AUGUSTO DE ALMEIDA PRADO GAVA TORÁCIO

APRENDIZADO DE REGRAS DE CLASSIFICAÇÃO COM OTIMIZAÇÃO POR NUVEM DE PARTÍCULAS MULTIOBJETIVO

CURITIBA
ABRIL 2008

UNIVERSIDADE FEDERAL DO PARANÁ
DEPARTAMENTO DE INFORMÁTICA

APRENDIZADO DE REGRAS DE CLASSIFICAÇÃO COM OTIMIZAÇÃO POR NUVEM DE PARTÍCULAS MULTIOBJETIVO

Dissertação de mestrado em informática apresentada ao programa de pós-graduação em informática da Universidade Federal do Paraná como requisito parcial à obtenção do título de mestre em informática.

Orientadora: Profª. Drª. Aurora Pozo

AUGUSTO DE ALMEIDA PRADO GAVA TORÁCIO

CURITIBA
ABRIL 2008

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

RESUMO

Este trabalho apresenta um algoritmo de aprendizado de regras de classificação que faz uso da técnica de otimização por nuvem de partículas multiobjetivo para obter um conjunto de melhores regras para a classificação, como uma alternativa aos algoritmos de seleção por cobertura, procurando evitar a perda da qualidade das regras. No treinamento, com uma base de dados de entrada, as regras são representadas como as posições das partículas que voam num espaço de busca, seguindo suas melhores posições ou as melhores posições dos seus vizinhos. As melhores regras são memorizadas no classificador, com base nos conceitos de dominância de Pareto, o que permite que sejam selecionadas regras com diversas características, que podem ser definidas pelo usuário. As regras são selecionadas para o classificador ao mesmo tempo em que são descobertas e, diferentemente dos algoritmos de cobertura, a seleção não retira exemplos cobertos da base. Por isso, as regras encontradas podem ser lidas de modo isolado, pois o classificador obtido é não ordenado e as regras não perdem qualidade. Foram feitos experimentos mostrando a diferença de comportamento de diferentes valores dos parâmetros de nuvem de partículas na exploração do espaço e na obtenção da fronteira de Pareto. Outro experimento mostrou que um aumento no número de regras selecionadas na fronteira, em geral, trouxe maior desempenho de classificação, com os objetivos de confiabilidade positiva e negativa. Comparações com demais algoritmos da literatura mostram que o algoritmo proposto é competitivo com relação à área abaixo da curva ROC (*Receiver Output Curve)* e número de regras no classificador. Além disso, as regras selecionadas têm alto grau de suporte e precisão relativa ponderada (*Wracc*), o que denota que as regras são importantes mesmo isoladas. Dessa forma, produz-se um conjunto, que é bom para a classificação, com regras muito boas, que podem trazer conhecimentos, mesmo se analisadas de forma isolada.

ABSTRACT

This work presents an algorithm of classification-rule learning that uses the multiple objective particle swarm optimization technique to obtain a set of the best rules to classification as an alternative way to covering algorithms, avoiding the loss of quality of the rules. In the training, with an input dataset, the rules are represented as the positions of the particles that fly in a search space, following their best positions or their neighbors' best positions. The best rules are memorized in the classifier, based in concepts of Pareto's dominance, which permits the selection of rules with several characteristics that can be defined by the user. The rules are selected to the classifier in the same time that they are discovered and, differently of the cover algorithms, the selection does not remove examples from the dataset. Therefore, the found rules can be read in isolation, because the obtained classifier is non-ordered and the rules do not lose quality. An experiment was conducted, showing the difference of the behavior of the obtainment of Pareto's front and space exploration, varying values of the particle swarm parameters. Another experiment showed that an increase in the number of selected rules in the front, generally, brought greater classification performance, with the objectives of positive and negative confidence. Comparisons with other algorithms of the literature show that the proposed algorithm is competitive related to the area under ROC (*Receiver Output Curve)* and number of rules in the classifier. Besides, the selected rules have high support and weighted relative accuracy (Wracc), which denotes the rules are important even isolated. In this way, it produces a set that is good for classification, with very good rules that can bring knowledge even if they are analyzed in isolation.

1 INTRODUÇÃO

A capacidade de perceber a informação, classificá-la e gerar novo conhecimento é a forma com que nós interagimos com o ambiente. Há muitos anos, o homem procura entender como esse processo funciona. Com o advento de computadores, tem sido de grande interesse a tentativa de simular essa capacidade, tanto do meio acadêmico, como do industrial. Com isso, uma nova área de pesquisa surge na computação — o Aprendizado de Máquina (AM).

O Aprendizado de Máquina é uma área da Inteligência Artificial (IA) que busca a construção de sistemas capazes de adquirir o conhecimento de forma automática, bem como o desenvolvimento de técnicas computacionais de aprendizagem [1].

De modo geral, há dois principais tipos de aprendizado: indutivo e dedutivo. A indução é um processo cognitivo que, a partir do exame de um certo número de casos, chega à formulação de uma lei geral, cujo alcance se estende para além dos casos considerados. Quando todos os possíveis casos são analisados, a indução é dita perfeita, sendo suas conclusões evidentemente exatas. Quando isso não acontece, a indução é dita imperfeita, possuindo apenas um valor estatístico. A dedução, por sua vez, contrapõe-se à indução no sentido de que é um processo cognitivo que consiste em chegar a conclusões partindo de determinadas premissas iniciais. Para isso, é preciso que a inferência se desenvolva segundo parâmetros lógicos e, para que as conclusões sejam verdadeiras, é preciso que os postulados sejam válidos [2].

O processo de aprendizagem indutiva, dessa forma, consiste no raciocínio originado de um conceito específico partindo para um conceito mais generalizado. Nosso cérebro se utiliza largamente da indução para gerar conhecimento novo ou prever eventos futuros. Assim, a tarefa do algoritmo de aprendizado é induzir um classificador cujo objetivo é rotular, com uma boa precisão, novos casos [3]. Nesse processo, as hipóteses geradas por essa inferência indutiva podem ou não ser verdadeiras.

Para se induzir, parte-se efetuando um raciocínio sobre exemplos fornecidos como entrada. Cada exemplo é um conjunto de atributos do problema junto a seus valores, denotando uma dada situação ou um caso. Com esses casos, cria-se uma idéia mais geral, que pode servir para predizer casos não conhecidos. Com isso, dada uma nova situação, é possível predizer algumas características que a acompanha. Essas características que são preditas recebem o nome de classe ou atributo-objetivo. Dessa forma, podemos dividir o processo indutivo em aprendizado supervisionado e não-supervisionado [4].

No aprendizado não-supervisionado, o indutor procura encontrar semelhanças entre os exemplos, de qualquer natureza, de modo a determinar se eles podem ser agrupados de alguma maneira, pois não se conhece os rótulos, ou classes, dos exemplos. É um processo que necessita de bastante atenção à escolha, quantidade ou qualidade dos exemplos, pois isso pode atrapalhar de forma significante o processo de aprendizagem [1].

No aprendizado supervisionado, conhece-se a verdadeira classe dos exemplos externos, também chamados de exemplos de treinamento. Com esses exemplos, o indutor tenta aprender, ou melhor, induzir um conceito mais amplo, de forma que possibilite a classificação de exemplos não conhecidos. Para classes discretas, esse processo denomina-se *classificação.* Para classes contínuas, é chamado de *regressão* [1].

O foco deste trabalho se situa no aprendizado de regras de classificação, que é parte do aprendizado supervisionado simbólico. Neste tipo de aprendizado, cada exemplo é descrito por um conjunto fixo pré-determinado de características chamadas de atributos. Sendo assim, o aprendizado determina regras que formam uma relação entre os seus atributos e suas classes. No aprendizado de regras de classificação, essas regras geralmente estão na forma de {atributo Operador valor} (lógica proposicional) ou em lógica de primeira ordem que possuem a enorme vantagem de serem muito mais compreensíveis ao ser humano do que outras formas de representação. Por esse motivo, é utilizado largamente na mineração de dados. Dessa forma, inúmeras outras áreas do conhecimento podem se beneficiar com este tipo de aprendizado.

O processo de aprendizado de regras objetiva capacitar a máquina a classificar corretamente exemplos positivos e negativos do conceito descrito com a utilização de um conjunto de regras. Dessa forma, o algoritmo é corrigido a cada erro durante o treinamento, podendo guiar sua indução em direção aos acertos. A desvantagem é que, normalmente, precisa-se de uma quantidade expressiva de exemplos rotulados para se obter um bom classificador [5].

Nesse processo, alguns algoritmos de aprendizado buscam obter um conjunto de regras classificadoras, baseando-se na cobertura da base de treinamento. Em geral, isso é feito retirando-se da base os exemplos já cobertos para que assim as novas regras aprendidas se concentrem apenas nos exemplos não cobertos. Isso pode fazer com que boas regras sejam deixadas de lado na seleção, pois esses algoritmos, também chamados de algoritmos de cobertura, selecionam as regras com o melhor desempenho na sub-base. Em geral, esses classificadores são ordenados (nos quais há uma ordem das regras para a classificação), o que prejudica a análise das regras, pois uma regra não pode ser vista de forma isolada. E, mesmo nos casos de classificadores não ordenados, há ainda a perda da qualidade das regras.

Outra forma de se realizar o aprendizado, de modo que não haja essa perda de qualidade das regras, é buscar por todas as regras da base, ordená-las por qualidade e fazer uma sub-seleção desse conjunto. Isso tende a elevar o desempenho do classificador e de suas regras. No entanto, além do fato de precisarmos de duas etapas para a classificação (geração e sub-seleção), a sub-seleção é geralmente feita através da cobertura das regras (igual à forma anterior), retirando exemplos da base de treinamento, caindo no mesmo problema de exclusão de algumas boas regras.

Isso nos motivou a propor um método de classificação que faz um tipo diferente de seleção de regras, no qual não se tira exemplos da base de treinamento. Além disso, procuramos classificadores não-ordenados, que facilitam a interpretação das regras, fazendo com que possam ser lidas isoladamente.

Outro problema da segunda abordagem é que ao se trabalhar com problemas muito complexos, com uma grande quantidade de características

simbólicas, a explosão combinatória é grande. Supondo que o problema defina *n* valores de atributos, e que para cada atributo, existem *m* valores simbólicos possíveis, dessa forma, o espaço no qual se faz a busca é da ordem m^n. Assim, com cada atributo a mais no problema, há grande aumento da complexidade do mesmo. Porém, determinar quais atributos são relevantes é uma tarefa complexa e encontrar o melhor classificador dentro desse contexto é um problema NP-difícil.

Logo, pesquisar todos os exemplos possíveis do problema para encontrar um melhor classificador é um processo impraticável. Isso nos leva à utilização de heurísticas para encontrar uma solução, não necessariamente ótima, para o problema. Essas funções são associadas a funções objetivo que são funções que avaliam a qualidade do classificador ou de hipóteses do problema. Essas funções objetivo são de suma importância para o processo, uma vez que todas as decisões e escolhas do algoritmo são tomadas com base na qualidade de cada regra de classificação. Por isso, a definição da função de aptidão que será utilizada para a avaliação das regras deve ser muito bem elaborada. Mesmo assim, em muitos casos, apenas uma métrica de qualidade não é suficiente para se chegar à solução do problema, pois há casos em que mais de uma característica é importante e desejável na regra ou classificador. Para esses casos, a qualidade das regras deve ser calculada como uma resultante dos vários objetivos. Vilfredo Pareto estudou esses conceitos multiobjetivo e estabeleceu a idéia de dominância de Pareto [6]. Esse conceito é utilizado no aprendizado multiobjetivo de regras.

No entanto, aspectos multiobjetivo são difíceis de tratar com heurísticas normais e buscas locais, pois esses problemas contêm geralmente espaços de busca muito grandes e altamente complexos [7]. Assim, tem sido freqüente a utilização de meta-heurísticas para a solução deste tipo de problema. No entanto, é necessário dar uma abrangência global ao algoritmo, de forma a evitar os máximos locais, concentrando-se em uma otimização global. Para isso, pode-se utilizar meta-heurísticas populacionais, como os algoritmos genéticos [8], nuvem de partículas [9] e colônia de formigas [10], que possuem essa característica e tornam o problema da multiobjetividade muito mais simples.

No entanto, para essa tarefa de aprendizado, o problema das meta-heurísticas competitivas, como o algoritmo genético, é que estas tendem todas as suas soluções a uma mesma espécie. Isso é prejudicial quando se quer encontrar várias boas soluções. Nesse caso, técnicas que preservem a diversidade da população, como soluções de caráter cooperativo, são preferíveis, como nuvem de partículas. Esta técnica já foi previamente utilizada para classificação em [11] e [12]. Entretanto, esses trabalhos não exploram características de múltiplos objetivos, além de serem algoritmos de cobertura, gerando classificadores ordenados.

Por isso, este trabalho propõe um processo de aprendizado de regras baseado na meta-heurística de nuvem de partículas com múltiplos objetivos. Esse processo procura por boas soluções nesse espaço exponencial que são então selecionadas, segundo os conceitos de dominância de Pareto [6], diferentemente da seleção por cobertura, para integrar um classificador não-ordenado de novas instâncias, sendo que as etapas de geração e seleção das regras ocorrem ao mesmo tempo. Além disso, nosso trabalho também introduz outra forma de classificação de instâncias não-ordenadas: a votação por confiança com voto negativo, procurando trazer maior precisão ao nosso classificador não-ordenado.

Este trabalho está organizado em seis capítulos. No Capítulo 2, são descritos os principais conceitos do aprendizado de regras de classificação. No Capítulo 3, é descrita a técnica de nuvem de partículas e sua variante com múltiplos objetivos. No Capítulo 4, é apresentado nosso método de aprendizado de regras e, no Capítulo 5, são apresentados os vários experimentos e comparações feitos com o algoritmo. Por fim, no Capítulo 6, são apresentadas as conclusões sobre o trabalho.

2 APRENDIZADO DE REGRAS

Este capítulo descreve os principais conceitos de aprendizado de regras de classificação, apresentando também alguns trabalhos da área.

2.1 SUBÁREA DO APRENDIZADO DE MÁQUINA

Conforme explicado no Capítulo 1, o aprendizado de regras é parte da área de aprendizado de máquina. Existe, nessa área, uma grande quantidade de paradigmas e algoritmos, cada qual se aplicando melhor em dadas situações. Por isso, é possível observar que não existe um único algoritmo que apresente o melhor desempenho para todos os problemas, pois cada um possui vantagens e desvantagens sob determinado aspecto. Em geral, todos esses métodos podem ser agrupados em subáreas de AM, cada qual possuindo seu próprio vocabulário e métodos. Como exemplo dos principais paradigmas de aprendizado, temos [1]:

- Simbólico: constrói representações simbólicas de um conceito, tipicamente na forma de regras, árvores de decisão, expressão lógica ou rede semântica. Tem a vantagem de ser facilmente compreensível.
- Estatístico: utiliza modelos estatísticos para definir a que classe um determinado exemplo pertence. Tem como característica os modelos probabilísticos, que determinam a probabilidade de um exemplo pertencer a uma classe, ao invés de determinar a qual delas ele pertence.
- Baseado em exemplos: armazena exemplos cuja classificação se conhece. Quando um novo exemplo precisa ser classificado, este é comparado com os exemplos armazenados e sua classificação é dada pelos exemplos que mais se assemelham a este. É também chamado de indutor preguiçoso.
- Conexionista: Utiliza conexões entre conceitos para criar um modelo classificador. São utilizados nós com pesos, conectados entre si, de

forma a codificar o conhecimento. Os pesos entre os nós determinam a classe do exemplo.

- Evolutivo: utiliza o conceito de evolução dos seres vivos para chegar a um modelo classificador. Vários classificadores competem entre si, sendo que os mais aptos sobrevivem e se reproduzem para as próximas gerações.

Nosso problema se concentra no aprendizado simbólico, pois procura representações simbólicas na forma de regras de classificação, e também no paradigma evolutivo, com a utilização da técnica de nuvem de partículas, que possui características evolucionárias, com a exceção de não ser competitiva e sim cooperativa.

2.2 APRENDIZADO DE REGRAS DE CLASSIFICAÇÃO

Aprendizado de regras de classificação é o processo de se encontrar, através de um conjunto de exemplos de treinamento, um conjunto de regras que pode ser usado para a classificação ou predição. Uma regra de classificação é um par $< condição \Rightarrow classe >$, onde ***condição*** é o conjunto de condições para que o problema seja classificado como um dos valores do conjunto de classes. Uma regra de classificação de apenas duas classes seria algo como:

SE <CONDIÇÃO> ENTÃO <sim> SENÃO <não>,

onde *condição* pode ser qualquer expressão de lógica booleana.

As regras são comumente utilizadas na mineração de dados, por sua simplicidade, intuicionismo, modularidade e por serem obtidas diretamente da base de dados [13]. Por isso, a indução de regras já se estabeleceu, de forma isolada, como um componente básico de muitos sistemas de aprendizado de máquina (AM). Além disso, foi a primeira técnica de AM a fazer parte de aplicações comerciais bem sucedidas [14].

No processo de aprendizado de regras, vincula-se a aprendizagem a uma base de treinamento. Essa base contém exemplos de predições que o sistema deve fazer. Com base nesses exemplos, procura-se identificar um padrão, para

que um sistema classificador possa ser aprendido. Com isso, quando surgir uma nova entrada no sistema, este saberá classificá-la, mesmo que nunca a tenha aprendido. Assim, o algoritmo deve receber uma base de treinamento de entrada, com os valores dos atributos e classes, e retornar um conjunto de regras de classificação para o problema.

Resumidamente, por exemplo, dados os exemplos de treinamento:

1. [Aparência=Sol, Temperatura=quente, umidade=alta, Vento=não -> Jogar=Não],
2. [Aparência=Sol, Temperatura=quente, umidade=alta, Vento=sim -> Jogar=Não],
3. [Aparência=Chuva, Temperatura=frio, umidade=alta, Vento=não -> Jogar=SIM],

o algoritmo deve ser capaz de induzir situações como:

Se Aparência = Sol **Então** Jogar = Não.

Assim, qualquer nova instância, mesmo que desconhecida, que possua *aparência= sol,* será classificada como *Não*. Evidentemente, a indução pode ser imperfeita.

Nos últimos anos, muitas técnicas surgiram para lidar com o aprendizado de regras. Em geral, podemos classificá-las em três grupos principais: separar-para-conquistar, dividir-para-conquistar e baseado em regras de associação [15].

