-se a forma escolhida pelo Slicer como método de classificação (Se-Então-Senão).

3.1.2 – Descrição da classificação usando Slicer

Cada regra obtida pelo Slicer Básico determina um hiperplano.

Um hiperplano divide o espaço em duas partes e, no caso do Slicer Básico, o lado apontado pelo sentido do vetor ortogonal que o define é composto de exemplos de uma única classe.

Aplicando-se este conceito, pode-se criar vários subespaços, cada qual indicando uma determinada classe.

Feito isso, qualquer vetor de atributos tem sua classe determinada apenas pela localização espacial que possui.

A fase de classificação do Slicer tem esse objetivo: dado um vetor de atributos, usa-se o conjunto de regras obtido na fase de aprendizado do Slicer Básico para determinar a classe a qual pertence, pela simples localização do subespaço ao qual pertence.

Para efeito ilustrativo considere o conjunto de regras obtido na Seção 3.1.1. A partir do respectivo conjunto de hiperplanos (ver Definição 12), determinam-se subespaços, cada qual, associado à classe da respectiva regra. A Figura 3.10 mostra os 4 subespaços resultantes em que: subespaços 1 e 2 correspondem à classe 1; subespaço 3 à classe 2 e subespaço 4 à classe 2 por padrão.

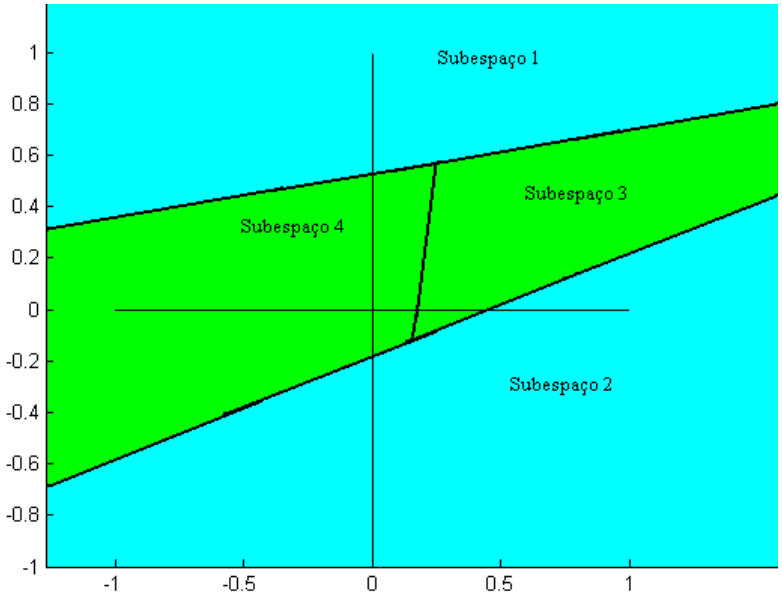


Figura 3.10 - Representação dos subespaços determinados pelas regras geradas pelo Slicer Básico: subespaços 1 e 2 correspondem à classe 1; subespaços 3 e 4 à classe 2.

A Figura 3.10 é obtida das regras geradas na fase de aprendizado. As regiões delimitadas pelos hiperplanos deixam claro como determinar as classes.

De acordo com o algoritmo de classificação (Quadro 2) busca-se no conjunto de regras, na mesma ordem em que são criadas, qual a primeira regra a ser satisfeita pelo vetor de atributos. Neste caso toma-se a classe desta regra como sendo a classe procurada. Caso contrário associa-se ao vetor de atributos a classe da última regra determinada na fase de aprendizado.

Slicer Básico - Algoritmo de Classificação
Entradas: conjunto de regras Ω e o vetor de atributos ϑ a ser classificado;
Saída: classe χ atribuída a ϑ.
Passo 1 - Faça j=1e m=$\|\Omega\|$ (cardinalidade de Ω);
Passo 2 - Calcula-se $s = < T^j, \vartheta > - \alpha^j$ (produto interno entre o vetor da regra e o vetor de atributos a ser classificado menos o limitante);
Passo 3 - Se $s \geq 0$ atribui-se a classe β^j a χ e finaliza-se o algoritmo;
Passo 4 - Se $s < 0$ e $j < m$, j=j+1 e vá para o Passo 2;
Passo 5 – Se $s < 0$ *e* $j = m$ atribui-se a classe β^j a χ e finaliza-se o algoritmo.

Quadro 2 – Algoritmo de classificação.

Execução demonstrativa de classificação pelo Slicer Básico:

A Tabela 3.3 apresenta o conjunto de vetores a ser utilizado na execução demonstrativa da classificação pelo Slicer Básico.

Considerem-se as regras geradas pelo Slicer Básico na execução demonstrativa da seção 3.1.1.

Verifica-se para 4 vetores de atributos o processo de classificação.

Sejam os vetores de atributos indicados na Tabela 3.4, na qual seguiu-se a nomenclatura x e y para os atributos em exemplos de apenas duas dimensões.

Tabela 3.4 – Vetores de atributos para classificação.

Vetores (ϑ)	x	y	Classe
$\vartheta 1$	1	0	?
$\vartheta 2$	1	1	?
$\vartheta 3$	0.5	0.2	?
$\vartheta 4$	-1	0	?

Para o vetor $\vartheta 1$:

Passo 1- Faça j=1 e m=3.

Passo 2 – Calcula-se *s* para o vetor $\vartheta 1$ e ω^1, $s = 0.1135$.

Passo 3 – Como $s \geq 0$, determina-se β^1 como classe do vetor $\vartheta 1$ e termina-se o algoritmo.

Para o vetor $\vartheta 2$:

Passo 1- Faça j=1 e m=3.

Passo 2 – Calcula-se *s* para o vetor $\vartheta 2$ e ω^1, $s = -0.4040$.

Passo 4 - Como $s < 0$ e j<m, faça j=j+1=2 e vai para o Passo 2.

Passo 2 – Calcula-se *s* para o vetor $\vartheta 2$ e ω^2, $s = 0.0982$.

Passo 3 – Como $s \geq 0$, determina-se β^2 como classe do vetor $\vartheta 2$ e termina-se o algoritmo.

Para o vetor $\vartheta 3$:

Passo 1- Faça j=1 e m=3.

Passo 2 – Calcula-se *s* para o vetor $\vartheta 3$ e ω^1, $s = -0.0937$.

Passo 4 - Como $s < 0$ e $j < m$ faça $j = j + 1 = 2$ e vai para o Passo 2.

Passo 2 – Calcula-se *s* para o vetor $\vartheta 3$ e ω^2, $s = -0.1344$.

Passo 4 - Como $s < 0$ e $j < m$, faça $j = j + 1 = 3$ e vai para o Passo 2.

Passo 2 – Calcula-se *s* para o vetor $\vartheta 3$ e ω^3, $s = 0.0416$.

Passo 3 – Como $s \geq 0$, determina-se β^3 como classe do vetor $\vartheta 3$ e termina-se o algoritmo.

Para o vetor $\vartheta 4$:

Passo 1- Faça $j = 1$ e $m = 3$.

Passo 2 – Calcula-se *s* para o vetor $\vartheta 4$ e ω^1, $s = -0.3015$.

Passo 4 - Como s < 0 e $j < m$, faça $j = j + 1 = 2$ e vai para o Passo 2.

Passo 2 – Calcula-se *s* para o vetor $\vartheta 4$ e ω^2, $s = -0.1152$.

Passo 4 - Como s < 0 e $j < m$, faça $j = j + 1 = 3$ e vai para o Passo 2.

Passo 2 – Calcula-se *s* para o vetor $\vartheta 4$ e ω^3, $s = -0.1644$.

Passo 5 - Como $s < 0$ e $j = m$, determina-se β^3 como classe do vetor $\vartheta 4$ e termina-se o algoritmo.

Na Tabela 3.5 têm-se todos os cálculos efetuados nesta execução demonstrativa, inclusive as classes atribuídas aos vetores.

Tabela 3.5 – Cálculos e resultados obtidos na demonstração de classificação.

Vetores(ϑ)	*x*	*Y*	ω^1	ω^2	ω^3	χ
$\vartheta 1$	1	0	0.1135	-----	------	2
$\vartheta 2$	1	1	-0,4040	0,0982	------	2
$\vartheta 3$	0.5	0.2	-0,0937	-0,1344	0,0416	1
$\vartheta 4$	-1	0	-0,3015	-0,1152	-0,1644	1

Na Figura 3.11 fica clara a associação entre a posição dos vetores e suas classes correspondentes.

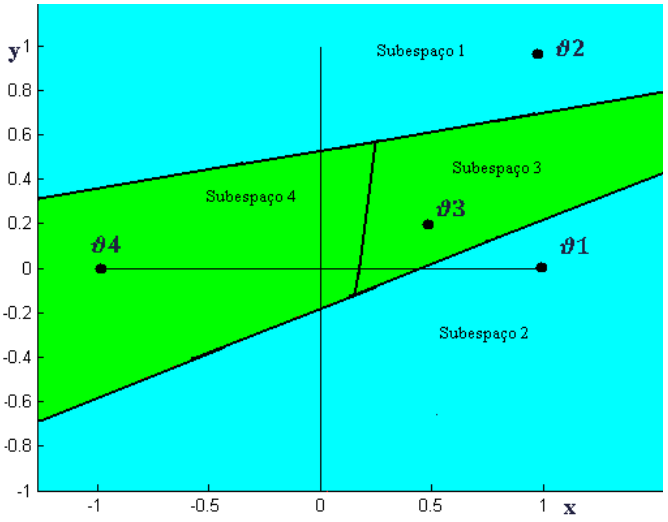


Figura 3.11 – Representação dos vetores classificados.

3.1.3 - Custo computacional do Slicer Básico

Um dos aspectos considerados em avaliação de ferramentas computacionais é o seu custo de execução, o qual é mensurado em termos de tempo de execução (tempo necessário para emitir uma solução) e o volume de memória (memória necessária para suportar as demandas da ferramenta).

Certamente o custo computacional é dependente do tamanho do problema, mas uma análise assintótica, pelo menos em termos qualitativos de tempo, oferece uma medida de reconhecida importância para efeitos comparativos.

A teoria envolvida neste tipo de análise já é bem desenvolvida [Aho & Ullman 1992], e o custo é medido em termos do operador O(.).

A notação "big-oh" foi desenvolvida para manter ocultos alguns fatores tais como:

- o número médio de instruções de máquina que um determinado compilador gera;

- o número médio de instruções de máquina que uma determinada máquina executa por segundo.

Assim quando se diz que um programa demora $O(n)$ tempo, significa que o programa demora uma constante vezes n tempo, em que n é a cardinalidade do conjunto de exemplos a ser usado no treinamento do algoritmo.

Em [Aho & Ullman 1992] são apresentados alguns procedimentos padrões que podem ser seguidos para efeitos de análise de algoritmos:

1) Analisar cada linha do algoritmo em termos de operações matemáticas simples e atribuições, sendo cada igual a O(1);
2) todas as operações aninhadas são escritas na forma:

   $O(constante\ do\ laço)O(\text{soma dos O(.)do bloco interno ao laço})$;
3) são válidas as operações com o operador O(.):
   - $O\big(g(n)f(n)\big) = O\big(g(n)\big)O\big(f(n)\big)$;
   - $O\big(n^i\big) + O\big(n^j\big) = O\big(n^i\big), se\ i > j,$ ou seja, termos de menor grau não importam;
   - $O(kn) = O(n),$ ou seja, constantes não importam.

Assim, sejam N a cardinalidade do conjunto de exemplos Λ e P seu número de atributos.

Teorema: O custo computacional do Slicer Básico segundo o método $O(.)$ é tal que:

$$O(Slicer\ Básico) = O(N^2).$$

Baseado nos conceitos em [Aho&Ullman 1992], todas operações que não usarem N ou P como parâmetros são considerados como $O(1)$ e as demais escritas em função de N e P. Como constantes serão eliminadas pela regra de maior potência então, na medida do possível, nem serão acrescidas as equações.

Prova: no desenvolvimento desta demonstração matemática:

Sendo calculado para o pior caso, ou seja, separa-se exemplo a exemplo, um por vez, sendo necessários N regras, N execuções.

Sejam, pois, os custos de cada passo do algoritmo:

Passo 1: corresponde a declarações simples, portanto:

$$O(Passo1) = 1;$$

Passo 2: três laços de repetição estão envolvidos, 1 deles aninhado:

- inicialização do centroide: P repetições;
- somatório dos atributos: PN repetições;
- cálculo das médias: P repetições;

Assim, $O(Passo2) = O(P) + O(PN) + O(P) = O(PN) = O(N)$.

Passo 3: três laços de repetição estão envolvidos, 1 deles aninhado:

- inicialização das distâncias: N repetições;
- cálculo dos quadrados das diferenças: PN repetições;
- extração das raízes: N repetições;

Assim, $O(Passo3) = O(P) + O(PN) + O(P) = O(PN) = O(N)$.

Passo 4: um laço de repetição:

- busca do maior distância: N repetições;

Assim, $O(Passo4) = O(N)$.

Passo 5: um laço de repetição:

- diferença entre exemplo mais distante e centroide: N repetições;

Assim, $O(Passo5) = O(N)$.

Passo 6: dois laços de repetição estão envolvidos:

- inicialização das projeções: N repetições;
- cálculo das projeções: PN repetições;

Assim, $O(Passo6) = O(N) + O(PN) = O(PN) = O(N)$.

Passo 7: 5 atribuicoes simples e um bloco com 3 laços de repetição, portanto:

Passo 7.1: corresponde a declarações simples, portanto:

$$O(Passo\ 7.1) = 1;$$

Passo 7.2: três laços de repetição estão envolvidos, 2 deles aninhado:

- busca de θ^*: N repetições;

- cálculo de $\overline{\Psi}$: PN repetições;

- cálculo de $\overline{\overline{\Psi}}$: PN repetições;

Assim, $O(Passo7.2) = O(N) + O(PN) + O(PN) = O(PN) = O(N)$.

Passo 7.3: corresponde a declarações simples, portanto:

$$O(Passo\ 7.3) = 1;$$

Passo 7.4: corresponde a declarações simples, portanto:

$$O(Passo\ 7.4) = 1;$$

Passo 7.5: corresponde a declarações simples, portanto:

$$O(Passo\ 7.5) = 1;$$

Passo 7.6: corresponde a declarações simples, portanto:

$$O(Passo\ 7.6) = 1;$$

Como considera-se o pior caso, ou seja, uma iteração dos passos 2 ao 10 para cada exemplo, o Passo 7 executa seus passos intermediários somente uma vez então pela regra da soma

$$O(Passo7) = O(1) + O(N) + O(1) + O(1) + O(1) + O(1) = O(N).$$

Passo 8: um laço de repetição:

- atribuição em ω^i: P repetições; assim, $O(Passo8) = O(P)$.

Passo 9: um laço de repetição:

- comparação do limitante com projeções e eliminação do exemplo: N repetições;

Assim, $O(Passo9) = O(N)$.

Passos 2 ao 9: 8 blocos sequenciais, pela regra da soma, assim

$O(Passo2ao9) = O(N) + O(N) + O(N) + O(N) + O(N) + O(N) + O(P) + O(N) = O(N)$.

Passo 2 ao 10: considerando o pior caso, com N iterações sendo necessárias,

- N iterações dos passos 2ao9: N repetições; assim

$O(Passo2ao10) = O(Passos2ao9)O(N) = O(Passo2ao10) = O(N)O(N) = O(N^2)$.

Passo 1 ao 10: inicialização e bloco dos passos 2ao10

$O(Passo1ao10) = O(Passo1)O(Passo2ao10) = O(1)O(N^2) = O(N^2)$.

Como visto P é uma constante e foi desprezado em todos os cálculos finais.

Como N tende ao infinito, resulta:

$O(Passo1ao10) = O(N^2)$, ou seja, $O(Slicer\ Básico) = O(N^2)$. Está provado.

3.2 – Slicer-O

O conjunto de regras obtido pelo Slicer Básico representa o conhecimento extraído durante a execução do respectivo algoritmo de aprendizado. Este conhecimento é obtido por meio da determinação de hiperplanos separadores de subconjuntos de classe única, recursivamente, até o total esvaziamento do conjunto de exemplos (treinamento). Na fase de classificação estas regras são testadas uma a uma, na mesma sequência em que foram criadas até que uma regra atenda a condição de parada, qual seja, o resultado da função hiperplano for maior ou igual a zero. Neste caso se atribui a classe associada a esta regra ao exemplo sendo classificado. Caso nenhuma regra atenda a condição de parada, atribui-se a classe da última regra testada.

Porém, quando se produzem estas regras não há conhecimento prévio para o domínio de conhecimento a ser aprendido. Após a sua criação, o próprio conjunto de regras é um conhecimento adquirido. Pode-se então fazer uso deste conhecimento para se tentar um conjunto otimizado, através da verificação de quais regras obtêm melhor desempenho no conjunto de treinamento.

O objetivo desta investigação é verificar se os procedimentos de: 1) reorganização da ordem em que as regras são testadas; 2) descarte de algumas regras; possibilitam encontrar um conjunto de regras com menor cardinalidade e com um desempenho equivalente.

3.2.1 – Descrição do Slicer-O

No processo de otimização cada uma das regras geradas pelo Slicer Básico é avaliada por meio de duas medidas de avaliação: número de acertos e número de erros. A regra que obtêm melhor avaliação é retirada do conjunto de regras original e inserida no novo conjunto e os exemplos classificados por esta regra são retirados do conjunto de treinamento. Refaz-se este processo até que o conjunto de treinamento esteja vazio.

O Quadro 3 apresenta o algoritmo de otimização e, em seguida, sua execução é ilustrada passo-a-passo, mostrando a representação geométrica da execução e resultados.

Slicer-O - Algoritmo de treinamento

Entrada: conjunto de exemplos Λ; conjunto de regras $\boldsymbol{\Omega}$, índice μ do exemplo mais distante e o limite de erros ν.

Saída: novo conjunto de regras $\boldsymbol{\Omega}^*$.

Passo 1: $\Omega^* = \emptyset$.

Passo 2: Faça contador de regras $j = 0$; número de regras $i = |\Omega|$;

maior número de acertos $\zeta = 0$; índice da regra vencedora $\xi = 0$.

Passo 3: Faça $j = j + 1$.

Passo 4: Se $j > i$, vai para o Passo 8 (se terminou de verificar todas regras).

Passo5: Faça $\overline{\Upsilon} = \{\lambda^g | (< \mathrm{T}^j, \Gamma^g > -\alpha^j) \geq 0, \beta^j = \chi^\mu, 1 \leq g \leq |\Lambda|\}$;

$$\overline{\overline{\Upsilon}} = \{\lambda^g | (< \mathrm{T}^j, \Gamma^g > -\alpha^j) \geq 0; \beta^j \neq \chi^\mu, 1 \leq g \leq |\Lambda|\}.$$

Passo 6: Se $|\overline{\overline{\Upsilon}}| > \nu$, vai para o Passo 3.

Passo 7: Se $|\overline{\Upsilon}| > \zeta$, $\zeta = |\overline{\Upsilon}|$, $\xi = j$, vai para o Passo 3.

Passo 8: Acrescenta a regra ω^j em $\boldsymbol{\Omega}^*$: $\Omega^* \leftarrow \Omega^* + \{\omega^j\}$;

Exclua ω^j de $\boldsymbol{\Omega}$: $\Omega \leftarrow \Omega - \{\omega^j\}$; e

exclua do conjunto de exemplos Λ todos exemplos classificados corretamente por ω^j:

$$\Lambda \leftarrow \Lambda - \{\lambda^g | (< \mathrm{T}^j, \Gamma^g > -\alpha^j) \geq 0; \beta^j = \chi^\mu; 1 \leq g \leq |\Lambda|\}.$$

Passo 9: Se $|\Lambda| > 0$, vai para o Passo 2.

Passo 10: Fim : $\boldsymbol{\Omega}^*$ é o novo conjunto de regras, W reduzido e reordenado.

Quadro 3 - Algoritmo Slicer-O.

Execução ilustrativa do treino do Slicer-O:

Seja o conjunto de regras obtido pelo Slicer Básico, considerando o conjunto de exemplos mostrado na Figura 3.12. A Tabela 3.6 relaciona as regras com seus acertos e erros obtidos no momento de sua criação, ou seja, quando da criação da regra durante o treinamento do Slicer Básico, um hiperplano é gerado, e este hiperplano separa um subconjunto de exemplos, todos de uma única classe, o que significa que o número de acertos é igual a cardinalidade deste subconjunto e que o número de erros é igual a zero, pois este subconjunto não possui elementos de outra classe.

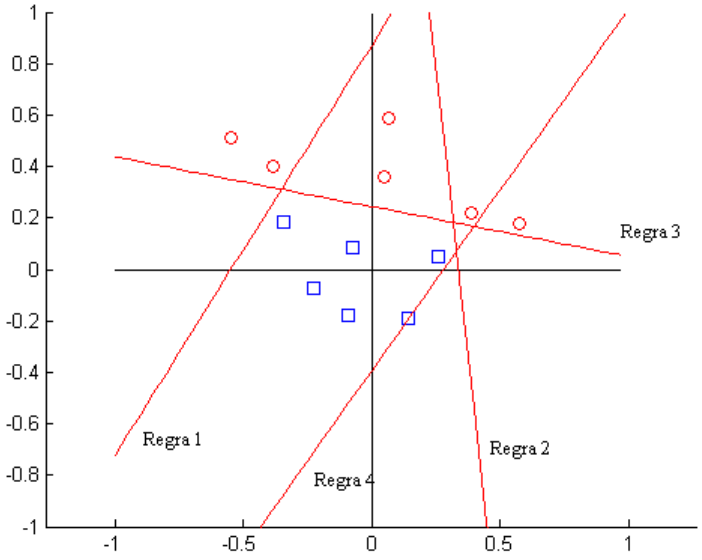


Figura 3.12- Conjunto de exemplos e regras geradas pela execução do Slicer Básico.

Tabela 3.6- Regras geradas pelo Slicer Básico e o número de acertos(cardinalidade do subconjunto separado pela regra) e erros(sempre zero).

Regra	Acertos	Erros
1	2	0
2	2	0
3	2	0
4	6	0

Ao se executarem as mesmas regras para o conjunto de exemplos Λ, após execução dos passos 2 ao 5, obtêm-se valores diferentes, conforme Tabela 3.7.

Tabela 3.7 - Regras executadas sobre o conjunto inteiro de exemplos (Passo 5).

Regra	Acertos	Erros
1	2	0
2	2	0
3	6	0
4	6	5

Passa-se então para a seleção de regras, escolhendo a que obtêm maior número de acertos e menor número de erros (looping dos passos 3 ao 7). Na demonstração a regra 3 cumpre essa exigência. Retira-se do conjunto de exemplos todos os exemplos que são separados por esta regra, e coloca-se esta regra como nova regra 1 em um novo conjunto, retirando-a do conjunto antigo de regras.

Conforme Tabela 3.8 percebe-se a situação das regras restantes sobre o conjunto restante de exemplos (em uma nova execução dos passos 3 a 7).

Tabela 3.8 - Regras executadas sobre o conjunto inteiro de exemplos.

Regra	Acertos	Erros
1	0	0
2	0	0
3	6	0

Neste novo conjunto de regras, pode-se perceber que a regra 3 obtém melhor resultado. Assim é selecionada como regra a ser transferida para o novo conjunto de regras, como regra 2.

Então, retira-se do conjunto de exemplos todos os exemplos classificados corretamente por esta regra.

O conjunto de exemplos que se obtêm é o conjunto vazio. Como não há necessidade de se obter mais regras, desprezam-se as regras restantes no conjunto de regras antigo e termina-se o processo de otimização substituindo o antigo conjunto de regras pelo novo conjunto obtido.

Percebe-se na Figura 3.13 e na Tabela 3.9 que o novo conjunto possui 2 regras apenas, número menor que o anterior (4 regras).

Tabela 3.9 - Regras do Slicer-O.

Regra	Acertos	Erros
1	6	0
2	6	0

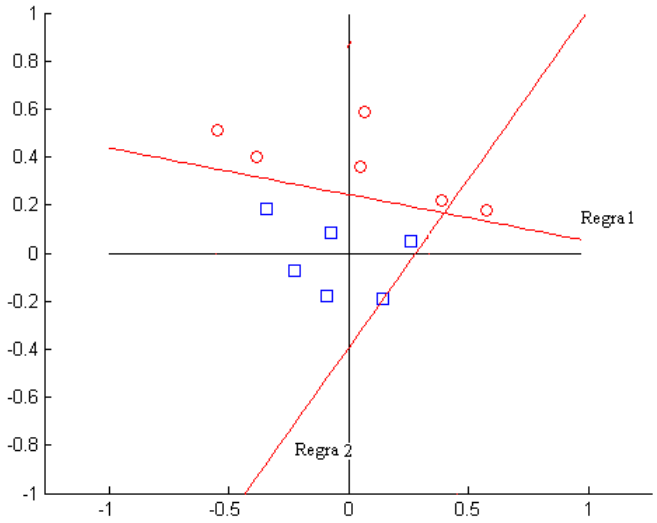
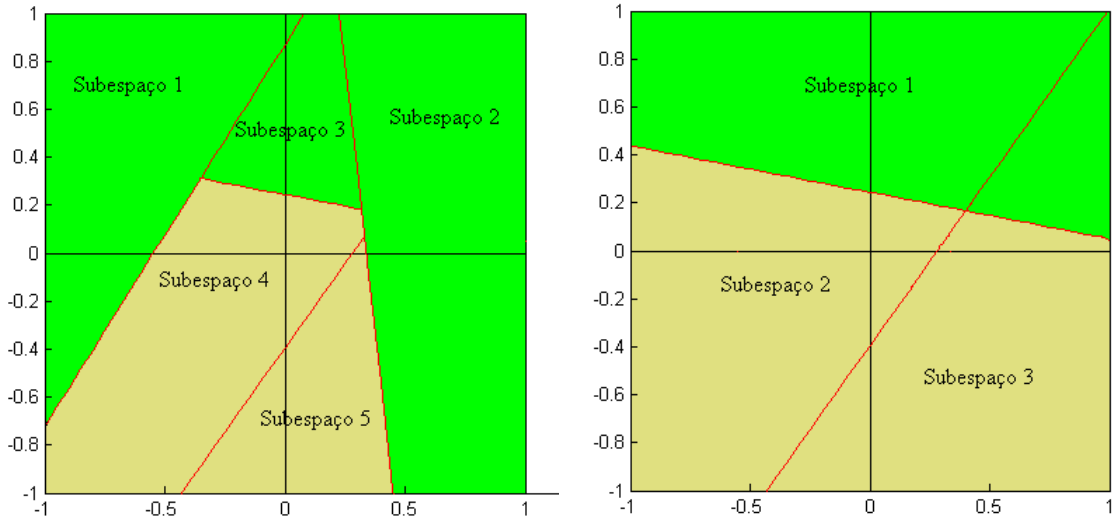


Figura 3.13 - Novo conjunto de regras gerado pelo Slicer-O.

Com esta redução de regras, obtém-se um tempo médio de classificação menor, e com um número menor de regras, tem-se menor uso de memória.

Espera-se que a divisão do espaço de classificação obtido pelo Slicer-O seja mais conveniente do que a obtida sem otimização (Slicer Básico), conforme pode-se notar na Figura 3.14.

Com os bons resultados obtidos por esta versão nos testes, optou-se por utilizá-lo em todas as avaliações de desempenho dos demais algoritmos apresentados neste trabalho.



(a) (b)

Figura 3.14 – Comparação dos subespaços encontrados por:
(a) Slicer Básico; (b) Slicer-O.

3.3 – Slicer-F

Domínios de conhecimento são representados na prática por bases de dados, formadas por exemplos. No caso de aprendizado supervisionado estes exemplos possuem uma classe associada.

Mesmo sendo geradas por especialistas ou automaticamente por algum sistema computacional, estas bases podem conter erros nos valores dos dados (ruídos) ou falhas (atributos faltantes) [Weiss & Kulikowski 1990] [Briscoe & Caelli 1996].

Segundo [Witten & Frank 2005] os principais motivos da ocorrência de atributos faltantes são:

- valor não se aplica ao caso;
- dados digitados ou colhidos erroneamente;
- base originada de diferentes fontes, nas quais o valor do atributo não é relevante;
- ruído ou defeito na base.

Segundo [Weiss & Kulikowski 1990] algumas estratégias utilizadas para permitir o uso de tais bases são:

- atribuir um valor (padrão ou calculado por alguma heurística) ao valor faltante;
- eliminar exemplos com atributo(s) faltante(s);
- admitir atributo faltante, adaptando os tratamentos dados aos atributos (cálculos, comparações).

Nesta seção descreve-se um conjunto de algoritmos para tratamento de base de dados que admite atributos faltantes em exemplos. Esta versão do Slicer, admitindo atributos faltantes, alarga o espectro de domínios de conhecimento que podem ser admitidos pelo sistema de classificação Slicer.

Nas subseções seguintes além da descrição do sistema de classificação Slicer, demonstra-se a execução do algoritmo, ilustrado por figuras, tabelas de dados e cálculos permeado com as descrições pertinentes.