No primeiro grupo, também chamado de estratégia de cobertura, o procedimento de busca é, geralmente, um algoritmo guloso que a cada iteração, encontra a melhor regra e remove os exemplos da base cobertos por ela. Esse processo é reiterado sobre os exemplos que restaram na base [16]. Ou seja, encontra uma melhor regra, separa as instâncias cobertas por essa regra e conquista recursivamente as demais instâncias até que não existam instâncias a conquistar.

Para formarmos um classificador nessa estratégia, criamos um conjunto dessas melhores regras. Este conjunto pode ser ordenado (no qual a ordem das regras de classificação é importante) ou não-ordenado (no qual não importa a ordem em que as regras devem classificar a entrada). Num classificador ordenado, também chamado de lista de decisão, a classificação é dada pela primeira regra que cobre a premissa. Em um não-ordenado, as regras que cobrem a premissa devem entrar num acordo e atribuir uma classe à entrada.

Um clássico algoritmo desse grupo é o CN2 [17]. Esse algoritmo induz uma lista de decisão — portanto ordenada —, usando entropia como sua heurística de busca. Há também uma versão estendida do algoritmo que induz uma lista não ordenada de regras e usa a correção de erro de Laplace como função de avaliação, conforme descrito em [14].

No segundo grupo, dividir-para-conquistar, é construído um classificador global seguindo uma abordagem de refinamentos consecutivos do problema. O resultado é geralmente expresso como uma árvore de decisão, que divide completamente o espaço de instância [18]. O grande representante desse grupo é o algoritmo C4.5 [19], que é quase um padrão na comparação de algoritmos de aprendizagem simbólica. Usa o ganho de informação como medida de qualidade para construir uma árvore de decisão, e também, um passo posterior de poda da árvore, baseada em redução de erro. Nessa abordagem, cada braço da árvore pode ser considerado uma regra [18] e não há sobreposição de regras. Os pontos fortes da indução de regras sobre a indução de árvores de decisão são sua inteligibilidade e o pouco espaço de armazenamento. Porém, o processo de indução de regras tende a ser mais lento que os processos de indução de árvores [1].

O terceiro grupo oferece uma alternativa aos dois primeiros grupos. Consiste na utilização de classificadores baseados em regras de associação, ou classificadores associativos [20]. Geralmente, esses classificadores são obtidos com a utilização de buscas exaustivas no espaço das regras possíveis recolhendo todas as regras que satisfazem certas condições. O problema dessa abordagem é que, geralmente, o número de regras geradas é gigantesco. Alguns trabalhos, como o Roccer [18] e MORLEA [21], procuram reduzir esse conjunto, por meio de uma sub-seleção de regras. Porém, esses algoritmos possuem características separar-para-conquistar.

2.2.1 Terminologia do Aprendizado de Regras

Há, neste tipo de aprendizado, alguns termos amplamente utilizados no que diz respeito à tarefa de classificação. Nesta seção, definimos os mais importantes:

a) Exemplo:

Um exemplo é uma tupla de valores de atributos, descrevendo o objeto do problema, um caso especial ou registro de uma ocasião do problema. O conjunto de exemplos utilizado para o treinamento do indutor recebe o nome de base de treinamento. O conjunto de exemplos utilizado para testar a qualidade do indutor é denominado base de testes.

b) Atributo

São as características dos objetos do problema. Com base nessas características, o algoritmo induz um classificador sobre o atributo-objetivo. Nesse ambiente, definimos um atributo como pertencente a um desses quatro tipos, definidos por suas escalas como em [22] e [23]:

1. Nominal ou categórica;
2. Ordinal;
3. Intervalar;
4. Razão;

As duas primeiras escalas são ditas não-métricas, pois não exibem nenhuma relação com valores numéricos. Os atributos categóricos representam categorias, como por exemplo, <cor=verde>. Vêm sempre na forma <atributo=valor>. Os atributos ordinários são parecidos com os atributos categóricos, porém possuem uma ordem pré-estabelecida. Um exemplo pode ser visto com um atributo de temperatura. Esse atributo pode representar, por exemplo, três categorias: quente, morno e frio. Nesse caso, fica evidente haver uma ordem estabelecida como quente > morno > frio.

As duas escalas restantes são ditas escalas métricas, pois, nesse caso, os atributos têm um sentido numérico; os atributos intervalares representam um intervalo; possuem escalas constantes de medida, possuindo ordem; o ponto zero, nesse caso, é arbitrário; a soma e o produto não fazem sentido algum com esse

tipo de escala. Como exemplo, temos a temperatura em graus Celsius, onde o zero não significa falta de temperatura.

O último tipo, atributo com escala de razão, representa uma escala numérica contínua, onde qualquer operação matemática é possível. O zero, nesse caso, não é arbitrário, e sim absoluto. Nessa escala, enquadram-se os números reais e medidas como o peso de um objeto.

c) Classe

Classe é o atributo objetivo do problema. É a característica que se quer classificar, ou, de outra forma, é a variável dependente dos outros atributos.

d) Entrada

A entrada para um algoritmo de aprendizado é um conjunto de exemplos de treinamento e teste. Com esse conjunto, o algoritmo induz um classificador ou tira conclusões mais genéricas.

e) Saída

A saída de um algoritmo de aprendizado supervisionado é um conjunto de definições, regras, ou árvore, que prediz situações novas. Para uma tarefa de classificação, a saída é um classificador.

f) Ruído

É qualquer tipo de imperfeição nos dados da base. Isso pode ser causado por um erro na obtenção dos dados, na sua transformação, ou até classes rotuladas incorretamente [1].

g) Bias de aprendizado

É a preferência de uma hipótese sobre outra. De fato, por haver nos problemas de aprendizado uma grande quantidade de hipóteses consistentes, todos os indutores possuem alguma forma de bias de aprendizado. Segundo [3], aprendizado sem bias é impossível.

Há alguns algoritmos, como o nosso, por exemplo, que se concentram no aprendizado de regras categóricas para tarefas de classificação. Por causa disso, os atributos métricos devem ser transformados em atributos categóricos para que o aprendizado seja possível. Geralmente, isso é feito representando um intervalo numérico como um nome, como por exemplo, <atributo = (>= a e < b)>. Esse

exemplo nos diz que o atributo está no intervalo [a,b). Para isso, é desejável um prévio conhecimento do problema para então efetuar um processo de discretização, para que assim, os valores numéricos sejam agrupados em faixas de valores, permitindo identificá-las com um rótulo de categoria. As regras categóricas são mais fáceis de definir e trabalhar, porém, com elas, não é fácil analisar inter-relações entre os valores dos atributos, como *frio* < *morno* < *quente,* por exemplo. Isso seria possível com a utilização de atributos ordinários ou intervalares.

2.3 AVALIAÇÃO DE REGRAS

Sendo imperfeita a indução do classificador, é necessária a avaliação de suas regras, pois o algoritmo pode cometer erros de classificação. Para que possamos considerar uma regra (ou classificador) boa ou ruim, fazemos uso de certas métricas de avaliação. Considerando cada regra no formato $L \rightarrow R$, sendo *L* a condição e *R* a classe, podemos obter a matriz de contingência desta regra, para uma situação de duas classes, conforme mostra a Tabela 2.3.1. Para um problema com mais de duas classes, é obtida uma matriz de contingência para cada classe. Nesse caso, qualquer referência à classe *R* é interpretada como classe positiva e qualquer referência à outra classe é negativa.

Nessa tabela, *L* é o conjunto de exemplos os quais a regra classifica como positivos e o seu complemento $\overline{L}$, o conjunto de exemplos os quais a regra classifica como negativos. *R* são os exemplos que pertencem à classe positiva. $\overline{R}$ são os exemplos que pertencem à classe negativa. Assim, Vp (verdadeiro positivo) é o conjunto $L \cap R$, ou seja, o conjunto dos exemplos de classe positiva que a regra classifica como positiva. Da mesma forma, Fn (falso negativo) é o conjunto dos exemplos de classe positiva os quais a regra classifica como negativa; Fp (falso positivo), os exemplos de classe negativa os quais são classificados como positivos; por fim, Vn (verdadeiro negativo), os exemplos de classe negativa que são classificados como negativos. Para complementar, r é o

número de exemplos do conjunto R, $r = |R|$; da mesma forma: $\bar{r} = |\overline{R}|$, $l = |L|$, $\bar{l} = |\overline{L}|$; *n* é o número total de exemplos.

Tabela 2.3.1 — matriz de contingência de uma regra $L \to R$.

	L	$\overline{L}$	
R	Vp	Fn	r
$\overline{R}$	Fp	Vn	$\bar{r}$
	l	$\bar{l}$	

Com base nessa matriz, é possível descrever uma série de medidas de avaliação de classificadores. A Tabela 2.3.2 apresenta alguns exemplos das medidas mais utilizadas [1].

Tabela 2.3.2 — Medidas de avaliação de regras		
$prel(L \to R) = \frac{Vp}{l}$,	(1)	Confiabilidade positiva
$prel(L \to R) = \frac{Vp+1}{l+2}$,	(2)	Confiabilidade positiva com correção de Laplace
$nrel(L \to R) = \frac{Vn}{\bar{l}}$,	(3)	Confiabilidade negativa
$nrel(L \to R) = \frac{Vn+1}{\bar{l}+2}$,	(4)	Confiabilidade negativa com correção de Laplace
$cob(L \to R) = \frac{l}{n}$,	(5)	Cobertura
$\sup(L \to R) = \frac{Vp}{n}$,	(6)	Suporte
$sens(L \to R) = \frac{Vp}{r}$,	(7)	Sensitividade
$esp(L \to R) = \frac{Vn}{\bar{r}}$,	(8)	Especificidade
$precT(L \to R) = \frac{Vn+Vp}{n}$,	(9)	Precisão total

2.4 ESTRUTURA DE ALGORITMO DE APRENDIZADO DE REGRAS

Geralmente, a estrutura de um algoritmo de aprendizado de regras pode ser esquematizada em um modelo de três camadas.

A primeira camada engloba o algoritmo de descoberta de regras. Esse algoritmo tem a função de encontrar as melhores regras de classificação aprendidas da base de dados. É nesse passo, que se faz o uso de algoritmos evolucionários como nuvem de partículas, algoritmos genéticos, e também de algoritmos de geração de regras de associação, como o Apriori [24].

A segunda camada é ocupada pelo algoritmo de escolha de regras, que tem a função de escolher, dentre as regras geradas pelo algoritmo da primeira camada, aquelas que irão integrar o classificador. Dependendo da forma como é feita essa escolha, podemos ter classificadores ordenados ou não-ordenados. Por exemplo, se um algoritmo de cobertura, do tipo separar-para-conquistar, cada vez que encontrar uma regra para o classificador, retirar todos os exemplos da base de dados cobertos por aquela regra, então, nesse caso, só há sentido em se utilizar um classificador ordenado [25]. Do contrário, se os exemplos retirados forem somente aqueles cobertos corretamente pela regra, ou seja, aqueles que possuam a mesma classe que a regra prediz, não há sentido em se utilizar um classificador ordenado [25]. Da mesma forma, quando não se retiram exemplos da base, como ocorre com a nossa proposta, a classificação deve ser feita de modo não ordenado.

A terceira camada é a etapa da validação do algoritmo. Com isso, temos noção se o classificador gerado é bom ou ruim para o objetivo, conhecemos seu valor de precisão ou outra medida estatística e percebemos seu poder de predição.

Para se obter medidas estatísticas do classificador, é preciso estimá-las segundo algum processo estatístico, pois não temos a exata noção de como o classificador se portará frente a instâncias com as quais ele nunca tomou contato. O método da ressubstituição, que consiste em se testar o classificador na própria base de treinamento, é extremamente otimista e dá uma medida estatística

totalmente aparente [1]. Para estimar uma medida mais verdadeira, podemos fazer uso de alguns métodos estatísticos tais como [1]:

Validação simples – divide-se a base de dados em uma porcentagem *p* de exemplos para a base de treinamento, ficando uma porcentagem (1 – *p*) para a base de teste, sendo $p > ½$.

Amostragem aleatória – são induzidas *L* hipóteses, sendo $L << N$, a partir de cada conjunto de treinamento com *t* exemplos aleatórios. Após *L* iterações, o erro é calculado como a média dos erros das hipóteses.

Validação cruzada – é um meio termo entre a validação simples e o deixa-um-fora. Consiste em se dividir a base de dados em *r* partes mutuamente exclusivas de tamanhos aproximadamente iguais a n/r exemplos. Faz-se o teste nas r partições, usando, para cada teste, as outras (r-1) partições para treinamento. A medida estatística é calculada a partir da média aritmética das *r* execuções.

Validação cruzada estratificada – equivalente à validação cruzada, porém, nesse caso, a proporção das classes da base original é mantida para todas as partições.

Deixa-um-fora (*Leave-one-out*) – é um caso especial de validação cruzada. Seria equivalente à validação cruzada em *n* partições, onde *n* é o número de exemplos do conjunto original. Assim, o classificador é induzido com (n – 1) exemplos e testado no exemplo que resta. Esse processo é repetido para todos os exemplos da base, ou seja, n vezes. É um método muito dispendioso, utilizado, geralmente, para pequenas bases de dados.

Bootstrap – O conjunto inicial é um conjunto do mesmo tamanho da base original, onde os exemplos são escolhidos aleatoriamente. Pode haver repetição de exemplos, de forma que os exemplos que não aparecem nessa base de treinamento perfazem a base de teste. O processo é repetido certo número de vezes, sendo as medidas estatísticas estimadas como a média das medidas obtidas nas iterações.

2.5 CLASSIFICAÇÃO COM REGRAS ORDENADAS

Para classificar novos exemplos, o classificador ordenado tenta, em ordem, cada regra do conjunto até que alguma satisfaça as condições da nova instância. Logo, a ordem das regras é um conceito fundamental. Isso significa que as regras do conjunto não têm sentido isoladamente, com exceção da primeira regra, pois formam apenas uma regra, na forma: SE X ENTÃO Y SENÃO SE W ENTÃO Z... e assim sucessivamente, até a última regra do classificador.

Como exemplo, considere o seguinte classificador ordenado:

Se pena = sim então classe = ave
Senão se pernas = dois então classe = humano
Senão...

A regra ‘se pernas = dois então classe = humano’, quando considerada sozinha, não está correta, pois aves também têm duas pernas. O problema piora com um grande número de regras, sendo muito difícil para um especialista entender o contexto de uma regra [14].

Logo, se um dos objetivos de seu classificador é a descoberta de regras interessantes dentro do contexto de classificação, esse tipo de classificador não é bom. Nesse caso, um classificador não-ordenado é preferido.

2.6 CLASSIFICAÇÃO COM REGRAS NÃO ORDENADAS

Com esse tipo de classificador, todas as regras devem ser testadas, e as que satisfizerem a condição da instância a ser classificada são coletadas. Com estas regras, aplica-se algum método de escolha da classe, como por exemplo:

Aleatório (A) — consiste em se escolher aleatoriamente uma regra daquelas coletadas para classificar a instância [13].

Primeira Regra (PR) — escolhe-se a primeira regra daquelas coletadas para se classificar a instância [13].

Cobertura da classe (CC) — procedimento mais usual, onde o objetivo seria o de analisar qual das classes é a mais coberta pelas regras coletadas. Por exemplo,

considere uma regra que cobre as classes {sim,não}, respectivamente, {15,1}, uma outra regra que cobre {20,0}, e uma regra que cobre {1,10}. Note que a classe mais coberta é {sim}, que seria a classe escolhida pelo classificador. No entanto, esse método tem a desvantagem de desconsiderar possíveis regras de exceção que cobrem poucos exemplos [14].

Votação simples (VS)— método no qual cada regra vota na classe desejada. A classe campeã é a classe que o classificador escolhe [13].

Votação de Confiança (VC) — Equivalente ao método de votação simples, mas, nesse caso, cada regra vota com a confiança que tem. Por exemplo, considere as classes {sim,não} e duas regras. A primeira prediz a classe da instância como {sim}, a segunda como {não}. Apesar disso, a confiabilidade positiva da primeira regra é 0,75 e a da segunda é 0,64. Assim, a classe {sim} ganha, pois sua pontuação é maior [13].

Votação de Confiança com voto negativo (VCVN) — equivalente à votação de confiança, mas, nesse caso, há também o voto negativo, que abaixa o valor da classe. Logo, por exemplo, uma regra tem confiabilidade positiva 0,75 e confiabilidade negativa 0,80. No caso, para cada instância que a regra satisfaz as condições, ela vota com 0,75 de confiança. Mas, para cada instância que a regra não satisfaz as condições, ela também vota, baixando o valor de sua classe em 0,80. Nesse método, todas as regras interagem, não somente as regras que satisfizeram as condições. Isso retira a possibilidade de não se classificar os exemplos não cobertos por nenhuma regra do classificador. Uma melhor descrição desta abordagem pode ser vista no Apêndice C.

Menor falso positivo (MFP) — Nesse caso, a regra que dentre as coletadas tiver a menor taxa de falso positivo é escolhida para classificar a instância. A lógica desse método é justamente evitar os falsos positivos, que dependendo do objetivo, pode ser um grande problema [13].

2.7 ANÁLISE ROC

Em aprendizado de regras de classificação, procura-se gerar regras que, coletivamente, têm bom desempenho para a classificação. Grande parte dos métodos de aprendizado de regras existentes busca otimizar as classificações, maximizando a precisão em uma base de treinamento [13]. No entanto, estudos recentes demonstraram problemas ao se utilizar a precisão como métrica para a indução ou avaliação de classificadores [26]. O classificador pode ser induzido ao erro quando as classes não estão balanceadas, por exemplo. E, quando não se pode utilizar custos de erro de classificação porque não são conhecidos ou porque a distribuição das classes não é conhecida, a solução é utilizar classificadores probabilísticos [13].

Uma das análises mais utilizadas para se medir o desempenho de um classificador probabilístico é a análise da característica operacional do receptor (ROC — Receiver *Operating Characteristic*). A análise ROC é um método que vem da Teoria de Decisão Estatística e foi originalmente utilizado na Segunda Guerra Mundial para análises de imagens de radar [27]. No entanto, começou a ser utilizado na mineração de dados apenas no final da década de 90, como um bom método de avaliação de classificadores probabilísticos.

O gráfico ROC equivale-se ao gráfico da sensitividade x especificidade, uma vez que relaciona a taxa de verdadeiros positivos *Vp,* no eixo *y*, com a taxa de falsos positivos *Fp*, no eixo *x*, de cada regra. Quanto mais se aproximar do ponto (0,1), nesse caso (*Fp, Vp*), melhor é a regra, conforme mostra a Figura 2.1. Minimizar *Fp* significa maximizar *Vn*. Logo, o ponto (0,1) equivale-se ao ponto (1,1) no plano sensitividade x especificidade.

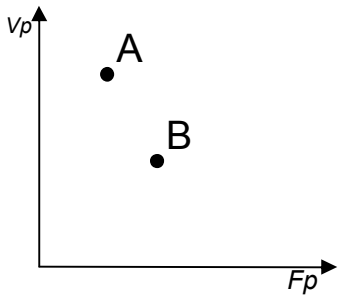

Figura 2.1 — Comparação de classificadores na curva ROC (A melhor que B).

Essa análise é bastante utilizada para testar o desempenho de classificadores, pois relaciona seus erros e acertos. É possível traçar, num gráfico ROC, uma única regra, um classificador (formado, ou não, por um conjunto de regras) ou até um classificador parcial. Vários pontos traçados no gráfico formam uma curva ROC, que nos dá uma idéia do desempenho geral do classificador.

Para comparar duas curvas ROC, a medida mais utilizada é a área abaixo da curva, ou AUC (*Area Under Curve*). Quanto maior essa medida, melhor o desempenho do classificador [28] [29]. Como AUC é a porção da área do primeiro quadrante, seu valor ficará sempre entre 0 e 1. No entanto, qualquer classificador que aleatoriamente tentar adivinhar a classe de uma instância produzirá a linha diagonal que liga os pontos (0,0) e (1,1) no gráfico ROC. Nesse caso, a AUC desse classificador é 0,5. Logo, nenhum classificador realístico deve obter uma AUC menor que 0,5 [13]. No entanto, é possível obter AUC menor, se o classificador for equivocado e obtiver pontos de desempenho muito ruins, quando, por exemplo, regras que predizem positivo obtêm desempenho muito melhor predizendo negativo.

Em experimentos, há um conjunto finito de pontos que formam a curva. Por isso, a AUC é aproximada. O método mais comum de se calcular a área é o método trapezoidal, que consiste em se somar as áreas dos trapézios que se formam entre os pontos da curva. A Figura 2.2 apresenta os trapézios entre os pontos da curva ROC. A soma das áreas desses trapézios forma a AUC.

Para medir a tendência de desempenho de um classificador probabilístico, podemos visualizá-lo em um gráfico ROC, calculando sua AUC. Para isso, com a utilização de um valor limiar, transformamos um classificador de pontuação, como é o caso da utilização de um esquema de votação, em um classificador binário [13].

Com um método de votação, por exemplo, tem-se um conjunto de regras classificadoras, que votam, exemplo a exemplo, cada qual em sua classe. Dessa forma, para cada instância da base, é obtido um grau numérico. Se o grau da instância ultrapassa o limiar, o classificador produz um sim, senão, não. Nesse caso, cada valor de limiar equivale a um classificador diferente, gerando um ponto

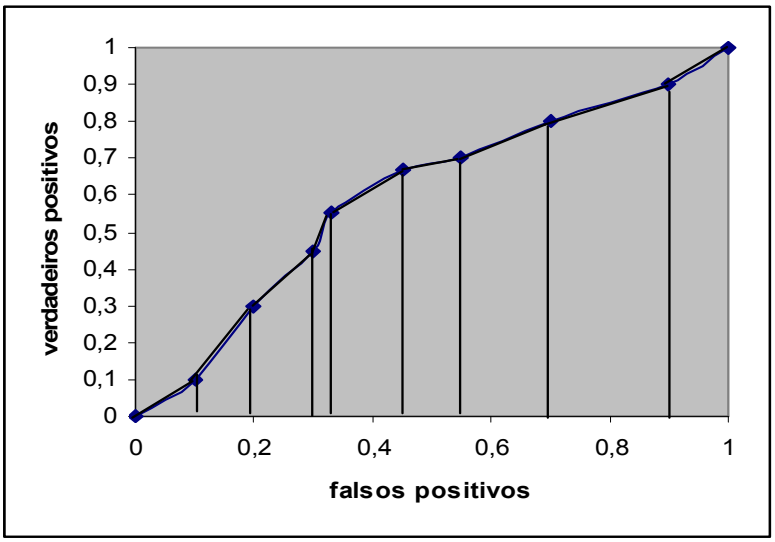

Figura 2.2 – Trapézios da curva ROC formando a AUC.

diferente no plano ROC. Dessa forma, variando o limiar de $-\infty\, a\, +\infty$, temos a avaliação de todos os possíveis classificadores equivalentes ao nosso classificador de pontuação, produzindo uma curva no plano ROC, podendo assim ser calculada a área abaixo dessa curva (AUC) [13] [30]. É evidente que, nesse caso, deve-se traçar um gráfico ROC para cada classe da base, pois as instâncias devem estar na forma de problemas binários (*sim* e *não*). Um exemplo do cálculo e elaboração da curva ROC para uma classe de um classificador é descrito no Apêndice A.

2.8 CONSIDERAÇÕES, OUTROS TRABALHOS E MOTIVAÇÃO

Recentemente, tem-se aumentado o interesse em se aplicar o conceito de dominância de Pareto ao aprendizado de máquina, inspirado por desenvolvimentos de sucesso em otimização evolucionária multiobjetivo. Esses trabalhos incluem seleção multiobjetivo de características, seleção multiobjetivo de modelos, redes de funções de base radial, treinamento de perceptrons multicamada, árvores de decisão e sistemas inteligentes [31]. Na literatura, poucos trabalhos lidam com algoritmos evolucionários multiobjetivo para aprendizado de regras, dentre eles, [32], [33], [34] e [35]. O primeiro se concentra na fase de

seleção de regras, apresentando um algoritmo genético multiobjetivo de seleção de regras para encontrar menores conjuntos de regras com maiores precisões que os conjuntos de regras extraídos heuristicamente. O algoritmo tem o objetivo de maximizar a precisão e minimizar o número de regras. Em [34], são discutidas a seleção e geração multiobjetivo de regras de associação com o NSGA-II (Non-Dominated Sorting Genetic Algorithm). Em [35], procura-se encontrar regras com características específicas que são, geralmente, rejeitadas por algoritmos tradicionais.

Em [36], um algoritmo evolucionário de otimização multiobjetivo com conceitos de Pareto é utilizado para descobrir regras de classificação interessantes para uma classe alvo. Ele apresenta uma implementação do NSGA com confiabilidade positiva e sensitividade como seus objetivos. Esse trabalho é estendido em [33], usando uma meta-heurística para produzir conjuntos de regras de classificação interessantes. Para isso, foi introduzida uma medida de dissimilaridade das regras para promover a diversidade na população.

Um exemplo de um algoritmo multiobjetivo fora do conceito evolutivo, do tipo separar-para-conquistar é o Apriori-Roccer [16] [18], que utiliza a curva ROC, ou seja, a sensitividade e a especificidade, para selecionar as regras. Ele obteve um ótimo desempenho de classificação e AUC com várias bases de dados, competindo com o já famoso CN2 [17], também do tipo separar-para-conquistar, e sua forma não ordenada de classificador [14].

A questão é que, conforme já mencionado, os algoritmos separar-para-conquistar necessitam da constante atualização da base, retirando os exemplos já cobertos por outras regras. Sendo as bases arquivos do disco, por exemplo, isso pode causar muita demora no processo. Além disso, o fato de se retirar os exemplos da base de dados, pode vir a gerar um caimento progressivo da qualidade das regras. Isso porque as regras que são aprendidas a partir dessa sub-base são aparentes, pois não consideram todos os exemplos de treinamento, e, por isso, suas medidas de precisão ou erro, não são válidas no contexto do problema. E, mesmo avaliando as partículas com a base completa, deixa-se de lado na aprendizagem muitas outras boas regras.

Além disso, com esses algoritmos, os classificadores são listas de decisão, que, sem dúvida, são boas para a classificação, mas não permitem a procura de regras interessantes, para especialistas, por exemplo, pois as regras do classificador, com exceção da primeira regra, não podem ser lidas isoladamente. Para a mineração de dados, por exemplo, isso é realmente importante. Ainda assim, isso é resolvido com a substituição das listas de decisão por classificadores não-ordenados, porém há o problema da perda da qualidade das regras.

Há ainda os algoritmos que necessitam das três camadas da estrutura bem separadas, como é o caso do Apriori-Roccer, por exemplo, onde há o processo separado de geração de regras, bem como ordenação, para então fazer a seleção das regras do classificador.

Por isso, aqui é apresentada uma idéia de como resolver o problema do aprendizado de regras de uma forma mais direta, onde a geração e seleção de regras ocorrem simultaneamente, sem a necessidade de se modificar a base de dados a cada regra aprendida. Por esse motivo, os classificadores gerados são formados por regras não ordenadas, que têm sentido isoladamente, e não perdem qualidade, pois foram aprendidas em todo o contexto da base de treinamento. Além disso, nosso algoritmo explora conceitos multiobjetivo de duas a qualquer quantidade de funções-objetivo definida pelo usuário, utilizando a dominância de Pareto para a seleção das regras, de forma a propiciar uma boa qualidade do classificador. Para isso, fazemos uso da técnica de otimização por Nuvens de Partículas, que é descrita no próximo capítulo.

3 NUVEM DE PARTÍCULAS

Otimização por Nuvem de Partículas, PSO (*Particle Swarm Optimization*), é uma técnica de otimização proposta por James Kennedy e Russel Eberhart em 1995 [9]. O método foi descoberto através da simulação de um modelo social simplificado análogo ao modelo de cardume de peixes ou bando de pássaros à procura de alimento [37].

Em uma nuvem de partículas, não existe um controle central. Cada partícula atua e toma decisões com base em informações locais e globais, como demais técnicas de Vida Artificial [37]. Mais abstratamente, uma partícula seria um estado de pensamento, representando nossas crenças e atitudes. Uma mudança de pensamento, dessa forma, corresponderia a um movimento da partícula. Ajustamos nossas crenças com base nos outros; avaliamos os estímulos do ambiente, comparamos com as nossas crenças, e imitamos o estímulo [37]. Avaliação, comparação e imitação são importantes propriedades do comportamento social humano e, por isso, são a base para a nuvem de partículas, que utiliza esses conceitos na adaptação a mudanças no ambiente e na resolução de problemas complexos [38]. Na simulação, o comportamento de cada indivíduo é afetado pelas experiências dos outros indivíduos.

O conceito de Nuvem de Partículas (PSO) pertence à categoria de Inteligência de Enxames (*Swarm Intelligence*). Além disso, tem raízes na Vida Artificial e na Computação Evolucionária (CE). Se, por exemplo, relacionarmos CE e PSO, veremos que as duas técnicas têm alguns pontos em comum. Uma nuvem consiste em uma população de indivíduos que representam uma solução para um problema de otimização. Através de modificações probabilísticas e iterativas dessas soluções, procuramos por uma solução ótima. A diferença entre os dois conceitos reside em como mudar a população ou nuvem de uma geração para outra. Na Computação Evolucionária, essa mudança de uma geração para a outra é feita com base nos operadores genéticos de seleção, cruzamento e mutação. Além disso, as espécies morrem e são substituídas a cada geração. Em Nuvem de Partículas, isso é feito de acordo com as fórmulas de atualização da

velocidade e posição, descritas na próxima seção. Nessa técnica, as partículas se movimentam; não morrem, nem são substituídas. O objetivo é alcançado, em PSO, através de uma busca cooperativa, ao passo que, em CE, de uma busca competitiva [37].

3.1 O ALGORITMO DE NUVEM DE PARTÍCULAS

Em PSO, a população é chamada nuvem. Uma nuvem é um número de partículas que se movem em um espaço n-dimensional, dentro de um subespaço de busca *S*. Há um conceito de vizinhança para cada partícula, onde $V(p)$ consiste no conjunto de todas as partículas vizinhas da partícula p. Dada uma partícula i, se $i \in V(p)$, então p conhece i e pode tomar decisões com base nas posições da partícula i [37].

Cada partícula p, numa dada iteração t, tem uma posição em R^n, $\vec{x}(t)$, e uma velocidade de deslocamento nesse espaço, $\vec{v}(t)$. Possui também uma pequena memória contendo sua melhor posição já alcançada, $\vec{p}(t)$, e a melhor posição, $\vec{g}(t)$, já alcançada até agora pelos conhecidos de p, ou seja, o melhor $\vec{p}(t)$ de todas as partículas pertencentes a $V(p)$. É importante que se diga que $\vec{x}(t)$, $\vec{v}(t)$, $\vec{p}(t)$ e $\vec{g}(t)$ são vetores n-dimensionais.

Cada partícula é uma solução potencial para o problema. O objetivo do algoritmo é movimentar essas partículas a fim de fazer com que elas se tornem soluções ótimas do problema. Para essa movimentação, uma partícula tem três opções [39]:

- Seguir seu próprio caminho;
- Seguir em direção a sua melhor posição já encontrada ($\vec{p}(t)$);
- Seguir em direção à melhor posição do melhor vizinho ($\vec{g}(t)$);

Para isso, é necessário calcular a aptidão de cada partícula, isto é, encontrar um valor que indique o quão próxima do ótimo é a solução representada pela posição dessa partícula. Dessa forma, temos uma função de aptidão. Como cada posição da partícula representa uma solução candidata para o problema, a

aptidão de uma partícula *p,* representada como $\alpha(p)$, é uma função da posição da partícula, ou seja, $\alpha : S \subseteq R^n \rightarrow R$. Assim, temos:

- Para um problema de maximização: num dado tempo *t*, uma partícula *i* é melhor que uma partícula *j*, se $\alpha(\vec{x}_i(t)) > \alpha(\vec{x}_j(t))$.
- Para um problema de minimização: num dado tempo *t*, uma partícula *i* é melhor que uma partícula *j*, se $\alpha(\vec{x}_i(t)) < \alpha(\vec{x}_j(t))$.

3.1.1 O Corpo do Algoritmo

A nuvem é iniciada no tempo $t = 0$, espalhando-se as partículas aleatoriamente no espaço *S*. Para cada partícula, fazemos $\vec{p}(0) = \vec{g}(0) = \vec{x}(0)$. Após isso, inicia-se o processo iterativo.

A velocidade e posição das partículas na próxima iteração são calculadas pelas seguintes fórmulas de atualização:

$$\vec{v}(t+1) = \omega.\vec{v}(t) + \phi_1(\vec{p}(t) - \vec{x}(t)) + \phi_2(\vec{g}(t) - \vec{x}(t)), \quad (3.1.1)$$

$$\vec{x}(t+1) = \vec{x}(t) + \vec{v}(t+1), \quad (3.1.2)$$

onde ϕ_1 e ϕ_2 são coeficientes que determinam a influência do melhor da partícula ($\vec{p}(t)$) e do melhor global ($\vec{g}(t)$), respectivamente, na fórmula da velocidade $\vec{v}(t+1)$. O coeficiente ω é a inércia da partícula, ou seja, o quanto sua velocidade anterior influencia na velocidade atual, equivalendo à autoconfiança da partícula. Pode ser definido um parâmetro de velocidade máxima (*vmax)* que limita o valor da velocidade da partícula, para que esta não se afaste demais do restante da nuvem.

Depois de atualizadas as velocidades e posições de todas as partículas, $\vec{p}(t+1)$ e $\vec{g}(t+1)$ são calculados e passa-se à próxima iteração ou termina-se a execução.

O pseudocódigo do algoritmo é apresentado na Figura 3.1.

1. Para cada partícula *i* da nuvem, faça:
   a. Inicia $\vec{x}_i$ com uma solução aleatória para o problema.
   b. Inicia $\vec{v}_i$ com uma velocidade aleatória < V_MAX.
   c. $\vec{p}_i \leftarrow \vec{x}_i$
   d. $\vec{g}_i \leftarrow \vec{x}_i$
2. Enquanto não atingir um critério de parada:
   a. para cada partícula *i* da nuvem, faça:
      i. $\vec{v}_i \leftarrow \omega.\vec{v}_i + \phi_1(\vec{p}_i - \vec{x}_i) + \phi_2(\vec{g}_i - \vec{x}_i)$
      ii. $\vec{x}_i \leftarrow \vec{x}_i + \vec{v}_i$
      iii. $\vec{p}_i \leftarrow melhor\ entre\ \vec{p}_i\ e\ \vec{x}_i$
      iv. $\vec{g}_i \leftarrow melhor(\vec{p}_j), com\ j \in V(i)$
3. Retorna melhor $\vec{g}$ dentre todas as partículas.

Figura 3.1 – Algoritmo de Nuvem de Partículas

3.2 APLICAÇÕES

Embora seja ainda uma técnica recente, PSO já é aplicada com sucesso em vários problemas de otimização. Tem sido utilizada para treinar redes neurais no lugar do conhecido método de retropropagação, em menor tempo e com mesma eficiência [9] [40]. Outras aplicações de PSO relatadas na literatura são:

- Otimização de funções de controle de lógica nebulosa [41];
- Análise de tremores humanos (treinamento de uma rede neural) [42];
- Evolução de agentes em jogos [43];
- Controle reativo de tensão elétrica e potência [44], [45];
- Roteamento de Redes de Sensores Ad-hoc [46].

3.3 CONTROLANDO A CONVERGÊNCIA

Um dos principais problemas que, por ventura, podem acontecer em uma otimização com nuvem de partículas, é a convergência prematura da nuvem para

soluções subótimas. Isso acontece quando as partículas da nuvem se aglomeram em uma região do espaço, não tendo, então, estímulo para voar sobre outras regiões não exploradas. Há muitos trabalhos na literatura que lidam com a seleção de parâmetros do algoritmo a fim de contornar este problema, como [38], [37], [47], [48], [49], [50].

3.3.1 Seleção de Parâmetros

Nesse modelo, há uma série de parâmetros que podem ser utilizados para se conseguir controlar a convergência ou atingir outro objetivo importante. Conforme visto nas seções anteriores, os parâmetros que podem ser definidos são: $\omega, \phi_1, \phi_2 \; e \; v\max$, além da topologia da vizinhança. As configurações dos parâmetros determinam como o algoritmo efetuará a otimização do espaço de busca.