3.3.1 – Descrição do Slicer-F

Para que o Slicer possa aceitar atributos faltantes, é necessário que se façam algumas adaptações nos seus procedimentos de cálculos e comparações.

Portanto, a partir da versão do Slicer Básico é desenvolvido um sistema mais robusto, uma versão Slicer para atributos faltantes.

Representa-se neste trabalho atributos faltantes pelo número -9999 (nos algoritmos) e pelo sinal de interrogação nas bases de domínios.

As operações que são adaptadas são as seguintes:

1 - cálculo do centroide: os atributos faltantes não entram na somatória dos atributos e nem são considerados na divisão final (Quadro 4, Passo 6);
2 - cálculo de distâncias Euclidianas: os termos que calculam com atributos faltantes são considerados iguais a zero (Quadro 5, Passos 6 e 7);
3 - diferença entre vetores (na criação de vetor ortogonal): os termos com atributos faltantes resultam zero (Quadro 6, Passos 5 e 6);
4 - produto interno (nos cálculos das projeções e na classificação): os termos de produto interno com atributo faltante resultam zero (Quadro 7, Passos 6 e 7).

As operações acima são justificadas pelas seguintes heurísticas:

1 - caso se dividisse o somatório dos atributos de todos exemplos pelo número total de exemplos, o centroide obtido teria os seus elementos minorados, posicionando muito fora do possível centro de massa dos exemplos.
2 - a heurística adotada aqui busca localizar a menor distância Euclidiana entre as possíveis para exemplos com atributo faltante, e para se obter esse resultado os termos que contenham atributo faltante são considerados igual a zero;
3 - a heuristica proposta deseja que o vetor gerado seja ortogonal ao eixo do atributo faltante, prevendo que o hiperplano a ser gerado manterá este exemplo totalmente de um dos lados;
4 - para manter a coerência da heurística do caso anterior,

Estas heurísticas se aplicam a exemplos com atributos faltantes, centroide com atributo faltante (gerado quando todos exemplos possuem o mesmo atributo faltando), vetor ortogonal com componente faltante (derivado do calculo entre centroide com componente faltante e exemplo com atributo faltante).

A seguir estão indicadas estas alterações, na forma de algoritmos.

Algoritmo de cálculo de vetor centroide Π com atributo faltante

Entrada: conjunto de exemplos Λ
Saída: vetor centroide Π

Passo 1 – faça contador de atributos $j = 0$; $vlaf = -9999$;

Passo 2 – faça $j = j + 1$; contador de exemplos $i = 0$; somatório $s = 0$; total de atributos somados $t = 0$; $\pi^j = vlaf$;

Passo 3 – se $j > P$ vai para o Passo 10;

Passo 4 – faça $i = i + 1$;

Passo 5 – se $i > N$ vai para o Passo 8;

Passo 6 – se $\gamma^{i,j} = vlaf$ vai para o Passo 4;

Passo 7 – faça $s = s + \gamma^{i,j}$; $t = t + 1$; vai para o Passo 4;

Passo 8 – se $t > 0$ faça $s = s/t$;

Passo 9 – se $t > 0$ faça $\pi^j = s$; vai para o Passo 2;

// caso algum atributo seja faltante em todos os exemplos, gera um componente faltante
// no centroide

Passo 10 - termina algoritmo.

Quadro 4 - Algoritmo de cálculo de centroide com atributo faltante.

Algoritmo de cálculo do conjunto de distâncias Euclidiana Δ com atributo faltante

Entradas: conjunto de exemplos Λ e centroide Π.
Saída: conjunto de distâncias Euclidianas Δ.

Passo 1 – faça $i = 0$; $vlaf = -9999$;

Passo 2 – faça contador de exemplos $i = i + 1$; contador de atributos $j = 0$; somatório $s = 0$;

Passo 3 – se $i > N$ vai para o Passo 10;

Passo 4 – faça $j = j + 1$;

Passo 5 – se $j > P$ vai para o Passo 9;

Passo 6 – se $\gamma^{i,j} = vlaf$ vai para o Passo 4;

Passo 7 - se $\pi^j = vlaf$ vai para o Passo 4;

Passo 8 – faça $s = s + \left(\gamma^{i,j} - \pi^j\right)^2$; vai para o Passo 4;

Passo 9 – faça $\delta^i = s^{0.5}$; vai para o Passo 2

Passo 10 - termina algoritmo.

Quadro 5 - Algoritmo de do conjunto de distâncias para atributo faltante.

Algoritmo de cálculo de vetor ortogonal Φ com atributo faltante

Entradas: conjunto de exemplos Λ; vetor centroide Π;
índice μ do exemplo mais distante do vetor centroide Π.
Saída: vetor ortogonal Φ.

Passo 1 – faça $j = 0$; $vlaf = -9999$;

Passo 2 – faça $j = j + 1$;

Passo 3 – se$j > P$ vai para o Passo 8;

Passo 4 – inicializa a posição j do vetor ortogonal $\phi^j = 0$;

Passo 5 – se $\pi^j = vlaf$ vai para o Passo 2;

Passo 6 – se $\gamma^{\mu,j} = vlaf$ vai para o Passo 2;

Passo 7 – $\phi^j = \gamma^{\mu,j} - \pi^j$;

Passo 8 – termina algoritmo.

Quadro 6 - Algoritmo de cálculo do vetor ortogonal com atributo faltante.

Algoritmo de cálculo do conjunto de projeções Ψ com atributo faltante

Entradas: conjunto de exemplos Λ e vetor ortogonal Φ.
Saída: conjunto de projeções Ψ.

Passo 1 – faça $i = 0$; $vlaf = -9999$;

Passo 2 – faça contador de exemplos $i = i + 1$; contador de atributos $j = 0$; somatório $s = 0$;

Passo 3 – se $i > N$ vai para o Passo 10;

Passo 4 – faça $j = j + 1$;

Passo 5 – se $j > P$ vai para o Passo 9;

Passo 6 – se $\gamma^{i,j}$ = vlaf; vai para o Passo 4;

Passo 7 - se $\phi^j = vlaf$; vai para o Passo 4;

Passo 8 – faça $s = s + (\gamma^{i,j} \times \phi^j)$; vai para o Passo 4;

Passo 9 – faça $\psi^i = s$; vai para o Passo 2

Passo 10 - termina algoritmo.

Quadro 7 - Algoritmo de cálculo de conjunto de projeções com atributos faltantes.

Execução demonstrativa do Slicer-F:

A Tabela 3.10 apresenta o conjunto de treinamento a ser utilizado na execução demonstrativa do Slicer-F. A Figura 3.15 corresponde à representação geométrica deste conjunto de exemplos.

Tabela 3.10 - Conjunto de exemplos para treinamento do Slicer-F (atributo faltante na linha indicada).

$\lambda^g\{g=1,...,6\}$	x	y	χ^g	
λ^1	0.14	0.39	1	
λ^2	0.28	0.16	1	
λ^3	-0.19	0.23	1	
λ^4	0.57	-9999	2	←
λ^5	0.35	0.30	2	
λ^6	-0.14	0.07	2	

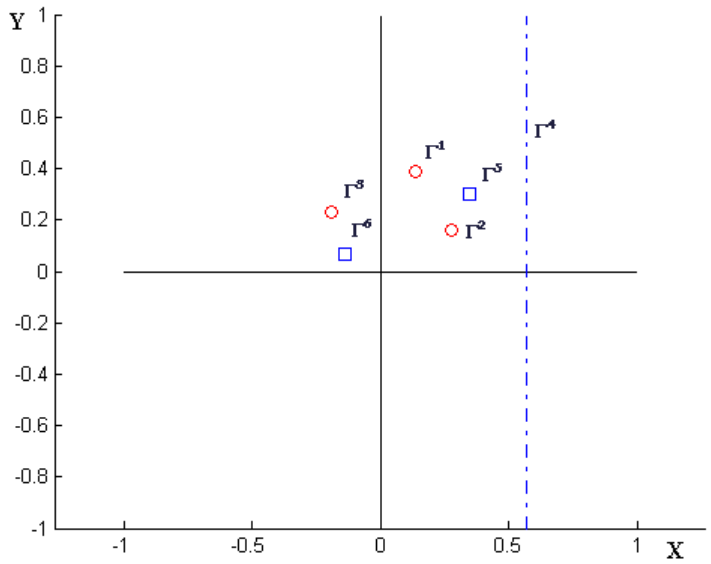


Figura 3.15 - Representação geométrica do conjunto de exemplos Λ, inclusive de um exemplo (Γ^4) com atributo faltante representado pela linha pontilhada.

As linhas da Tabela 3.11 correspondem ao rastreamento das variáveis durante a execução demonstrativa do algoritmo na primeira iteração do Slicer-F (passo 2 ao passo 10 do Quadro 1). Note-se que este processo deve ocorrer mais três vezes, além da mostrada, para que ao final tenha-se um total de 4 regras definidas.

Tabela 3.11 – Resultados de cálculos para cada um dos passos da primeira iteração do algoritmo do Slicer-F.

Passos	Situação das variáveis
Passo 1	j=0; Λ={((0.14,0.39),1),((0.28,0.16),1),((–0.19,0.23),1), ((0.57,–9999),2),((0.35,0,30),2),((–0.14,0.07),2)}
Passo 2	Π=(0.1683, 0.2300)
Passo 3	Δ={ 0.1625, 0.1318, 0.3583, 0.4017,0.1947,0.3473}
Passo 4	μ=4
Passo 5	Φ=(0.4017,0.000)
Passo 6	Ψ={0.0562, 0.1125, –0.0763, 0.2289, 0.1406, –0.0562}; ε=0.0100
Passo 7	Passo 7.1 : E=2; θ= 0.2289; κ= 0.2189. Passo 7.2 : $\bar{\Psi} = \{0.1406\}$; $\bar{\bar{\Psi}} = \{\ \}$. Passo 7.3 : ($\lvert\bar{\Psi}\rvert > 0$ e $\lvert\bar{\bar{\Psi}}\rvert = 0$) então Passo 7.4. Passo 7.4 : $\lvert\bar{\bar{\Psi}}\rvert = 0$ então θ= 0.1406; κ= 0.1306; Passo 7.2. Passo 7.2 : $\bar{\Psi} = \{\ \}$; $\bar{\bar{\Psi}} = \{0.1125\}$. Passo 7.3 : ($\lvert\bar{\Psi}\rvert = 0$ e $\lvert\bar{\bar{\Psi}}\rvert > 0$) então Passo 7.4. Passo 7.4 : $\lvert\bar{\bar{\Psi}}\rvert > 0$ então Passo 7.5. Passo 7.5 : Limite inferior $\kappa = \bar{\bar{\psi}}^1 = 0.1125$. Passo 7.6 : Limitante $\rho = (\theta + \kappa)/2 = (0.1406 + 0.1125)/2 = 0.1265$.
Passo 8	$j=j+1$; Ω = { Φ, ρ, χ^{μ}} = {(0.4017, 0.0000), 0.1265, 2.0000}
Passo 9	Λ={((0.14,0.39),1),((0.28,0.16),1),((–0.19,0.23),1),((0.35,0,30),2), ((–0.14,0.07),2)}
Passo 10	Λ não é vazio então Passo 2.

Acompanhamento passo-a-passo da execução para a primeira iteração:

Passo 1 - Faz-se j = 0.

Passo 2 - Calcula-se o centroide Π do conjunto de exemplos Λ (Quadro 4), mostrado na Figura 3.16. De acordo com o Passo 2 da Tabela 3.11, Π = (0.1683, 0.2300).

Passo 3 - Calcula-se o conjunto Λ das distâncias de todos os exemplos para este centroide Π (Quadro 5). De acordo com o Passo 3 da Tabela 3.11, Δ ={ 0.1625, 0.1318, 0.3583, 0.4017,0.1947,0.3473}.

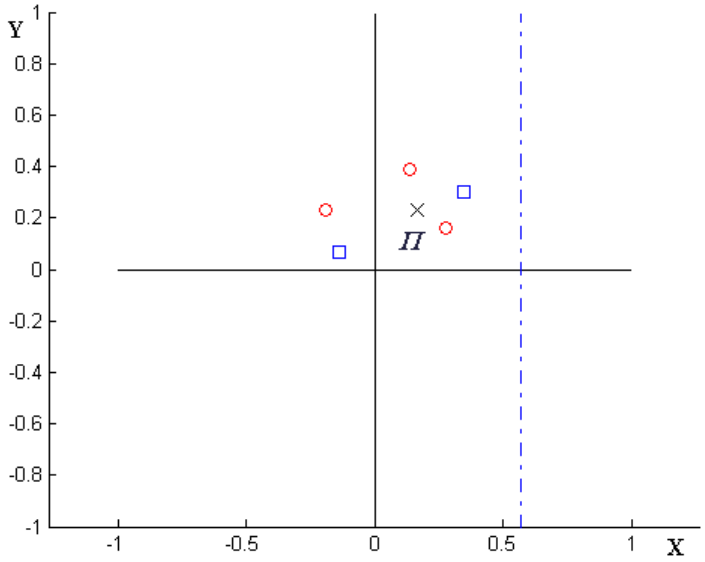


Figura 3.16 - Representação do centroide Π em bases com atributo faltante.

Passo 4 – Localiza-se o índice μ do exemplo mais distante (γ^{μ}) do centroide Π (Definição 4). De acordo com o Passo 4 da Tabela 3.11, $\mu = 4$.

A Figura 3.17, de acordo com os Passos 3 e 4, representa o centroide Π e o exemplo de índice μ do exemplo com maior distância ao centroide Π. A circunferência de centro em Π (centroide) e passando pelo exemplo Γ^{μ} (exemplo mais distante) envolve os demais exemplos, deixando claro que este exemplo efetivamente é o mais distante.

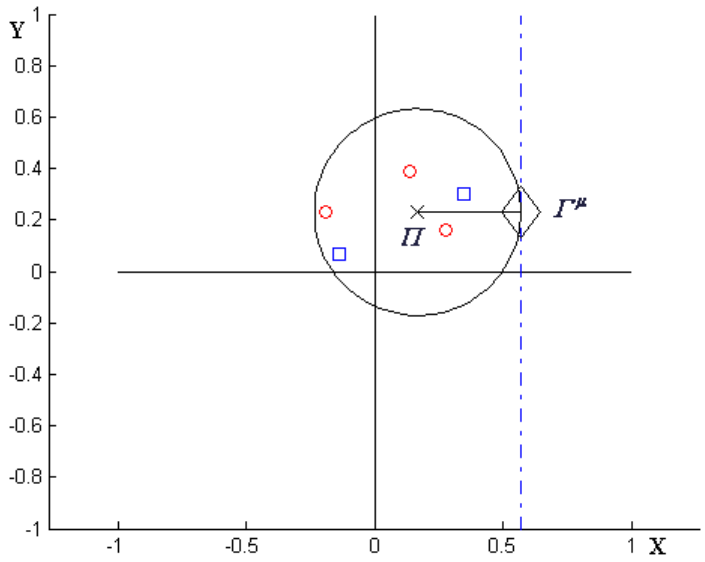


Figura 3.17 - Representação do centroide Π e do exemplo Γ^{μ}, circunferência centrada em Π envolvendo exemplos diferentes de Γ^{μ}, em bases com atributo faltante.

Passo 5 - Calcula-se o vetor Φ, pela diferença dos vetores Γ^{μ} e o centroide Π, de acordo com o Quadro 6, conforme mostrado na Figura 3.18. De acordo com o Passo 5 da Tabela 3.11, Φ = (0.4017,0.000).

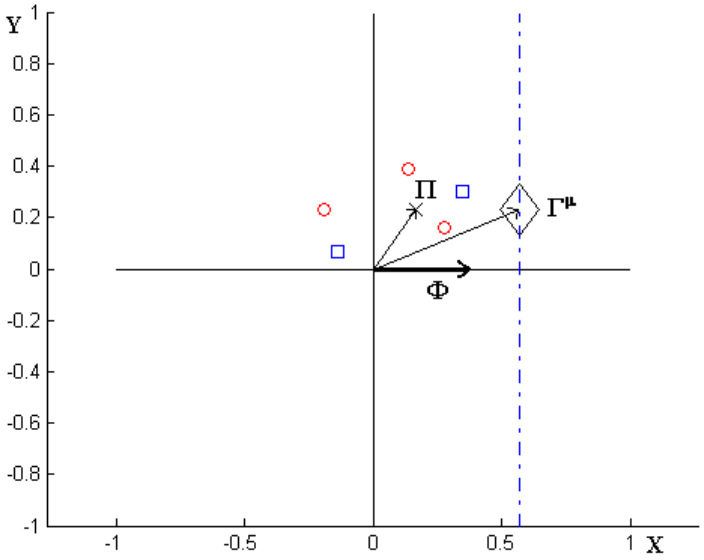


Figura 3.18 – Representação do vetor Φ ($\Phi = \Gamma^{\mu} - \Pi$).

Passo 6 – Gera-se o conjunto Ψ de projeções dos exemplos de Λ. Estas projeções são calculadas pelo produto interno entre Φ e o vetor de atributos Γ do exemplo, de acordo com o Quadro 7 (Figura 3.19). Observe que as projeções estão em uma reta paralela ao vetor Φ e seus valores crescem no sentido deste vetor. Observe também que as projeções não estão na mesma escala dos vetores de atributo, mas em escala equivalente.

Calcula-se a variável $\varepsilon = intproj/2$, em que *intproj* corresponde ao valor do menor intervalo entre duas projeções consecutivas.

No caso Ψ = {0.0562, 0.1125, –0.0763, 0.2289, 0.1406, –0.0562} e ε = 0.0010.

Passo 7 – Busca do limitante do hiperplano sendo criado:

Passo 7.1 – Atribui-se: a E a classe que está sendo separada, ou seja, $E = \chi^{\mu} = 2$; ao limitante θ o valor da maior projeção em Ψ e ao limitante κ o valor de θ menos ε, ou seja, $\theta = max(\Psi)$ e $\kappa = \theta - \varepsilon$. No caso E=2, θ = 0.2289 e κ = 0.2189.

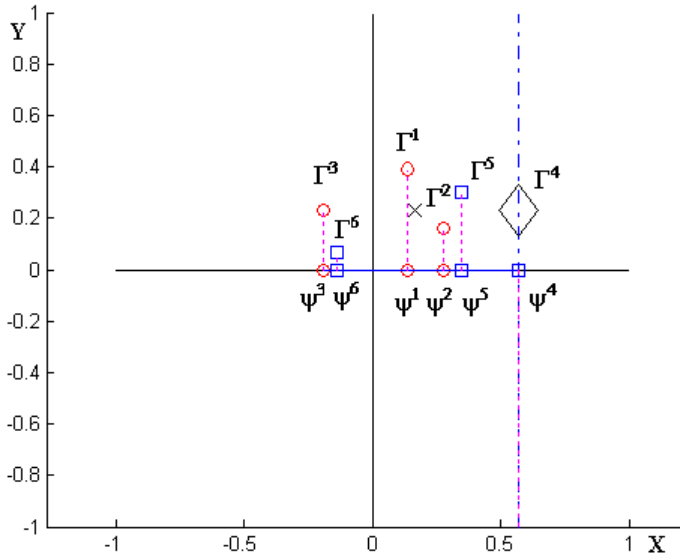


Figura 3.19 - Conjunto ordenado das projeções Ψ.

Passo 7.2 – Criam-se os conjuntos de projeções $\bar{\Psi}$ e $\bar{\bar{\Psi}}$. Uma sequência de procedimentos é necessária para determinação destes conjuntos.

- Inicialmente cria-se Ψ^* com os elementos de Ψ menores que θ, ou seja, $\Psi^* = \Psi - \{\theta\}$;

- Em seguida localiza-se θ^* como sendo a maior projeção em Ψ^*, ou seja, $\theta^* = max(\Psi^*)$; e

- Finalmente cria-se $\bar{\Psi}$ com as projeções iguais a θ^* e de classe igual a χ^{μ}; e $\bar{\bar{\Psi}}$ com as projeções iguais a θ^* e de classe diferente de χ^{μ}. No caso, $\Psi^* = \{0.0562, 0.1125, -0.0763, 0.1406, -0.0562\}$,

$\theta^* = 0.1406$, $\bar{\Psi} = \{0.1406\}$; $\bar{\bar{\Psi}} = \{\ \}$.

Passo 7.3 – Se ambos os conjuntos estivessem vazios então passaria-se para o Passo 7.6, porém como o conjunto $\bar{\Psi}$ possui elementos, passa-se para o Passo 7.4.

Passo 7.4 – Como $\bar{\bar{\Psi}}$ não possui elementos, então significa que todas as projeções são da mesma classe da classe de busca, e se continua a busca, atualizando θ e κ e passando para o Passo 7.2. No caso $\theta \leftarrow 0.1406$; $\kappa \leftarrow 0.1306$.

Passo 7.2 – Criam-se os conjuntos de projeções $\bar{\Psi}$ e $\bar{\bar{\Psi}}$. Usando a mesma sequência de procedimentos explicada anteriormente.

No caso, $\Psi^* = \{0.0562, 0.1125, -0.0763, -0.0562\}$, $\theta^* = 0.1125$, $\bar{\Psi} = \{\ \}$; $\bar{\bar{\Psi}} = \{0.1125\}$.

Passo 7.3 – Se ambos os conjuntos estivessem vazios então passaria-se para o Passo 7.6, porém como o conjunto $\bar{\bar{\boldsymbol{\Psi}}}$ possui elementos, passa-se para o Passo 7.4.

Passo 7.4 – Se o conjunto $\bar{\boldsymbol{\Psi}}$ tivesse elementos, então significaria que todas as projeções são da mesma classe da classe de busca, e se continuaria a busca, atualizando θ e κ e passando para o Passo 7.2, porém como $\bar{\boldsymbol{\Psi}}$ não possui elementos, passa-se para o Passo 7.5.

Passo 7.5 – Quando este passo é executado significa que encontrou-se uma projeção de classe diferente da classe de busca *E*. Então atualiza-se o limitante inferior *k* e passa-se para o Passo 7.6. No caso limite inferior $\kappa = \overline{\overline{\psi^1}} = 0{,}1125$;

Passo 7.6 - Calcula-se o limitante do hiperplano, ou seja, $\rho = (\theta + \kappa)/2 = (0.1406 + 0.1125)/2 = 0.1265$.

Os valores assumidos pelas variáveis κ, θ e ρ são representados na Figura 3.20.

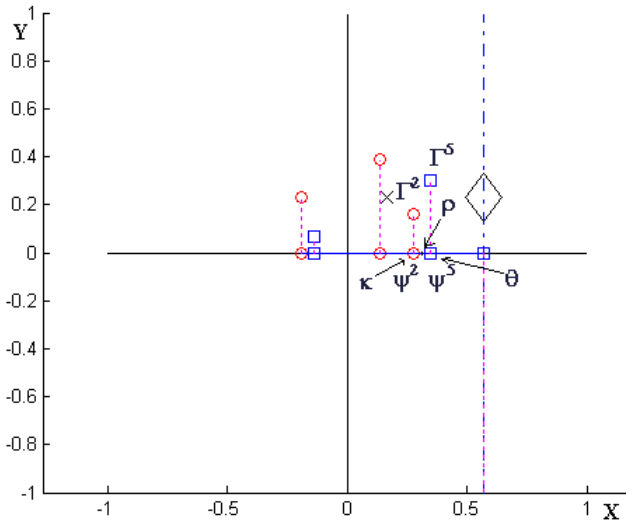


Figura 3.20 - Representação de κ, θ e ρ.

Sempre ao final do Passo 7 é determinado o hiperplano H (Definição 5) a partir do vetor $\boldsymbol{\Phi}$ (Passo 5) e do limitante ρ (Passo 7). Este hiperplano é o separador do subconjunto de classe única igual à classe de busca E. Na Figura 3.21 pode-se ver a representação deste hiperplano e do conjunto de exemplos separados.

Passo 8 – Incrementa-se j e cria-se a regra ω^j utilizando-se o vetor $\boldsymbol{\Phi}$, o limitante ρ e a classe χ^{μ}, conforme Definição 12.

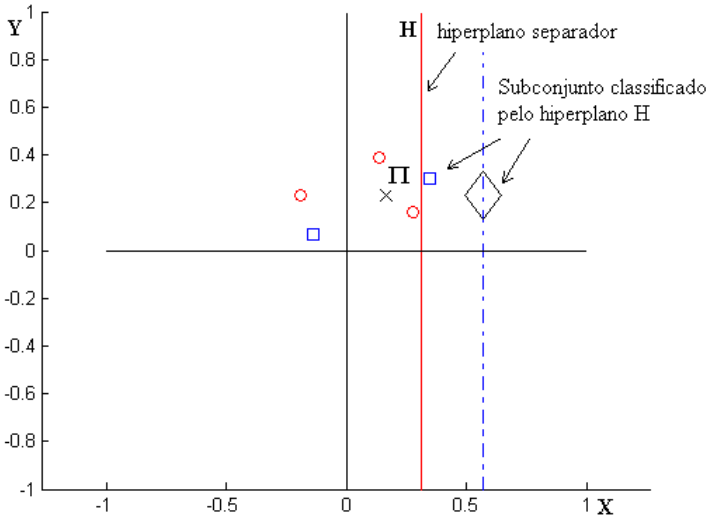


Figura 3.21 - Representação do hiperplano separador H e do subconjunto separado.

Passo 9 - Retira-se do conjunto Λ os exemplos que são separados por H como ilustrado na Figura 3.21.

Passo 10 – Como o conjunto Λ ainda não é vazio, reinicia-se o processo retornando ao Passo 2.

Este processo é repetido mais três vezes gerando mais três regras, ou seja, o número de regras é igual ao número de execuções dos passos 2 ao 10 do Quadro 1 (algoritmo de treinamento do Slicer Básico).

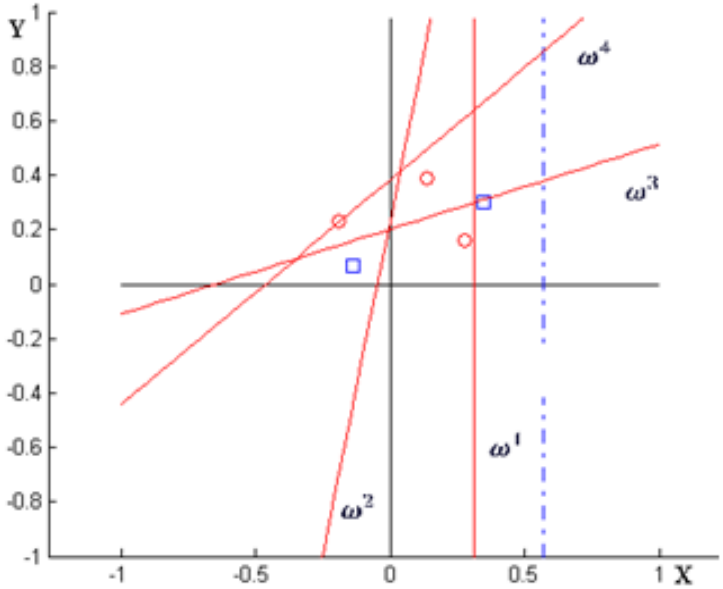


Figura 3.22 - Representação dos hiperplanos gerados pelo Slicer Básico.

Ao final do processo tem-se um conjunto de regras Ω, construído no passo 8 e de acordo com a Definição 12 de formação da regra. Estas regras estão na Tabela 3.12 e são representadas na Figura 3.22 por meio dos respectivos hiperplanos separadores.

Tabela 3.12 – Regras obtidas ao final do aprendizado do Slicer-F.

Regras Ω	Vetor ortogonal T	Limitante α	Classe β
ω^1	(0.4017, 0.0000)	0.1265	2
ω^2	(0.2575, -0.0525)	-0.0121	1
ω^3	(0.0250, -0.0800)	-0.0161	2
ω^4	(-0.1900, 0.2300)	0.0880	1

Nao é possivel comparar os desempenhos da execução do Slicer Básico com a execução do Slicer-F pois é impossível ao Slicer Básico executar sobre bases com atributos faltantes. Por outro lado, o Slicer-F é totalmente equivalente ao Slicer Básico no caso da base de dados não ter atributos faltantes. Conclui-se que o Slicer-F é mais robusto que o Slicer Básico e por isso assume-se que toda implementação posterior estará se utilizando do Slicer-F ao invés do Slicer Básico.

3.4 – Slicer-R

Como o vetor ortogonal obtido pelo Slicer Básico não garante produzir um hiperplano ótimo, investiu-se na pesquisa de uma possível melhora neste vetor. Deste modo pensou-se na rotação do vetor. Para esta rotação são levados em consideração todos os exemplos classificados corretamente pelo hiperplano gerado pelo vetor inicial.

A seguir é apresentada a descrição desta versão, seu algoritmo e uma comparação entre o Slicer Básico e o Slicer-R.

3.4.1 - Descrição do Slicer-R

Considere-se a Figura 3.23, que representa uma possível situação após o térrmino da execução do Passo 7 (Slicer Básico) em um conjunto de exemplos. Estão representados: um conjunto de exemplos e seu respectivo centroide, exemplo mais distante, vetor ortogonal e hiperplano separador.