Uma solução usual genérica é definir $\phi_1 = \phi_2 = 2$ e o coeficiente de inércia como $\omega = 0{,}8$. A velocidade máxima *vmax* pode ser definida como 10% da dimensão média do espaço de busca [37].

No entanto, cada aplicação, com objetivos mais específicos, tem uma configuração dos parâmetros própria. Deve-se, portanto, verificar os efeitos dos vários parâmetros, a fim de escolher aquele que melhor se adapta ao problema. Alguns trabalhos que abordam a seleção de parâmetros de PSO são [51], [52] e [53].

Nas seções seguintes, é abordado o efeito de cada um dos parâmetros no processo de otimização.

3.3.1.1 Parâmetros ϕ_1 e ϕ_2

Esses dois parâmetros definem a influência do melhor da partícula e melhor global na velocidade da partícula.

Se ϕ_1 for muito maior que ϕ_2, a partícula tende a escolher as direções que a levem próxima de seu próprio ótimo, ao invés de procurar ótimos globais. Ou

seja, a partícula sente-se mais atraída pela direção de seu ótimo do que pela direção da melhor posição encontrada pela vizinhança. Por sua vez, quando ϕ_2 é maior que ϕ_1, a partícula tende a se deslocar para a direção do ótimo global. Essa direção da partícula tem relação com o ponto de atração, dado pela fórmula [37]:

$$\frac{\phi_1 \vec{p}(t) + \phi_2 \vec{g}(t)}{\phi_1 + \phi_2}.$$

O caso particular, quando $\phi_2 = 0$, converte todas as partículas em escaladores-da-montanha independentes, pois o coeficiente de aprendizado social $\phi_2(\vec{p}(t) - \vec{x}(t))$ é 0. A partícula encontra a melhor posição vizinha a sua e caminha até lá. Isso continua até um melhor local ser encontrado. No outro caso, $\phi_1 = 0$, a nuvem se transforma em um escalador-da-montanha global, pois há apenas um único ponto de atração para todas as partículas da vizinhança g(t). Caso $\phi_1 = \phi_2$, todas as partículas serão atraídas pela média de p(t) e g(t).

Outro fato diz respeito à magnitude dos parâmetros. Quanto maior forem ϕ_1 e ϕ_2, maior será a aceleração da partícula. Dessa forma, as partículas reagem rapidamente a qualquer mudança da busca. Definindo-os com valor baixo, as partículas reagirão com demora às mudanças e farão grandes curvas para mudar de direção, passando longe, provavelmente, do ponto de atração [37].

3.3.1.2 Coeficiente de Inércia ω

O coeficiente de inércia regula a influência da velocidade anterior da partícula na nova velocidade. Com um ω baixo, apenas um pequeno momento é preservado. Assim, a partícula consegue mudar de direção rapidamente. Quando ω é alto (>1), temos uma demora ou dificuldade na mudança de direção da partícula, pois sua velocidade anterior influencia bastante na velocidade atual, implicando numa convergência mais lenta. Porém, quando $\omega = 0$, o conceito de velocidade anterior desaparece. Assim, a partícula se move a cada geração de forma independente do movimento anterior [37] [38].

3.3.1.3 Velocidade Máxima

A velocidade máxima procura evitar a explosão e divergência. Com a utilização do coeficiente de inércia, no entanto, ela se tornou desnecessária. Bom, pelo menos a convergência pode ser assegurada sem este parâmetro [48]. No entanto, por exemplo, se uma partícula está posicionada no final do espaço de busca e *p(t)* e *g(t)* estão posicionados no outro lado do espaço, a partícula pode obter uma velocidade de quatro vezes o tamanho do espaço de busca, quando $\phi_1 = \phi_2 = 2$, o que não tem sentido. Nesse caso, a *vmax* pode ajudar a economizar tempo de computação, evitando cálculos de aptidão e desaceleração da partícula [37]. Aspectos do efeito da velocidade máxima são analisados em [37], [52] e [53].

3.3.1.4 Topologia da Vizinhança

Basicamente, a topologia da vizinhança influencia a convergência da otimização. Quanto menor a média da distância topológica entre duas partículas quaisquer, mais rápida é a convergência.

Duas classes de vizinhança podem ser distinguidas:

- Vizinhança física ou geográfica, que relaciona os vizinhos às partículas que estão mais próximas geograficamente no espaço de busca.
- Vizinhança social, que estabelece uma lista de partículas vizinhas para cada partícula da nuvem, não sendo, necessariamente, as partículas mais próximas geograficamente.

Alguns aspectos da influência da topologia da vizinhança na otimização de PSO são vistos em [54]. A Figura 3.2 mostra alguns exemplos de topologia para a vizinhança.

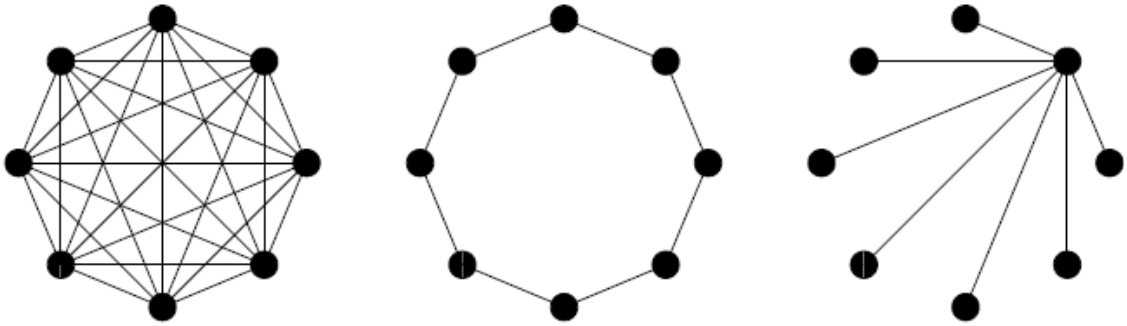

Figura 3.2 — Topologias de Vizinhança.

3.4 NUVEM DE PARTÍCULAS PARA PROBLEMAS MULTIOBJETIVO

A técnica de nuvem de partículas tem sido utilizada, com sucesso, na resolução de problemas de otimização monoobjetivo discretos e contínuos. No entanto, há problemas em que temos de otimizar não apenas um aspecto, mas dois ou mais. Nesse caso, trata-se de um problema de otimização multiobjetivo, onde temos mais de uma função de aptidão. Para resolver esse tipo de problema, deve-se então considerar todos os objetivos. Sendo assim, se $\Pi \subseteq S \subseteq R^n$ é um subespaço praticável do espaço de busca, temos de encontrar $\vec{x} \in \Pi$, tal que:

$$\alpha(\vec{x}) \leq \alpha(\vec{y}), \qquad \textit{para problemas de minimização,}$$

$$\alpha(\vec{x}) \geq \alpha(\vec{y}), \qquad \textit{para problemas de maximização,}$$

para cada $\vec{y} \in \Pi$ e para cada $\alpha \in \Phi$, onde Φ é o conjunto de funções de aptidão que devem ser otimizadas. Logo, $|\Phi|$ é o número de objetivos que queremos otimizar.

Porém, para alguns problemas, não há tal solução. As melhores soluções são aquelas que se aproximam da perfeição, formando uma fronteira de valores conhecida como Fronteira de Pareto [6]. Assim, a solução para problemas multiobjetivo é encontrar soluções ótimas de Pareto. Essas soluções são encontradas buscando-se por soluções não dominadas do problema, que são aquelas que não são piores que nenhuma outra solução. Assim, formalizando:

- Uma solução $\vec{x} \in \Pi$ para um problema multiobjetivo é não dominada se não houver nenhuma solução $\vec{y} \in \Pi$ que domine $\vec{x}$.
- Uma solução $\vec{x} \in \Pi$ domina fortemente outra solução $\vec{y} \in \Pi$ ($\vec{x} \prec \vec{y}$) se (minimização):

  $\forall \alpha \in \Phi, \quad \alpha(\vec{x}) \leq \alpha(\vec{y}),$ e

  $\exists \alpha \in \Phi, \quad \alpha(\vec{x}) < \alpha(\vec{y}).$ (3.2.1)
- Uma solução $\vec{x} \in \Pi$ domina fracamente outra solução $\vec{y} \in \Pi$ ($\vec{x} \prec= \vec{y}$) se (minimização):

  $\forall \alpha \in \Phi, \quad \alpha(\vec{x}) \leq \alpha(\vec{y}).$

Grande parte dos recentes trabalhos em algoritmos multiobjetivo é formulada em termos da não dominância e soluções ótimas de Pareto [55]. A Figura 3.3 mostra um exemplo de soluções na Fronteira de Pareto para um problema biobjetivo.

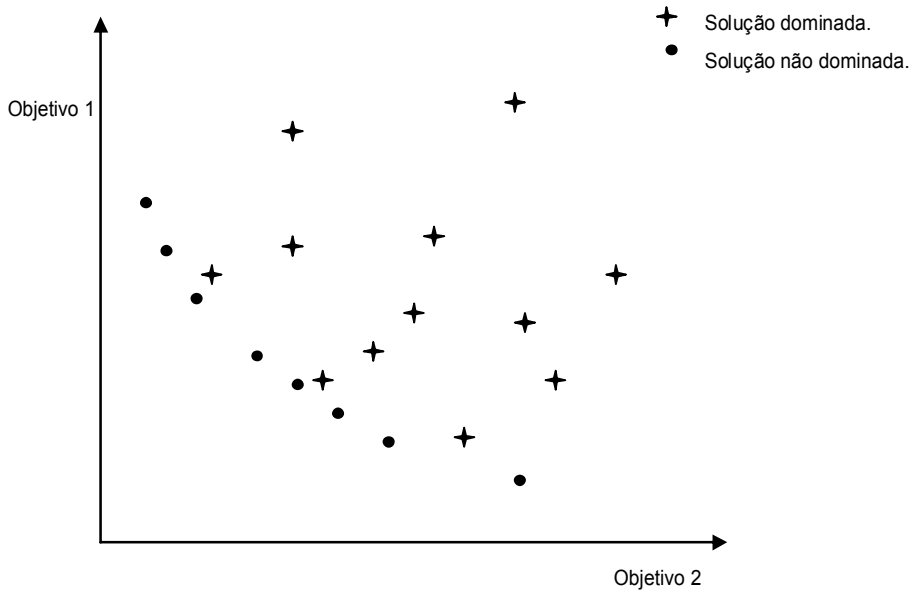

Figura 3.3 — Soluções não dominadas formando uma fronteira em direção ao ponto ótimo (0,0).

Um algoritmo de nuvem de partículas para a solução de problemas multiobjetivo foi apresentado por Coello e Lechuga [56]. No MOPSO (*Multiple Objective Particle Swarm Optimization),* o que muda em relação ao PSO é o

número de funções de aptidão e o conceito de melhor global. Nesse caso, uma solução é melhor que outra se esta solução domina a outra. Logo, não existe um melhor global, mas um conjunto de soluções não dominadas.

Dessa forma, o $\vec{g}(t)$ da partícula deve ser uma das partículas não dominadas do problema, até o tempo *t*. Para isso, o registro histórico das melhores soluções encontradas por uma partícula pode ser usado para guardar soluções não dominadas (similarmente à noção de elitismo usada na computação evolucionária multiobjetivo) [56].

O algoritmo se baseia na idéia de haver um repositório global para que cada partícula deposite suas experiências de movimentação a cada ciclo. Se alguma experiência da partícula não for dominada por nenhuma solução do repositório, a solução é incluída.

Esse repositório é usado pelas partículas para que escolham um líder para seguir. Assim, para que as partículas não escolham, todas, o mesmo líder, existe um mecanismo baseado na geração de hipercubos, dividindo-se o espaço objetivo. Dessa forma, cada partícula não dominada possuirá uma região do espaço objetivo reservada para ela. Com isso, toda partícula cujos objetivos se posicionarem dentro desse espaço seguirá a partícula não-dominada dona da região. Essa é a diferença básica entre PSO e o MOPSO. O Pseudocódigo do algoritmo MOPSO é apresentado na Figura 3.4 [56].

Existem várias formas de uma partícula escolher sua partícula líder. Aqui, apresentamos uma delas: a menor distância-sigma [57], que é a forma utilizada no nosso algoritmo. Outros métodos da escolha de um guia global para otimização multiobjetivo com nuvem de partículas podem ser vistos em [55].

O método da distância sigma divide o espaço objetivo em linhas da posição da partícula à origem. Isso é feito calculando o vetor sigma, abaixo, para cada partícula do repositório e da nuvem:

$$\sigma = \begin{pmatrix} f_1(x)^2 - f_2(x)^2 \\ f_2(x)^2 - f_3(x)^2 \\ \vdots \\ f_N(x)^2 - f_1(x)^2 \end{pmatrix} / (f_1(x)^2 + f_2(x)^2 + f_3(x)^2 + \cdots + f_N(x)^2),$$

1. Para cada partícula *i* da nuvem, faça:
   a. Inicia $\vec{x}_i$ com uma solução aleatória para o problema.
   b. Inicia $\vec{v}_i$ com uma velocidade aleatória < V_MAX.
   c. $\vec{p}_i \leftarrow \vec{x}_i$
2. Avalia partículas.
3. Encontra soluções não dominadas, guardando-as no Repositório.
4. Divide-se o espaço de busca entre as soluções do Repositório.
5. Enquanto não atingir um critério de parada:
   a. para cada partícula *i* da nuvem, faça:
      i. $\vec{v}_i \leftarrow \omega.\vec{v}_i + \phi_1(\vec{p}_i - \vec{x}_i) + \phi_2(\vec{R}_h - \vec{x}_i)$
         1. obs.: $\vec{R}_h$ é uma partícula do repositório, escolhida através da divisão do espaço objetivo entre as soluções não dominadas.
      ii. $\vec{x}_i \leftarrow \vec{x}_i + \vec{v}_i$
      iii. Avalia partícula.
         1. obs.: Ao invés de um único valor, a partícula terá um valor para cada objetivo do problema.
      iv. $\vec{p}_i \leftarrow melhor\ entre\ \vec{p}_i\ e\ \vec{x}_i$
         1. obs.: Utilizando a dominância de Pareto, uma solução é melhor que outra se domina-la.
   b. Atualiza Repositório com as partículas não dominadas.
   c. Divide-se o espaço de busca, encontrando $\vec{R}_h$ das partículas
6. Retorna Repositório.

Figura 3.4 — Pseudocódigo do MOPSO [56].

onde N é o número de objetivos do problema; f_i é a i-ésima função de aptidão, para *i* de 1 a N. Nesse caso, a partícula R_h seria a partícula do repositório que detém o vetor sigma com a menor distância euclidiana do vetor sigma da partícula da nuvem. A Figura 3.5 mostra o comportamento dessa divisão em um espaço 2-objetivo.

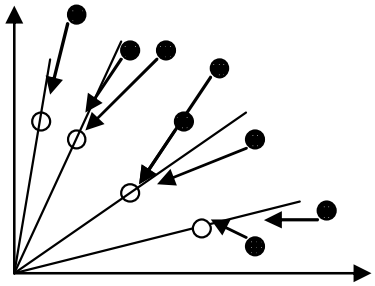

Figura 3.5. Método Sigma. Pontos brancos são partículas não-dominadas. Pontos pretos são as partículas da nuvem. As flechas indicam qual partícula não-dominada é selecionada para cada partícula da nuvem.

Este capítulo descreveu a técnica de nuvem de partículas e sua variante multiobjetivo, como foram definidos por seus autores. No próximo capítulo, é descrita nossa técnica de nuvem de partículas multiobjetivo para o aprendizado de regras de classificação.

4 DESCRIÇÃO DO ALGORITMO

Neste capítulo, é feita a descrição do algoritmo, desde a representação da regra, como a nuvem é iniciada, os passos do algoritmo até sua validação.

Nosso algoritmo é uma implementação do MOPSO para aprendizado de regras e pode ser definido como uma função $f(Ex[M,N],P,G,f_1,f_2,\cdots,f_K)$, sendo *K* >= 2 o número de objetivos do problema e f_i a i-ésima função objetivo, com *i* de 1 a *K*. No caso, para um problema de dois objetivos, o algoritmo seria $f(Ex[M,N],P,G,f_1,f_2)$. O parâmetro *Ex[M,N]* é o conjunto de exemplos da base de treinamento para M exemplos de N atributos. P representa o número de partículas a ser utilizado para cada classe. G é o número de gerações. A função retorna um conjunto $C[M_1,N]$ que é o classificador que contém M_1 regras.

Com essa configuração, os parâmetros da equação da velocidade são aleatórios, o limiar da fronteira *E* é zero. No entanto, nos casos em que se queira definir valores diferentes para esses parâmetros, podemos definir o algoritmo como $f(Ex[M,N],P,G,\omega,\phi_1,\phi_2,E,f_1,f_2,\cdots,f_K)$.

Nas seções seguintes, é feita a descrição detalhada do algoritmo.

4.1 REPRESENTAÇÃO DA REGRA

Em nosso algoritmo, seguimos a abordagem de *Michigan* para a representação de uma partícula. Nessa abordagem, cada partícula é uma regra, não um classificador. A evolução é feita com as regras, não com os classificadores. Outra abordagem possível seria a de *Pittsburgh*, onde cada partícula representa um classificador completo, ou seja, um conjunto de regras. A vantagem de *Michigan* é que o indivíduo é bem mais simples e seus operadores mais intuitivos. Porém, possui a desvantagem de não considerar interações entre as regras, o que seria possível com a de *Pittsburgh* [58].

Dessa forma, uma regra é representada pela posição da partícula no espaço de busca. Este espaço é definido pela base de dados. Se uma base de dados tem N atributos, o espaço de busca é N-dimensional. Além disso, o

tamanho de cada dimensão é definido pelo número de valores possíveis de cada atributo da base. Assim, a primeira dimensão teria o número de valores que são possíveis para o primeiro atributo da base; a segunda dimensão, para o segundo atributo; e assim por diante. Para representar os casos em que o atributo não aparece na regra, há ainda a possibilidade de um valor coringa '?' para cada atributo da base, com exceção do atributo objetivo.