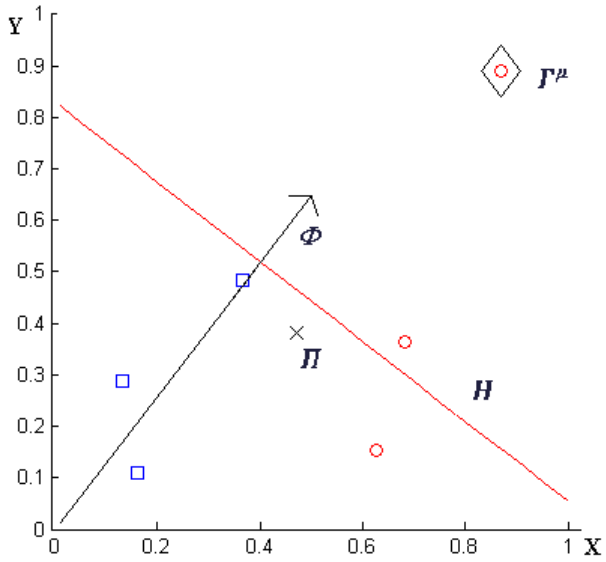


Figura 3.23 - Representação de exemplos (*Λ*), centroide (*Π*), exemplo mais distante (*Γ*$^{\mu}$), vetor ortogonal (*Φ*) e hiperplano separador (*H*).

Esta versão/aprimoramento do Slicer Básico se insere no algoritmo logo após a criação do hiperplano separador H (Passo7). Verificam-se quantos exemplos são corretamente classificados por este vetor e se o número for maior que 1 busca-se um novo vetor ortogonal Φ^* que gere um hiperplano separador H^*. Para a criação do hiperplano H^* deve-se primeiro calcular um novo vetor ortogonal Φ^* que é a rotação de Φ.

Para o cálculo do vetor rotação Φ^*, cada exemplo classificado corretamente por H gera um módulo de vetor, calculado pela diferença do vetor exemplo ao centroide. Em seguida faz-se a média de todos esses módulos de vetores obtidos. Na Figura 3.24 estão representados os dois vetores (Γ^2 e Π) e ($\Gamma^{\mu} - \Pi$) cujos módulos são geradores do vetor rotação Φ^*. Espera-se que esta rotação revele uma tendência de existir mais exemplos de mesma classe se feito um corte do espaço nesta nova direção.

Na Figura 3.24 os vetores são representados fora de escala, possibilitando uma melhor apreciação das características relativas entre tais entidades.

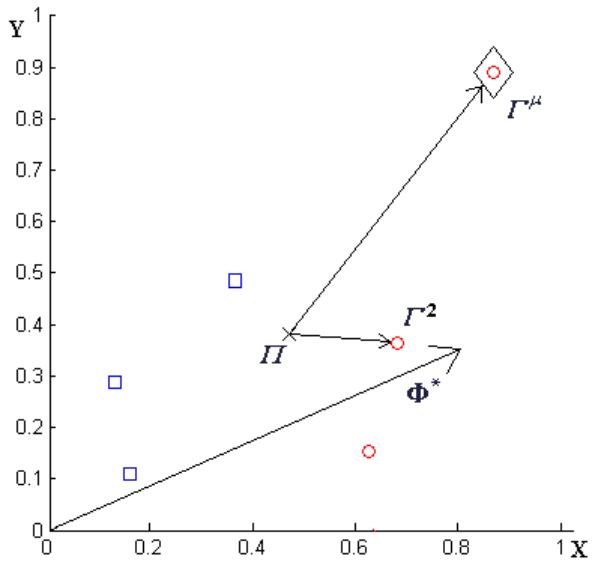


Figura 3.24 - Representação qualitativa dos vetores (Γ^2 e Π) e ($\Gamma^{\mu} - \Pi$) geradores do vetor Φ^*.

Se o hiperplano H^* obtido pelo vetor Φ^* classificar, corretamente, mais exemplos que H, então o substitui, senão é descartado. Como pode-se notar na Figura 3.25 o hiperplano H^* classifica 3 exemplos, contra 2 separados pelo hiperplano H, logo os argumentos Φ^* e ρ^* substituem Φ e ρ criados anteriormente.

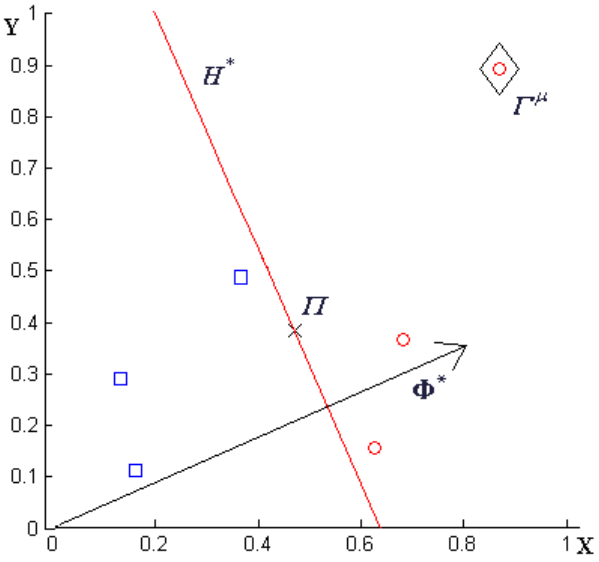


Figura 3.25 - Representação de exemplos (Λ), centroide (Π), exemplo mais distante (Γ^{μ}), vetor ortogonal (Φ^*) e hiperplano separador (H^*).

Slicer-R - Algoritmo de Aprendizado

Entrada: conjunto de exemplos Λ.

Saída: conjunto de regras Ω.

Passo 1 – Faça j = 0 .

Passo 2 - Calcula-se o centroide Π do conjunto de exemplos Λ.

Passo 3 – Calcula-se o conjunto de distâncias Euclidianas Δ.

Passo 4 – Localiza-se o índice μ do exemplo mais distante do centroide Π.

Passo 5 – Calcula-se vetor ortogonal Φ.

Passo 6 – Calculam-se o conjunto de projeções Ψ e margem ε.

Passo 7 – Busca do limitante:

Passo 7.1- Faça: classe de busca *E* igual a classe de λ^{μ}, $E = \chi^{\mu}$;

limite superior θ igual a Max(Ψ), $\theta = max(\Psi)$;

limite inferior κ igual a θ menos ε, $\kappa = \theta - \varepsilon$.

Passo 7.2 - Busca os conjuntos de projeções $\overline{\Psi}$ e $\overline{\overline{\Psi}}$.

Passo 7.3 - Se $|\overline{\Psi}| = 0$ e $|\overline{\overline{\Psi}}| = 0$, ou seja, se não há mais projeções a considerar, vá para o Passo 7.6.

Passo 7.4 – Se $|\overline{\overline{\Psi}}| = 0$, ou seja, todas as projeções são da mesma classe que a classe da busca, então:

Faça: limite superior θ igual ao primeiro elemento de $\overline{\Psi}$, $\theta = \bar{\psi}^1$; e

limite inferior κ igual a θ menos ε, $\kappa = \theta - \varepsilon$. Vá para o Passo 7.2.

Passo 7.5 - Faça limite inferior $\kappa = \bar{\bar{\psi}}^1$ e vá para o Passo 7.6.

Passo 7.6 - Faça limitante $\rho = (\theta + \kappa)/2$.

Passo 8 – Determina-se o número nroclas de exemplos que são corretamente classificados por este hiperplano:

$nroclas = |\{\Gamma^g | (< \Phi, \Gamma^g >) \geq \rho, \chi^{\mu} = \chi^g, 1 \leq g \leq N\}|$.

Se $nroclas = 1$, ou seja, só 1 exemplo foi separado corretamente, não há exemplos para se produzir uma rotação. Vá para Passo 14.

Passo 9 – Cria-se um novo vetor ortogonal Φ^*:

Passo 9.1 - Calculam-se os módulos dos vetores das diferenças entre cada vetor classificado corretamente e o centroide. Tira-se a média de todos estes módulos. Esse novo vetor é o vetor rotação. Ou seja:

Quadro 8 – Algoritmo do Slicer-R.

$$\Phi^* = \sum \|\Gamma^g - \Pi\| / nroclas\,, \forall\, \Gamma^g \in \{\Gamma^g | (< \Gamma^g, \Phi >\geq \rho), \chi^\mu = \chi^g, 1 \leq g \leq N\}.$$

Passo 10 – Calculam-se o novo conjunto de projeções Ψ e ε.

Passo 11 – Busca do limitante ρ^*:

Passo 11.1- Faça: classe de busca *E* igual a classe de λ^μ, $E = \chi^\mu$;

limite superior θ igual a Max(Ψ), $\theta = max(\Psi)$;

limite inferior κ igual a θ menos ε, $\kappa = \theta - \varepsilon$.

Passo 11.2 - Busca dos conjuntos de projeções $\overline{\Psi}$ e $\overline{\overline{\Psi}}$.

Passo 11.3 - Se $|\overline{\Psi}| = 0$ e $\left|\overline{\overline{\Psi}}\right| = 0$, ou seja, se não há mais projeções a considerar, vá para o Passo 11.6.

Passo 11.4 – Se $\left|\overline{\overline{\Psi}}\right| = 0$, ou seja, todas as projeções são da mesma classe que a classe da busca, então:

Faça: limite superior θ igual ao primeiro elemento de $\overline{\Psi}$, $\theta = \bar{\psi}^1$; e

limite inferior κ igual a θ menos ε, $\kappa = \theta - \varepsilon$. Vá para o Passo 11.2.

Passo 11.5 - Faça limite inferior $\kappa = \bar{\bar{\psi}}^1$ e vá para o Passo 11.6.

Passo 11.6 - Faça limitante $\rho^* = (\theta + \kappa)/2$.

Passo 12 - Determina-se o número $nroclas^*$ de exemplos que são corretamente classificados por este hiperplano:

$nroclas^* = |\{\Gamma^g | (< \Phi^*, \Gamma^g >) \geq \rho^*, \chi^\mu = \chi^g, 1 \leq g \leq N\}|$.

Se $nroclas^* < nroclas$, ou seja, o novo hiperplano classifica corretamente um número menor de exemplos que o anterior, vá para o Passo 14.

Passo 13 – Caso o novo hiperplano classifique corretamente um número maior de exemplos que o anterior, então substitui-se o vetor e o limitante anteriores pelos calculados pela rotação, ou seja:

$\Phi \leftarrow \Phi^*$ e $\rho \leftarrow \rho^*$.

Passo 14 – Incrementa-se j e cria-se a regra ω^j, baseado no vetor ortogonal Φ, limitante ρ e classe χ^μ.

Passo 15 - Excluem-se do conjunto de exemplos Λ todos os exemplos que possuem projeção maior que o limitante, ou seja, exclui-se todos os exemplos que pertençam à classe χ^μ: $\Lambda \leftarrow \Lambda - \{\lambda^g | H(\lambda^g) \geq 0; 1 \leq g \leq N\}$.

Passo 16 - Caso o conjunto de exemplos Λ não esteja vazio, volta-se ao passo 2; senão finaliza-se o algoritmo.

Quadro 9 – continuação do algoritmo do Slicer-R.

Na Figura 3.26 pode-se ver claramente a rotação feita no vetor $\boldsymbol{\theta}$ e a consequente melhora no desempenho do hiperplano H.

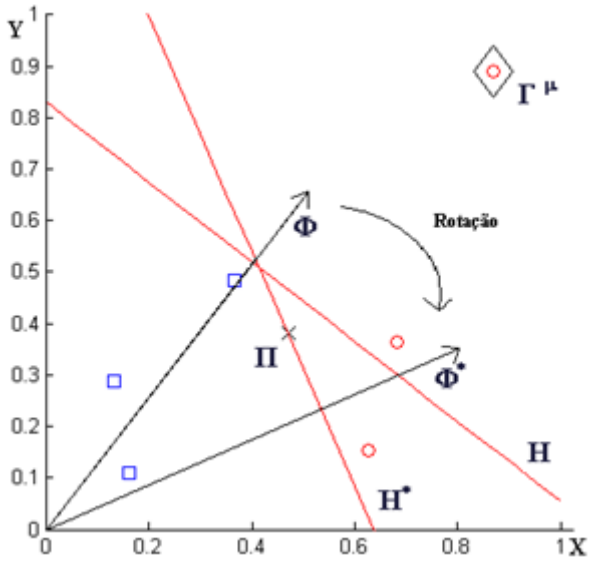
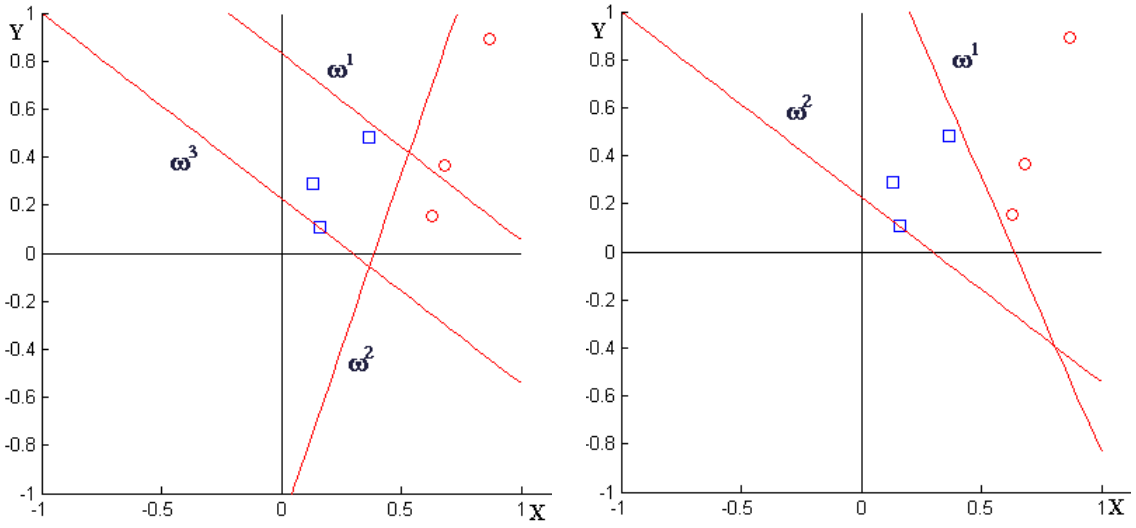


Figura 3.26 - Representação simbólica dos vetores Φ e Φ^*, dos respectivos hiperplanos H e H^* e o sentido da rotação.

3.4.2- Considerações sobre o Slicer-R

Pode-se ver na comparação do conhecimento obtido pelo Slicer Básico e pelo Slicer-R sobre a mesma base de dados, mostrado na Figura 3.27, que realmente é possível se obter um conjunto de hiperplanos mais compacto usando Slicer-R.



(a) (b)

Figura 3.27 – Conjunto de hiperplanos obtidos pela execução de:
(a) Slicer Básico; (b) Slicer-R.

Observe que os hiperplanos ω^1 e ω^2 que separam exemplos do tipo "bolinha" obtidos pelo Slicer Básico foram substituídos pelo hiperplano ω^1 (com as mesmas funções dos anteriores).

3.5 – Slicer-G

Algumas bases de dados possuem ruídos ou mesmo erros nos dados, podendo causar perda de desempenho do sistema classificador Slicer.

Para tentar minimizar tal perda foi criada a versão Slicer-G. "Gap" é um intervalo entre projeções composto de classes diferentes da classe de seus extremos. O uso da identificação e descarte de "gaps", permite uma maior generalização do método Slicer, fazendo com que, na geração de regras, seja possível identificar e descartar algum nível de ruído.

A seguir é apresentada a descrição desta versão, seu algoritmo e uma comparação entre o Slicer Básico e o Slicer-G.

3.5.1 - Descrição do Slicer-G

Considere a Figura 3.28 e note-se um conjunto ordenado de projeções, representado pelo segmento de reta inclinado e passando pela origem. As projeções estão em ordem crescente de ψ^3 para ψ^4 e a classe associada a cada uma das projeções está indicada pelo símbolo utilizado, ou seja, quadrado indica classe 1 e círculo indica classe 2. Nota-se ainda que o vetor Γ^3 é o ponto mais distante do centroide. Note-se que a reta de projeções pode conter mais de uma projeção com mesmo valor, e estas projeções podem estar associadas a classes diferentes.

Neste cenário, gap pode ser entendido como as projeções ψ^1 e ψ^2, ambas de classe diferente de ψ^3 e compõem um conjunto envolvido por projeções de classe igual a classe de ψ^3. Observe-se que a relação entre as projeções é consequência direta da relação entre o conjunto de exemplos que as mesmas representam, ou seja, os exemplos representados por ψ^1 e ψ^2são também envolvidos espacialmente por exemplos de ψ^3.

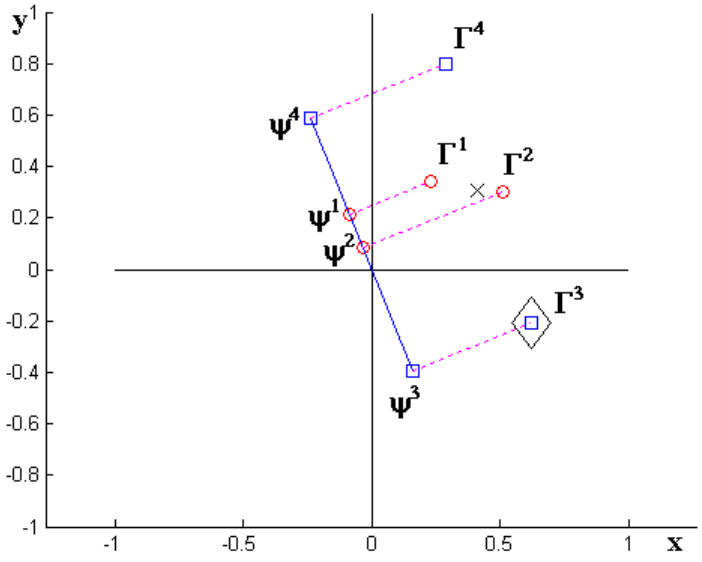


Figura 3.28 - Conjunto ordenado das projeções Ψ.

O Slicer-G difere do Slicer Básico somente no tocante à busca do limitante (Passo 7) e na passagem de parâmetros.

A idéia básica concentra-se na análise de transições de classes associadas às projeções presentes sobre o vetor ortogonal que determina o hiperplano. Iniciando por uma projeção base, as projeções são visitadas em ordem decrescente em termos de suas classes.

Dependendo do tipo de transição, ou seja, de acordo com a cardinalidade do conjunto de projeções associado à transição de classes o algoritmo decide pelo descarte de tal conjunto quando do cálculo do hiperplano.

Note-se que os algoritmos Slicer Básico e Slicer-G só diferem na passagem de parâmetros e na descrição do Passo 7. Inclusive, considerando-se o caso em que os parâmetros nrogap e tamgap são iguais a zero, tais algoritmos tornam-se idênticos. Isso possibilita a inclusão de novas características ao sistema Slicer por meio da passagem de parâmetros.

Slicer-G - Algoritmo de Aprendizado

Entrada: conjunto de exemplos Λ, nrogap, tamgap.

Saída: conjunto de regras Ω.

Passo 1 – Faça j = 0 .

Passo 2 - Calcula-se o centroide Π do conjunto de exemplos Λ.

Passo 3 – Calcula-se o conjunto de distâncias Euclidianas Δ.

Passo 4 – Localiza-se o índice μ do exemplo mais distante do centroide Π.

Passo 5 – Calcula-se vetor ortogonal Φ.

Passo 6 – Calculam-se o conjunto de projeções Ψ e ε.

Passo 7 – Busca do limitante com gap:

Passo 7.1- Faça: classe de busca *E* igual a classe de λ^{μ}, $E = \chi^{\mu}$;

limite superior θ igual a Max(Ψ), $\theta = max(\Psi)$;

limite inferior κ igual a θ menos ε, $\kappa = \theta - \varepsilon$.

$ng = nrogap$, $tg = tamgap$, $ga = 0$, $gf = 0$.

Passo 7.2 - Busca os conjuntos de projeções $\overline{\Psi}$ e $\overline{\overline{\Psi}}$.

Passo 7.3 - Se $|\overline{\Psi}| = 0$ e $|\overline{\overline{\Psi}}| = 0$, ou seja, se não há mais projeções a considerar, vá para o Passo 7.11.

Passo 7.4 – Se $|\overline{\overline{\Psi}}| = 0$, ou seja, todas as projeções são da mesma classe que a classe da busca, então:

Faça: limite superior θ igual ao primeiro elemento de $\overline{\Psi}$, $\theta = \bar{\psi}^1$;

limite inferior κ igual a θ menos ε, $\kappa = \theta - \varepsilon$;

ga=1; gf=0 ; vá para o Passo 7.2.

Quadro 10 – Algoritmo do Slicer-G.

Passo 7.5 - Se $|\overline{\Psi}| > 0$, faça gf=1.

Passo 7.6 - Se $ga = 0$ então

Faça: Flag de gap aberto $ga = 1$;

Número de gaps ainda admitidos $ng = ng - 1$;

Tamanho admitido para o gap aberto $tg = tamgap$;

Limite inferior igual ao primeiro elemento de $\overline{\overline{\Psi}}$, $\kappa = \bar{\bar{\psi}}^1$.

Passo 7.7 - Se $ng < 0$, vá para o Passo 7.11. (excedeu limite de gaps)

Passo 7.8 - Se $tg < |\overline{\overline{\Psi}}|$ então vá para o Passo 7.11. (excedeu tamanho máximo gap)

Passo 7.9 - Se $gf = 1$ então faça $gf = 0$; $ga = 0$; $\theta = \bar{\psi}^1$; $tg = 0$; $\kappa = \theta - \varepsilon$; vá para o Passo 7.2. (gap fechado).

Passo 7.10 - Faça $tg = tg - |\overline{\overline{\Psi}}|$ e vá para o Passo 7.2. (abre gap).

Passo 7.11 - Faça limitante $\rho = (\theta + \kappa)/2$.

Passo 8 – Incrementa-se j e cria-se a regra ω^j, baseado no vetor ortogonal Φ, limitante ρ e classe χ^{μ}.

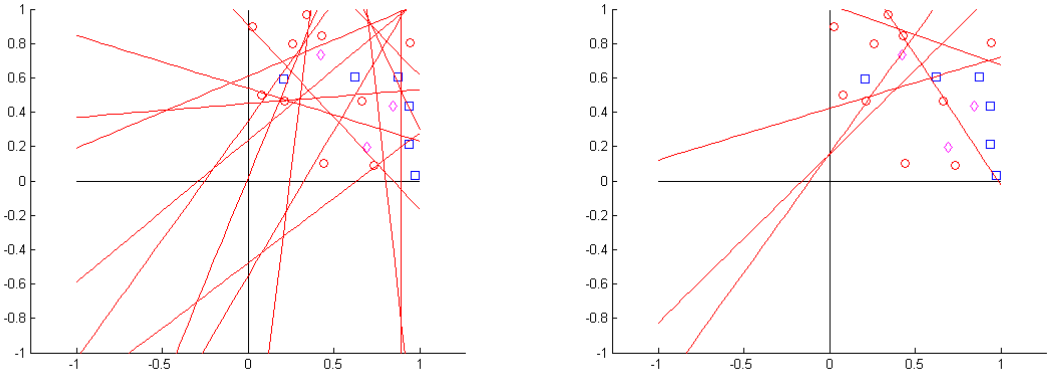
Passo 9 - Exclui-se do conjunto de exemplos Λ todos os exemplos que possuem projeção maior que o limitante: $\Lambda \leftarrow \Lambda - \{\lambda^g | H(\lambda^g) \geq 0; 1 \leq g \leq N\}$.

Passo 10 - Caso o conjunto de exemplos Λ não esteja vazio, volta-se ao Passo 2; senão finaliza-se o algoritmo.

Quadro 11 – (continuação) Algoritmo do Slicer-G.

3.5.2 - Considerações sobre o Slicer-G

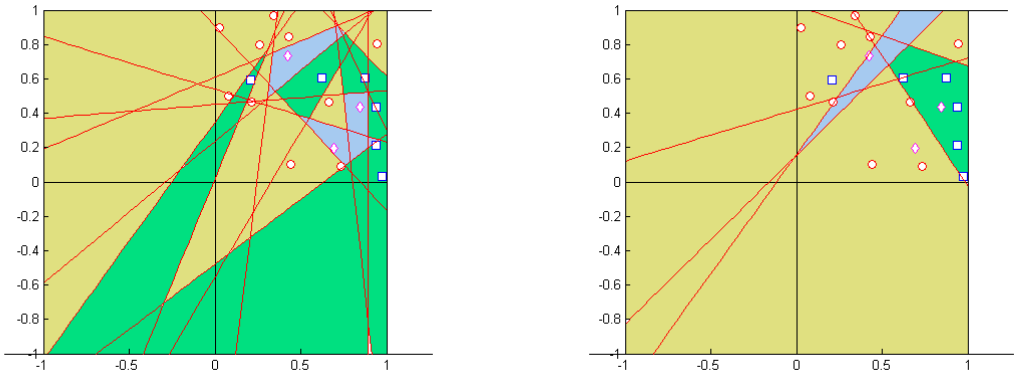
Na Figura 3.29 estão representados os conjuntos de regras produzidos pelo Slicer Básico e pelo Slicer-G sobre a mesma base de treinamento. O Slicer-G usou como parâmetros: nrogap=1 e tamgap=1, ou seja, admite um subconjunto de ruídos e com o máximo de 1 elemento. Pode-se notar que existe uma diferença significativa entre os números de hiperplanos gerados: 5, no Slicer-G e 13, no Slicer Básico. Benefícios do Slicer-G: localização e descarte de ruídos; redução de custos computacionais na fase de classificação.



(a) (b)

Figura 3.29 – Conjunto de hiperplanos obtidos pela execução de:
(a) Slicer Básico; (b) Slicer-G.

Na Figura 3.30 pode-se notar as divisões do espaço geradas pelos Slicer Básico e Slicer-G. Percebe-se que o Slicer Básico possui uma distribuição muito mais complexa em relação à obtida pelo Slicer-G.



(a) (b)

Figura 3.30 – Divisões do espaço gerados pelo:
(a) Slicer Básico; (b) Slicer-G.

3.6 - Considerações finais sobre o método Slicer e suas versões

O método Slicer Básico cria, recursivamente, em sua fase de treinamento, por meio de centroide e do exemplo mais distante, um conjunto de hiperplanos com classe

associada. Este conjunto posteriormente é utilizado para se fazer classificação de novos vetores de atributos através da aplicação de regras se-então-senão.

Este método tem custo computacional da ordem de $O(N^2)$ e gera uma classificação de explicação fácil e compreensível.

São várias as versões apresentadas, cada qual com características distintas, porém complementares: Slicer-F, Slicer-R, Slicer-G e Slicer-O. Tais características resumidamente são:

- Slicer-F: admite bases de dados com atributos faltantes, por meio de adaptações em vários processos, deixando o método mais robusto e com espectro de uso ampliado.
- Slicer-R: rotaciona os vetores geradores dos hiperplanos, permitindo tanto a melhora do desempenho destes hiperplanos como a redução da cardinalidade do conjunto final de hiperplanos.
- Slicer-G: admite ruídos nas bases de dados e consegue aumentar a generalização dos hiperplanos, por meio do uso de gaps, também conseguindo redução no número final de hiperplanos gerados.
- Slicer-O: atua sobre o conjunto de hiperplanos produzido pelo Slicer, reordenando-o e reduzindo sua cardinalidade final.

Como pode-se ver, as versões do Slicer buscam sempre uma melhora no desempenho e redução da cardinalidade do conjunto de hiperplanos gerado.

O Slicer é caracterizado como sistema classificador, pois admite opções feitas pelo usuário na sua execução, tal como mostra a Figura 3.31. Nela pode-se ver que a versão Slicer-F encampa o Slicer Básico e é permitido ao usuário optar entre o treinamento usando essa versão Slicer-F (emcampando a versão Slicer Básico), Slicer-F + Slicer-R, Slicer-F + Slicer-G, Slicer-F + Slicer-R + Slicer-G. Quando do uso da opção Slicer-G, nas duas situações, permite-se definir qual o número máximo de gaps admitido e qual o tamanho máximo para cada um deles.

Ao término desta fase de treinamento, um conjunto de hiperplanos está bem definido para uso no processo de classificação.

Este mesmo conjunto é então submetido ao Slicer-O e obtém-se um novo conjunto de hiperplanos, mais compacto, o qual também é utilizado no processo de classificação.

As contribuições apresentadas neste capítulo podem ser comentadas sobre perspectivas distintas. De uma forma geral descreve-se um Sistema Classificador consistindo em um sistema de aprendizagem e um sistema de classificação.

O Sistema Classificador Slicer, tal como representado na Figura 3.31, possui 2 subsistemas:

1) entrada: Λ (conjunto de treinamento), processado pelo motor de aprendizagem de máquina, saída: conjunto de regras;

2) entrada: vetor de atributos e regras, processado pelo motor de classificação, saída: classe correspondente.