Para melhor visualização, um exemplo é apresentado, com base na Tabela 4.1.1. Vê-se que a base possui catorze exemplos e cinco atributos. **'Jogar'** é o atributo objetivo da base. Logo, as partículas podem se movimentar num espaço de cinco dimensões onde:

Tabela 4.1.1 — Base de treinamento

índice	Aparência	Temperatura	Umidade	Vento	Jogar
1	Sol	quente	alta	Não	Não
2	Sol	quente	alta	Sim	Não
3	nublado	quente	alta	Não	Sim
4	chuvoso	fresco	alta	Não	Sim
5	chuvoso	frio	normal	Não	Sim
6	chuvoso	frio	normal	Sim	Não
7	nublado	frio	normal	Sim	Sim
8	Sol	fresco	alta	Não	Não
9	Sol	frio	normal	Não	Sim
10	chuvoso	fresco	normal	Não	Sim
11	Sol	fresco	normal	Sim	Sim
12	nublado	fresco	alta	Sim	Sim
13	nublado	quente	normal	Não	Sim
14	chuvoso	fresco	alta	Sim	Não

1. Atributo **aparência** com os valores: '?'(0), sol (1), nublado (2) e chuvoso (3).
2. Atributo **temperatura** com os valores: '?' (0), quente (1), fresco (2) e frio (3).
3. Atributo **umidade** com os valores: '?' (0), alta (1) e normal (2).
4. Atributo **vento** com os valores: '?' (0), não (1) e sim (2).
5. Atributo **jogar** (classe) com os valores: não (0) e sim (1).

Os números em parênteses são os identificadores dos valores de cada atributo. Logo, a regra **Se Aparência = Sol E Vento = Sim Então Jogar = SIM,** seria representada como **[Sol, ?, ?, Sim, SIM],** que por sua vez seria equivalente a **[1,0,0,2,1]**.

Com essa representação numérica, é possível utilizar as mesmas fórmulas da velocidade e posição do MOPSO.

A forma de representação da regra na partícula é obtida diretamente das informações contidas na base de treinamento, como atributos e valores possíveis para cada atributo. É um processo automático, não sendo necessária nenhuma intervenção.

4.2 INICIAÇÃO DA PARTÍCULA

Nessa etapa, as partículas são criadas na nuvem e espalhadas pelo espaço N-dimensional. Isso pode ser feito de várias formas. Aqui, apresentamos três delas. Em nossos experimentos, somente a iniciação por cobertura (seção 4.2.3) foi utilizada, com dois atributos na condição.

4.2.1 Iniciação Aleatória

Esse é o tipo mais simples de iniciação de partículas. Basicamente, consiste de se colocar valores aleatórios nos atributos de cada partícula, dessa forma, espalhando-as no espaço de busca.

Possui várias desvantagens como a formação, dependendo da base de dados, de regras muito específicas, dificultando o aprendizado e a evolução. Além disso, corre-se o perigo de produzir regras que não cobrem nenhum exemplo, ou que estejam muito agrupadas com outras regras, prejudicando a exploração do espaço. Nesse caso, é desejável utilizar uma roleta para a escolha do valor do atributo, dando ao valor coringa ‘?’ maior prioridade.

4.2.2 Iniciação por Mutação da Mais Genérica

Esse tipo de iniciação tenta corrigir alguns problemas da iniciação aleatória. Consiste, basicamente, em iniciar todos os atributos da partícula, com exceção da classe, com a forma genérica **‘?’.** Em seguida, de forma aleatória, alteram-se

alguns atributos da partícula. Isso permite a geração de partículas mais genéricas, controlando o problema da especialização da partícula, e, além disso, reduzindo o número de regras que não cobrem exemplos da base, pois por ser mais genérica, tende a abranger mais exemplos.

A escolha dos atributos é feita com base numa roleta, onde os atributos mais freqüentes da base e o valor coringa '?' têm maior probabilidade de serem sorteados.

4.2.3 Iniciação por Cobertura

A iniciação por cobertura tenta controlar o problema das regras específicas e da cobertura dos exemplos da base. Primeiramente, cada partícula é iniciada com um exemplo da base, escolhido de forma aleatória.

Em seguida, as partículas são generalizadas até atingirem certo número definido de atributos especificados. Por exemplo, o exemplo 3 da base da Tabela 4.1, **[nublado,quente,alta,Não,SIM]**, poderia ser reduzido, definido o número de atributos igual a dois, a **[nublado, ?,?,Não,SIM].** Da mesma forma, o exemplo 5 **[chuvoso,frio,normal,Não,SIM]**, definido o número de atributos igual a um, poderia ser reduzido para **[?,frio,?,?,SIM]**.

Com isso, tem-se a certeza de que a regra cobre ao menos um exemplo da base de dados. Isso auxilia na melhoria da qualidade das regras já na nuvem inicial.

4.3 MOVIMENTAÇÃO DAS PARTÍCULAS

A movimentação das partículas é feita com base nas fórmulas de atualização da velocidade e posição, com topologia de vizinhança completa, ou seja, cada partícula é vizinha de todas as outras:

$$\vec{v}_i \leftarrow \omega.\vec{v}_i + \phi_1(\vec{p}_i - \vec{x}_i) + \phi_2(\vec{g}_i - \vec{x}_i) \qquad (4.3.1)$$

$$\vec{x}_i \leftarrow \vec{x}_i + \vec{v}_i \qquad (4.3.2)$$

Cada atributo *i* da partícula é calculado isoladamente dos outros, utilizando-se seu valor numérico. Nesse caso, o parâmetro *vmax* não é considerado, pois não há sentido em se pensar em limitação da velocidade da partícula. Essa velocidade máxima, num espaço limitado, poderia se equivaler à taxa de mutação de AG. Essa taxa de mutação, quando alta, pode fazer com que espécies muito boas se tornem ruins na população. No nosso caso, isso não seria um problema, pois as melhores soluções estão salvas no grupo dos melhores globais.

Como exemplo, considere a base da Tabela 4.1.1 e a partícula [1,0,0,2,SIM]. Consideremos também sua velocidade inicial [0,0,0,0,0], sua melhor posição atingida [2,0,0,2,SIM] e partícula guia [2,1,2,2,SIM]. Nesse caso, sua velocidade seria alterada para:

$$\vec{v} = \omega[0,0,0,0,0] + \phi_1([2,0,0,2,SIM] - [1,0,0,2,SIM]) + \phi_2([2,1,2,2,SIM] - [1,0,0,2,SIM]) \rightarrow$$
$$\vec{v} = [0,0,0,0,0] + \phi_1[1,0,0,0,0] + \phi_2[1,1,2,0,0].$$

Considerando $\phi_1 = 2$ e $\phi_2 = 2$, temos:

$$\vec{v} = [2,0,0,0,0] + [2,2,4,0,0] = [4,2,4,0,0].$$

Logo, a próxima posição será:

$$\vec{x} = [1,0,0,2,SIM] + [4,2,4,0,0] = [5,2,4,2,SIM].$$

Aqui, surge um problema. Não há a posição [5,2,4,2,SIM] no espaço. Poderíamos contornar essa situação desqualificando essa solução na função de aptidão, dando-lhe valor zero. No entanto, isso seria uma computação desnecessária, mantendo na nuvem um indivíduo que sabemos ser inválido. Outra idéia seria a de limitar o espaço, trazendo todo atributo que excede a fronteira para seu limite. Assim, a posição da partícula ficaria em [3,2,2,2,SIM], que é uma partícula válida. No entanto, isso reduziria drasticamente a diversidade da nuvem, pois com o avanço da otimização, várias partículas ultrapassariam os limites do espaço. Para contornar isso, o novo valor do atributo que foi excedido poderia ser escolhido de forma aleatória, ou podemos pensar em eixos circulares do espaço.

Eixo circular significa dizer que quando uma partícula excede o limite de um eixo (atributo) é como se ela tivesse voltado ao começo e movimentado mais o restante excedido. No caso de nossa partícula exemplo, temos para o primeiro atributo: 5 – 3 (valor máximo do atributo) = 2 (quantidade excedente). Logo, o

valor do atributo aparência seria igual a 1 (sol). Isso seria equivalente à função de resto de divisão (mod), pois 5 mod 4, que é o número de valores para o atributo, é igual a 1. Assim, a posição da partícula passaria de [5,2,4,2,SIM] para [1,2,1,2,SIM].

Outra forma de se fazer essa movimentação seria fazer a partícula andar na posição contrária do eixo quando chegasse a seu limite. No caso do primeiro atributo, a partícula chegaria a 3 e, em seguida, ao invés de ir para a posição quatro e cinco, voltaria para a posição dois e um. Logo, para esse caso a partícula passaria da posição [5,2,4,2,SIM] para [1,2,0,2,SIM].

Nosso algoritmo utiliza a movimentação com eixo circular. Por esse motivo, *vmax* torna-se completamente dispensável.

4.4 FUNÇÕES DE APTIDÃO

As funções de aptidão das partículas são funções que identificam a qualidade das regras. Relacionam-se, geralmente, com sua quantidade de acertos na base de dados, como por exemplo, a confiabilidade positiva (taxa de acerto dos exemplos que a regra diz serem positivos), confiabilidade negativa (taxa de acerto dos exemplos que a regra diz serem negativos), sensitividade (taxa de acerto dos exemplos positivos da base) e especificidade (taxa de acerto dos exemplos negativos da base). Em nossos experimentos, são observados esses quatro objetivos otimizados como confiabilidade positiva com negativa (com correção de Laplace) e sensitividade com especificidade. Pode-se utilizar também qualquer quantidade de objetivos e qualquer objetivo da Tabela 2.3.2, bem como qualquer outro objetivo criado pelo usuário.

A confiabilidade positiva talvez obtenha melhores resultados quando associada à votação por confiança (VC). Isso porque são selecionadas as regras que mais têm certeza no que dizem positivamente. Dessa forma, com a votação por confiança, a regra só se pronunciará proporcionalmente à sua própria confiança e se cobrir o exemplo. No entanto, pode ocorrer de muitos exemplos não serem cobertos. Com a sensitividade, muitos falsos positivos podem aparecer,

pois as regras cobrem muitos exemplos, inclusive negativos. Em geral, procura-se contrabalançar a sensitividade com a especificidade. No entanto, mesmo assim, para um esquema de votação, e em especial, votação por confiança, a sensitividade com especificidade deve gerar votos falsos positivos, visto que os pontos extremos (1,0) e (0,1) fazem parte da fronteira. A votação por confiança com voto negativo (VCVN), geralmente, faz sentido quando associada à maximização da confiabilidade negativa, pois as regras terão mais certeza ao dizer não, e votarão ***não*** proporcionalmente a essa certeza. Unindo-a à confiabilidade positiva, evita-se que o classificador deixe de classificar instâncias não cobertas, uma vez que a regra vota mesmo quando não cobre o exemplo. Logo, nos casos em que as fronteiras descobertas não cobrem corretamente os exemplos da base, é recomendável utilizar alguma forma de voto negativo, para que não se deixe de classificar exemplos.

4.5 GRUPOS GLOBAIS

No algoritmo, há dois grupos ou repositórios globais: o grupo dos melhores globais, que é formado pelas partículas não dominadas da nuvem, e o grupo das partículas do classificador, que é formado pelas partículas que foram selecionadas para integrar o classificador. Para cada grupo, há um limiar que determina, para cada função de aptidão, o valor mínimo aceito para que a regra pertença ao grupo. Em geral, o limiar é zero, para que a dominância seja equivalente à dominância de Pareto, descrita na equação 3.2.1. Porém, este limiar pode ser livremente alterado para se tentar resultados melhores.

Uma partícula pertence a um grupo se for uma partícula não dominada com relação a um limiar *E*. Para isso, uma partícula $\vec{x}$ domina outra partícula $\vec{y}$, se:

$$\forall \alpha \in \Phi, \quad (\alpha(\vec{x})/(1-E)) \leq \alpha(\vec{y}) \text{ , e}$$
$$\exists \alpha \in \Phi, \quad (\alpha(\vec{x})/(1-E)) < \alpha(\vec{y}). \qquad (4.5.1)$$

sendo Φ o conjunto das funções de aptidão da otimização e $0 <= E < 1$.

O grupo dos melhores globais influencia diretamente a movimentação das partículas no espaço, uma vez que os guias de cada partícula são escolhidos

desse grupo. O outro grupo é o próprio classificador, que é o resultado do algoritmo, usado também para a validação.

Logo, uma mesma partícula pode, por exemplo, não ser aceita como melhor global, mas aceita como regra de classificação, e vice-versa, dependendo do valor do limiar. Se os limiares dos dois grupos forem iguais, então os grupos serão iguais.

Com o aumento do limiar, o número de partículas do grupo aumenta, podendo, no caso do classificador, levar a resultados mais precisos. Experimentos com o limiar do classificador são discutidos na Seção 5.3.

Há ainda a possibilidade de incluir uma regra ou não em um grupo com base em um suporte mínimo, vencida a condição de dominância. Esse caso de seleção não é abordado neste trabalho, pois consideramos o suporte mínimo zero.

4.6 CORPO DO ALGORITMO

O algoritmo começa pela criação da nuvem e iniciação das partículas. Isso é feito considerando as informações da base de dados. É interessante mencionar que uma otimização com 10 partículas, para um problema de duas classes, criará duas nuvens com 10 partículas, e terá dois grupos dos melhores globais e dois grupos classificadores, um para cada classe. Se fossem três classes, seriam três nuvens de 10 partículas e assim sucessivamente.

Depois de espalhadas as partículas, começa-se o processo iterativo para o número de gerações definido, ou até atingir a convergência, se este critério de parada for escolhido. Assim, a cada geração, avalia-se a aptidão de cada partícula da nuvem, com base nos objetivos definidos, utilizando-se a base de treinamento.

Em seguida, tenta-se inserir cada partícula no grupo de melhores globais e no grupo do classificador, referentes à classe da partícula. Se a partícula não for dominada por nenhuma outra partícula do grupo, nos conceitos de dominância definidos na equação 4.5.1, e se já não existir uma partícula igual, inclui-se a partícula no grupo. Nesse processo, se a partícula dominar qualquer outra do grupo, esta última deve ser retirada.

O passo seguinte é a movimentação das partículas. Nesse passo, é escolhido o melhor global da partícula, pelo método da distância sigma. Esse método nos garante certa diversidade. Porém, para ter certeza de que as partículas não escolherão sempre o mesmo líder, criamos um tipo de roleta com as distâncias sigma. Assim, quanto menor a distância entre os vetores sigma de uma partícula *p* da nuvem com o de uma partícula não-dominada *P*, maior é a probabilidade de *P* ser escolhida como guia de *p*.

Com esse guia, com a melhor posição da partícula e com sua posição atual, realiza-se o cálculo da velocidade com a Equação 4.3.1. Com essa velocidade, a posição da partícula é atualizada na forma da Equação 4.3.2, utilizando-se eixos circulares. Os parâmetros ω, ϕ_1, ϕ_2, por padrão, são escolhidos aleatoriamente cada vez que se faz um movimento de uma partícula. No entanto, podem ser definidos valores fixos.

Após a movimentação das partículas, começa uma nova geração com a avaliação das regras, inclusão nos grupos globais e movimentação. Ao término das gerações, as regras nos grupos do classificador de cada classe são unidas em um só grupo. Esse grupo é retornado pelo algoritmo como o classificador encontrado.

Em seguida, é feito o processo de cálculo da AUC do classificador para cada classe, utilizando-se preferencialmente votação de confiança com voto negativo. O pseudocódigo do algoritmo é apresentado na Figura 4.1.

1. Criação das nuvens e iniciação das partículas.
2. Enquanto não atingir um critério de parada:
    a. Avalia as partículas.
    b. para cada partícula *i* da nuvem, faça:
        i. Tenta inserir a partícula em cada um dos grupos globais, segundo limiares E1 e E2.
        ii. Escolha do líder global da partícula, do grupo dos melhores globais.
        iii. Movimentação da partícula.
            a. $\vec{v}_i \leftarrow \omega.\vec{v}_i + \phi_1(\vec{p}_i - \vec{x}_i) + \phi_2(\vec{R}_h - \vec{x}_i)$
            b. $\vec{x}_i \leftarrow \vec{x}_i + \vec{v}_i$
3. Retorna união dos grupos classificadores.

Figura 4.1 — Pseudocódigo do algoritmo de aprendizado de regras com o MOPSO.

4.7 CRITÉRIO DE PARADA

O critério de parada do algoritmo mais simples de ser implementado é definir um número limite de gerações. Com isso, o algoritmo pára assim que esse número for alcançado. O problema desse critério é evitar a melhora do algoritmo quando este ainda está longe da convergência, ou, de forma contrária, desperdiçar tempo em otimização desnecessária, quando o algoritmo já convergiu.

Para evitar isso, é possível definir um critério relativo à convergência do algoritmo. Nesse caso, o algoritmo pára se não obtiver certa quantidade de regras novas durante um período de gerações seguidas. Essa quantidade pode ser uma porcentagem da quantidade total de regras possíveis de serem descobertas em uma geração, que é o número de partículas vezes o número de classes. Dessa forma, dois parâmetros precisam ser definidos. O primeiro é o parâmetro que expressa a porcentagem do número de regras possíveis na geração. Conforme dito, esse valor tem relação com o número de partículas da nuvem. O segundo parâmetro representa o número de gerações seguidas que deverão obter número de regras novas inferior ao do primeiro parâmetro para o algoritmo parar. A desvantagem dessa abordagem é que, além do número a mais de parâmetros, o

tempo de execução nas maiores bases pode ser muito alto, sendo que, às vezes, a melhora não é significante.

4.8 VALIDAÇÃO DO RESULTADO

Terminada a otimização da base de dados, temos, no grupo do classificador, as regras que foram escolhidas para, em conjunto, classificar as novas entradas. Mas, para termos uma medida da precisão desse classificador para possíveis novas entradas que não se encontram na base de treinamento, temos de validá-lo segundo um método estatístico.

A otimização é toda feita com a base de treinamento. O classificador gerado é então validado com a base de testes. Isso mostra a capacidade do classificador em classificar instâncias totalmente novas, que não estavam presentes na base de treinamento.

4.9 COMPLEXIDADE DO ALGORITMO

O algoritmo de treinamento é um processo iterativo de número de gerações vezes o número de partículas. Logo, podemos considerar, a priori, a ordem de complexidade do algoritmo como *O (P.G)*, sendo P o número definido de partículas e G o número de gerações. No entanto, cada partícula é avaliada com os exemplos da base de treinamento. Dessa forma, sendo N o número de exemplos da base de treinamento, podemos considerar *P.G.N* o número de comparações. Ainda assim, há a comparação de cada partícula com as partículas dos repositórios globais. Dessa forma, de modo mais simplificado, sendo A o número de partículas médio do repositório, então a complexidade pode ser formulada como *P.G.N + P.G.A.* Ainda assim, isso é feito para cada atributo do problema. Sendo *M* o número de atributos da base, assim, o número de comparações fica em *P.G.M.N + P.G.M.A.*

4.10 RESTRIÇÕES E PRINCIPAIS OBJETIVOS

É importante salientar que o algoritmo proposto só lida com atributos categóricos. Sendo assim, é exigida a discretização dos atributos contínuos da base antes da aprendizagem. Logo, atributos que devem formar regras na forma *salário > 300,* por exemplo, precisam estar em forma categórica na base de dados. Isso simplifica bastante a aprendizagem, porém há certa perda de informação.

Outra desvantagem do algoritmo, a priori, é a quantidade de parâmetros a configurar para a otimização. Embora possamos já padronizar alguns parâmetros, como o limiar do grupo do classificador $E = 0$ (que influencia na seleção das regras para o classificador), e ainda definir algum critério de parada que não o número de gerações, bem como a eliminação do *vmax*, ainda assim, o algoritmo apresenta uma alta quantidade de parâmetros da nuvem de partículas a ser configurada pelo usuário.

Mesmo assim, acreditamos que o algoritmo de nuvem de partículas oferece vantagem sobre a utilização de algoritmo genético, pois:

- O algoritmo genético tem a capacidade de manter na população os melhores indivíduos. Com o passar das gerações, vários integrantes da mesma espécie estão presentes na população. Para nosso caso, isso é prejudicial;
- Com a manipulação dos parâmetros, conseguimos evitar esta uniformidade da população, mantendo ainda, certa inteligência evolutiva, com a utilização dos guias globais e locais do repositório;

O objetivo é que uma partícula nunca seja igual de uma geração para outra e não colida com nenhuma outra partícula. Mesmo com elitismo, isso não é tão facilmente alcançado com AG, justamente porque a população tende a evoluir para uma só espécie.

5 EXPERIMENTOS

Para verificar o desempenho do algoritmo, foram realizados diversos experimentos com as bases de dados da Tabela 5.1. Essas bases de dados foram obtidas na UCI [59]. Seus atributos contínuos foram todos discretizados, pois o algoritmo só trabalha com valores discretos. Além disso, as bases que possuíam mais de duas classes foram reduzidas a problemas de duas classes, pegando-se a classe de menor freqüência como positiva, juntando-se as demais como classe negativa. Note que a tabela traz o número de regras possíveis para cada base de dados. Com isso, é possível perceber que os experimentos foram feitos com bases bem pequenas, no caso de *haberman* com mil regras possíveis, até bases muito grandes, no caso de *ionosphere*, passando dos 200 nonilhões de regras possíveis.

Tabela 5.1 — Bases de dados da UCI.

índice	Base	Num. atributos	num. exemplos	Classe maj. (%)	Num. regras possíveis
1	breast	10	683	65,00	1.800.000.000
2	bupa	7	345	57,98	77.760
3	ecoli	8	336	89,58	1.458.000
4	german	21	1.000	70,00	$3,99 \times 10^{13}$
5	glass	10	214	92,07	10.206.000
6	haberman	4	306	73,53	1.000
7	heart	14	270	55,55	192.893.400
8	ionosphere	34	351	64,10	$2,60 \times 10^{32}$
9	new-thyroid	6	215	96,06	26.400
10	pima	9	768	65,10	6.667.920

Todas as partículas foram iniciadas com iniciação por cobertura com dois atributos na condição. Para o critério de parada, foi utilizado um limite de gerações, por sua facilidade de implementação e análise.

Em todos os experimentos deste trabalho nos quais foi feito o cálculo de desempenho do classificador, foi utilizada a validação cruzada estratificada em dez partições. Assim, a base de dados é dividida em dez partições. A cada execução, é utilizada uma partição para teste, sendo as nove partições restantes utilizadas como uma base de treinamento. Logo, são dez execuções para cada

classe da base, uma vez que os valores de AUC e regras de cada classe são calculados isoladamente. O resultado final desses valores é a média aritmética das dez bases.

As seções deste capítulo tratam e discutem cada um dos experimentos realizados com o algoritmo.

5.1 EXPERIMENTOS COM PARÂMETROS DE VELOCIDADE

Conforme explicado no capítulo 3, os parâmetros da velocidade exercem uma importante influência no desempenho de PSO e MOPSO. Porém, a priori, essa influência não é conhecida para o resultado do classificador. Visto que, nesse caso, não há uma lógica entre os termos e suas posições no espaço de busca, não há, assim, uma influência conhecida dos parâmetros $\omega, \phi_1 \; e \; \phi_2$ para alcançar a solução. Os estudos relacionados à influência desses parâmetros com nuvem de partículas, em geral, não contemplam aspectos multiobjetivo, além de serem formulados para problemas matemáticos, em um espaço infinito. Em nosso subespaço discreto, onde a proximidade geográfica das partículas não sugere nenhuma proximidade de aptidão, a influência dos parâmetros muda completamente.

Neste domínio, cada atributo do problema recebe um número no espaço de busca discreto. Mas, isso não significa que uma solução com distância mínima de outra boa solução, seja uma boa solução também. Como exemplo, vejamos duas regras R1 e R2, sendo:

R1 = [a1,a2,a2,a3,C1] e

R2 = [a2,a2,a2,a3,C1].

Note que as regras se diferenciam apenas no primeiro atributo. Supondo que R1 seja uma excelente regra encontrada, não há nada que assegure que R2 seja uma boa regra também, por sua proximidade com R1. Ou seja, R2 pode ser a pior regra da nuvem. Isso depende estritamente da base de dados utilizada e do problema abordado. Por isso, temos a idéia de que cada base tem uma diferente configuração de parâmetros para conseguir melhores resultados.

Assim, os parâmetros de PSO não exercem influência direta sobre o classificador, uma vez que o melhor desempenho deste classificador fica a cargo das regras que foram escolhidas para integrá-lo. No entanto, esses parâmetros exercem uma influência direta na exploração do espaço pela nuvem de partículas e, por conseguinte, na obtenção da fronteira.

Um experimento foi realizado com o nosso algoritmo, com as dez bases de dados da Tabela 5.1. Foram utilizadas 100 partículas e 50 gerações, com os parâmetros $\phi_1 \, e \, \phi_2 \in \{0{,}5;\ 1;\ 1{,}5;\ 2;\ aleatório[0..2]\}$. O valor de ω foi mantido fixo em 1 nesse experimento. Em todas as otimizações, com exceção da aleatória, $\phi_1 = \phi_2$. Na configuração de parâmetros aleatória, os valores de $\phi_1 \, e \, \phi_2$ são escolhidos separadamente para cada partícula a cada nova atualização da posição. As otimizações com todas as bases foram executadas 30 vezes.

A Tabela 5.1.1 apresenta a fronteira de limiar zero das melhores regras obtida pelo algoritmo para cada classe das bases. Esse desempenho foi calculado com base na *métrica S* [60] (ver Apêndice B). Essa métrica calcula o desempenho da fronteira com base na área formada acima da curva dos pontos não dominados. É uma métrica não cardinal, pois não é diretamente influenciada pelo número de soluções não-dominadas.

Os objetivos do algoritmo foram a confiabilidade positiva e a confiabilidade negativa (com correção de erro de Laplace). Analisando a tabela, é possível perceber que não houve diferença estatística entre os parâmetros na obtenção da fronteira com os parâmetros de 50 gerações e 100 partículas. O mesmo pode ser dito, com relação ao número de regras obtidas nas fronteiras, apresentado na Tabela 5.1.2.

Outro experimento realizado foi a verificação da influência do parâmetro da inércia ω na obtenção da fronteira de cada classe das bases da Tabela 5.1. O experimento foi executado trinta vezes, com os valores de $\omega \in \{0{,}5; 1{,}5; 1; 2; [0..1]\}$, usando 100 partículas em 50 gerações. Os parâmetros $\phi_1 \, e \, \phi_2$ foram escolhidos de forma aleatória a cada atualização da partícula.

A Tabela 5.1.3 mostra que também não houve diferenças estatísticas de um valor de ω para outro, com relação ao desempenho da fronteira obtida em 50 gerações com 100 partículas. O mesmo pode ser visto com o número de regras das fronteiras apresentados na Tabela 5.1.4.

Tabela 5.1.1 — Fronteiras para as classes das bases. Influência de ϕ_1 e ϕ_2.

	(0,5;0,5)	**(1,5;1,5)**	**(1;1)**	**(2;2)**	**(0..2;0..2)**
Breast C1	77,76 (3,55)	76,51 (4,17)	75,41 (4,59)	77,99 (3,41)	75,94 (4,44)
breast C2	78,48 (2,51)	79,19 (3,01)	83,98 (3,34)	85,40 (1,84)	80,81 (3,87)
bupa C1	45,23 (0,60)	44,44 (1,05)	44,11 (1,02)	44,07 (0,97)	44,51 (1,13)
bupa C2	55,91 (2,23)	58,08 (1,49)	58,26 (1,52)	55,35 (0,86)	58,31 (1,68)
ecoli C1	31,06 (0,00)	31,05 (0,00)	30,06 (3,78)	31,05 (0,00)	31,05 (0,00)
ecoli C2	75,66 (2,12)	74,33 (2,63)	75,45 (2,72)	71,79 (2,59)	74,33 (2,05)
german C1	39,55 (0,60)	38,84 (1,46)	39,01 (1,17)	39,02 (1,62)	39,15 (0,98)
german C2	69,42 (1,92)	67,74 (2,13)	68,44 (1,90)	66,38 (2,58)	67,29 (2,75)
glass C1	10,40 (0,19)	10,40 (0,21)	10,46 (0,16)	10,46 (0,16)	10,47 (0,19)
glass C2	70,01 (2,35)	65,55 (6,20)	67,41 (3,95)	63,43 (5,34)	64,24 (6,29)
haberman C1	60,19 (0,00)	60,19 (0,01)	60,19 (0,01)	59,94 (0,95)	60,19 (0,00)
haberman C2	43,56 (0,00)	43,56 (0,00)	43,56 (0,02)	43,56 (0,00)	43,56 (0,00)
heart C1	72,49 (0,93)	72,39 (1,17)	73,87 (0,48)	73,65 (0,58)	73,80 (0,50)
heart C2	68,43 (0,39)	67,82 (0,68)	68,49 (0,67)	68,52 (0,40)	68,50 (0,57)
ionosphere C1	69,77 (1,82)	69,58 (1,74)	69,92 (1,91)	69,68 (2,10)	68,57 (1,95)
ionosphere C2	77,52 (12,44)	69,43 (13,95)	66,29 (16,72)	67,91 (14,96)	74,41 (12,94)
new-thyroid C1	29,69 (0,00)	28,82 (0,89)	29,63 (0,32)	29,45 (0,00)	29,69 (0,00)
new-thyroid C2	89,95 (0,00)	89,95 (0,00)	89,69 (1,41)	89,95 (0,00)	89,95 (0,00)
pima C1	66,08 (2,53)	64,75 (3,37)	64,27 (3,68)	64,54 (3,44)	65,78 (2,78)
pima C2	43,05 (0,88)	42,47 (0,90)	42,95 (1,06)	42,89 (0,90)	42,58 (1,06)
Média	**58,71 (20,58)**	**57,76 (20,12)**	**58,07 (20,43)**	**57,75 (20,38)**	**58,16 (20,30)**

Tabela 5.1.2 — Número de regras para as classes das bases. Influência de ϕ_1 e ϕ_2.

	(0,5;0,5)	(1,5;1,5)	(1;1)	(2;2)	(0..2;0..2)
Breast C1	1,83 (0,53)	1,90 (0,61)	2,03 (0,76)	1,90 (0,61)	1,97 (0,67)
breast C2	4,40 (0,67)	5,33 (0,88)	6,80 (0,89)	6,50 (0,90)	5,20 (1,03)
bupa C1	5,43 (0,68)	5,93 (1,93)	6,77 (2,31)	5,73 (1,60)	5,60 (1,77)
bupa C2	8,03 (2,98)	7,20 (1,49)	6,83 (1,39)	8,67 (2,71)	7,40 (1,45)
ecoli C1	10,87 (0,90)	11,47 (0,82)	11,57 (0,77)	11,90 (0,76)	10,80 (1,03)
ecoli C2	8,37 (2,57)	6,37 (2,74)	7,50 (3,01)	8,50 (4,48)	6,67 (2,73)
german C1	6,63 (1,69)	7,33 (2,12)	7,53 (2,06)	5,73 (1,66)	6,77 (1,72)
german C2	14,67 (3,29)	13,57 (2,80)	15,17 (3,67)	13,63 (3,38)	13,40 (2,67)
glass C1	1,63 (0,49)	1,63 (0,49)	1,77 (0,43)	1,77 (0,43)	1,83 (0,38)
glass C2	8,50 (6,55)	8,10 (2,80)	9,90 (3,53)	8,07 (2,07)	9,20 (3,21)
haberman C1	5,97 (0,18)	6,03 (0,18)	6,00 (0,26)	5,70 (1,02)	6,00 (0,00)
haberman C2	2,00 (0,00)	2,00 (0,00)	2,03 (0,18)	2,00 (0,00)	2,00 (0,00)
heart C1	9,97 (1,99)	10,63 (2,24)	15,03 (1,85)	14,10 (1,75)	14,23 (1,91)
heart C2	14,30 (1,64)	11,40 (1,81)	12,23 (1,72)	13,00 (1,62)	12,77 (1,45)
ionosphere C1	2,93 (0,69)	3,10 (1,03)	2,47 (1,04)	2,80 (0,85)	2,87 (1,17)
ionosphere C2	5,63 (1,61)	5,23 (1,25)	5,27 (1,17)	5,30 (1,53)	4,97 (1,25)
new-thyroid C1	2,80 (0,41)	2,97 (0,18)	2,93 (0,25)	3,00 (0,00)	3,00 (0,00)
new-thyroid C2	4,87 (0,43)	4,97 (0,18)	4,90 (0,31)	4,47 (0,57)	4,93 (0,25)
pima C1	12,97 (2,99)	11,00 (2,59)	12,50 (2,99)	12,10 (2,60)	11,43 (2,10)
pima C2	6,17 (0,83)	6,43 (0,86)	6,47 (0,86)	6,20 (1,00)	5,83 (1,12)
média	**6,90 (4,03)**	**6,63 (3,51)**	**7,29 (4,23)**	**7,05 (4,06)**	**6,85 (3,92)**

Tabela 5.1.3 — Fronteira obtida para as classes das bases. Influência de ω.

	0,5	**1,5**	**1**	**2**	**0..1**
breast C1	76,06 (4,06)	77,31 (3,70)	77,83 (3,37)	76,41 (4,51)	77,09 (3,53)
breast C2	81,08 (3,83)	81,73 (3,82)	81,55 (3,97)	80,50 (3,82)	81,49 (4,03)
Bupa C1	44,74 (0,99)	44,68 (1,05)	44,60 (1,01)	44,48 (1,00)	44,84 (0,91)
bupa C2	60,38 (1,37)	58,45 (1,59)	57,64 (1,08)	58,12 (1,38)	60,79 (1,05)
ecoli C1	30,56 (2,67)	30,56 (2,67)	31,05 (0,00)	31,05 (0,00)	31,05 (0,00)
ecoli C2	75,71 (2,01)	75,99 (2,03)	74,86 (2,36)	75,66 (2,32)	76,01 (2,24)
german C1	39,67 (0,85)	39,95 (0,56)	39,39 (0,74)	39,34 (1,14)	40,16 (0,51)
german C2	67,95 (1,84)	68,15 (2,47)	67,51 (2,21)	68,21 (2,93)	68,70 (2,38)
glass C1	10,46 (0,16)	10,47 (0,16)	10,45 (0,17)	10,50 (0,13)	10,51 (0,11)
glass C2	69,14 (3,31)	68,51 (3,53)	66,33 (3,89)	67,14 (3,91)	70,24 (2,31)
haberman C1	60,19 (0,03)	60,19 (0,03)	60,19 (0,03)	60,19 (0,03)	60,19 (0,04)
haberman C2	43,56 (0,03)	43,56 (0,03)	43,56 (0,03)	43,56 (0,03)	43,56 (0,03)
heart C1	73,78 (0,44)	73,68 (0,61)	73,67 (0,59)	73,87 (0,31)	73,38 (0,90)
heart C2	68,37 (0,38)	68,39 (0,41)	68,58 (0,49)	68,38 (0,53)	68,49 (0,26)
ionosphere C1	69,17 (1,92)	69,66 (1,77)	69,32 (1,49)	69,35 (2,34)	69,36 (1,81)
ionosphere	72,91 (14,94)	70,94 (13,70)	71,04 (14,53)	71,54 (14,98)	69,76 (14,06)
new-thyroid C1	29,51 (0,52)	29,63 (0,31)	29,63 (0,31)	29,57 (0,44)	29,16 (0,80)
new-thyroid C2	89,95 (0,08)	89,95 (0,08)	89,95 (0,08)	89,95 (0,08)	89,95 (0,08)
pima C1	66,08 (2,24)	65,20 (3,14)	64,23 (4,08)	65,23 (3,21)	66,13 (2,50)
pima C2	43,29 (0,60)	42,89 (1,21)	42,84 (0,85)	42,58 (1,04)	43,20 (0,94)
	58,63 (20,45)	**58,49 (20,47)**	**58,21 (20,35)**	**58,28 (20,34)**	**58,70 (20,44)**

Tabela 5.1.4 — Num. de regras da fronteira para as classes das bases. Influência de ω.