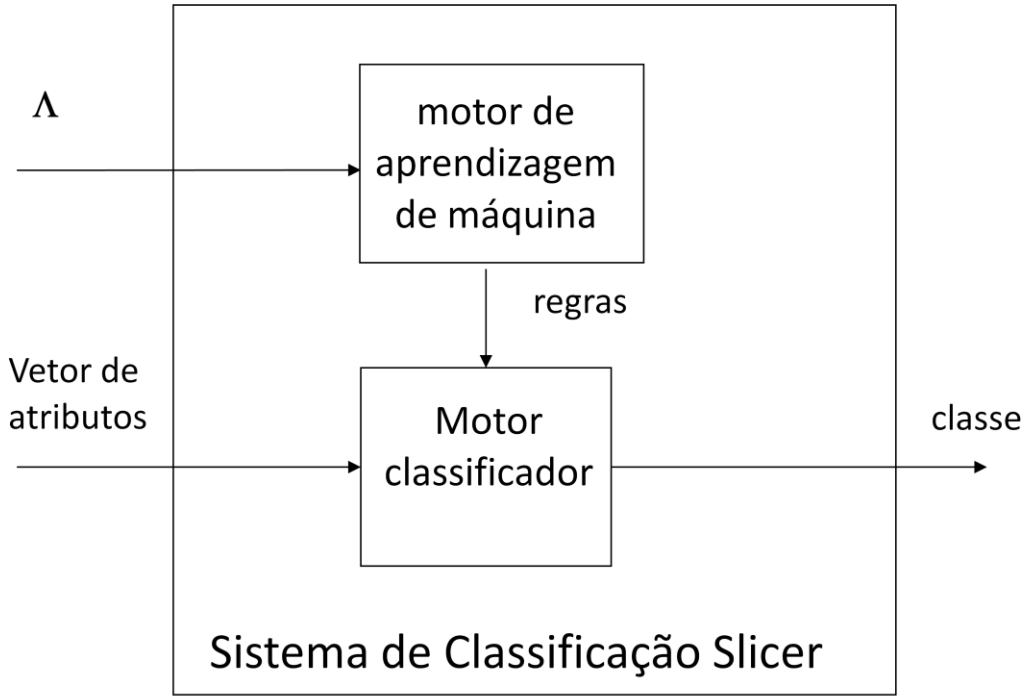


Figura 3.31- Arquitetura do Sistema Classificador Slicer.

O sistema de aprendizagem (Figura 3.32) consiste de:

- Escolha das versões do Slicer que participam do treinamento, no módulo de aprendizagem de máquina (motor de aprendizado), sendo que o Slicer Básico mais o Slicer-F são uma constante;
- Módulo de aprendizagem de máquina (motor de aprendizado) contendo as versões do Slicer Básico, Slicer-F, Slicer-G e Slicer-R;
- Conjunto de exemplos (**Λ**) dado como entrada para treinamento;
- Conjunto REGRAS obtido como saída pelo módulo MA;

- Módulo de otimização Slicer-O;
- Conjunto REGRAS-O obtido como saída do Slicer-O.

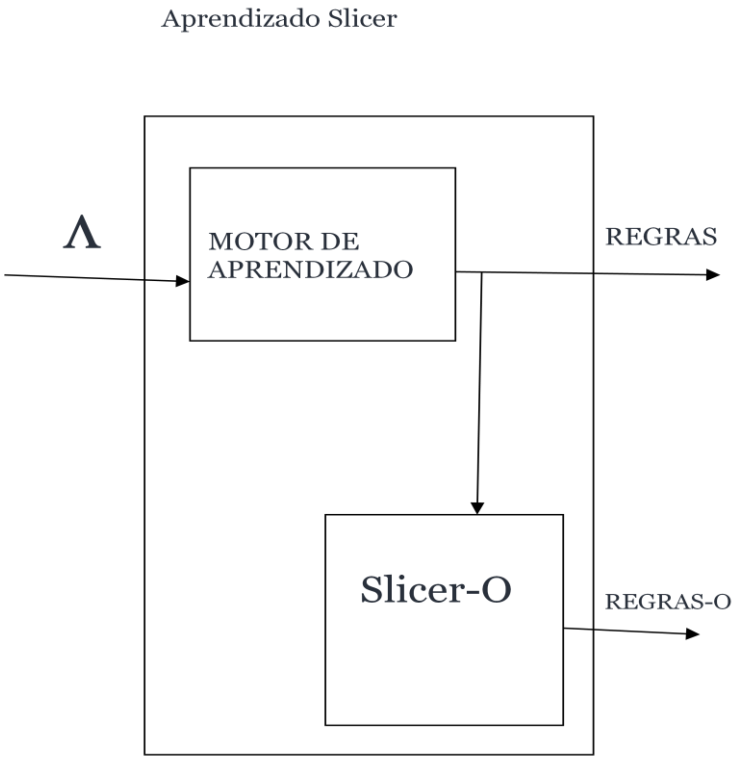


Figura 3.32 – Arquitetura do sistema de aprendizagem Slicer.

O sistema de classificação Slicer (Figura 3.33) consiste de:

- Escolha do conjunto de regras a ser utilizado, REGRAS ou REGRAS-O;
- Motor de classificação se-então-senão;
- Vetor de atributos a ser classificado ;
- Classe determinada.

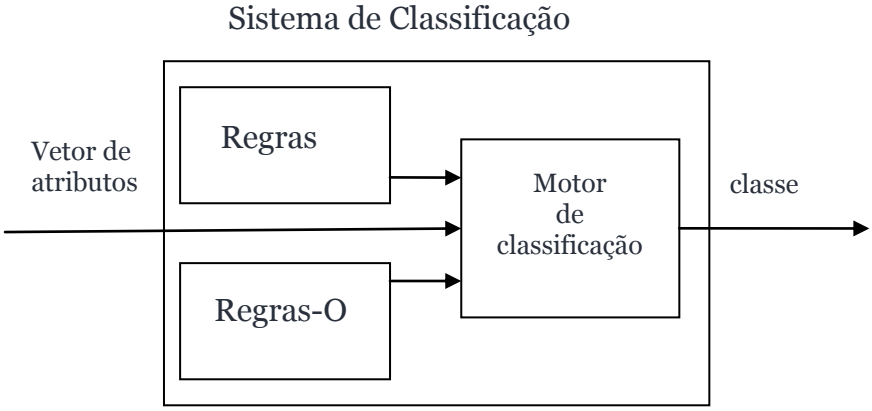


Figura 3.33 - Arquitetura do Sistema de Classificação Slicer.

As principais características do método de aprendizagem utilizado no Slicer Básico são:

- geométrico;
- término do treinamento em no máximo N iterações;
- supervisionado, indutivo e multiclasse;
- custo computacional $O(N^2)$.

Espera-se obter todos os benefícios do Slicer e suas versões em uma única execução, gerando-se um conjunto de hiperplanos compacto e eficiente.

Todos os conjuntos de regras obtidos do Slicer, em várias bases de avaliação, são utilizados como variáveis de entrada para um sistema fuzzy baseado em regras (SFBR) descrito no próximo capítulo.

Capítulo 4

Descrição do Método Slicer-Nebuloso

Uma das principais características do Slicer é a garantia de que seu tempo de treinamento só depende do número de exemplos da base de dados do domínio para ter seu término garantido e em tempo de classificação fornece uma possibilidade de visualização dos espaços classificatórios. O Slicer é um sistema que gera um conjunto de regras determinísticas, que dividem o espaço em subespaços bem definidos e associados a classes distintas. Além disso, suas versões possibilitam tratamento de atributos faltantes (Slicer-F) e ruídos (Slicer-G), além de permitir um aprendizado mais eficaz com uma melhor escolha de parâmetros (Slicer-R) e redução da base de conhecimento (Slicer-O).

Por outro lado, identificam-se nos próprios algoritmos de aprendizagem e classificação deficiências relevantes: 1) algoritmo de classificação associa uma das classes sem estabelecer uma transição entre os subespaços de cada classe; e 2) caso nenhum subespaço atenda à classificação, utiliza-se da classe da última regra, independente da proximidade de outros subespaços.

Para tentar transpor estas deficiências, investigou-se a criação de um sistema híbrido. Sistemas híbridos são comuns na literatura e buscam unir métodos de diferentes naturezas, reforçando as boas características e tentando eliminar deficiências de seus participantes.

O algoritmo resultante apresenta as seguintes características principais: 1) considera o conjunto de regras no todo e não sequencialmente; e 2) quando o exemplo está posicionado em região sem classe associada, a classsificação se dá por proximidade dos outros subsespaços.

Note-se que nesta proposta muda-se o paradigma de análise das regras para classificação. De hierárquico, no sistema Slicer, em que o processo equivale a uma busca iterativa pelo conjunto de regras até achar a regra classificadora, para geral, no sistema Slicer-Nebuloso, em que todas são verificadas simultaneamente.

Mesmo assim, métodos nebulosos sofrem do problema de situações em que o exemplo a ser classsificado não corresponde a nenhuma das regras definidas, quando

então se lança mão de uma classe a ser utilizada como padrão. Na proposta a ser explanada neste capítulo, o método supre esta dificuldade determinando a classe usando a distância do vetor de atributos aos subespaços definidos pelas regras nebulosas.

Nas próximas seções são explicados conceitos básicos da lógica nebulosa e de Sistemas Fuzzy de Classificação Baseados em Regras (SFCBR), feita uma breve descrição do método Wang-Mendel (WM) de geração de regras nebulosas e a seguir dada a descrição das fases de treinamento e classificação do método Slicer-Nebuloso, terminando com as considerações finais.

Durante este capítulo adotam-se conceitos, notações e termos utilizados em [Klir & Yuan 1995], [Pedrycz 1998] e [Pedrycz & Gomide 1998].

4.1 - Descrição do Método Slicer-Nebuloso

Basicamente o que se faz é transformar as entradas de treinamento, na forma de vetor de atributos, em vetor de produtos internos entre os vetores ortogonais, (representados por cada regra) e o vetor de atributos. Além disso, deve-se utilizar os limitantes das regras como marcador intermediário das partições.

Na hibridização proposta, utiliza-se o conjunto de regras gerado pelo Slicer, em qualquer das versões, mas sempre com Slicer-O, como sendo o conjunto de variáveis de entrada de um Sistema Fuzzy de Classificação Baseado em Regras (SFCBR). Mais especificamente, utiliza-se o vetor ortogonal indicado nas regras do Slicer, como sendo o gerador das variáveis de entrada do sistema nebuloso; e o limitante como sendo o ponto intermediário de suas partições.

4.1.1 - Fase treinamento

Nesta seção explica-se como os fundamentos de um SFCBR são aplicados na versão híbrida Slicer-Nebuloso.

Na versão Slicer-Nebuloso, os universos a serem particionados são os conjuntos das projeções de cada exemplo. Ou seja, cada regra do Slicer indica um vetor ortogonal e as projeções sobre cada um destes vetores são as entradas do Sistema Nebuloso.

Tome-se para ilustração o conjunto de exemplos da seção 3.1.1 e observe o vetor Φ indicado na Figura 3.5. Também considere a Figura 3.6 em que se notam todas as

projeções dos exemplos sobre o vetor Φ e a representação do limitante localizado, na Figura 3.7.

Este vetor está representado na reta, crescendo da esquerda para a direita, e particionado em 3 conforme indicado na Figura 4.1. Nesta figura vê-se que os valores das projeções mínima e máxima na reta são os extremos do intervalo do conjunto a ser particionado; e que o limitante é o ponto considerado como ponto central desta partição, sendo usado como referência para se nomear as partições.

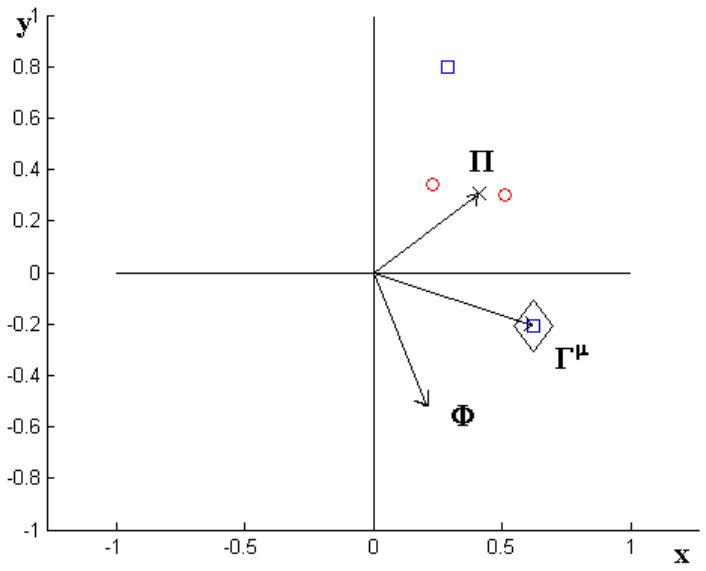


Figura 3.5 – Vetor Φ ($\Phi = \Gamma^{\mu} - \Pi$).

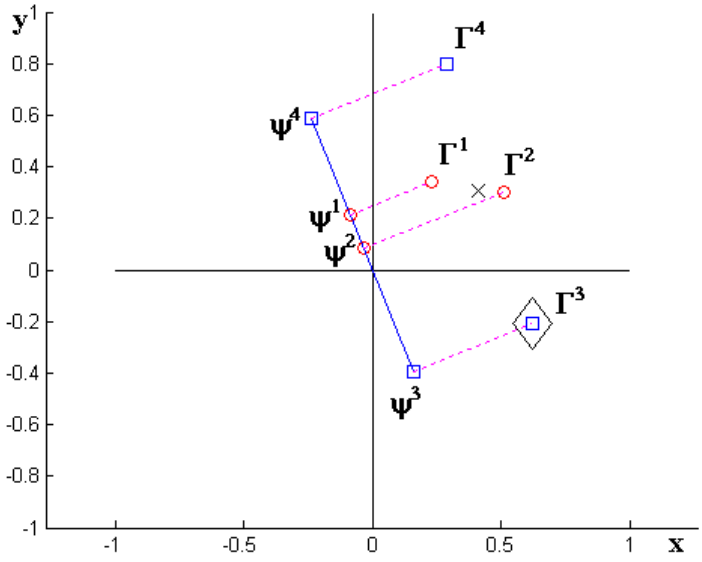


Figura 3.6 - Conjunto ordenado das projeções Ψ.

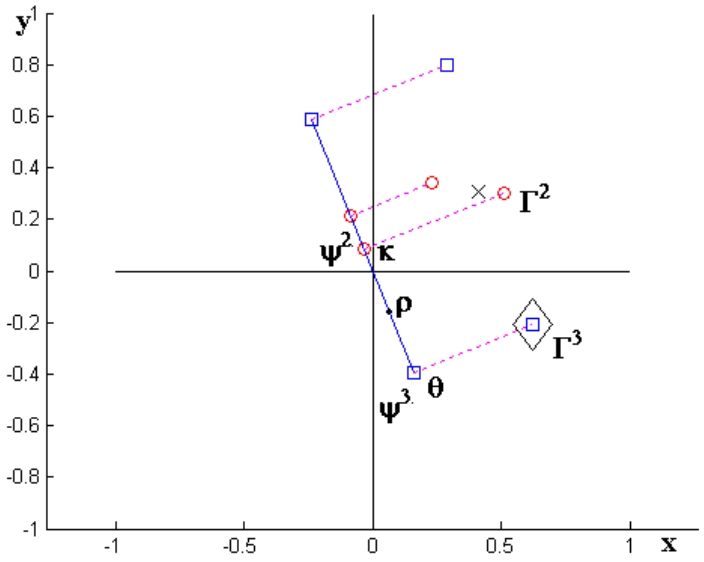


Figura 3.7 - Representação de κ, θ e ρ.

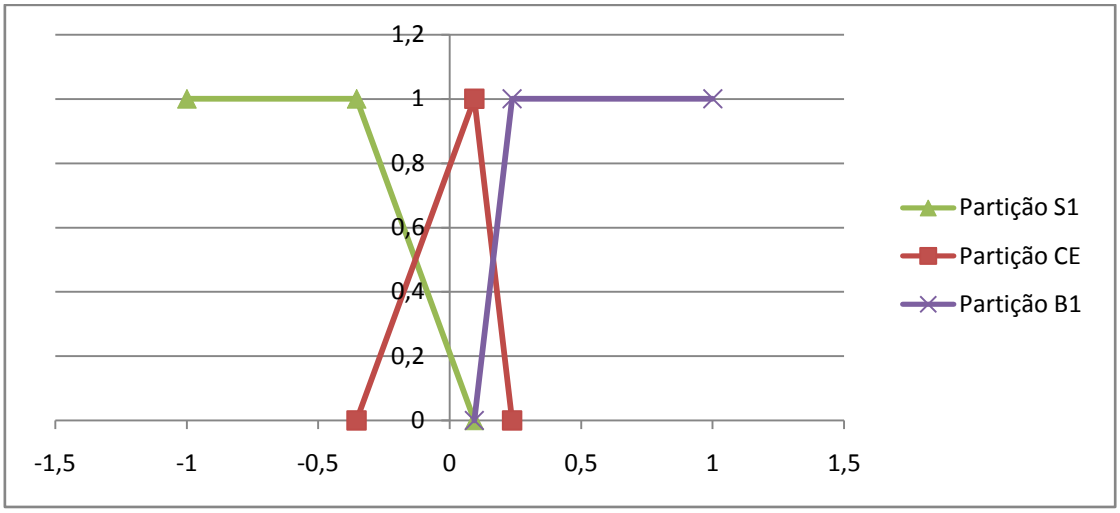


Figura 4.1 – Representação do vetor de projeções particionado.

No caso deste trabalho, as entradas que geram partições na verdade são as regras produzidas pelo Slicer, logo os valores máximo e mínimo são as projeções produzidas pelos exemplos utilizando estas regras.

Referências ao lado direito da partição indicam as partições que acontecem do lado direito do ponto máximo de CE e ao lado esquerdo, as que acontecem do lado esquerdo deste marcador.

Na Figura 4.2 pode ser observada a arquitetura de geração da BD e da BR, utilizando: 1) os exemplos projetados (gerados pelo Gerador de Projeções) e o algoritmo criaparticoes (Gerador de Partições) para a geração da BD; e 2) os exemplos projetados, da BD e do método WM para a geração da BR.

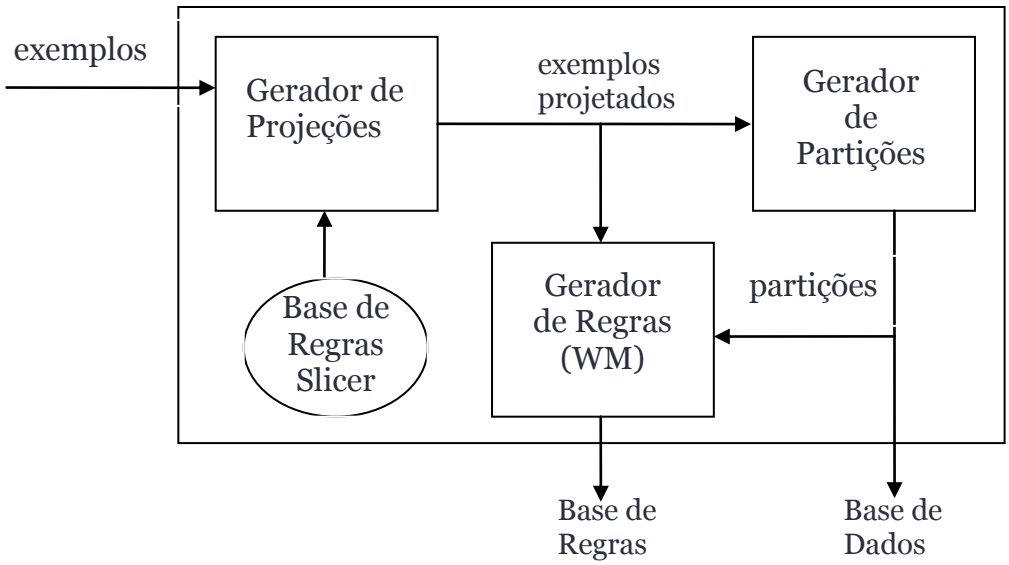


Figura 4.2 – Arquitetura da geração de BD e BR proposta.

A arquitetura do SFCBR tem como entradas para seu motor de inferência o exemplo projetado e as BD e BR, produzindo a classe determinada (Figura 4.5).

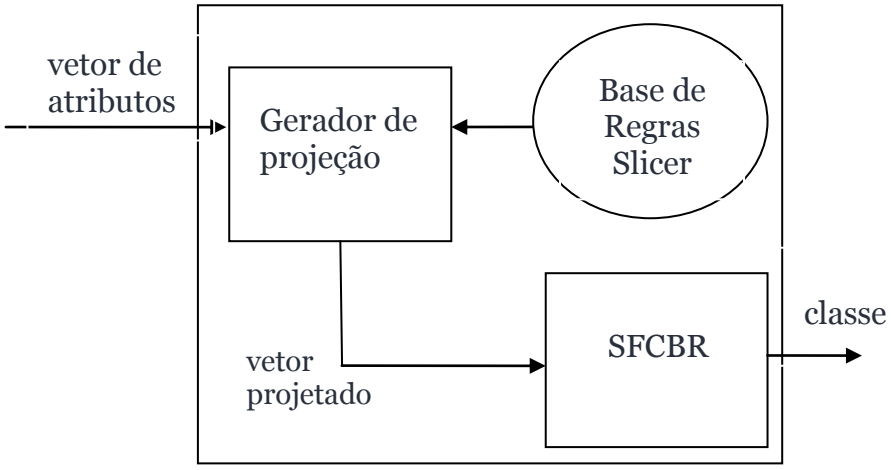


Figura 4.3 – Arquitetura do SFCBR proposto.

Note-se que a arquitetura do SFCBR tal como mostra a Figura 4.2. Neste trabalho a saída não necessita ser "defuzificada", ou seja, não há necessidade de ID.

A BD possui as informações a respeito das partições utilizadas no processo e é gerada no processo de aprendizagem. Neste trabalho é gerado no módulo *geraparticoes*. Recebe como entrada o conjunto de exemplos projetados e tem como saída o conjunto de partições DESCPART.

A BR contém todas as regras de inferência geradas para este problema. Neste

trabalho são geradas pelo método Wang-Mendel e têm como entrada o conjunto de exemplos projetados e o conjunto de partições (BD).

A MI se utiliza da BC para inferir a classe de um vetor de atributos. Calcula para este vetor de atributos os graus de relevância de todas as regras em BR, utilizando as partições de BD. Usa-se costumeiramente a regra vencedora como classificação. Em caso de empate utiliza-se a distância de partições como segundo critério.

Neste trabalho, os exemplos dados associados ao problema de classificação são mapeados por meio do produto interno dos vetores de seus elementos pelos vetores das regras vindas do Slicer (algoritmos *geraprojecoes e geraprojecao*), executado nos componentes Gerador de Projeções e Gerador de Projeção (Figura 4.2 e Figura 4.3).

O algoritmo de treinamento ou aquisição de conhecimento nebuloso engloba os algoritmos *geraprojecoes*, *geraparticao* e WM. O algoritmo de classificação engloba *geraprojecao* e *MotordeInferencia*.

4.1.1.1 - Modelagem do conhecimento

Nesta subseção apresenta-se o algoritmo de representação de conhecimento em regras nebulosas.

As operações descritas no Quadro 12 são brevemente descritas a seguir. São a base para a execução demonstrativa apresentada após o algoritmo. Os algoritmos auxiliares à compreensão estão expostos a seguir.

Para cada elemento do conjunto de exemplos, calcula-se o vetor de projeções equivalentes (Passo 1).

Localiza-se a menor e maior projeção (projmin e projmax) para cada regra Slicer. Cria as projeções intermediárias (projmed) por meio dos limitantes das regras Slicer. Baseados nestas informações (projmin, projmax e projmed) e no número de partições desejado cria-se o conjunto de descrição de partições DESCPART (Passo 2).

Usa-se o método WM para criar o conjunto de regras nebulosas tendo com entrada para cada exemplo um vetor de projeções usando as regras Slicer. Este conjunto é o conjunto final obtido (Passo 3). O algoritmo do método Wang-Mendel está no Quadro 13.

SLICER-NEBULOSO - ALGORITMO DE TREINAMENTO

Entradas: conjunto de regras criadas no Slicer Ω,

Conjunto de exemplos para treinamento Λ,

Número de partições W.

Saídas: conjunto de regras nebulosas RF,

Conjunto de descrições das partições DESCPART.

Passo 1 – Gerar as projeções de todos os exemplos nos vetores ortogonais de Ω:

$$Geraprojecoes(\Omega, \Lambda, \widehat{\Lambda}).$$

Passo 2 – Gerar as partições de todas as projeções em $\widehat{\Lambda}$:

$$Geraparticoes(\widehat{\Lambda}, \Omega, W, DESCPART).$$

Passo 3- Gera conjunto de regras nebulosas RF usando o método Wang-Mendel:

$$WM(\widehat{\Lambda}, DESCPART, RF).$$

Passo 4– Finaliza, tendo gerado a BD (DESCPART) e a BR (RF).

Quadro 12 – Algoritmo treinamento Slicer-Nebuloso.

Algoritmo Wang-Mendel

Entradas: conjunto de projeções $\widehat{\Lambda}$, conjunto de partições DESCPART.

Saidas: conjunto de regras nebulosas RF.

Passo 1- para cada elemento de $\widehat{\Lambda}$ criar uma regra nebulosa. Na criação dos antecedentes, atribui a cada projeção o conjunto nebuloso em que possuir a maior pertinência. Em caso de empate, escolhe o conjunto mais à esquerda. O consequente é a classe associada ao elemento de $\widehat{\Lambda}$;

Passo 2 – criar para todas as regras nebulosas um grau de relevância, que é a t-norma das pertinências de seus antecedentes;

Passo 3 - verificar todas as regras que possuem os mesmos antecendentes e manter apenas a que possuir o maior grau de relevância. Em caso de empate, manter a regra que vem primeiro;

Passo 4 – finalizar, dando saída em um conjunto de regras nebulosas RF.

Quadro 13 – Algoritmo do Método Wang-Mendel.

As entradas deste algoritmo são produzidas pelos algoritmos *geraprojecoes* e *geraparticoes*.

No algoritmo seguinte, estão os procedimentos adotados para a conversão dos exemplos em projeções. Que são os seguintes:

Considere o conjunto de exemplos Λ e o conjunto de regras Ω definidos como:

$$\Lambda = \{(\Gamma^g, \chi^g)\};$$

$$\Gamma^g = \left(\gamma^{g,i}\right); \text{ e}$$

$$\Omega = \left\{\left(\mathrm{T}^{\mathrm{j}}, \alpha^{\mathrm{j}}, \beta^{\mathrm{j}}\right)\right\} \text{ em que}$$

$$1 \leq g \leq N, 1 \leq i \leq P, 1 \leq j \leq R.$$

O novo conjunto de exemplos $\widehat{\Lambda}$ é definido como:

$$\widehat{\Lambda} = \left\{\left(\hat{\Gamma}^g, \hat{\chi}^g\right)\right\};$$

$$\hat{\Gamma}^g = \left(\hat{\gamma}^{g,j}\right); \text{ e}$$

$$\hat{\chi}^g = \chi^g; \text{ em que}$$

$$\hat{\gamma}^{g,j} =< \mathrm{T}^j, \Gamma^g >.$$

Observe-se que sobre o espaço de atributos age o Sistema Slicer. Sobre o espaço de projeções age o sistema Slicer-Nebuloso. O mapeamento entre estes espaços corresponde ao gerador de projeções (Figura 4.4). Seu algoritmo está no Quadro 14.

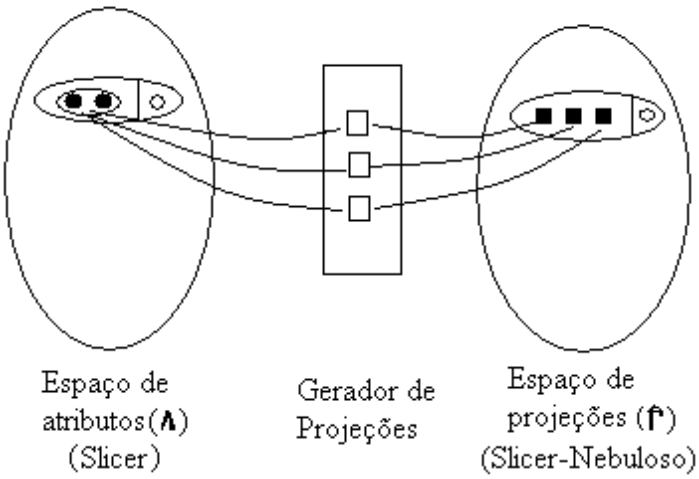


Figura 4.4 – Mapeamento entre espaço de atributos e espaço de projeções.

Geraprojecoes – Algoritmo Gerador de Projeções

Entradas: conjunto de regras criadas no Slicer Ω,

Conjunto de exemplos para treinamento Λ.

Saídas: conjunto de projeções $\widehat{\Lambda}$.

Passo 1 – Crie as constantes: número de regras em Ω, $I = |\Omega|$; número de exemplos em Λ, $K = |\Lambda|$; número de atributos nos vetores Γ do conjunto de exemplos Λ, $J = |\Gamma^1|$.

Inicialize o conjunto $\widehat{\Lambda}$, $\widehat{\Lambda} = \emptyset$.