	0,5	**1,5**	**1**	**2**	**0..1**
breast C1	1,83 (0,69)	1,90 (0,65)	1,93 (0,57)	2,03 (0,75)	1,90 (0,65)
breast C1	4,70 (0,74)	5,50 (0,92)	4,80 (0,79)	5,17 (0,93)	5,30 (1,00)
bupa C1	5,70 (1,53)	6,43 (2,19)	5,70 (2,70)	5,57 (1,28)	5,77 (1,36)
bupa C1	9,93 (1,46)	8,20 (1,89)	7,53 (1,31)	8,20 (1,35)	10,53 (1,12)
ecoli C1	10,97 (0,87)	11,33 (0,83)	11,23 (0,88)	11,20 (0,91)	11,43 (0,80)
ecoli C1	6,70 (2,05)	8,53 (3,45)	6,20 (2,26)	8,23 (3,68)	7,47 (2,46)
german C1	8,73 (2,43)	7,60 (1,82)	7,73 (1,86)	7,53 (2,54)	9,30 (3,06)
german C1	15,40 (3,87)	14,53 (3,16)	13,90 (2,71)	14,40 (3,30)	16,77 (3,45)
glass C1	1,77 (0,42)	1,80 (0,40)	1,73 (0,44)	1,87 (0,34)	1,90 (0,30)
glass C1	11,53 (5,17)	10,27 (3,60)	8,10 (2,99)	8,87 (3,20)	10,90 (2,44)
haberman C1	6,00 (0,00)	6,00 (0,00)	6,00 (0,00)	6,00 (0,00)	6,00 (0,00)
haberman C1	2,00 (0,00)	2,00 (0,00)	2,00 (0,00)	2,00 (0,00)	2,00 (0,00)
heart C1	14,13 (2,16)	14,70 (2,10)	14,33 (1,97)	14,50 (2,00)	13,03 (2,30)
heart C1	13,30 (1,53)	14,43 (1,58)	13,17 (1,81)	12,93 (1,57)	13,37 (1,78)
ionosphere C1	3,30 (0,97)	3,13 (1,12)	2,70 (0,86)	2,77 (1,05)	3,20 (1,11)
ionosphere C1	5,37 (1,35)	4,87 (1,23)	5,37 (1,58)	5,20 (1,33)	5,50 (1,31)
new-thyroid C1	3,00 (0,00)	3,00 (0,00)	3,00 (0,00)	3,00 (0,00)	3,00 (0,00)
new-thyroid C1	4,83 (0,64)	4,93 (0,36)	4,90 (0,40)	4,90 (0,30)	5,00 (0,00)
pima C1	12,10 (2,20)	12,47 (3,30)	11,53 (2,39)	12,13 (3,24)	12,67 (2,41)
pima C1	6,20 (1,14)	6,07 (0,63)	6,03 (1,14)	5,90 (0,87)	6,33 (1,07)
	7,37 (4,33)	**7,38 (4,32)**	**6,89 (4,03)**	**7,12 (4,12)**	**7,57 (4,42)**

No entanto, isso não quer dizer que não haja diferença em se utilizar parâmetros diferentes. É evidente que o objetivo maior se concentra na obtenção da fronteira, porém, há diferenças no que diz respeito a regiões do espaço exploradas ou à velocidade com que a fronteira é encontrada. Certas configurações de parâmetros têm uma rápida convergência, pois buscam diretamente a fronteira. Esses casos, possivelmente, apresentarão média de diversidade menor e, talvez, uma menor média de regras encontradas a cada geração. Do contrário, parâmetros que procuram atingir a fronteira com maior exploração do espaço possuem uma convergência mais lenta e, possivelmente, maior diversidade e média de regras encontradas por geração.

Como $\phi_1 = \phi_2$, com exceção do caso aleatório, o peso do melhor global sobre a regra é o mesmo peso do melhor da partícula, conforme visto na seção 3.3.1.1. Assim, possivelmente, todas as configurações testadas seguem curvas parecidas. Acreditamos que $\phi_1 = \phi_2$ seja preferível em casos de parâmetros pré-fixados, pois se $\phi_1 > \phi_2$, a partícula dá maior preferência ao seu melhor,

concentrando-se em máximos locais, e se $\phi_1 < \phi_2$, a partícula dá maior preferência aos melhores globais, tendendo o algoritmo a uma possível rápida convergência, que, de certa forma, nem sempre é desejada. Dessa forma, uma configuração aleatória pode priorizar em algum momento certo parâmetro e um parâmetro diferente em outro momento. Essa variação pode ser benéfica à otimização. Pode-se ainda variar essa influência de forma contínua, aumentando ou diminuindo parâmetros com o passar das gerações. Essa forma de variação não é tratada nesse trabalho.

Os gráficos da Figura 5.1 mostram a evolução da fronteira para as duas classes das bases *breast* e *heart*. É possível observar como cada parâmetro se saiu diferente com as classes de cada base.

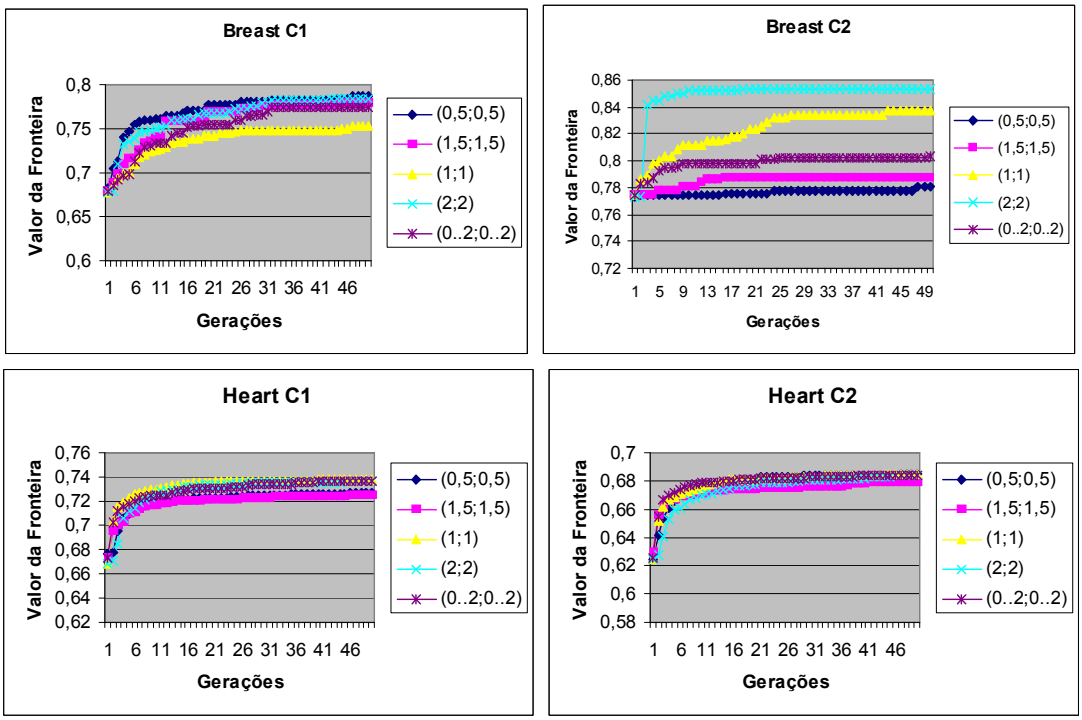

Figura 5.1 — Evolução das fronteiras para cada parâmetro nas bases *breast* e *heart*.

O papel do algoritmo gerador de regras é explorar o espaço de busca, a procura das melhores regras. Se o algoritmo não encontrar nenhuma boa regra, o classificador terá um desempenho muito ruim. Por isso, sem uma boa exploração do espaço e sem uma boa diversidade da nuvem, não é fácil obter bons

resultados de AUC, pois regras boas podem não ser encontradas pela nuvem de partículas.

Essa exploração do espaço de busca está intimamente ligada ao número de novas regras encontradas. Quanto maior esse número, maior a taxa de exploração da nuvem. Quando esse número de novas regras pára de crescer, podemos dizer que o algoritmo convergiu a uma solução (não necessariamente de forma espacial).

Diz-se então que uma dada execução explorou o espaço mais rapidamente que outra, se encontrou mais regras durante um mesmo número de gerações. Assim, à medida que o espaço explorado aumenta, a nuvem se aproxima da convergência. Essa afirmação se justifica, pois o número de regras possíveis de serem encontradas é limitado e quando se encontra esse número, não há mais a necessidade de continuar a otimização, pois nenhuma melhora é possível. Portanto, acreditamos que a melhor configuração de parâmetros para a exploração do espaço é aquela que consegue encontrar o maior número de regras em menor número de gerações. A diversidade tem sua importância nesse caso, pois com baixa diversidade, o algoritmo precisaria de maior número de gerações para atingir um mesmo número de regras encontradas.

5.1.1 Exploração do Espaço

Para termos noção da influência de $\phi_1 e \phi_2$ na exploração do espaço, apresentamos os valores da média de descoberta de novas regras a cada geração para o experimento anterior. O gráfico da Figura 5.2 mostra como os parâmetros se comportaram para a base *breast*. Nesse caso, uma configuração aleatória de 0 a 2 (a cada atualização da partícula) trouxe uma maior descoberta de novas regras. No gráfico, fica clara a constância da descoberta nessas cinqüenta gerações. Ao que parece, o algoritmo, com esses parâmetros, descobriria muitas outras regras se continuasse sua execução após a geração 50, diminuindo gradativamente seu potencial descobridor até convergir a um número máximo de regras.

Um exemplo da diferente configuração dos parâmetros para cada base é apresentado na Figura 5.3. Esse gráfico mostra o resultado do mesmo experimento da Figura 5.2, realizado agora com a base *heart*. Note no gráfico que, nesse experimento, a configuração de parâmetros que descobriu mais regras em 50 gerações foi a de $\phi_1 = 1 \, e \, \phi_2 = 1$.

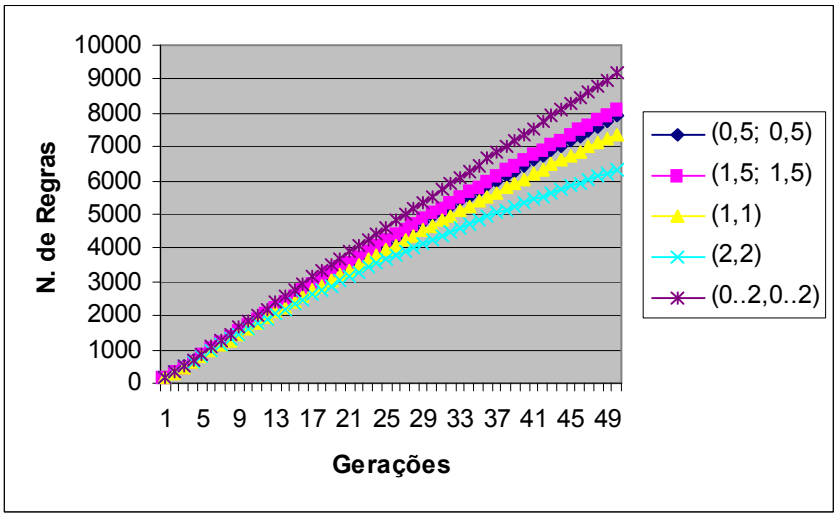

Figura 5.2 – Média do número de regras encontradas para base *breast* com relação a cada um dos valores de $\phi_1 \, e \, \phi_2$.

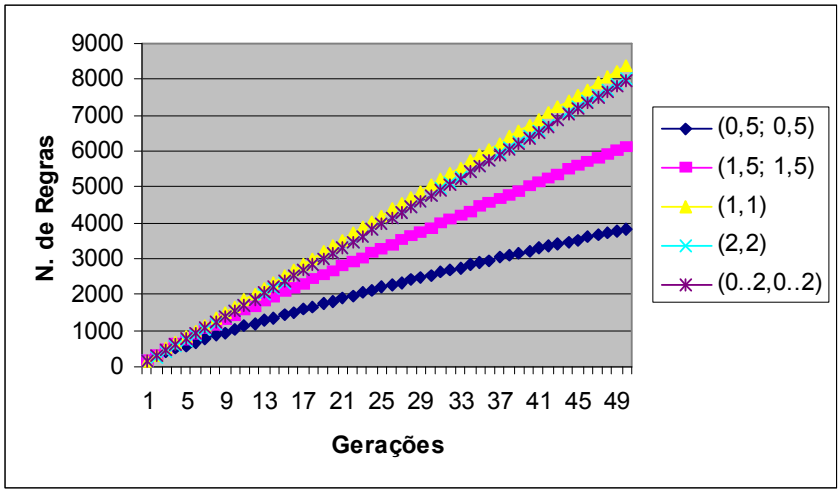

Figura 5.3 - Média do número de regras encontradas para base *heart* com relação a cada um dos valores de $\phi_1 \, e \, \phi_2$.

Já o gráfico da Figura 5.4 mostra como as execuções convergiram quando o número de regras se aproximou do número máximo para a base *haberman*

(1000). Note que, já na geração 20, há uma brusca diminuição do número de regras encontradas a cada geração, pois esse número, no caso dos parâmetros aleatórios, por exemplo, já excede a casa dos 900, se aproximando do total de 1000 regras.

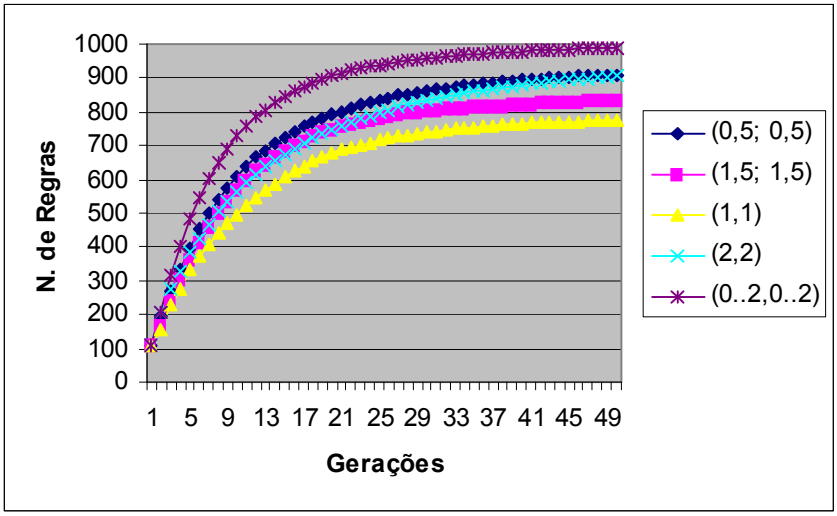

Figura 5.4 - Média do número de regras encontradas para base *haberman* com relação a cada um dos valores de ϕ_1 e ϕ_2.

A Tabela 5.1.5 apresenta a média de novas regras encontradas a cada geração para todas as bases da Tabela 5.1. O valor entre parênteses junto da média representa o desvio padrão das 30 execuções. O valor na coluna DV representa o desvio padrão entre a média de regras descobertas de cada geração. Portanto, um alto valor em DV mostra uma grande diferença entre as médias de uma geração para outra.

Tabela 5.1.5 — Número médio de descoberta de regras. Influência de ϕ_1 e ϕ_2.

Bases	(0,5;0,5)		(1,5;1,5)		(1,1)		(2,2)		(0..2,0..2)	
	MEDIA	**DV**	**MEDIA**	**DV**	**MEDIA**	**DV**	**MEDIA**	**DV**	**MEDIA**	**DV**
1	158,08 (13,68)	9,12	162,10 (11,87)	9,09	147,50 (10,53)	12,90	126,57 (12,65)	23,54	183,46 (5,54)	5,73
2	67,52 (12,28)	17,27	132,92 (12,68)	12,08	119,46 (15,68)	12,59	108,17 (13,96)	10,75	140,15 (13,87)	9,91
3	120,47 (11,05)	13,29	128,91 (15,54)	16,99	133,84 (13,15)	16,58	120,59 (14,46)	18,43	157,68 (10,01)	11,60
4	34,00 (19,80)	10,25	188,51 (3,06)	3,64	189,09 (3,56)	3,09	178,87 (7,67)	5,67	191,89 (2,16)	9,05
5	81,54 (15,42)	22,34	129,58 (7,13)	15,61	131,64 (11,31)	14,61	130,55 (8,79)	12,85	155,83 (6,70)	7,96
6	18,21 (0,71)	24,77	16,72 (0,82)	23,43	15,52 (1,55)	21,33	18,13 (0,51)	23,44	19,78 (0,12)	30,03
7	77,01 (14,10)	18,40	122,54 (14,50)	12,61	167,93 (6,27)	2,15	160,74 (7,18)	9,93	159,56 (9,14)	5,19
8	171,44 (16,29)	8,09	190,45 (7,42)	2,82	182,66 (10,50)	5,73	175,62 (17,18)	6,23	193,52 (2,65)	4,31
9	20,76 (4,35)	21,00	54,16 (8,56)	23,27	53,21 (10,18)	23,85	39,12 (6,51)	28,10	72,57 (7,11)	28,16
10	165,70 (7,92)	3,32	184,17 (4,86)	6,15	181,23 (6,83)	6,35	177,50 (5,81)	5,01	184,26 (3,72)	7,62
Média	**91,47**		**131,01**		**132,21**		**123,58**		**145,87**	

Analisando a Tabela 5.1.5, nota-se que para cada base de dados, uma configuração diferente de parâmetros obtém o melhor resultado, embora se sobressaia a configuração aleatória. Pelo modo de representação circular do espaço, no qual a partícula que excede os limites superiores de um atributo recomeça pelos limites inferiores na mesma proporção do excedido, a configuração aleatória evita que a velocidade da partícula caia em um processo repetitivo, encontrando as mesmas partículas de sempre. Além disso, a desigualdade dos parâmetros faz com que as partículas ora procurem seguir os líderes globais, ora procurem seguir seus máximos. A configuração baixa de $\phi_1 = 0{,}5 \; e \; \phi_2 = 0{,}5$ foi a pior, mostrando não ser boa para a exploração do espaço. Obviamente, a configuração dos parâmetros $\phi_1 = 0 \; e \; \phi_2 = 0$ manteria a nuvem constante por todas as gerações, visto que a velocidade inicial é nula e as partículas nunca se movimentam. Isso demonstra uma forte influência desses dois parâmetros no bom desempenho do algoritmo.

Nesta tabela, é possível notar também que nas bases menores, com poucas regras possíveis, o desvio padrão DV é bastante alto em relação à média, descrevendo, possivelmente, como é o caso da base 6 (*haberman)*, um processo de convergência, onde o número de regras encontradas a cada geração é drasticamente diminuído.

Outro aspecto a considerar ao analisar os parâmetros da otimização, é a sua influência na diversidade da nuvem. Para isso, foi realizado um experimento com as dez bases da Tabela 5.1, recolhendo as informações de diversidade na nuvem. Nossa métrica da diversidade é apenas a porcentagem de partículas distintas na nuvem, ou seja:

$$D = \frac{N_D}{N}, \qquad (5.1.1)$$

onde N_D é o número de partículas distintas da nuvem numa dada geração e N é o número total de partículas da nuvem. Assim, se todas as partículas da nuvem forem diferentes umas das outras, a diversidade dessa nuvem será 1. Evidentemente, quanto maior o número de regras possíveis para uma base de dados, maiores as chances de a diversidade ser grande, pois há muitas regras a representar.