Criar variáveis contador de regras ix, $ix = 0$; contador de exemplos kx, $kx = 0$.

Passo 2- Se $kx = K$ então vá para o Passo 7.

Passo 3 – Faça $kx = kx + 1$ e $\hat{\chi}^{kx} = \chi^{kx}$.

Passo 4 - Se $ix = I$ então vá para o Passo 2.

Passo 5 – Faça $ix = ix + 1$ e $\hat{\gamma}^{kx,ix} =< \Gamma^{kx}, \mathrm{T}^{ix} >$.

Passo 6 - Vá para o Passo 4.

Passo 7 - Finaliza, com $\widehat{\Lambda}$ construido.

Quadro 14 – Algoritmo Geraprojecoes.

O algoritmo a seguir descreve o processo de geração de partições. Este algoritmo está particularizado para os particionamentos em 3 e 7 partições, com o objetivo de facilitar o entendimento de seus resultados.

No algoritmo que gera partições (Geraparticoes) são usadas as heurísticas:

- valor intermediário da partição (CE) como sendo o valor limitante da regra crisp;
- os comprimentos de meia partição podem ser diferentes de cada lado de CE.

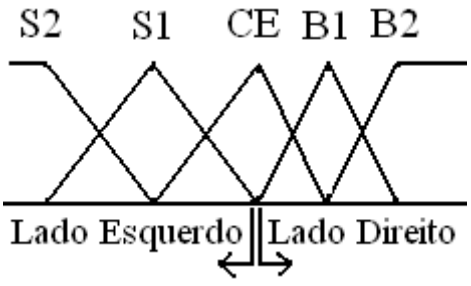


Figura 4.5 – Representação de partições e lados.

Cada meia partição do lado direito é igual a diferença entre a projeção no CE e a menor projeção entre todos os exemplos (projmin), dividido por W; enquanto que cada meia partição do lado esquerdo deste ponto tem o comprimento da diferença da maior projeção (projmax) menos a projeção neste ponto (projmed) dividido por W. As equações que definem estes lados estão no Passo 2 do Quadro 15. Pode-se entender melhor os conceitos de lado direito e esquerdo por meio da Figura 4.5.

Execução demonstrativa:

Considere o conjunto de exemplos Λ, Tabela 3.1, e o conjunto de regras Ω, Tabela 3.3, da execução demonstrativa da subseção 3.1.1. Em seguida é apresentado a execução demonstrativa do treinamento com o Slicer-Nebuloso.

Tabela 3.1 - Conjunto de treinamento $\boldsymbol{\Lambda}$.

λ^g {g=1,...,4}	x	y	χ^g
λ^1	0.23	0.34	1
λ^2	0.51	0.30	1
λ^3	0.62	-0.21	2
λ^4	0.29	0.80	2

Tabela 3.3 – Regras obtidas ao final do aprendizado do Slicer Básico.

Regras $\boldsymbol{\Omega}$	Vetor ortogonal T	Limitante α	Classe β
ω^1	$\mathrm{T}^1 = (0.2075, -0.5175)$	$\alpha^1 = 0.0940$	$\beta^1 = 2$
ω^2	$\mathrm{T}^2 = (-0.0533, 0.3200)$	$\alpha^2 = 0.1685$	$\beta^2 = 2$
ω^3	$\mathrm{T}^3 = (0.1400, -0.0200)$	$\alpha^3 = 0.0244$	$\beta^3 = 1$

Considerando W=1, ou seja, número de partições igual a 3.

Passo 1- Cria o conjunto de projeções $\widehat{\Lambda}$:

$$Geraprojecoes(\Omega, \Lambda, \widehat{\Lambda}).$$

Geraparticoes - Algoritmo Gerador de Partições

Entradas: conjunto de projeções $\widehat{\Lambda}$, conjunto de regras criadas no Slicer Ω, constante W.
Saídas: conjunto de descrições das partições DESCPART.

Passo 1 – Crie as constantes:
número de regras em Ω, $I = |\Omega|$;
número de exemplos em $\widehat{\Lambda}$, $K = \left|\widehat{\Lambda}\right|$.
Busca maior e menor projeção para cada projeção em $\widehat{\Lambda}$:

$$projmax^i = \max(\hat{\gamma}^{k,i}), em\ que\ 1 \le i \le I, 1 \le k \le K;$$
$$projmin^i = min(\hat{\gamma}^{k,i}),\ em\ que\ 1 \le i \le I, 1 \le k \le K.$$

Cria projeções intermediárias:

$$projmed^i = \begin{Bmatrix} \alpha^i : se\ \alpha^i > projmin^i; \\ projmin^i + (projmin^i - \alpha^i) : caso\ contrário \end{Bmatrix}, em\ que\ 1 \le i \le I.$$

Passo 2 - Calcule os lados esquerdos e direitos de todas as partições:

$$ld^i = (projmax^i - projmed^i)/W;$$
$$le^i = (projmed^i - projmin^i)/W,$$

em que $1 \le i \le I$.

Passo 3 – Se $W = 1$ então:

$$part^{i,2} = \left((projmed^i - le^i), projmed^i, (projmed^i + ld^i)\right);$$
$$part^{i,1} = \left((projmed^i - le^i), (projmed^i - le^i), projmed^i\right);$$
$$part^{i,3} = \left(projmed^i, (projmed^i + ld^i), (projmed^i + ld^i)\right);$$

em que $1 \le i \le I$.
Vá $para\ o\ Passo\ 5$.

Passo 4 - Se $W = 3$ então:
Partição central:

$$part^{i,4} = \left((projmed^i - le^i), projmed^i, (projmed^i + ld^i)\right).$$

Partições à esquerda:

$$part^{i,3} = \left((projmed^i - 2le^i), (projmed^i - le^i), projmed^i\right);$$
$$part^{i,2} = \left((projmed^i - 3le^i), (projmed^i - 2le^i), (projmed^i - le^i)\right);$$
$$part^{i,1} = \left((projmed^i - 3le^i), (projmed^i - 3le^i), (projmed^i - 2le^i)\right).$$

Partições à direita:

$$part^{i,5} = \left(projmed^i, (projmed^i + ld^i), (projmed^i + 2ld^i)\right);$$
$$part^{i,6} = \left((projmed^i + ld^i), (projmed^i + 2ld^i), (projmed^i + 3ld^i)\right);$$
$$part^{i,7} = \left((projmed^i + 2ld^i), (projmed^i + 3ld^i), (projmed^i + 3ld^i)\right);$$

em que $1 \le i \le I$.
$Vá\ para\ o\ Passo\ 5$.

Passo 5 – Finaliza, devolvendo DESCPART.

Quadro 15 – Algoritmo Geraparticoes.

Neste algoritmo (geraprojecoes, Quadro 14) pode-se destacar:

Crie as constantes: número de regras em Ω, $I = |\Omega| = 3$;

número de exemplos em Λ, $K = |\Lambda| = 4$; número de atributos nos vetores Γ do conjunto de exemplos Λ, $J = |\Gamma^1| = 2$.

Tabela 4.1 – Projeções obtidas para cada vetor ortogonal T em Ω.

Exemplos	**x**	**y**	$\mathbf{T}^{\mathbf{1}}$	$\mathbf{T}^{\mathbf{2}}$	$\mathbf{T}^{\mathbf{3}}$
$\Gamma^{\mathbf{1}}$	0.23	0.34	-0.1282	0.0965	0.0254
$\Gamma^{\mathbf{2}}$	0.51	0.30	-0.0494	0.0688	0.0654
$\Gamma^{\mathbf{3}}$	0.62	-0.21	0.2373	-0.1002	0.0910
$\Gamma^{\mathbf{4}}$	0.29	0.80	-0.3538	0.2405	0.0246

Desta tabela pode-se retirar os seguintes intervalos para cada regra:

Regra-slicer1: -0.3538 a 0.2373;

Regra-slicer 2: -0.1002 a 0.2405; e

Regra-slicer 3: 0.0246 a 0.0910.

Passo 2 – Cria o conjunto de partições DESCPART:

$$Geraparticoes\left(\widehat{\Lambda}, \Omega, W, DESCPART\right).$$

Busca maior e menor projeção para cada regra em Ω (Passo 1 de *Geraparticoes*):

$$projmax^i = max\left(\tilde{\tilde{\gamma}}^{k,i}\right), em\ que\ 1 \leq i \leq I, 1 \leq k \leq K;$$

$$projmin^i = min\left(\tilde{\tilde{\gamma}}^{k,i}\right), em\ que\ 1 \leq i \leq I, 1 \leq k \leq K.$$

Conforme mostrado na Tabela 4.1, tem-se:

$$projmin^1 = -0.3538, projmin^2 = -0.1002 \text{ e } projmin^3 = 0.0246;$$

$$projmax^1 = 0.2373, projmax^2 = 0.2405 \text{ e } projmax^3 = 0.0910;$$

Cria projeções intermediárias (Passo 1 de *Geraparticoes*):

$$projmed^i = \begin{Bmatrix} \alpha^i : se\ \alpha^i > projmin^i; \\ projmin^i + \left(projmin^i - \alpha^i\right) : caso\ contrário \end{Bmatrix}, em\ que\ 1 \leq i \leq I.$$

$$projmed^1 = 0.0940, projmed^2 = 0.1685 \text{ e } projmed^3 = 0.0248;$$

Note-se que $\alpha^3 < projmin^3, então\ é\ recalculado.$

Tabela 4.2 – Partições geradas para as regras Slicer.

Regras Slicer Ω	S1	CE	B1
ω^1	-0.3538, -0.3538, 0.0940	-0.3538, 0.0940, 0.2373	0.0940, 0.2373, 0.2373
ω^2	-0.1002, -0.1002, 0.1685	-0.1002, 0.1685, 0.2405	0.1685, 0.2405, 0.2405
ω^3	0.0246, 0.0246, 0.0248	0.0246, 0.0248, 0.0910	0.0248, 0.0910, 0.0910

Na Figura 4.6 estão representadas geometricamente todas as partições da regra-Slicer 1. Na Figura 4.7 estão representadas geometricamente todas as partições da regra-Slicer 2. Na Figura 4.8 estão representadas geometricamente todas as partições da regra-Slicer 3.

Passo 3- Gera conjunto de regras nebulosas RF usando o método Wang-Mendel:

$$WM(\widehat{\Lambda}, DESCPART, RF).$$

Para cada exemplo cria-se uma regra nebulosa.

Localizam-se primeiro as pertinências de cada regra-Slicer aplicado ao exemplo, conforme Tabela 4.3. Note-se que as maiores pertinências estão marcadas com um asterisco do lado esquerdo.

Escolhem–se as partições que obtêm maior pertinência e montam-se as regras nebulosas (Figura 4.9).

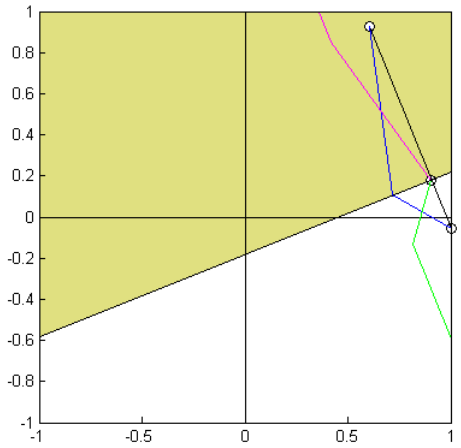
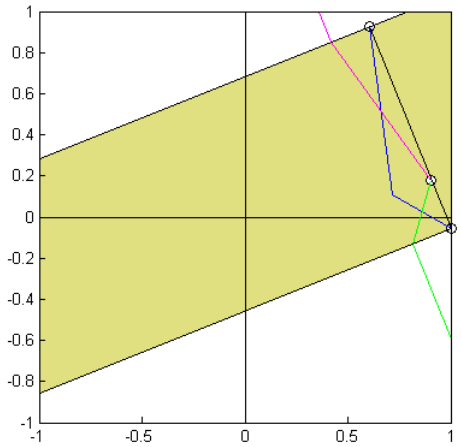
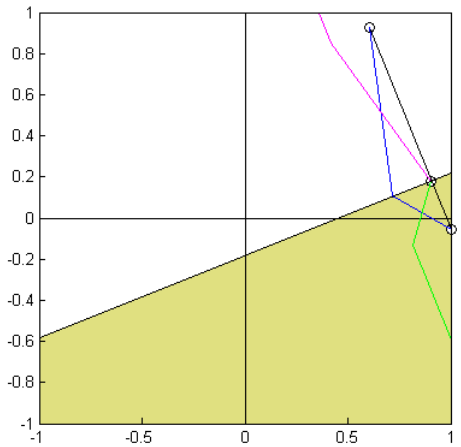
	**partição S1 da regra-slicer 1**
	**partição CE da regra-slicer 1**
	**partição B1 da regra-slicer 1**

Figura 4.6 – Partições da regra-slicer 1.

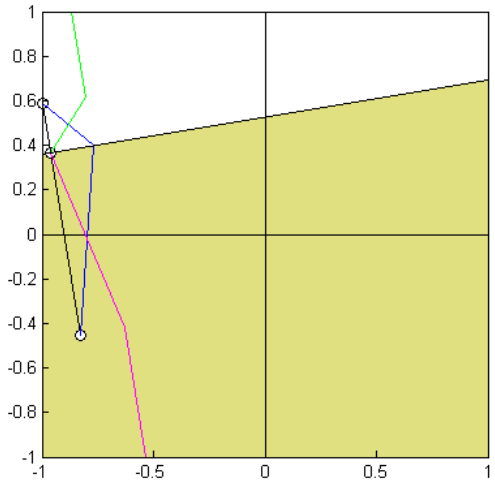
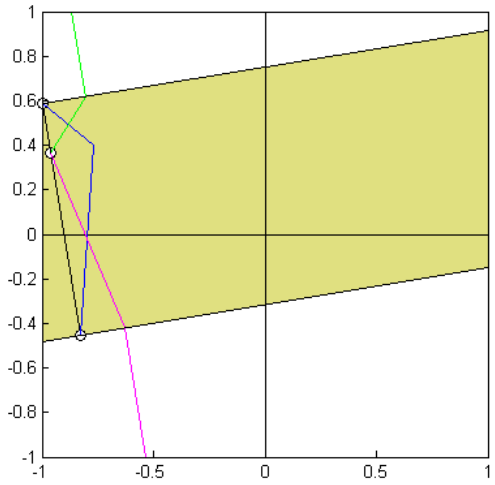
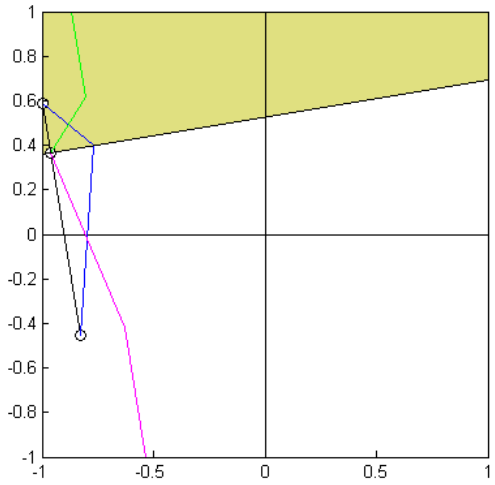
	**partição S1 da regra-slicer 2**
	**partição CE da regra-slicer 2**
	**partição B1 da regra-slicer 2**

Figura 4.7 – Partições da regra-slicer 2.

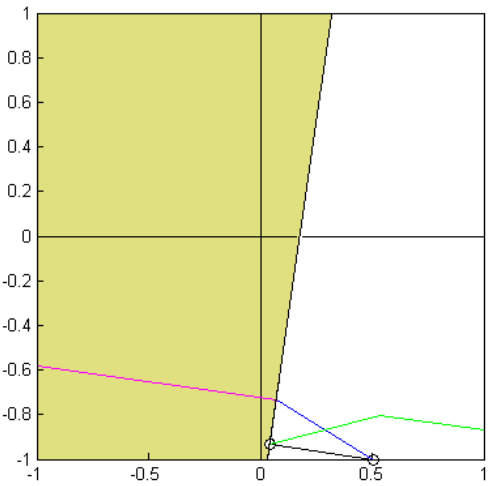
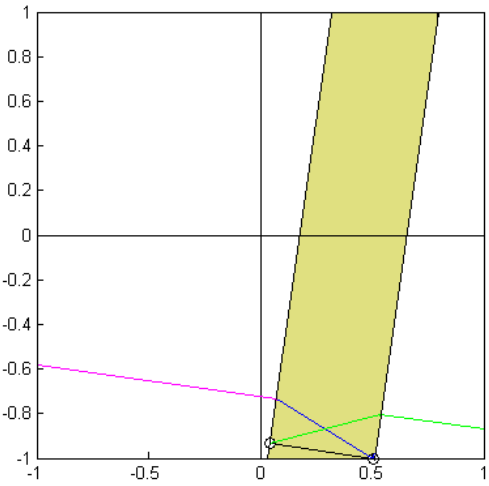
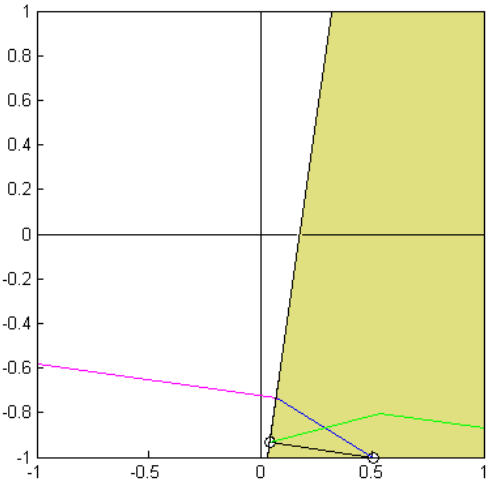
	**partição S1 da regra-slicer 3**
	**partição CE da regra-slicer 3**
	**partição B1 da regra-slicer 3**

Figura 4.8 – Partições da regra-slicer 3.

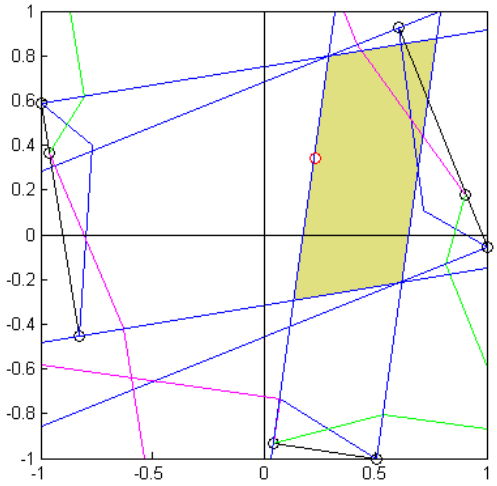
Tabela 4.3 – Pertinências dos exemplos em todas as partições de cada regra-slicer.

Exemplos	regra-slicer 1	Pertinência	regra-slicer 2	Pertinência	regra-slicer 3	Pertinência
1	-0.1282	S1(49.62%) *CE(50.38%)	0.0965	S1(26.80) *CE(73.20)	0.0254	*CE(99.09) B1(0.91)
2	-0.0494	S1(32.02%) *CE(67.98%)	0.0688	S1(37.10) *CE(62.90)	0.0654	CE(38.67) *B1(61.33)
3	0.2373	CE(0) *B1(100)	-0.1002	*S1(100) CE(0)	0.0910	CE(0) *B1(100)
4	-0.3538	*S1(100%) CE(0%)	0.2405	CE(0) *B1(100)	0.0246	*S1(100) CE(0)

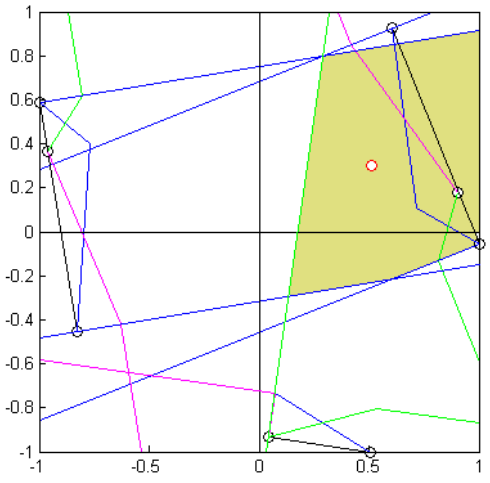
Observe que na Figura 4.8 as partições S1 e CE parecem ser definidas apenas por dois pontos. Na verdade, observando-se a Tabela 4.2 percebe-se que dois dos pontos que definem as partições de S1 e CE estão muito próximos dando impressão que são o mesmo ponto no gráfico, fazendo com que suas bordas se confundam em uma única linha.

Abaixo estão listadas as regras nebulosas obtidas:

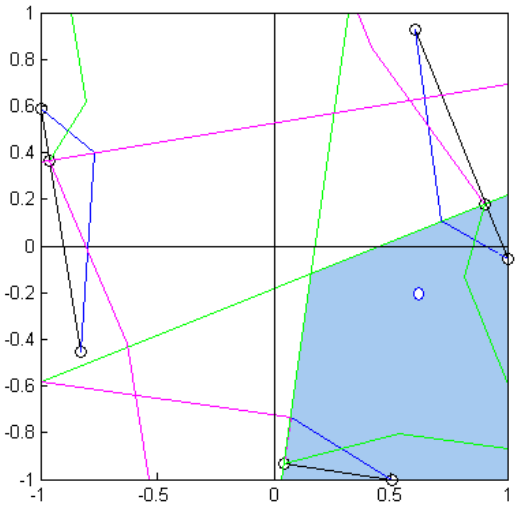
- Regra nebulosa 1: $CE^1 \cap CE^2 \cap CE^3 = 1$;
- Regra nebulosa 2: $CE^1 \cap CE^2 \cap B1^3 = 1$;
- Regra nebulosa 3: $B1^1 \cap S1^2 \cap B1^3 = 2$;
- Regra nebulosa 4: $S1^1 \cap B1^2 \cap S1^3 = 2$.



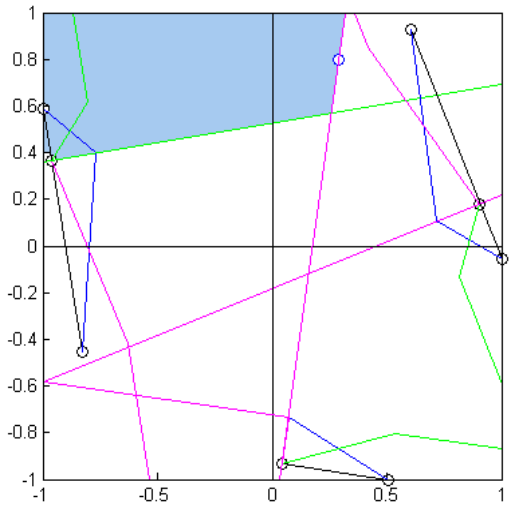
Regra nebulosa 1: $CE^1 \cap CE^2 \cap CE^3 = 1$



Regra nebulosa 2: $CE^1 \cap CE^2 \cap B1^3 = 1$



Regra nebulosa 3: $B1^1 \cap S1^2 \cap B1^3 = 2$



Regra nebulosa 4: $S1^1 \cap B1^2 \cap S1^3 = 2$

Figura 4.9 - Representação das regras nebulosas geradas.

Note-se a área em que as partições representando classes diferentes se sobrepõem, indicando regiões de decisão nebulosa (Figura 4.10.

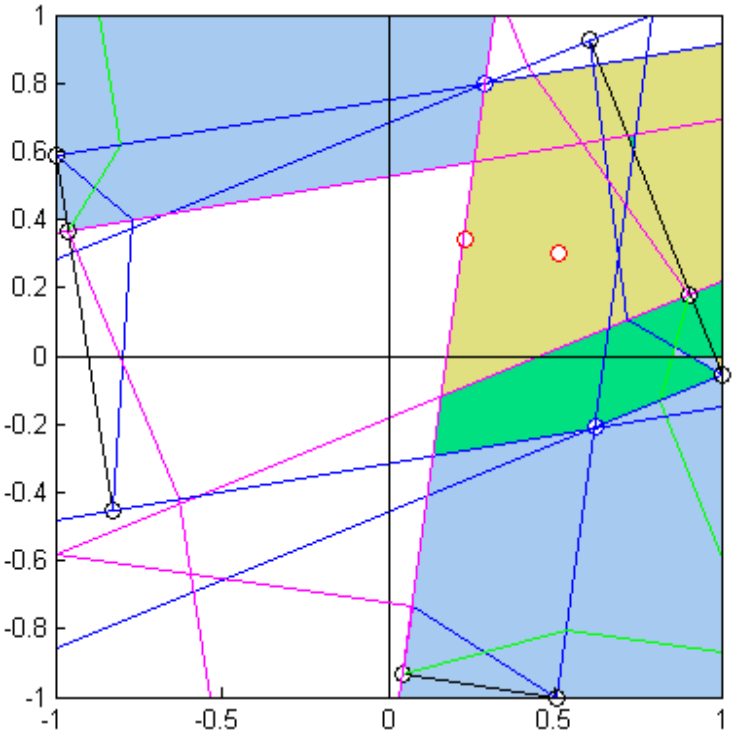


Figura 4.10 – Representação dos subespaços determinados pelas regras nebulosas.

4.1.2 - Classificação

O Sistema Slicer-Nebuloso, quando na fase de classificação, recebe o vetor de atributos e o respectivo vetor de projeções é produzido pelo *Geradordeprojecao*.

A classificação usando a base de regras nebulosa é feita pela inferência do vetor de projeções em todas as regras da base de regras.

A regra-nebulosa vencedora é a que obtiver maior grau de relevância. A regra-nebulosa vencedora fornece a classe, seu consequente, como classe a ser determinada para o vetor de atributos inicial. Em caso de empate a classe da entrada corresponde à classe associada ao espaço ao qual o vetor de projeções mais se aproxima.

A classificação ocorre através da análise do grau de relevância da regra, usando a t-norma produto algébrico.

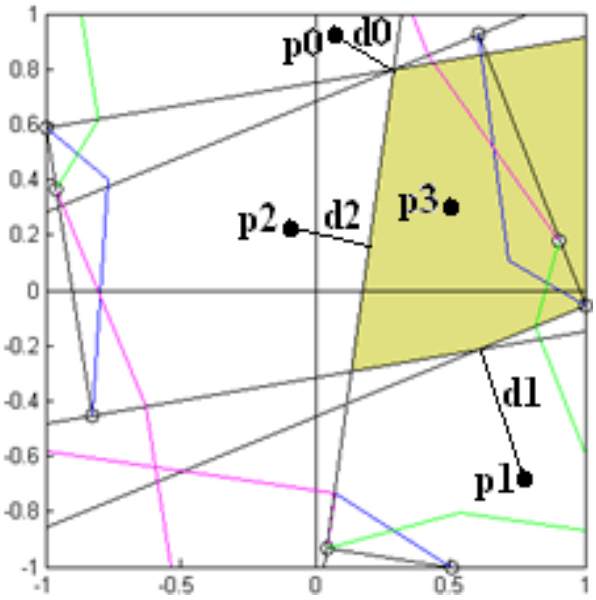


Figura 4.11 – Representação das distâncias (d0, d1 e d2) para 4 pontos (p0, p1, p2 e p3).

Na Figura 4.11 mostra-se as distâncias calculadas para 3 pontos. Para o ponto p3, a distância é igual a zero, pois neste caso o ponto está interno ao subespaço gerado pela regra.

SLICER-NEBULOSO - ALGORITMO DE CLASSIFICAÇÃO

Entradas: conjunto de regras criadas no Slicer Ω,

Vetor de atributos a ser classificado ϒ,

Conjunto de regras nebulosas RF,

Conjunto de descrições das partições DESCPART.

Saida: classe determinada para o vetor Υ, ν.

Passo 1 – Crie o vetor de projeções $\widehat{\Upsilon}$ usando o vetor ϒ e as regras de Ω.

$$Geraprojecao(\Omega, \Upsilon, \widehat{\Upsilon}).$$

Passo 2 – Calcula para cada regra nebulosa em RF o grau de relevância e a distância. Retorna a classe da regra que possuir maior grau e, em caso de empate, menor distância.

$$MotordeInferencia(DESCPART, RF, \widehat{\Upsilon}, \nu).$$

Passo 3- Finaliza, retornando a classe ν.

Quadro 16 - **Algoritmo de classificação Slicer-Nebuloso.**

O algoritmo abaixo calcula o vetor de projeções associado ao vetor de atributos a ser classificado.

Geraprojecao – Algoritmo Gerador de Projeção

Entradas: conjunto de regras criadas no Slicer Ω,
vetor de atributos Υ.
Saídas: vetor de projeções $\widehat{\Upsilon}$.

Passo 1 – Crie as constantes: número de regras em Ω, $I = |\Omega|$; número de atributos no vetor Υ, $J = |\Upsilon|$.
Criar variáveis contador de regras ix, $ix = 0$.
Passo 2 – Se $ix = I$ então vá para o Passo 5.
Passo 3 – Faça $ix = ix + 1$ e $\hat{v}^{ix} =< \Upsilon, \mathrm{T}^{ix} >$.
Passo 4 - Vá para o Passo 2.
Passo 5 - Finaliza, com $\widehat{\Upsilon}$ construido..

Quadro 17 – Algoritmo de geração de vetor de projeções para um único vetor de atributos.

No algoritmo *Motordeinferencia* a seguir, são usados dois critérios para escolha da regra nebulosa vencedora: 1) grau de relevância obtida pelo uso da t-norma produto algébrico; e 2) distância Euclidiana do vetor de atributos à partição no espaço.

MotordeInferencia - Algoritmo do Motor de Inferência

Entradas: conjunto de partições DESCPART, conjunto de regras nebulosas RF, vetor de projeções $\widehat{\Upsilon}$.
Saidas: classe determinada para o vetor $\widehat{\Upsilon}$, ν.

Passo 1- para cada regra nebulosa em RF calcular seu grau de relevância e sua distância de $\widehat{\Upsilon}$. O grau de relevância é a t-norma das pertinências de seus antecedentes; e a distância é a distância Euclidiana do vetor $\widehat{\Upsilon}$ às partições.
Passo 2 – selecionar a regra nebulosa que possuir maior grau de relevância e, em caso de empate, a que possuir menor distância.
Passo 3 - finalizar, retornando a classe do consequente da regra selecionada no Passo 2 como sendo a classe determinada ν.

Quadro 18 – Algoritmo do Motordeinferência.

Execução demonstrativa:

A seguir apresenta-se uma execução demonstrativa do procedimento via Sistema Slicer-Nebuloso.

Dado um vetor Υ de atributos (1, 1), seguem-se os passos para se obter a classificação:

Passo 1 – Criar o vetor de projeções $\widehat{\Upsilon}$ usando o vetor Υ e as regras de Ω.

Cada elemento do vetor $\widehat{\Upsilon}$ é produzido pelo produto interno entre o vetor $\boldsymbol{\Upsilon}$ e o vetor **T**, ou seja,

$\widehat{\Upsilon} = (< (0.2075, -0.5175), (1,1) >, < (-0.0533, 0.32), (1,1) >, < (0.14, -0.02), (1,1) >)$, que é igual a $\widehat{\Upsilon} = (0.31, 0.2667, 0.12)$.

Passo 2 – Calcular para cada regra nebulosa em RF o grau de relevância e a distância. Mantém salvo o índice da regra que possuir maior grau e menor distância (Tabela 4.4). Note-se que o vetor de projeções dado não pertence a qualquer uma das regras nebulosas, porém, ao se verificar as distâncias deste vetor aos subespaços das regras nebulosas, o subespaço correspondente à regra nebulosa 2 é o que possui menor distância, determinando a classe.

Tabela 4.4 – Graus e distâncias calculados para cada regra nebulosa.

Regra Nebulosa	Pertinência	Grau Relevância	Distância
$\mathbf{CE^1 \cap CE^2 \cap CE^3 = 1}$	Prod_alg(0,0,0)	0	0.0825
$\mathbf{CE^1 \cap CE^2 \cap B1^3 = 2}$	Prod_alg(0,0,1)	0	0.0059
$\mathbf{B1^1 \cap S1^2 \cap B1^3 = 2}$	Prod_alg(1,0,1)	0	0.0982
$\mathbf{S1^1 \cap B1^2 \cap S1^3 = 1}$	Prod_alg(0,0,0)	0	0.2360

Passo 3- Atribuir a Υ à classe do consequente da regra nebulosa vencedora do Passo 2, qual seja, o índice da regra nebulosa 2, correspondente à classe 2.

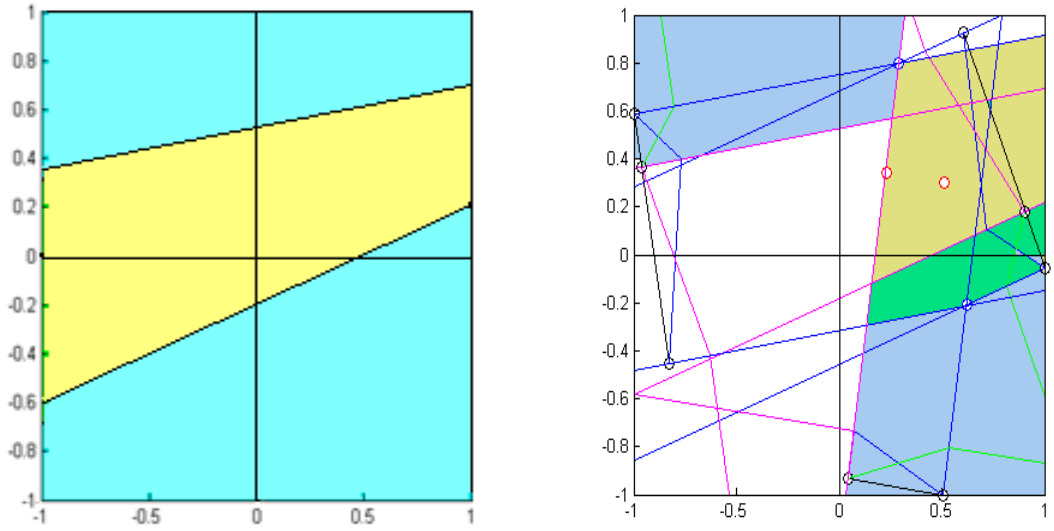
4.2 - Considerações sobre o método Slicer-Nebuloso

Nesta seção comparam-se as divisões de espaços geradas pelo Slicer Básico e pelo Slicer-Nebuloso, assim como apresentam-se as possibilidades deste sistema no tratamento do problema de dimensionalidade e discutem-se as expectativas quanto aos seus resultados.

Com o uso de raciocínio aproximado, pretende-se obter respostas aos pedidos de classificação com melhor possibilidade de acerto, pois: 1) passa-se a tratar do problema de classificação levando em conta a proximidade aos espaços definidos pelas regras nebulosas; e 2) em situações em que mais de uma classificação é possível, situação de ambiguidade, considera-se o quão relevante uma classificação é em relação às classificações concorrentes.

Percebe-se na Figura 4.14 que a divisão do espaço feita pelo Slicer-Nebuloso é mais complexa e admite áreas sobrepostas, sendo que as áreas não definidas estão sujeitas a medição de suas distâncias para determinação.

Percebe-se que as divisões de espaço geradas pelo Slicer Básico e pelo Slicer-Nebuloso, tais como mostradas na Figura 4.14, apresentam fortes diferenças entre si. Isto é devido ao fato de que em seus treinamentos é utilizado um conjunto de exemplos com apenas 4 elementos, servindo esta figura apenas como base para ilustração destes métodos.



(a) (b)

Figura 4.14 – Comparação dos subespaços encontrados por:
(a) Slicer Básico; (b) Slicer-Nebuloso.

O uso de regras Slicer (leiam-se hiperplanos) no lugar de variáveis de entrada em SFCBR permite que se faça uma mudança da dimensionalidade do problema, permitindo produzir uma divisão do espaço mais adequada à classificação. Por meio de um número de regras crisp (regras slicer) maior que o número de atributos do problema, consegue-se aumentar sua dimensionalidade e, por meio de número de regras crisp inferior ao de atributos, consegue-se diminuir esta dimensionalidade. Tal aspecto é interessante quando se pensa em problemas reais, nos quais a dimensionalidade passa a ser uma precupação.

É possível implantar um limite para o número de regras crisp desejadas, pois o Slicer fornece estas regras ordenadas segundo a eficiência na classificação, permitindo uma poda em seu número de modo bastante simples, apenas por descarte das regras que ultrapassam um limite desejado.

Espera-se do Slicer-Nebuloso uma melhor acurácia em relação à obtida pelo Slicer Básico e versões/extensões. Esta expectativa é devida à sua maior complexidade da divisão do espaço entre as classes e a possibilidade de exemplos existirem em áreas com determinação de classe em situação ambígua. Além disso, trata das regiões não cobertas pelas regras nebulosas, sendo a determinação da classe baseada na distância do vetor de projeções aos espaços definidos pelas partições de cada regra nebulosa.

Capítulo 5

Resultados

5.1 – Introdução

Uma implementação do Sistema Classificador SLICER, foi realizada no ambiente MatLab R2007a, da The MathWorks. As próximas seções descrevem a sua aplicação em domínios de conhecimento.

Neste capítulo apresentam-se os resultados obtidos pelo Slicer, em todas suas possíveis combinações de versões, e do Slicer–Nebuloso, com 3 e 7 partições utilizando os conjuntos de regras vindos das execuções do Slicer, em vários domínios de conhecimento.

Para efeito de avaliação e obtenção de resultados consideram-se 12 bases conhecidas advindas de [Asuncion & Newman 2007] e 4 domínios artificiais.

Após a obtenção de resultados, compara-se o desempenho aos apresentados em [Tran et. al. 2005], obtidos por diversos outros métodos.

5.2 - Metodologia

A metodologia de validação dos resultados é a 10 vezes 10-validação cruzada estratificada (manutenção da proporção de cada classe no conjunto de teste), ou seja, para cada conjunto de exemplos adotado para o processo de aprendizagem, o processo 10-validação cruzada estratificada é repetido 10 vezes, considerando em cada repetição um conjunto reordenado aleatoriamente.

Todas as bases são normalizadas, ficando seus valores de atributos entre 0 e 1.

Todos os resultados e número de regras geradas são acompanhados de desvio padrão.

Na Tabela 5.1 estão descritas as 11 bases sem atributos faltantes utilizadas na geração de resultados do Slicer e Slicer-Nebuloso. As 4 primeiras bases são artificiais e são descritas a seguir. As outras 7 bases são oriundas de [Asuncion & Newman 2007]. Percebe-se que são bases com número de exemplos, atributos e classes heterogêneos.

Tabela 5.1 – Bases de dados utilizadas, sem atributos faltantes.

DOMÍNIO	EXEMPLOS	ATRIBUTOS	CLASSE
Gauss00	400	2	2
Gauss20	400	2	2
Gauss50	400	2	2
Gauss75	400	2	2
Haberman's survival	306	3	2
Musk database 1	476	166	2
Spect heart 1	267	22	2
Spect heart 2	349	44	2
Hayes-Roth	150	3	3
Iris plants	150	4	3
Flags	194	28	8

Na Tabela 5.2 estão descritas as 5 bases com atributos faltantes utilizadas na geração de resultados do Slicer e Slicer-Nebuloso, quando da análise de domínios com atributos faltantes. São todas oriundas de [Asuncion & Newman 2007].

Tabela 5.2 – Bases de dados utilizadas, com atributos faltantes.

DOMÍNIO	EXEMPLOS	ATRIBUTOS	CLASSE
Pittsburgh bridges version 1 TORD	108	7	2
Pittsburgh bridges version 1 MATERIAL	108	7	3
Pittsburgh bridges version 1 SPAN	108	7	3
Pittsburgh bridges version 1 REL-L	108	7	3
Pittsburgh bridges version 1 TYPE	108	7	6

As 4 bases artificiais são representações de distribuições Gaussianas, cada uma com uma característica específica, a saber: 1) com nenhuma sobreposição, 2) com 20% de sobreposição, 3) com 50% de sobreposição e 4) com 75% de sobreposição, respectivamente. Estas bases possuem 2 distribuições gaussianas, com valor médio em 0 e desvio padrão em 1, normalizadas. As sobreposições são consideradas em função da distância entre os centros destas distribuições.

Considere que a sobreposição de 0%, indica que os centros das distribuições distam 1, sobreposição de 20% indica distância de 0,8, sobreposição de 50% indica distância de 0,5; e sobreposição de 75% indica distância entre os centros de 0,25.

Abaixo a Figura 5.1 mostra todas estas 4 distribuições.

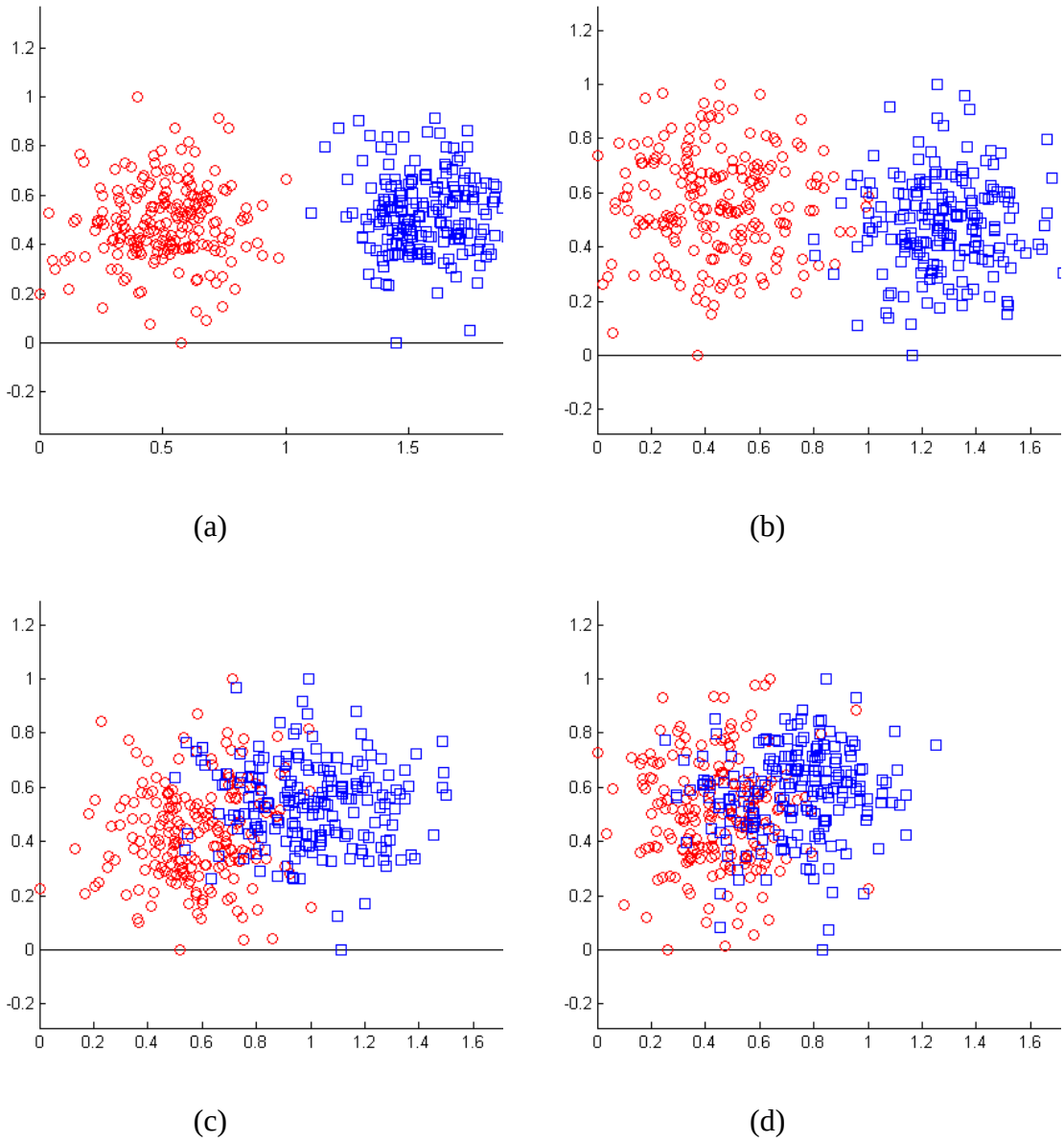


Figura 5.1 – Representação gráfica das distribuições Gaussianas: (a) sem sobreposição; (b) com 25% de sobreposição; (c) com 50% de sobreposição; e (d) com 75% de sobreposição.

5.3 – Experimentos e análise de resultados

O objetivo da avaliação é encontrar uma medida fiel das expectativas que devem ser associadas ao sistema proposto frente aos diferentes desafios do problema da classificação.

Todas estas avaliações ocorrem sobre os 16 domínios de conhecimento: sendo 12 oriundos de [Asuncion & Newman 2007], dentre os 31 constantes de [Tran et. al. 2005], e 4 domínios artificiais descritos na subseção anterior.

Com esse objetivo são desenvolvidos 3 experimentos: 1) avaliação das versões Slicer Básico (com Slicer-F embutido), Slicer-R, Slicer–G e Slicer-O; 2) avaliação das versões Slicer-Nebuloso com 3 e 7 partições; 3) comparação de desempenho com os resultados de [Tran et. al. 2005].

Os resultados dos experimentos 1 e 2 estão nas tabelas constantes do apêndice. Estas tabelas possuem uma marcação das comparações feitas, ou seja:

1. melhores acurácias entre a Tabela A2 e A1,
2. melhores acurácias entre a Tabela A3 e A1,
3. melhores acurácias entre a Tabela A4 e A2,
4. melhores acurácias entre a Tabela A4 e A3,
5. menores quantidades de regras geradas entre a Tabela A2 e A1,
6. menores quantidades de regras geradas entre a Tabela A3 e A1,
7. menores quantidades de regras geradas entre a Tabela A4 e A2,
8. menores quantidades de regras geradas entre a Tabela A4 e A3,
9. melhores acurácias entre as acurácias obtidas com as regras geradas pelo Slicer-O e as regras que são dadas como entrada ao Slicer-O,
10. menores quantidades de regras geradas pelo Slicer-O e as regras que são dadas como entrada ao Slicer-O (este conjunto não é marcado, pois em todos resultados obteve-se quantidade menor),
11. melhor(es) acurácia(s) obtida(s) por domínio no Experimento 1,
12. melhores acurácias entre Slicer-Nebuloso de 3 partições da Tabela A5 e Tabela A1,
13. melhores acurácias entre Slicer-Nebuloso de 3 partições da Tabela A6 e Tabela A2,
14. melhores acurácias entre Slicer-Nebuloso de 3 partições da Tabela A7 e Tabela A3,
15. melhores acurácias entre Slicer-Nebuloso de 3 partições da Tabela A8 e Tabela A4,

16. melhores acurácias entre Slicer-Nebuloso de 7 partições da Tabela A5 e Tabela A1,
17. melhores acurácias entre Slicer-Nebuloso de 7 partições da Tabela A6 e Tabela A2,
18. melhores acurácias entre Slicer-Nebuloso de 7 partições da Tabela A7 e Tabela A3,
19. melhores acurácias entre Slicer-Nebuloso de 7 partições da Tabela A8 e Tabela A4,
20. melhor(es) acurácia(s) obtida(s) por domínio no Experimento 2,
21. melhor(es) acurácia(s) obtida(s) por domínio entre os experimentos 1 e 2.

Cada apresentação de acurácia contém 3 valores: acurácia mínima, acurácia média e acurácia máxima, todas obtidas na execução 10 x 10 validação cruzada estratificada. A acurácia utilizada nas comparações é a acurácia média.

Cada apresentação de quantidades de regras geradas contém 3 valores: quantidade mínima, quantidade média e quantidade máxima, todas obtidas na execução 10 x 10 validação cruzada estratificada. A quantidade utilizada nas comparações é a quantidade média.

O Experimento 1 gerou as tabelas de A1 a A4 e o Experimento 2 gerou as tabelas de A5 a A8. O Experimentou 3 faz uso dos resultados obtidos em todas as tabelas, mas especificamente nas marcações indicadas na comparação 21.

Em todas as avaliações a expressão “A melhor-ou-igual B” significa que o termo A é maior que B ou que (A+1) é maior que B, indicando que A está muito próximo de B ou o ultrapassa.

As análises dos resultados são feitos em duas fases por experimento, primeiro para os domínios sem atributo faltante e depois em domínios com atributos faltantes.

Experimento 1

As combinações de versões: 1) Slicer Básico; 2) Slicer Básico mais Slicer-G; 3) Slicer Básico mais Slicer-R; e 4) Slicer Básico mais Slicer-G e Slicer-R, são avaliadas. A versão Slicer-O é avaliada em todos os conjuntos de regras geradas pelas combinações anteriores. Note-se que todas as versões do Slicer Básico possuem a melhoria Slicer-F implementada.

No experimento 1 são feitas 4 baterias de teste, em todos os 16 domínios disponíveis, gerando para cada domínio 8 acurácias e médias de regras:

1) Faz-se aprendizagem usando Slicer Básico + Slicer-F, o conjunto de regras resultante é avaliado e sua acurácia e número de regras são arquivados para média final e cálculo de desvio padrão; em seguida esse conjunto de regras é dado como entrada para o Slicer-O, resultando em novo conjunto de regras, que também é avaliado e tem sua acurácia e número de regras arquivados para média final e cálculo de desvio padrão.
2) Faz-se aprendizagem usando Slicer Básico + Slicer-F + Slicer-R, o conjunto de regras resultante é avaliado e sua acurácia e número de regras são arquivados para média final e cálculo de desvio padrão; em seguida esse conjunto de regras é dado como entrada para o Slicer-O, resultando em novo conjunto de regras, que também é avaliado e tem sua acurácia e número de regras arquivados para média final e cálculo de desvio padrão.
3) Faz-se aprendizagem usando Slicer Básico + Slicer-F + Slicer-G, o conjunto de regras resultante é avaliado e sua acurácia e número de regras são arquivados para média final e cálculo de desvio padrão; em seguida esse conjunto de regras é dado como entrada para o Slicer-O, resultando em novo conjunto de regras, que também é avaliado e tem sua acurácia e número de regras arquivados para média final e cálculo de desvio padrão.
4) Faz-se aprendizagem usando Slicer Básico + Slicer-F + Slicer-R + Slicer-G, o conjunto de regras resultante é avaliado e sua acurácia e número de regras são arquivados para média final e cálculo de desvio padrão; em seguida esse conjunto de regras é dado como entrada para o Slicer-O, resultando em novo conjunto de regras, que também é avaliado e tem sua acurácia e número de regras arquivados para média final e cálculo de desvio padrão.

Neste conjunto de experimentos são apresentados resultados com o objetivo de comparar as diferentes combinações de versões, visando deixar evidente que as versões incrementam a qualidade do algoritmo original.

Análise em domínios sem atributo faltante:

Na tabela 5.3 estão descritas as acurácias da execução do Slicer Básico em todos os domínios (sem atributos faltantes) considerados, além do número médio de hiperplanos gerados, e seus respectivos desvios padrões.

Tabela 5.3 – Resultados obtidos com o Slicer Básico (AC/DP) acurácia/desvio padrão, HP/DP: média de número de HPs gerados/desvio padrão. Domínios sem atributo faltante.

Domínio	AC/DP	HP/DP
Gauss00	0.9892 /0.0037	5.15 / 0.1285
Gauss20	0.9488 /0.0108	20.62 / 0.5381
Gauss50	0.9072 /0.0067	44.23 /1.1714
Gauss75	0.7118 /0.0174	119.98/ 1.7469
Haberman's survival	0.6353 /0.0583	91.58 / 1.6952
Musk database 1	0.8362 /0.0165	81.71/ 0.7816
Spect heart	0.9323 /0.0149	45.00/ 0.4195
Spect heart	0.8509 /0.0130	68.20 /1.4574
Hayes-Roth	0.6692 /0.0673	17.84 / 0.5333
Iris plants	0.9424 /0.0129	9.37 /0.2283
Flags	0.3636/ 0.0470	117.57 / 0.7417

As acurácias obtidas pela aplicação do Slicer Básico nos domínios indicados na Tabela 5.3 assumem valores acima de 80% para 7 dos 11 domínios listados.

Na comparação com o Slicer Básico (ver tabelas A1 e A2 no apêndice), em 11 bases, o Slicer-R obteve melhor-ou-igual acurácia em 10 (91%), e em 8 (73%) bases conseguiu produzir conjunto de regras menor. Estas medidas atestam que a busca de um melhor vetor ortogonal para geração de hiperplanos é relevante.

Na comparação com o Slicer Básico (ver tabelas A1 e A3 no apêndice), em 11 bases, o Slicer-G obteve melhor-ou-igual acurácia em 7 (64%), e nas 11 (100%) bases conseguiu produzir conjunto de regras drasticamente menor. Estas medidas confirmam que o tratamento de ruídos além de melhorar a acurácia ainda consegue reduzir significativamente o número de regras finais.

Na comparação com o Slicer-R (ver tabelas A2 e A4 no apêndice), em 11 bases, o Slicer-R + Slicer-G obteve melhor-ou-igual acurácia em 9 (82%), e em 11 (100%) bases conseguiu produzir conjunto de regras menor; e na comparação com o Slicer-G (ver tabelas A3 e A4 no apêndice), em 11 bases, o Slicer-R + Slicer-G obteve melhor-ou-igual acurácia em 9 (82%), e em 11 (100%) bases conseguiu produzir conjunto de regras menor. Estas medidas atestam que a utilização do Slicer-R junto com o Slicer-G consegue obter melhor acurácia e menor número de regras finais.

Na comparação com os resultados obtidos pelo conjunto de regras dado como entrada do Slicer-O (ver tabelas A1, A2, A3 e A4 no apêndice), em 44 testes nas 11 bases, o Slicer-O obteve melhor-ou-igual acurácia em 31 (70%) destes testes, e em todos (100%) testes conseguiu produzir conjunto de regras menor. Estas medidas atestam que a utilização do Slicer-O em 70% aumenta a acurácia e sempre garante diminuição do número de regras. É possível considerá-lo como um redutor do conjunto de regras para ser usado em outros sistemas classificadores.

Os números dos melhores resultados desta bateria, considerando a combinação de versões, são os seguintes:

- 1 na versão Slicer Básico;
- 1 na versão Slicer-O após Slicer Básico;
- 1 do Slicer Básico + Slicer-R;
- 2 do Slicer Básico + Slicer-G;
- 5 do Slicer-O após Slicer Básico + Slicer-G;
- 2 do Slicer Básico + Slicer-R + Slicer-G;
- 2 do Slicer-O após Slicer Básico + Slicer-G+ Slicer-R.

Percebe-se que o total ultrapassa 11, devido ao fato que em várias situações a melhor acurácia acontece em mais de uma combinação.

Análise em domínios com atributo faltante:

Tabela 5.4 – Resultados obtidos com o Slicer Básico + Slicer-F (AC/DP) acurácia/desvio padrão, HP/DP: média de número de HPs gerados/desvio padrão. Domínios com atributo faltante.

Domínio	Acurácias/DP	#HP/DP
Pittsburgh bridges version 1 TORD	0.8300/0.0235	17.09/0.3833
Pittsburgh bridges version 1 MATERIAL	0.5447/0.0574	14.62/0.3429
Pittsburgh bridges version 1 SPAN	0.5768/0.0358	26.33/0.3607
Pittsburgh bridges version 1 REL-L	0.5428/0.0342	30.41/0.6252
Pittsburgh bridges version 1 TYPE	0.3808/0.0403	38.88/0.4377

Na tabela 5.4 estão descritas as acurácias da execução do Slicer Básico em todos os domínios (com atributos faltantes) considerados, além do número médio de hiperplanos gerados, e seus respectivos desvios padrões. Note-se que os valores obtidos são da execução do Slicer Básico mais Slicer-F.

As acurácias obtidas pela aplicação do Slicer Básico mais Slicer-F nos 5 domínios indicados na Tabela 5.4 assumem valor acima de 80% para apenas 1 dos 5 domínios listados, e mais de 50% para 4 dos domínios, domínios com mais de 25% de exemplos com atributos faltantes. Estas medidas atestam que o uso do Slicer-F embutido no Slicer Básico pode recuperar conhecimento destas bases, desde que a proporção de atributos faltantes não ultrapasse determinado limite, limite este a ser calculado para cada base.

Na comparação com o Slicer Básico (ver tabelas A1 e A2 no apêndice), em 5 bases, o Slicer-R obteve melhor-ou-igual acurácia em 5 (100%), e em 4 (80%) bases conseguiu produzir conjunto de regras menor. Estas medidas atestam que a busca de um melhor vetor ortogonal para geração de hiperplanos é relevante.

Na comparação com o Slicer Básico (ver tabelas A1 e A3 no apêndice), em 5 bases, o Slicer-G obteve melhor-ou-igual acurácia em 3 (60%), e nas 5 (100%) bases conseguiu produzir conjunto de regras drasticamente menor. Estas medidas confirmam que o tratamento de ruídos além de melhorar a acurácia ainda consegue reduzir significativamente o número de regras finais.

Na comparação com o Slicer-R (ver tabelas A2 e A4 no apêndice), em 5 bases, o Slicer-R + Slicer-G obteve melhor-ou-igual acurácia em 4 (80%), e em 5 (100%) bases conseguiu produzir conjunto de regras menor; e na comparação com o Slicer-G (ver tabelas A3 e A4 no apêndice), em 5 bases, o Slicer-R + Slicer-G obteve melhor-ou-igual acurácia em 4 (80%), e em 3 (60%) bases conseguiu produzir conjunto de regras menor. Estas medidas atestam que a utilização do Slicer-R junto com o Slicer-G consegue obter melhor acurácia e menor número de regras finais.

Na comparação com os resultados obtidos pelo conjunto de regras dado como entrada do Slicer-O (ver tabelas no apêndice), em 20 testes nas 5 bases, o Slicer-O obteve melhor-ou-igual acurácia em apenas 1 (5%) destes testes, e em todos (100%) testes conseguiu produzir conjunto de regras menor. Estas medidas atestam que a utilização do Slicer-O em 95% diminui a acurácia em domínios com atributo faltante, mas sempre garante diminuição do número de regras. É possível considerá-lo como um redutor do conjunto de regras para ser usado em outros sistemas classificadores.

Os números dos melhores resultados desta bateria, considerando a combinação de versões, são os seguintes:

- 1 na versão Slicer Básico;
- 1 do Slicer Básico + Slicer-G;
- 1 do Slicer-O após Slicer Básico + Slicer-G;
- 2 do Slicer Básico + Slicer-R + Slicer-G.

Experimento 2

As versões Slicer-Nebuloso 3 partições e Slicer-Nebuloso 7 partições serão notadas como Slicer-N 3 e Slicer-N 7, respectivamente, para efeitos de abreviação.

As versões Slicer-N 3 e Slicer-N 7 são avaliadas em todos os domínios disponíveis, tendo como entrada o conjunto de regras gerado pelo Slicer-O em todas as combinações possíveis.

No experimento 2 são feitas 4 baterias de teste, em todos os 16 domínios disponíveis, gerando para cada domínio 8 acurácias e médias de regras:

- Faz-se aprendizagem usando Slicer Básico + Slicer-F, o conjunto de regras resultante é dado como entrada para o Slicer-O, resultando em novo conjunto de regras que é dado como entrada para o Slicer-N 3 e para o Slicer-N 7, e ambos geram seus conjuntos de regras e suas avaliações, arquivando suas acurácias e número de regras para média final e cálculo de desvio padrão.
- Faz-se aprendizagem usando Slicer Básico + Slicer-F + Slicer-R, o conjunto de regras resultante é dado como entrada para o Slicer-O, resultando em novo conjunto de regras que é dado como entrada para o Slicer-N 3 e para o Slicer-N 7, e ambos geram seus conjuntos de regras e suas avaliações, arquivando suas acurácias e número de regras para média final e cálculo de desvio padrão.
- Faz-se aprendizagem usando Slicer Básico + Slicer-F + Slicer-G, o conjunto de regras resultante é dado como entrada para o Slicer-O, resultando em novo conjunto de regras que é dado como entrada para o Slicer-N 3 e para o Slicer-N 7, e ambos geram seus conjuntos de regras e

suas avaliações, arquivando suas acurácias e número de regras para média final e cálculo de desvio padrão.

- Faz-se aprendizagem usando Slicer Básico + Slicer-F + Slicer-R + Slicer-G, o conjunto de regras resultante é dado como entrada para o Slicer-O, resultando em novo conjunto de regras que é dado como entrada para o Slicer-N 3 e para o Slicer-N 7, e ambos geram seus conjuntos de regras e suas avaliações, arquivando suas acurácias e número de regras para média final e cálculo de desvio padrão.

Neste conjunto de experimentos são apresentados resultados com o objetivo de comparar as versões Slicer-N 3 e Slicer-N 7, visando deixar evidente se as versões Slicer-N incrementam a qualidade do algoritmo original, ou de suas composições.

Análise em domínios sem atributo faltante:

Dos resultados obtidos pelo Slicer-N 3, no conjunto de 44 testes (ver tabelas A5, A6, A7 e A8 no apêndice), só em 11 (25%) casos obteve resultado melhor que alguma das combinações do experimento 1, excluidas as acurácias do Slicer-O, o que parece evidenciar a fragilidade desta versão.

Porém o Slicer-N 7, no conjunto de 44 testes produzidos (ver tabelas A5, A6, A7 e A8 no apêndice), obtém 20 (45%) acurácias maior-ou-igual às acurácias obtidas no experimento 1, excluidas as acurácias do Slicer-O, evidenciando que realmente esta versão pode ser interessante em alguns domínios.

Os números dos melhores resultados desta bateria, considerando as combinações vindas do experimento 1, são os seguintes:

- 6 para o Slicer-N 7 utilizando regras vindas do Slicer Básico;
- 1 para o Slicer-N 3 utilizando regras vindas do Slicer Básico + Slicer-R;
- 2 para o Slicer-N 7 utilizando regras vindas do Slicer Básico + Slicer-R;
- 1 para o Slicer-N 3 utilizando regras vindas do Básico + Slicer-G;
- 1 para o Slicer-N 7 utilizando regras vindas do Básico + Slicer-G + Slicer-R.

Análise em domínios com atributo faltante:

Dos resultados obtidos pelo Slicer-N 3, no conjunto de 20 testes (ver tabelas A5, A6, A7 e A8 no apêndice), só em 2 (10%) casos obteve resultado melhor que alguma das combinações do experimento 1, excluidas as acurácias do Slicer-O, o que parece evidenciar a fragilidade desta versão.

Porém o Slicer-N 7, no conjunto de 20 testes produzidos (ver tabelas A5, A6, A7 e A8 no apêndice), obtém 15 (75%) acurácias maior-ou-igual às acurácias obtidas no experimento 1, excluidas as acurácias do Slicer-O, evidenciando que realmente esta versão pode ser interessante em alguns domínios.

Os números dos melhores resultados desta bateria, considerando as combinações vindas do experimento 1, são os seguintes:

- 1 para o Slicer-N 7 utilizando regras vindas do Slicer Básico;
- 1 para o Slicer-N 7 utilizando regras vindas do Básico + Slicer-G;
- 3 para o Slicer-N 7 utilizando regras vindas do Básico + Slicer-G + Slicer-R.

Experimento 3

Os autores de [Tran et. al. 2005] simularam e compararam nove métodos de uso mais comum adotados em problemas de classificação, tais como: RM (Reduced Multivariate Polynomials); redes neurais com ativação de perceptron por SINH, COSH, TANH, RAMP e STEP; SVM (Support Vector Machines); k-NN (k-Nearest Neighbor); MLP (Multi Layer Perceptron).

Executaram exaustivos testes de localização dos melhores parâmetros para cada método, em cada domínio, e os avaliaram usando 10 x 10-validação cruzada estratificada.

Nos resultados de comparação em [Tran et. al. 2005], conclui-se que embora o SVM obtenha as melhores acurácias, seus tempos de treinamento são muito maiores que dos outros métodos testados. Os métodos SVM e k-NN também são, entre os métodos testados, os que usam as maiores quantidades de memória para guardar seu conhecimento.

Comparou-se o Slicer (sua melhor acurácia entre os experimentos 1 e 2) com outros 3 métodos distintos já considerados em [Tran et. al. 2005], por serem bem disseminados na literatura, a saber:

- SVM ("Support Vector Machine") baseado em programação quadrática, busca melhor generalização por meio da minimização do limite do erro de generalização. Usa-se vários kernels na atualidade, sendo que o utilizado em [Tran et. al. 2005] é o kernel polinomial para casos multiclasse, também implementado em Matlab;
- k-NN ("k-Nearest Neighbor") baseado em exemplos, busca classificar um novo padrão por meio da classe mais presente na k-vizinhança. Utilizou-se da implementação em Matlab no artigo supra citado;
- MLP ("Multi Layer Perceptron") baseado em redes neurais artificiais, seus resultados vêm da execução de MLP com uma camada invisível.

A Tabela 5.5 apresenta as acurácias associadas ao Slicer e aos métodos SVM, MLP e k-NN, quando aplicados a domínios sem atributos faltantes. Nas acurácias do Slicer estão indicadas também os desvios padrões. Asterisco indica o melhor resultado. Áreas sombreadas indicam resultados na faixa de até 5% abaixo do melhor resultado.

Nota-se na Tabela 5.5 que SVM detém 57% dos melhores desempenhos, Slicer 29%, k-NN 14% e MLP 0%. Interessante se notar que os resultados do Slicer diferem no máximo em 5% do melhor resultado em 57% dos domínios testados, domínios sem atributos faltantes, indicados na Tabela 5.5.

Nota-se também que as acurácias do Slicer possuem baixo valor de desvio padrão, indicando pouca variação nestes resultados.

Tabela 5.5 – Desempenhos dos métodos Slicer, SVM, k-NN e MLP em bases de dados sem atributos faltantes.

Domínio	**Slicer**	**SVM**	**k-NN**	**MLP**
Haberman's survival	0.6904±0.0552	0.7340*	0.7217	07120
Musk database 1	0.8435 ±0.0196	0.9952*	0.9380	0.6948
Spect heart 1	0.9720 ±0.0107*	0.8254	0.8050	0.8108
Spect heart 2	0.8842± 0.0161*	0.8841	0.8479	0.8171
Hayes-Roth	0.6692 ±0.0673	0.8660*	0.6071	0.7587
Iris plants	0.9513±0.0110	0.9640*	0.9583	0.9520
Flags	0.4212±0.0415	0.5294	0.6358*	0.5475

A Tabela 5.6 apresenta as acurácias associadas ao Slicer e aos métodos SVM, MLP e k-NN, quando aplicados a domínios com atributos faltantes. Nas acurácias do Slicer estão indicadas também os desvios padrões. Asterisco indica o melhor resultado. Áreas sombreadas indicam resultados na faixa de até 5% abaixo do melhor resultado.

Nota-se na Tabela 5.6 que k-NN detém 60% dos melhores desempenhos, SVM 20%, MLP 20% e Slicer 0%. Interessante se notar que os resultados do Slicer em nenhum destes domínios conseguiu sequer ficar na faixa de 5% do melhor resultado, em domínios com atributos faltantes, indicados na Tabela 5.6.

Nota-se também que as acurácias do Slicer possuem baixo valor de desvio padrão, indicando pouca variação nestes resultados.

Tabela 5.6 – Desempenhos dos métodos Slicer, SVM, k-NN e MLP em bases de dados com atributos faltantes.

Domínio	**Slicer**	**SVM**	**k-NN**	**MLP**
Pitts 1 TORD	0.8300 ±0.0235	0.8986*	0.8514	0.8529
Pitts 1 MATERIAL	0.5553±0.0633	0.8571	0.9257*	0.9000
Pitts 1 SPAN	0.6054±0.0454	0.8100	0.8417*	0.7883
Pitts 1 REL-L	0.5580± 0.0489	0.6771	0.7043*	0.6814
Pitts 1 TYPE	0.4658±0.0390	0.6186	0.6071	0.6443*

Como um dos critérios que norteou o desenvolvimento do Slicer foi criar um método com baixo custo computacional, analisou-se alguns custos computacionais encontrados na literatura:

- segundo [Elizondo 2006] o custo computacional dos métodos de teste de separabilidade linear:
  - CLS [Tajine et al. 1997] é $O(CLS) = O(N^P)$; e
  - Quick Hull [Preparata & Shamos 1985] é $O(Quick\ Hull) = O\left(N^{P-1}log(N)\right)$.
- sabe-se ainda que o custo computacional do SVM com kernel polinomial é $O(SVM\ Polinomial) = O(N^{Px})$ [Hellerstein & Servedio 2007], em que x é variável dependendo o kernel, logo percebe-se que o SVM é vitima do mal da dimensionalidade.
- $O(Slicer) = O(N^2)$ demonstrado nesta dissertação

Dadas estas informações, percebe-se o quão menos custoso é o Slicer em relação aos métodos referenciados.

5.4 - Considerações sobre os resultados

Devido aos resultados obtidos com Slicer-F, Slicer-G e Slicer-R pode-se afirmar que a integração destas versões ao Slicer Básico resultam em um sistema classificador robusto, admitindo domínios com atributos faltantes e ruídos e produzindo base de regras compactas.

O Slicer-O mostra-se capaz de obter redução em qualquer um dos conjuntos de regras a que é submetido nos experimentos mostrando-se de grande utilidade na redução de custos de memória.

O sistema híbrido Slicer-Nebuloso quando analisada sua versão com 7 partições, nota-se que seus resultados são melhores do que os obtidos no experimento 1, em vários domínios. Quando analisa-se os resultados da versão Slicer-Nebuloso com 3 partições, percebe-se que esta versão não obtém resultados equivalentes, alcançando valores bem abaixo das expectativas.

Nas comparações de desempenho, percebe-se que o Slicer atinge, em domínios sem atributos faltantes, a segunda posição com 29%. SVM é o melhor com 57% e o k-NN em terceiro com 14%. Já em domínios com atributos faltantes, o Slicer nem despontou, não conseguindo nem sequer ficar na faixa de 5% do melhor resultado. No entanto, vale ressaltar que o k-NN guarda a base de treinamento inteira na memória enquanto o SVM possui um custo computacional extremamente alto, conforme exposto em [Tran et. al. 2005].

CAPÍTULO 6

Conclusões

É proposto neste trabalho um sistema classificador indutivo, geométrico, multiclasse, que possui parada garantida, simplicidade de implementação, aplicável a problemas não linearmente separáveis, que gera conhecimento facilmente visualizável (explanável) e que obteve resultados interessantes.

O sistema classificador proposto usa de conceitos geométricos e de separabilidade linear para obter vetores que dêem suporte à criação de hiperplanos. Este sistema classificador é de simples programação e garante terminar o treinamento em no máximo N iterações, gerando uma base de conhecimento (conjunto de regras) de fácil explanação, o que facilita sua compreensão.

Para efeito de análise de desempenho são considerados os resultados em [Tran et. al. 2005]. Nesse artigo os autores comparam 9 métodos de classificação em 31 bases de domínios de conhecimento encontradas em [Asuncion & Newman 2007]. Para avaliação de desempenho comparam-se os resultados obtidos em 12 domínios de conhecimento classificados através de 3 métodos (SVM, k-NN e MLP). Tanto o sistema proposto quanto os métodos SVM, k-NN e MLP obtêm bons resultados nos domínios considerados.

Nos resultados obtidos nota-se que o Slicer consegue em 57% das bases usadas para comparação resultados dentro da faixa de 5% do melhor desempenho, confirmando a potencialidade da proposta.

6.1 - Contribuições

Este trabalho investigou, propôs, implementou e testou várias idéias, sendo estas contribuições passíveis de serem extrapoladas para outras pesquisas. Entre outras, destacam-se as seguintes contribuições principais:

- um novo método de classificação, geométrico, com tempo de treinamento na ordem de $O(N^2)$ e com resultados de fácil interpretação geométrica;

- um método de classificação que permite o uso em bases com ruído e atributos faltantes;
- uma metodologia de rotação de vetor;
- uma metodologia de localização e descarte de ruídos em tempo de treinamento;
- uma metodologia de adaptações a fim de admitir atributos faltantes no treino e na classificação;
- uma investigação da validade de se reordenar regras obtidas, com o fim de se reduzir seu número, e possivelmente melhorar sua acurácia;
- uma proposta de mudança de dimensionalidade de um problema, por conversão dos exemplos originais em novos exemplos, resultantes do produto interno entre o vetor de atributos e os vetores de conversão (regras obtidas pelo Slicer-O);
- uma proposta de solução do problema de indefinição de classificação em sistemas nebulosos, pelo uso do conceito de distância das partições, passando a existir além do critério convencional de localização da regra vencedora, um segundo critério desempatador;
- um método que permite aos sistemas nebulosos extrapolarem as partições geradas da dimensão do problema para outra dimensão, talvez permitindo maior definição dos espaços descritos pelas regras de inferência.

Todos estes métodos, metodologias e propostas foram implementados e obtiveram resultados satisfatórios nas avaliações.

6.2 - Resultados alcançados

Nos resultados obtidos da avaliação do Slicer Básico conclui-se que possui acurácia como classificador, obtendo 57% de resultados acima de 80% de acurácia. Porém a versão Slicer-F não obteve resultados promissores, mas deve-se notar que os domínios utilizados para avaliação possuem mais de 25% de exemplos com atributos faltantes, e neste cenário obteve-se resultados acima de 80% de acurácia em algumas bases.

Ambas versões Slicer-R e Slicer-G conseguiram acurácias melhores que o Slicer Básico e ambas conseguiram redução no número final de regras produzidas. A combinação das duas também se mostrou satisfatória.

O Slicer-O sempre obtém conjunto de regras mais compacto do que o que lhe é dado como entrada, embora em domínios com atributos faltantes sua acurácia tenha caído. Como um dos objetivos do Slicer-O é reduzir o conjunto de regras para uso com outros classificadores, pode-se dizer que obteve pleno sucesso nesse quesito.

Ou seja, em linhas gerais, experimentos mostraram que os investimentos nas versões Slicer-G, Slicer-R e Slicer-O foram justificadas. Indicaram também que a versão Slicer-F embora interessante no conceito, deve ter suas adaptações repensadas, pois seus resultados não foram convincentes.

Experimentos mostram ainda que o investimento na versão Slicer-Nebuloso com 7 partições obteve retornos interessantes, 55% melhores que os melhores resultados das outras versões do Slicer, mostrando que é uma proposta com possibilidades reais. Já a versão Slicer-Nebuloso com 3 partições mostrou-se inadequado.

Percebe-se por meio de experimentos que o Slicer se equipara com os outros métodos (SVM, kNN, MLP), em domínios sem atributos faltantes, obtendo melhor resultado em 2 destes. Porém entre os domínios que possuem atributos faltantes seu desempenho não se mostra equivalente.

Observando em particular a comparação entre Slicer e SVM, percebe-se que o SVM consegue resultados superiores, porém deve-se lembrar que o SVM é vitima do mal da dimensionalidade; por outro lado, observando-se em particular a comparação entre o Slicer e o k-NN percebe-se a superioridade do k-NN em domínios com atributos faltantes, mas vale lembrar que o k-NN guarda todo conjunto de exemplos na memória, indicando alto consumo de memória.

6.3 - Pesquisas futuras

Investigações de possíveis melhorias nos algoritmos e parâmetros, buscando aumentar a acurácia e reduzir o conjunto de regras final. Entre estas melhorias estão:

- investigar novos limites para Slicer-G, com outros números para nrogap e tamgap;
- investigar outras formas de gerar a rotação do vetor no Slicer-R;
- criação de uma versão Slicer-C, utilizando os vértices do "convex hull" [Preparata & Shamos 1985] do conjunto de exemplos como possíveis geradores de vetor ortogonal;

- investigar uma variação no Slicer-Nebuloso, buscando obter as possíveis classes de um novo exemplo em termos de porcentagem de cada;
- investigar novos parâmetros para as partições do Slicer-Nebuloso;
- redução do número das regras nebulosas geradas pelo método Wang-Mendel;
- limitar o número de regras geradas pelo Slicer-O passadas como entrada para o Slicer-Nebuloso;
- implementar filtro nas variáveis do Slicer-Nebuloso, eliminando variáveis que pioram ou não afetam a acurácia final do Slicer-Nebuloso;
- investigar a possibilidade de criar um Slicer-Nebuloso "on-line", atualizando sua base de regras periodicamente;
- inplementar no processo de classificação (tanto no Slicer quanto no Slicer-Nebuloso) outras formas de resolver o impasse quando a base de regras não consegue determinar a classe, utilizando possivelmente heurísticas como freqüência, distância, classe padrão e erro na classificação;
- implementar ponderação nas regras slicer quando usadas como entrada no Slicer-Nebuloso;
- gerar regras nebulosas através de outros métodos [Nozaki et al. 1997];
- testar outros operadores t-norma;
- comparar os métodos fuzzy clássico e Slicer-Nebuloso, ambos considerando distância e erro como formas de determinação de classe em situações de indeterminação;
- utilizar testes estatísticos na comparação dos resultados desta dissertação e nos novos resultados a serem obtidos [Demšar 2006].

REFERÊNCIAS BIBLIOGRÁFICAS

1) [Aho & Ullman 1992] Aho, Alfred.V., Ullman, Jeffrey D., “Foundations of Computer Science”, W. H. Freeman and Company, USA, 1992.

2) [Asuncion & Newman 2007] Asuncion, A. & Newman, D.J., UCI Machine Learning Repository [http://www.ics.uci.edu/~mlearn/MLRepository.html], Irvine, CA: University of California, School of Information and Computer Science, 2007.

3) [Briscoe & Caelli 1996] Briscoe, G., Caelli, T., “A Compendium of Machine Learning”, Volume 1: Symbolic Machine Learning, Ablex Publishing Corp, New Jersey, USA, 1996.

4) [Cormen et al. 2002] Cormen, Thomas H. et al., “Algoritmos: teoria e prática”, ed. Rio de Janeiro: Elsevier, 2002.

5) [Cristiani & Taylor 2000] Cristiani, N., Shawe-Taylor, J., “An Introduction to Support Vector Machines and other kernel-based learning methods”, Cambridge University Press, USA, 2000.

6) [de Berg et al. 2000] de Berg, M., van Kreveld, M., Overmans,M., and Schwarzkopf, O., “Convex Hulls: Mixing Things”, San Francisco, CA, 1995.

7) [Demšar 2006] Demšar, J., “Statistical comparisons of classifiers over multiple data sets”, Journal of Machine Learning Research. vol. 7, 1-30, 2006.

8) [Dobkin & Gunopulos 1995] Dobkin, D.P., Gunopulo, D., “Concept Learning with geometric hypotheses”,Annual Workshop on Computational Learning Theory, Proceedings of the eighth annual conference on Computational learning theory, pp 329-336, 1995.

9) [Duda et al. 2001] Duda, R. O., Hart, P. E., Stork, D. G., “Pattern Classification”, John Wiley & Sons, USA, 2001.

10) [Elizondo 2006] Elizondo, D., “The linear separability problem: some testing methods”, IEEE Transactions on Neural Networks, vol. 17, no. 2, pp 330-344, 2006.

11) [Gates 1972] Gates, G.W., ”The reduced nearest neighbor rule”, IEEE Transactions on Information Theory, vol.IT 18, no. 3, pp 431-433, 1972.

12) [Hand et al. 2001] Hand, David, Mannila, Heikki, Smyth, Padhraic, “Principles of Data Mining”, MIT Press, Cambridge, 2001.

13) [Hart 1968] Hart, P.E., “The condensed nearest neighbor rule”, IEEE Transactions on Information Theory, vol.IT 14, no. 3, pp 515-516, 1968.

14) [Hellerstein & Servedio 2007] Hellerstein L. and Servedio R.A. “On PAC learning algorithms for rich Boolean function classes”, Theoretical Computer Science, 384, pp 66–76, 2007.

15) [Klir & Yuan 1995] Klir, George J., Yuan, Bo, “Fuzzy Sets and Fuzzy Logic, Theory and Applications”, Prentice may, New Jersey, 1995.

16) [Mitchell 1997] Mitchell, T., “Machine Learning”, McGraw-Hill Series in Computer Science, McGraw-Hill, 1997.

17) [Neto e Nicoletti 2005] Neto, Luiz G. P., Nicoletti, Maria do Carmo, “Introdução às Redes Neurais Construtivas”, EdUFSCar, 2005.

18) [Nilsson 1965] Nilsson, N. J., “The Mathematical foundations of Learning Machines”, McGrall-Hill Systems Science Series, McGraw-Hill, USA, 1965.

19) [Nozaki et al. 1997] K. Nozaki, H. Ishibuchi, H. Tanaka, “A Simple but Powerful Heuristic Method for Generating Fuzzy Rules From Numerical Data”, Fuzzy Sets and Systems, Vol. 86, No. 3, pp. 251-270, 1997.

20) [Pedrycz 1998] Pedrycz, Witold, “Computational Intelligence: An Introduction”, CRC Press LLC, Boca Raton, 1998.

21) [Pedrycz & Gomide 1998] Pedrycz, Witold; Gomide, Fernando, “An Introduction to Fuzzy Set Analysis and Design”, MIT Press, Cambridge, 1998.

22) [Poulard 1995] Poulard, H., “Barycentric correction procedure: a fast method of learning threshold units”, Proc. WCNN´95, vol.1, pp. 710-713, 1995.

23) [Preparata & Shamos 1985] Preparata, F.P., Shamos, M.I., “Computational Geometry – An Introduction”, texts and monographs in computer science, Springer-Verlag, 1985.

24) [Ruján & Marchand 1989] Ruján, P., Marchand, M., “A geometric approach to learning in neural networks”, IJCNN 1989, vol. 2, pp 105-109, 1989.

25) [Tajine et al. 1997] Tajine, M., Elizondo, D., Fiesler, E., Korczak, J., “The Class of Linear Separability Method”, ESANN’1997, 1997.

26) [Tajine & Elizondo 1998] Tajine, M., Elizondo, D., “Growing methods for constructing recursive deterministic perceptron neural networks and knowledge extraction”, Artificial Intelligence, Elsevier 102, pp 295-322, 1998.

27) [Theodoridis & Koutroumbas 1999] Theodoridis, S., Koutroumbas, K., “Pattern recognition”, Academic Press, USA, 1999.

28) [Tran et al. 2005] Tran, Q. L., Toh, K. A., .Srinivasan, D., Wong, K. L. and Low, S. Q. C., “An empirical Comparasion of Nine Pattern Classifiers”, IEEE Trans. Syst. Man. Cyb.-part B:Cyb, vol.35, no.5, pp.1079-1091, 2005.

29) [Tuma et AL. 2008] Tuma, C.C.M., Horta, A.C.L., Zangirolami, T.C., Nicoletti, M.C., “Identifying the growth phase of S. Pneumoniae cultivations using a linear discriminant machine learning algorithm”, XVII COBEQ 2008, 2008.

30) [Vance & Ralescu 2006] Vance, D., Ralescu, A., “The Hyperplane Algorithm - A Decision Tree Using Oblique Lines”, Proceedings of the 17th Midwest Artificial Intelligence and Cognitive Science Conference (MAICS), 2006.

31) [Wang & Mendel 1992] Wang, L., Mendel, J., ”Generating fuzzy rules by learning from examples”, IEEE Trans. On SMC, vol. 22, pp 1414-1427, 1992.

32) [Webb 2007] Webb, G.I., “Discovering significant patterns”, Machine Learning Journal, Springer Netherlands, julho/2007, Vol. 68, Number 1, pp 1-33, 2007.

33) [Weiss & Kulikowski 1990] Weiss, Sholom M., Kulikowski, Casimir A., “Computer systems that learn”, Morgan Kaufmann Publishers, Inc., 1990.

34) [Witten & Frank 2005] Witten, Ian H., Frank, Eibe, “Data mining, practical machime learning tools and techniques”, Elsevier, 2005.

35) [Young & Downs 1998] Young S. and Downs, T., “CARVE – a constructive algorithm for real-valued examples”, IEEE Transactions on Neural Networks, vol. 9, no. 6, pp 1180-1190, 1998.

APÊNDICE

A seguir se encontram as 8 tabelas com os resultados dos 2 experimentos descritos no Capítulo 5.

Todas as tabelas possuem 3 acurácias para cada teste (mínima, média e máxima), desvio padrão da acurácia média, 3 valores para HP (mínimo, médio, máximo), desvio padrão para o HP médio. Quando as medidas são de testes do Slicer-Nebuloso, segundo experimento, no lugar de HP tem-se regras.

Experimento 1

Tabela A1 – Resultados para Slicer Básico + Slicer-F e Slicer-O.

Domínio	Slicer Básico				Slicer-O			
	Acurácias	dp	HP	dp	Acurácias	dp	HP	dp
Gauss00	0.9850 0.9892 0.9975	0.0037	5.00 5.15 5.40	0.1285	0.9850 !$ 0.9950 1.0000	0.0042	3.00 3.17 3.60	0.2283
Gauss20	0.9300 0.9488 0.9700	0.0108	19.40 20.62 21.60	0.5381	0.9300 ! 0.9517 0.9675	0.0108	17.00 17.74 18.60	0.5004
Gauss50	0.8975 0.9072 0.9150	0.0067	42.30 44.23 46.20	1.1714	0.8950 ! 0.9093 0.9175	0.0068	34.80 37.11 38.40	1.1113
Gauss75	0.6875 0.7118 0.7500	0.0174	117.30 119.98 123.60	1.7469	0.6675 0.7007 0.7425	0.0204	100.80 104.25 106.70	1.6741
Haberman's survival	0.4795 0.6353 0.7040	0.0583	89.60 91.58 95.10	1.6952	0.4521 ! 0.6268 0.6823	0.0628	73.80 75.47 78.50	1.3929
Musk database 1	0.8116 0.8362 0.8696	0.0165	80.70 81.71 83.10	0.7816	0.7778 ! 0.8295 0.8647	0.0238	68.90 70.19 71.80	0.9803
Spect heart 1	0.9140 0.9323 0.9624	0.0149	44.30 45.00 45.60	0.4195	0.8763 0.9113 0.9516	0.0257	33.00 33.94 35.50	0.7200
Spect heart 2	0.8255 0.8509 0.8632	0.0130	66.30 68.20 71.00	1.4574	0.8208 0.8373 0.8538	0.0114	52.80 54.65 56.80	1.1209
Hayes-Roth	0.5556 $ 0.6692 0.7818	0.0673	16.60 17.84 18.50	0.5333	0.3611 0.6140 0.7297	0.0994	12.30 13.15 14.30	0.6217
Iris plants	0.9103 0.9424 0.9586	0.0129	9.10 9.37 9.70	0.2283	0.9103 ! 0.9404 0.9592	0.0129	7.80 8.13 8.50	0.2052
Flags	0.2957 0.3636 0.4425	0.0470	115.90 117.57 118.50	0.7417	0.3077 ! 0.3581 0.4485	0.0440	103.10 104.97 107.50	1.2775
Pittsburgh bridges version 1 TORD	0.7941 $ 0.8300 0.8824	0.0235	16.40 17.09 17.60	0.3833	0.3824 0.4773 0.5980	0.0592	7.40 9.18 10.30	0.8807
Pittsburgh bridges version 1 MATERIAL	0.4242 0.5447 0.6053	0.0574	14.20 14.62 15.30	0.3429	0.1579 0.2421 0.3256	0.0533	5.10 5.72 6.40	0.4020
Pittsburgh bridges version 1 SPAN	0.5373 0.5768 0.6479	0.0358	25.70 26.33 26.80	0.3607	0.1084 0.1596 0.2075	0.0265	11.10 11.45 11.80	0.2156
Pittsburgh bridges version 1 REL-L	0.5000 0.5428 0.6092	0.0342	29.60 30.41 31.20	0.6252	0.1667 0.2141 0.2667	0.0308	8.80 9.46 10.90	0.5352
Pittsburgh bridges version 1 TYPE	0.3056 0.3808 0.4353	0.0403	38.00 38.88 39.50	0.4377	0.0781 0.1247 0.1912	0.0302	10.40 11.29 12.10	0.5394

A marca ! indica que a acurácia do Slicer-O é melhor-ou-igual a do conjunto de regras que o originou.
A marca $ indica que a acurácia é a melhor desta bateria de testes.
Acurácias com fundo ressaltado indicam melhores resultados entre os dois experimentos.

Tabela A2 – Resultados para Slicer Básico+Slicer-F+Slicer-R e Slicer-O.

Domínio	Slicer Básico+Slicer-F+Slicer-R				Slicer-O			
	Acurácias	dp	HP	dp	Acurácias	dp	HP	dp
Gauss00	0.9775 * 0.9855 0.9950	0.0047	6.00 6.37 6.70	0.2410	0.9775! 0.9860 0.9950	0.0044	5.60 6.10 6.60	0.2720
Gauss20	0.9350 * 0.9463 0.9550	0.0056	20.70 21.08 21.80	0.3370	0.9375! 0.9478 0.9600	0.0062	17.70 18.39 19.30	0.4949
Gauss50	0.8900 * 0.9075 0.9375	0.0121	38.90* 40.51 42.30	1.0024	0.8950 ! 0.9075 0.9325	0.0096	34.30 35.43 36.50	0.7086
Gauss75	0.6775 * 0.7095 0.7400	0.0167	112.60* 113.97 115.50	1.0469	0.6800! 0.7070 0.7425	0.0179	97.20 100.01 102.20	1.5023
Haberman's survival	0.5411 * 0.6499 0.6895	0.0413	83.40 * 85.31 87.30	1.1870	0.4863 0.6390 0.6751	0.0560	70.20 72.60 75.40	1.5735
Musk database 1	0.8019 *$ 0.8435 0.8744	0.0196	69.00 * 69.49 70.00	0.3270	0.8068 ! 0.8367 0.8696	0.0214	59.50 61.58 62.60	0.8207
Spect heart 1	0.9032 * 0.9301 0.9570	0.0167	39.00 * 39.70 40.40	0.4266	0.8495 0.9086 0.9516	0.0258	30.40 31.37 32.10	0.5883
Spect heart 2	0.8113 * 0.8533 0.8821	0.0206	57.30* 58.68 60.30	1.0068	0.7972 0.8401 0.8679	0.0217	47.80 48.82 50.50	0.8518
Hayes-Roth	0.5278 0.6541 0.7500	0.0721	14.90* 15.74 16.40	0.4800	0.4444 0.6065 0.7222	0.0812	12.80 13.78 14.60	0.5192
Iris plants	0.9310 * 0.9438 0.9524	0.0068	8.90 9.43 10.00	0.3407	0.9241 ! 0.9438 0.9592	0.0101	8.00 8.44 8.70	0.2375
Flags	0.2957* 0.3654 0.4513	0.0530	111.60* 113.15 114.30	0.8261	0.2435 0.3480 0.4194	0.0541	100.80 102.44 103.90	0.9308
Pittsburgh bridges version 1 TORD	0.7843 * 0.8202 0.8627	0.0234	14.90* 15.82 17.30	0.6194	0.2843 0.4368 0.5392	0.0671	5.70 8.42 10.00	1.2032
Pittsburgh bridges version 1 MATERIAL	0.4211 * 0.5421 0.6579	0.0678	13.30* 13.87 14.50	0.3951	0.0233 0.1789 0.3158	0.0766	4.10 4.97 5.80	0.4734
Pittsburgh bridges version 1 SPAN	0.4717 * 0.5678 0.6620	0.0515	23.70 * 25.13 26.20	0.6943	0.0625 0.1190 0.1930	0.0407	9.50 10.25 11.20	0.4500
Pittsburgh bridges version 1 REL-L	0.4634 * 0.5373 0.6207	0.0456	29.40 30.57 31.50	0.6885	0.2000 0.2721 0.3467	0.0480	10.20 10.55 11.10	0.2872
Pittsburgh bridges version 1 TYPE	0.2917 * 0.3890 0.4783	0.0547	36.40 * 36.99 37.80	0.3961	0.0313 0.0904 0.1667	0.0377	9.30 10.20 11.30	0.5441

A marca * indica que a acurácia é melhor-ou-igual que a obtida na Tabela A1, e o HP é menor.
A marca ! indica que a acurácia do Slicer-O é melhor-ou-igual a do conjunto de regras que o originou.
A marca $ indica que a acurácia é a melhor desta bateria de testes.
Acurácias com fundo ressaltado indicam melhores resultados entre os dois experimentos.

Tabela A3 – Resultados para Slicer Básico+Slicer-F+Slicer-G e Slicer-O.

Domínio	Slicer Básico+Slicer-F+Slicer-G				Slicer-O			
	Acurácias	dp	HP	dp	Acurácias	dp	HP	dp
Gauss00	0.9600 0.9715 0.9850	0.0079	3.00 * 3.12 3.40	0.1661	0.9575! 0.9698 0.9850	0.0083	2.90 3.08 3.30	0.1470
Gauss20	0.9350 * 0.9477 0.9700	0.0090	6.90* 7.22 7.50	0.1600	0.9450 !$ 0.9565 0.9750	0.0075	4.70 5.16 5.50	0.2245
Gauss50	0.8925 * 0.9072 0.9300	0.0102	11.50* 12.09 12.90	0.3780	0.8925 ! 0.9090 0.9275	0.0097	8.60 9.03 9.80	0.3822
Gauss75	0.6875 * 0.7205 0.7600	0.0195	33.50* 34.74 36.00	0.6020	0.7025 !$ 0.7242 0.7675	0.0197	23.00 24.11 25.00	0.5576
Haberman's survival	0.5274 * 0.6746 0.7112	0.0604	25.10* 26.30 27.80	0.7849	0.5548! $ 0.6904 0.7329	0.0552	16.10 17.05 18.70	0.7487
Musk database 1	0.6377 0.7391 0.8454	0.0610	38.30* 39.10 40.30	0.6678	0.6715 ! 0.7382 0.8454	0.0596	25.80 27.48 28.30	0.6911
Spect heart 1	0.9462 *$ 0.9720 0.9839	0.0107	10.80* 11.29 11.60	0.2427	0.9301 0.9608 0.9839	0.0148	7.90 8.21 8.70	0.3145
Spect heart 2	0.8491 * 0.8797 0.9151	0.0177	19.50* 20.87 21.80	0.6754	0.8538 !$ 0.8842 0.9104	0.0161	13.30 13.91 15.10	0.5718
Hayes-Roth	0.4167 0.5388 0.6545	0.0825	7.90* 8.50 9.10	0.4074	0.3611 0.5113 0.6216	0.0781	5.90 6.51 6.90	0.3618
Iris plants	0.9241 *$ 0.9513 0.9658	0.0110	3.20* 3.29 3.50	0.0831	0.9241 !$ 0.9513 0.9658	0.0110	3.00 3.00 3.00	0.0000
Flags	0.2530 0.3416 0.4086	0.0450	52.00* 53.96 57.00	1.4961	0.2410 ! 0.3407 0.4019	0.0515	30.50 31.04 32.00	0.4409
Pittsburgh bridges version 1 TORD	0.7843 0.8182 0.8529	0.0215	5.90* 6.29 6.70	0.2625	0.7745 0.8024 0.8333	0.0152	4.30 4.63 5.00	0.2002
Pittsburgh bridges version 1 MATERIAL	0.3488 * 0.5474 0.6579	0.0793	7.00* 7.44 7.90	0.2577	0.3023! $ 0.5500 0.6579	0.1041	4.60 5.24 5.80	0.3826
Pittsburgh bridges version 1 SPAN	0.3860 0.5497 0.6833	0.0764	11.10* 11.65 12.30	0.3106	0.3125 0.3870 0.5181	0.0577	5.40 6.49 7.10	0.4527
Pittsburgh bridges version 1 REL-L	0.4667 *$ 0.5566 0.6322	0.0506	13.10* 13.78 14.80	0.5325	0.3733 0.4641 0.5402	0.0502	7.00 8.07 9.30	0.7198
Pittsburgh bridges version 1 TYPE	0.3194 * 0.3822 0.4471	0.0404	18.30 * 19.14 20.20	0.5625	0.1324 0.2192 0.2941	0.0492	7.90 9.58 11.00	0.8841

A marca * indica que a acurácia é melhor-ou-igual que a obtida na Tabela A1, e o HP é menor.
A marca ! indica que a acurácia do Slicer-O é melhor-ou-igual a do conjunto de regras que o originou.
A marca $ indica que a acurácia é a melhor desta bateria de testes.
Acurácias com fundo ressaltado indicam melhores resultados entre os dois experimentos.

Tabela A4 - Resultados para Slicer Básico+Slicer-F+Slicer-R+Slicer-G e Slicer-O.

Domínio	Slicer Básico+Slicer-F+Slicer-R+Slicer-G				Slicer-O			
	Acurácias	dp	HP	dp	Acurácias	dp	HP	dp
Gauss00	0.9725 *# 0.9850 0.9925	0.0056	2.30*# 2.51 2.70	0.0943	0.9725! 0.9850 0.9950	0.0057	2.00 2.02 2.10	0.0400
Gauss20	0.9325 *# 0.9478 0.9600	0.0079	6.30 *# 6.65 7.00	0.2062	0.9300 ! 0.9510 0.9625	0.0094	5.20 5.81 6.10	0.2625
Gauss50	0.8975 *# 0.9132 0.9325	0.0110	10.00 *# 10.57 11.00	0.3257	0.8975!$ 0.9177 0.9350	0.0108	8.30 8.75 9.10	0.2579
Gauss75	0.6800 * 0.6998 0.7300	0.0144	31.70 *# 32.78 34.20	0.7652	0.6825 ! 0.7070 0.7525	0.0192	23.10 23.87 24.40	0.4076
Haberman's survival	0.5274 *# 0.6763 0.7360	0.0575	23.00 *# 23.80 24.30	0.3464	0.5274 ! 0.6876 0.7432	0.0605	16.30 16.88 17.20	0.2482
Musk database 1	0.6957 # 0.7874 0.8502	0.0496	32.20 *# 32.80 34.80	0.7014	0.6763 0.7696 0.8261	0.0532	22.70 23.81 25.00	0.6862
Spect heart 1	0.9301 * 0.9484 0.9785	0.0143	10.40 *# 10.79 11.20	0.2879	0.9086 ! 0.9495 0.9731	0.0180	7.80 8.17 8.50	0.2238
Spect heart 2	0.8443 *# 0.8774 0.9009	0.0197	14.80 *# 16.01 16.60	0.5029	0.8443 ! 0.8731 0.9009	0.0186	11.80 12.55 13.10	0.4478
Hayes-Roth	0.5000 # 0.5602 0.6444	0.0478	7.40*# 7.98 8.60	0.3124	0.4444 ! 0.5576 0.6667	0.0699	6.00 6.33 6.60	0.1792
Iris plants	0.9241 *#$ 0.9513 0.9658	0.0110	3.10*# 3.19 3.30	0.0539	0.9241 !$ 0.9513 0.9658	0.0110	3.00 3.00 3.00	0.0000
Flags	0.2788 *#$ 0.3690 0.4485	0.0558	48.70*# 50.69 52.50	1.2778	0.2959 0.3535 0.3971	0.0348	28.30 29.14 30.80	0.7592
Pittsburgh bridges version 1 TORD	0.7843 *# 0.8113 0.8333	0.0211	5.20 *# 5.57 5.90	0.1792	0.7451 0.7984 0.8431	0.0310	4.10 4.63 5.00	0.2830
Pittsburgh bridges version 1 MATERIAL	0.4474 *# 0.5421 0.6579	0.0532	6.90 * 7.61 8.30	0.4110	0.3947 0.5211 0.6053	0.0503	4.60 5.59 6.20	0.4989
Pittsburgh bridges version 1 SPAN	0.3509 *#$ 0.5949 0.6901	0.0943	10.20*# 10.66 11.10	0.2905	0.2394 0.3946 0.5352	0.0895	5.60 6.59 7.30	0.4784
Pittsburgh bridges version 1 REL-L	0.4512 0.5221 0.5867	0.0424	12.00 *# 12.93 14.00	0.5533	0.3333 0.4061 0.4833	0.0460	5.80 7.34 8.20	0.7116
Pittsburgh bridges version 1 TYPE	0.3194 *#$ 0.4110 0.4848	0.0472	18.00* 19.16 20.50	0.7526	0.1970 0.2986 0.4242	0.0684	7.70 9.41 10.40	0.9481

A marca * indica que a acurácia é melhor-ou-igual que a obtida na Tabela A2, e o HP é menor.
A marca # indica que a acurácia é melhor-ou-igual que a obtida na Tabela A3, e o HP é menor.
A marca ! indica que a acurácia do Slicer-O é melhor-ou-igual a do conjunto de regras que o originou.
A marca $ indica que a acurácia é a melhor desta bateria de testes.
Acurácias com fundo ressaltado indicam melhores resultados entre os dois experimentos.

Experimento 2

Tabela A5- Resultados para Slicer-Nebuloso usando hps vindos de Slicer Básico+ Slicer-F

Domínio	Slicer-Nebuloso 3 partições				Slicer-Nebuloso 7 partições			
	Acurácias	dp	Regras	dp	Acurácias	dp	Regras	dp
Gauss00	0.7400 0.8018 0.8525	0.0291	10,90 12,99 16,20	1.3568	0.9525 * 0.9797 0.9975	0.0146	48,10 51,80 59,40	3,8348
Gauss20	0.8325 0.8525 0.8775	0.0138	94.50 100.72 108.90	4.5834	0.9275 *# 0.9470 0.9650	0.0109	233.40 242.11 248.90	4.8399
Gauss50	0.7775 0.8145 0.8575	0.0215	173.70 181.31 187.20	4.2630	0.8875 *# 0.9035 0.9175	0.8807	308.70 313.51 319.00	3.7843
Gauss75	0.6275 0.6603 0.6875	0.0186	257.50 262.80 268.30	3.0659	0.6600 # 0.6768 0.7100	0.0155	347.30 348.15 349.20	0.5954
Haberman's survival	0.5342 * 0.6629 0.7004	0.0447	226.00 232.86 242.60	4.7204	0.5616 *# 0.6645 0.6931	0.0369	247.60 251.32 261.00	4.3027
Musk database 1	0.7633 0.8155 0.8502	0.0292	434.80 437.14 438.40	1.1775	0.7585 0.8261 0.8841	0.0434	451.90 452.37 453.00	0.3195
Spect heart 1	0.8978 * 0.9237 0.9462	0.0159	129.30 159.16 173.90	15.3512	0.8871 0.9177 0.9516	0.0186	110.70 133.50 158.80	13.8534
Spect heart 2	0.8066 * 0.8486 0.8962	0.0293	245.00 245.37 245.70	0.1847	0.8113 0.8340 0.8585	0.0158	245.80 245.80 245.80	0.0000
Hayes-Roth	0.4722 0.5664 0.6667	0.0610	43.70 46.54 48.10	1.2714	0.3056 # 0.6015 0.7297	0.1133	40.90 44.89 47.10	2.0270
Iris plants	0.6000 0.7437 0.8231	0.0600	18.80 19.76 21.10	0.8616	0.9034 0.9198 0.9521	0.0146	53.00 56.94 63.20	3.0542
Flags	0.2857 * 0.3864 0.4516	0.0561	175.30 178.15 179.90	1.4179	0.3462 *# 0.4212 0.4779	0.0415	176.80 179.45 181.80	1.4800
Pittsburgh bridges version 1 TORD	0.7451 0.8043 0.8529	0.0369	28.60 36.62 42.10	4.2951	0.7843 *# 0.8281 0.8529	0.0231	55.60 61.95 66.20	3.6759
Pittsburgh bridges version 1 MATERIAL	0.3953 0.5000 0.6053	0.0601	18.30 20.15 23.60	1.5292	0.3421 0.4842 0.6053	0.0679	36.90 45.36 52.60	4.9358
Pittsburgh bridges version 1 SPAN	0.4561 0.5211 0.5783	0.0421	24.70 26.31 30.30	1.5287	0.4906 * 0.6009 0.6901	0.0658	52.40 55.08 56.70	1.3029
Pittsburgh bridges version 1 REL-L	0.3867 0.4876 0.5698	0.0617	26.00 28.53 32.90	1.8826	0.4933 * 0.5552 0.6667	0.0556	56.20 60.80 65.90	2.9189
Pittsburgh bridges version 1 TYPE	0.2027 0.2630 0.3333	0.0419	30.10 34.72 37.60	2.0478	0.3108 * 0.4178 0.5067	0.0554	66.30 69.47 72.50	1.7838

A marca * indica que a acurácia é melhor-ou-igual que a obtida na Tabela A1, na segunda coluna.
A marca # indica melhor acurácia do experimento 2.
Acurácias com fundo ressaltado indicam melhores resultados entre os dois experimentos.

Tabela A6- Resultados para Slicer-Nebuloso usando Slicer Básico+Slicer-F+Slicer-R.

Domínio	Slicer-Nebuloso 3 partições				Slicer-Nebuloso 7 partições			
	Acurácias	dp	Regras	dp	Acurácias	dp	Regras	dp
Gauss00	0.9125 0.9340 0.9625	0.0152	28,50 30,74 33,00	1,4562	0.9775 *# 0.9918 0.9975	0.0057	98,90 107,90 119,30	5,6877
Gauss20	0.7825 0.8182 0.8475	0.0192	96.90 102.56 107.10	3.0764	0.9325 * 0.9423 0.9550	0.0061	239.80 245.34 249.80	3.8648
Gauss50	0.8000 0.8117 0.8250	0.0087	168.60 174.76 180.90	3.5601	0.8775 * 0.8993 0.9100	0.0092	305.60 310.78 314.00	2,5266
Gauss75	0.6375 0.6683 0.7000	0.0217	258.60 260.94 265.30	1.8964	0.6450 0.6700 0.6950	0.0151	346.00 346.98 348.00	0.5862
Haberman's survival	0.5068 * 0.6540 0.6954	0.0523	224.60 230.14 239.50	4.9176	0.5342 * 0.6544 0.7004	0.0449	245.90 250.77 261.30	4.3308
Musk database 1	0.7971 0.8169 0.8406	0.0176	425.30 428.37 433.10	1.9611	0.7971 * 0.8386 0.8744	0.0254	451.70 452.06 452.50	0.2871
Spect heart 1	0.8925 *# 0.9263 0.9516	0.0192	154.90 174.62 183.80	9.9231	0.8817 0.9140 0.9409	0.0185	143.40 164.47 179.20	11.1149
Spect heart 2	0.8066 0.8429 0.8774	0.0231	244.50 244.86 245.60	0.2871	0.7877 0.8410 0.8726	0.0251	245.80 245.80 245.80	0.0000
Hayes-Roth	0.4444 0.5388 0.6111	0.0484	44.10 45.74 48.80	1.4820	0.3889 0.5865 0.7818	0.1071	37.70 42.11 48.30	3.3987
Iris plants	0.6828 0.7937 0.8844	0.0658	18.70 22.06 24.30	1.8880	0.8966 # 0.9280 0.9524	0.0175	57.90 65.42 69.60	4.2682
Flags	0.2651 * 0.3755 0.4425	0.0587	175.40 178.06 180.80	1.6409	0.3269 * 0.4075 0.4690	0.0480	176.60 179.45 182.30	1.6064
Pittsburgh bridges version 1 TORD	0.7549 0.8014 0.8511	0.0290	20.10 31.32 37.30	4.8481	0.7843 * 0.8152 0.8627	0.0263	47.30 57.91 61.90	4.4332
Pittsburgh bridges version 1 MATERIAL	0.3023 0.4184 0.5263	0.0650	12.60 15.79 18.20	1.8727	0.4474 0.5289 0.6053	0.0445	35.20 44.58 50.60	4.3241
Pittsburgh bridges version 1 SPAN	0.3521 0.4849 0.6167	0.0843	19.60 21.76 25.90	1.7800	0.4912 * 0.6009 0.6833	0.0624	43.70 48.06 51.90	2.2245
Pittsburgh bridges version 1 REL-L	0.4063 0.4972 0.5930	0.0706	31.60 34.75 37.70	2.4422	0.4667 * 0.5511 0.6667	0.0627	65.60 67.05 69.20	1.345
Pittsburgh bridges version 1 TYPE	0.2500 0.3151 0.4242	0.0519	27.90 32.75 35.40	2.3863	0.3472 * 0.4301 0.5000	0.0475	63.00 66.36 69.00	1.5825

A marca * indica que a acurácia é melhor-ou-igual que a obtida na Tabela A2, na segunda coluna.
A marca # indica melhor acurácia do experimento 2.
Acurácias com fundo ressaltado indicam melhores resultados entre os dois experimentos.

Tabela A7 – Resultados para Slicer-Nebuloso usando Slicer-O sobre Slicer Básico+Slicer-F+Slicer-G.

Domínio	Slicer-Nebuloso 3 partições				Slicer-Nebuloso 7 partições			
	Acurácias	dp	Regras	dp	Acurácias	dp	Regras	dp
Gauss00	0.7675 0.8168 0.8475	0.0232	10.30 11.81 13.20	0.8117	0.9600 * 0.9795 1.0000	0.0137	37.70 41.34 46.90	3.0819
Gauss20	0.7150 0.7513 0.7925	0.0233	24.00 27.58 31.00	2.3051	0.9275 * 0.9397 0.9500	0.0075	84.20 89.32 98.00	4.2216
Gauss50	0.6975 0.7200 0.7550	0.0190	44.00 50.15 54.30	3.1617	0.8700 0.8847 0.9050	0.0119	137.70 152.79 161.30	6.5370
Gauss75	0.6050 0.6422 0.6875	0.0210	119.10 122.68 129.60	2.9546	0.6375 0.6695 0.6975	0.0190	257.00 264.20 269.90	4.1400
Haberman's survival	0.4795 0.6236 0.6715	0.0621	81.60 88.42 96.30	4.3464	0.5479 0.6422 0.6787	0.0375	201.40 208.82 216.20	4.6606
Musk database 1	0.7101 * 0.7691 0.8309	0.0317	319.20 346.78 362.30	11.9600	0.7826 * 0.8242 0.8792	0.0300	443.80 447.18 449.00	1.3257
Spect heart 1	0.8065 0.8645 0.9086	0.0308	62.20 76.06 86.30	6.7149	0.8495 0.8855 0.9194	0.0203	178.60 183.16 185.90	2.2752
Spect heart 2	0.7925 *# 0.8698 0.9387	0.0447	85.80 97.44 107.90	7.1582	0.8302 0.8462 0.8726	0.0155	242.40 243.37 244.80	0.6084
Hayes-Roth	0.3889 0.5038 0.6000	0.0694	26.20 30.07 34.30	2.3656	0.5000* 0.5789 0.7091	0.0673	47.30 49.01 50.90	0.9843
Iris plants	0.7415 0.8561 0.9116	0.0458	8.10 8.80 9.30	0.4313	0.8767 0.9143 0.9521	0.0233	21.80 24.39 26.50	1.4342
Flags	0.3304 * 0.3837 0.4409	0.0337	166.40 169.73 171.30	1.3914	0.3163* 0.3855 0.4634	0.0466	174.80 177.96 180.60	1.6323
Pittsburgh bridges version 1 TORD	0.6078 0.7372 0.8085	0.0632	13.10 16.91 19.00	1.7913	0.7157 0.7688 0.8137	0.0306	43.60 47.14 50.60	2.6231
Pittsburgh bridges version 1 MATERIAL	0.3939 0.4632 0.5526	0.0529	18.20 22.80 28.60	3.1126	0.4186 * 0.5395 0.6053	0.0646	51.10 59.10 68.40	5.7432
Pittsburgh bridges version 1 SPAN	0.4000 0.4819 0.5422	0.0391	17.70 24.04 28.60	2.8932	0.4386 * 0.5663 0.6747	0.0678	43.90 50.79 55.90	3.0713
Pittsburgh bridges version 1 REL-L	0.4267 0.5110 0.5977	0.0501	26.50 35.96 42.80	5.2227	0.4688 * 0.5566 0.6322	0.0471	58.40 65.28 72.70	4.3185
Pittsburgh bridges version 1 TYPE	0.3472 * 0.4178 0.4559	0.0341	26.10 35.71 40.80	4.1748	0.4028 *# 0.4658 0.5303	0.0390	53.40 63.08 68.20	4.1868

A marca * indica que a acurácia é melhor-ou-igual que a obtida na Tabela A3, na segunda coluna.
A marca # indica melhor acurácia do experimento 2.
Acurácias com fundo ressaltado indicam melhores resultados entre os dois experimentos.

Tabela A8 – Resultados para Slicer-Nebuloso usando hps vindos Slicer-O vindo Slicer Básico+Slicer-F+Slicer-R+Slicer-G.

Domínio	Slicer-Nebuloso 3 partições				Slicer-Nebuloso 7 partições			
	Acurácias	dp	Regras	dp	Acurácias	dp	Regras	dp
Gauss00	0.7650 0.8227 0.8675	0.0326	6.00 6.34 7.10	0.3323	0.9350 0.9650 0.9950	0.0179	22.00 23.03 25.20	1.1010
Gauss20	0.7475 0.7695 0.7950	0.0146	26.60 30.95 35.50	2.7933	0.9225 * 0.9380 0.9675	0.0127	91.70 100.81 110.20	6.4221
Gauss50	0.6850 0.7205 0.7525	0.0186	41.90 46.56 49.60	2.4467	0.8750 0.8917 0.9075	0.0107	138.80 148.73 156.40	5.9942
Gauss75	0.6225 0.6538 0.6925	0.0231	115.20 120.88 123.80	2.4641	0.6525 0.6733 0.7000	0.0166	259.40 262.67 267.30	2.6401
Haberman's survival	0.5342 0.6288 0.6701	0.0408	83.10 90.31 98.70	3.9906	0.5479 0.6341 0.6534	0.0287	203.30 208.54 215.70	3.5663
Musk database 1	0.6957 0.7633 0.8357	0.0472	275.00 298.60 318.10	11.4884	0.7874 *# 0.8401 0.9082	0.0299	437.80 442.26 444.60	2.0490
Spect heart 1	0.8011 0.8651 0.9247	0.0352	74.90 85.92 99.60	8.2693	0.8656 0.8914 0.9247	0.0216	163.20 182.03 185.70	6.3452
Spect heart 2	0.8160 0.8491 0.8962	0.0273	90.70 97.27 103.90	3.8427	0.8160 0.8552 0.8726	0.0200	240.40 242.38 243.50	0.8412
Hayes-Roth	0.3889 0.5288 0.6531	0.0736	28.90 30.76 32.60	1.1074	0.4722 * 0.5707 0.6111	0.0411	48.00 50.01 50.90	0.8105
Iris plants	0.7279 0.8472 0.8980	0.0460	7.80 8.76 9.20	0.4758	0.8836 0.9136 0.9521	0.0202	21.90 24.39 26.40	1.4391
Flags	0.3163 * 0.3709 0.4602	0.0444	165.70 167.98 170.30	1.5721	0.2530 * 0.3855 0.4715	0.0649	175.40 177.47 179.90	1.4812
Pittsburgh bridges version 1 TORD	0.6064 0.7530 0.8235	0.0610	14.30 18.32 20.30	1.7371	0.7157 0.7875 0.8529	0.0457	48.00 52.80 58.10	2.9048
Pittsburgh bridges version 1 MATERIAL	0.3684 0.4816 0.5814	0.0525	21.40 27.29 32.60	3.3872	0.4545 *# 0.5553 0.6579	0.0633	52.60 62.84 74.10	5.5467
Pittsburgh bridges version 1 SPAN	0.3684 0.5181 0.6269	0.0680	20.70 25.94 29.00	2.4163	0.4912*# 0.6054 0.6620	0.0454	42.00 50.68 56.90	3.9219
Pittsburgh bridges version 1 REL-L	0.4531 * 0.5331 0.6782	0.0663	26.40 32.82 38.20	3.6235	0.5000 *# 0.5580 0.6667	0.0489	60.80 63.64 69.40	2.9255
Pittsburgh bridges version 1 TYPE	0.2353 0.3425 0.4353	0.0568	24.10 35.50 43.80	5.7196	0.3824* 0.4438 0.5647	0.0531	51.50 64.45 74.20	6.2771

A marca * indica que a acurácia é melhor-ou-igual que a obtida na Tabela A4, na segunda coluna.
A marca # indica melhor acurácia do experimento 2.
Acurácias com fundo ressaltado indicam melhores resultados entre os dois experimentos.