O experimento realizado refletiu essa idéia. O gráfico da Figura 5.5 mostra o cálculo da diversidade a cada geração para a base *breast*. É importante notar que com a configuração aleatória, nesse caso, as regras se mantiveram diferentes umas das outras. Isso não quer dizer que o resultado de AUC seria melhor ou que a fronteira seria obtida primeiro, pois as regras podem ser diferentes, mas não se modificarem.

Com a base *heart*, o resultado não foi muito diferente. Com essa base, algumas configurações obtiveram nuvens iniciais com diversidade bem baixa, que foi crescendo com o passar das gerações, conforme mostra o gráfico da Figura 5.6.

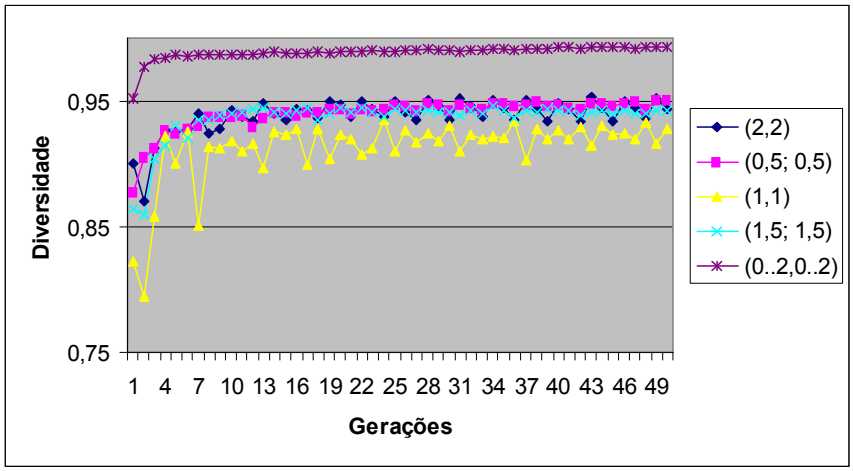

Figura 5.5 - Média da diversidade da nuvem para base *breast* com relação a cada um dos valores de ϕ_1 *e* ϕ_2.

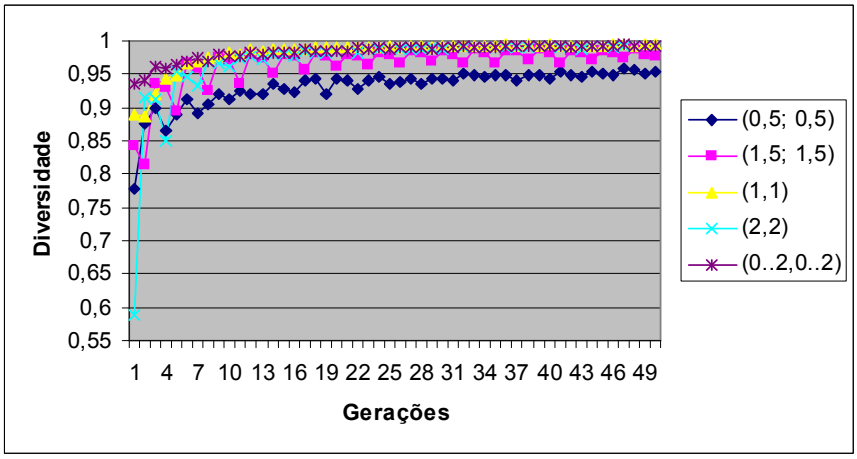

Figura 5.6 - Média da diversidade da nuvem para base *heart* com relação a cada um dos valores de ϕ_1 *e* ϕ_2.

Um aspecto interessante, observado com a base *haberman*, é o desempenho da configuração de parâmetros $\omega = 1, \phi_1 = 1 \, e \, \phi_2 = 1$. Note que, no gráfico da Figura 5.4, essa configuração obtém o pior desempenho de descoberta de regras. Isso pode ter sido gerado, de forma especulativa, por uma constância na fórmula da velocidade e posição, que faz com que as partículas passem pelas mesmas posições de sempre. Isso talvez seja a causa do efeito observado com a diversidade da nuvem para essa configuração, conforme mostra a Figura 5.7. Isso pode ser evitado com uma configuração de parâmetros aleatória.

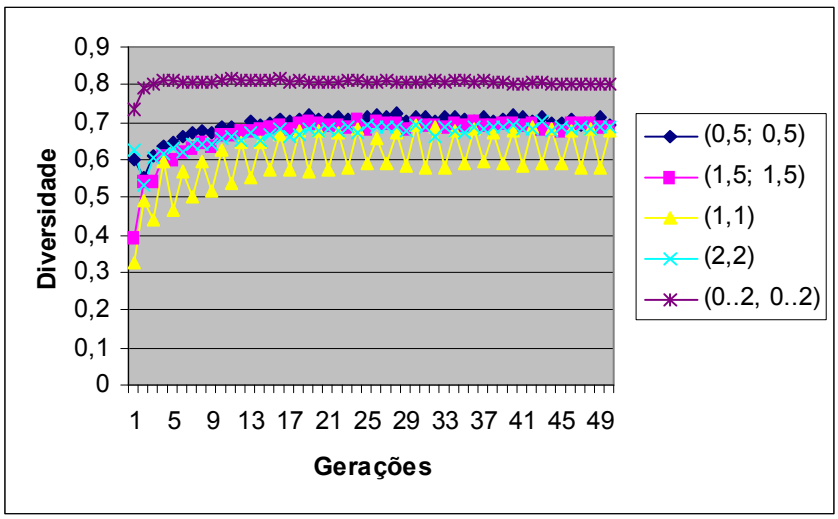

Figura 5.7 - Média da diversidade da nuvem para base *haberman* com relação a cada um dos valores de ϕ_1 e ϕ_2.

Como já foi dito, uma boa taxa de diversidade não significa que o algoritmo terá uma AUC maior, mas apenas possibilita que a exploração do espaço seja maior, pois para obter um maior número de regras encontradas, é preciso ter uma boa taxa de diversidade.

O mesmo pode ser dito com relação ao número de regras distintas descobertas. Quanto maior o número de regras descobertas, maior a exploração do espaço feita pela nuvem. Mas, isso não quer necessariamente dizer que a AUC do classificador será boa, ou que a fronteira obtida será mais próxima à fronteira de Pareto, pois pode haver áreas do espaço importantes que não tenham sido exploradas. Porém, como medida genérica auxiliar para avaliar o desempenho do algoritmo, a métrica é válida, pois a obtenção de bons resultados com esse algoritmo está, geralmente, vinculada com a idéia de maior exploração do espaço e maior diversidade da nuvem.

A Tabela 5.1.6 mostra a média e o desvio padrão dos resultados dos experimentos com diversidade da nuvem. Nesse caso, vê-se a superioridade da configuração de parâmetros aleatória de 0 a 2, que obteve o maior valor médio de diversidade com todas as bases do experimento. Pode-se notar também o baixíssimo desvio padrão DV com essa configuração. Isso mostra que a diversidade se manteve constante do início ao fim da otimização.

A baixa diversidade das otimizações com a base *haberman* pode ser explicada se observarmos que o número de regras possíveis com essa base é pequeno. Efeito contrário a esse pode ser observado nas bases *pima*, *german* e *ionosphere*, onde o algoritmo apresentou uma diversidade bem próxima a 1, com desvio padrão DV baixo. Isso porque o universo de regras possíveis é tão grande que a probabilidade das regras repetirem é baixa. Além disso, pode-se ver também a inferioridade da configuração de parâmetros $\phi_1 = 0{,}5 \; e \; \phi_2 = 0{,}5$ em relação às demais, devido, possivelmente, à pouca interação e mobilidade das partículas.

Tabela 5.1.6 — Porcentagem da diversidade da nuvem. Influência de $\phi_1 \; e \; \phi_2$.

	(0,5;0,5)		(1,5;1,5)		(1,1)		(2,2)		(0..2,0..2)	
bases	**Média**	**DV**	**Média**	**DV**	**Média**	**DV**	**Média**	**DV**	**Média**	**DV**
breast	93,99 (2,97)	1,29	93,60 (2,64)	1,70	91,23 (3,09)	2,67	93,82 (2,76)	1,41	98,87 (0,64)	0,01
bupa	84,07 (3,59)	3,85	95,71 (1,38)	4,93	93,36 (2,16)	5,58	93,57 (2,22)	5,60	97,29 (1,17)	0,02
ecoli	92,98 (2,06)	2,72	94,30 (2,54)	2,85	93,45 (2,41)	3,02	93,67 (3,03)	3,41	98,80 (0,56)	0,01
german	98,35 (0,80)	1,30	99,40 (0,34)	1,44	99,37 (0,30)	1,17	98,97 (0,33)	2,79	99,75 (0,15)	0,01
glass	87,59 (3,99)	2,28	90,32 (3,41)	2,09	89,66 (2,87)	2,35	90,09 (2,65)	2,15	97,56 (0,86)	0,01
haberman	69,35 (4,05)	3,31	67,15 (4,00)	5,43	60,38 (5,98)	7,48	66,82 (2,72)	2,99	80,60 (2,01)	0,01
heart	93,02 (2,05)	3,05	96,45 (1,06)	3,36	98,20 (0,65)	2,40	96,94 (0,84)	6,08	98,34 (0,63)	0,01
ionosphere	99,65 (0,38)	0,10	99,67 (0,32)	0,27	98,61 (1,05)	1,33	99,80 (0,22)	0,19	99,87 (0,21)	0,00
new-thyroid	56,40 (4,63)	0,83	65,03 (5,33)	2,47	62,42 (5,66)	2,21	65,06 (5,10)	2,05	90,85 (1,44)	0,03
pima	98,40 (0,67)	2,12	98,88 (0,60)	2,04	98,85 (0,60)	2,65	98,77 (0,51)	2,26	99,20 (0,51)	0,01
Média Total	**87,38**		**90,05**		**88,55**		**89,75**		**96,11**	

Outro experimento realizado foi a verificação da influência do parâmetro da inércia ω na exploração do espaço e na diversidade. A Tabela 5.1.7 mostra o resultado da análise de descoberta de regras sob a influência do parâmetro ω. O experimento foi executado trinta vezes, com os valores de $\omega \in \{0{,}5; 1{,}5; 1; 2; [0..1]\}$, usando 100 partículas em 50 gerações. Os parâmetros $\phi_1 \; e \; \phi_2$ foram escolhidos de forma aleatória a cada atualização da partícula.

Tabela 5.1.7 — Análise de descoberta de regras. Influência de ω.

Bases	0,5		1,5		1		2		0..2	
	MEDIA	**DV**	**MEDIA**	**DV**	**MEDIA**	**DV**	**MEDIA**	**DV**	**MEDIA**	**DV**
1	175,21 (9,90)	7,95	175,53 (8,21)	8,39	182,68 (5,24)	5,88	179,80 (5,29)	6,95	155,70 (9,13)	16,62
2	111,90 (13,70)	16,63	126,33 (11,04)	17,33	137,85 (12,95)	9,73	130,55 (13,11)	13,54	84,78 (14,07)	29,00
3	143,16 (9,08)	17,79	138,88 (11,08)	21,76	156,63 (9,26)	11,60	155,39 (8,27)	14,08	117,24 (11,85)	28,59
4	175,15 (5,11)	6,11	185,95 (3,96)	7,64	191,34 (1,94)	8,70	189,82 (2,22)	8,28	147,40 (11,52)	11,58
5	132,82 (8,14)	16,92	145,17 (8,64)	14,38	153,86 (8,77)	9,48	151,75 (9,25)	9,64	103,49 (9,89)	28,14
6	19,18 (0,49)	29,41	19,69 (0,18)	30,08	19,81 (0,11)	30,49	19,69 (0,18)	29,90	17,76 (0,48)	27,64
7	121,45 (11,11)	15,05	119,87 (10,26)	20,00	158,96 (6,79)	4,86	153,62 (7,52)	5,05	86,65 (12,01)	28,62
8	188,30 (8,59)	4,47	191,28 (4,98)	4,85	194,44 (1,54)	4,24	191,58 (3,60)	4,35	179,47 (9,81)	6,83
9	58,67 (9,50)	29,57	58,04 (5,94)	33,57	72,48 (6,83)	28,07	61,49 (7,18)	33,42	40,19 (4,56)	32,20
10	175,00 (5,06)	5,97	178,91 (6,43)	6,44	182,94 (5,57)	7,23	182,81 (6,05)	7,30	158,40 (8,64)	8,37
Média	**130,08**		**133,97**		**145,10**		**141,65**		**109,11**	

É possível observar, analisando os resultados, que mantendo a inércia constante em um, obtemos melhores médias de regras encontradas, e que, nesse caso, uma configuração aleatória perde em todas as bases analisadas. Com a inércia constante, a velocidade anterior da partícula é sempre considerada para a próxima posição.

Na Tabela 5.1.8, são apresentados os resultados da influência da inércia sobre a diversidade da nuvem. Nota-se, novamente, o melhor desempenho da configuração $\omega = 1$ com relação à diversidade da nuvem.

5.1.2 Considerações do experimento

Nesta seção, pudemos ver como os parâmetros da velocidade influenciam na exploração do espaço e, por conseguinte, na geração das regras de classificação.

Tabela 5.1.8 — Análise da diversidade da nuvem. Influência de ω.

bases	0,5		1,5		1		2		0..2	
	Média	**DV**	**Média**	**DV**	**Média**	**DV**	**Média**	**DV**	**Média**	**DV**
breast	97,91 (1,38)	0,55	98,04 (1,17)	0,46	98,71 (0,85)	0,52	98,58 (0,61)	0,44	92,44 (3,40)	3,57
bupa	89,41 (2,91)	1,57	95,25 (1,48)	1,94	97,03 (1,04)	2,29	95,39 (1,44)	1,96	73,15 (6,44)	10,35
ecoli	97,35 (1,12)	0,62	97,32 (1,55)	0,82	98,82 (0,54)	0,85	98,75 (0,50)	0,82	88,91 (4,66)	4,46
german	96,67 (1,65)	0,19	99,48 (0,32)	0,34	99,76 (0,11)	0,48	99,62 (0,24)	0,43	88,48 (4,47)	3,82
glass	90,93 (2,61)	2,76	94,91 (2,09)	1,73	97,57 (1,02)	1,23	97,10 (1,46)	1,11	76,76 (5,25)	10,55
haberman	76,07 (2,73)	2,41	79,19 (2,75)	1,89	81,36 (2,64)	1,28	79,62 (3,01)	1,56	60,84 (3,16)	8,13
heart	84,90 (4,28)	4,79	85,88 (3,54)	6,37	98,17 (0,54)	1,29	98,17 (0,69)	1,23	66,59 (6,67)	13,53
ionosphere	99,19 (1,10)	0,27	99,64 (0,52)	0,16	99,92 (0,13)	0,07	99,77 (0,23)	0,10	96,55 (2,29)	1,66
new-thyroid	77,52 (4,62)	6,14	82,17 (2,06)	5,78	90,79 (1,56)	2,98	83,02 (2,66)	5,30	56,34 (3,38)	15,37
pima	97,82 (1,26)	0,64	98,97 (0,64)	0,75	99,16 (0,66)	0,85	99,21 (0,57)	0,90	93,69 (2,56)	1,74
Média Total	**90,78**		**93,09**		**96,13**		**94,92**		**79,38**	

Nota-se que existe a influência dos parâmetros da velocidade das partículas na obtenção da fronteira, exploração e diversidade no espaço. No entanto, essa influência depende da base de dados utilizada. Dessa forma, não se tem uma conclusão definitiva a respeito da melhor configuração de parâmetros para o aprendizado de regras. No entanto, com relação à exploração e diversidade $\omega = 1$ e $\phi_1 \; e \; \phi_2$ aleatórios de 0 a 2, foram os melhores parâmetros.

Na próxima seção, vemos a influência do número de partículas e do número de gerações na exploração do espaço.

5.2 EXPERIMENTO COM NÚMERO DE PARTÍCULAS E GERAÇÕES

Outros parâmetros que influenciam as medidas de fronteira, de diversidade e número de regras encontradas por geração são o número de partículas e o número de gerações da otimização. A Tabela 5.2.1 mostra os resultados da fronteira obtidos para 10 partículas com 10, 50 e 100 gerações e com 50 e 100

partículas em 10 gerações. Os resultados mostram o comportamento com os parâmetros $\phi_1 \, e \, \phi_2 \in \{0{,}5;\ 1;\ 1{,}5;\ 2;\ aleatório[0..2]\}$ para as bases da Tabela 5.1. Os objetivos do algoritmo foram a confiabilidade positiva (Laplace) e a confiabilidade negativa (Laplace). Os experimentos foram executados trinta vezes e o parâmetro ω foi fixado em 1.

Analisando a tabela, é possível notar que os resultados de fronteira foram muito próximos para todas as configurações de parâmetros de velocidade. No entanto, é interessante observar a relação da fronteira com o crescimento do número de partículas e gerações. Nota-se que um número maior de partículas trouxe maior área e número de regras na fronteira que o mesmo número em gerações. A utilização de 50 partículas em 10 gerações, por exemplo, trouxe melhor fronteira que com 10 partículas em 100 gerações. Além disso, o tempo computacional no primeiro caso é muito menor que no segundo caso. Comparativamente, de forma empírica, o tempo de 100 partículas em 10 gerações foi menor que o tempo de 10 partículas em 100 gerações.

Dessa forma, aumentar o número de partículas é melhor que aumentar o número de gerações. Isso é válido até a obtenção da Fronteira de Pareto. A partir daí, nenhuma melhora é possível. Quanto mais se aproxima da verdadeira fronteira, menos diferença há entre aumentar o número de partículas ou gerações.

Analisemos, agora, a influência desses parâmetros sobre a média do número de regras encontradas por geração. Primeiramente, é natural se pensar que quanto maior o número de partículas, maior será o número de regras encontradas, pois há mais candidatos à solução. Além disso, é esperado que essa média diminua com o aumento do número de gerações, uma vez que, com o passar das gerações, o algoritmo tende a convergir, diminuindo o número de novas regras encontradas. Evidentemente, o número total de regras encontradas é maior a cada geração, porém a média de crescimento diminui.

Tabela 5.2.1 — Análise da fronteira da nuvem variando número de partículas e gerações. Influência de ϕ_1 e ϕ_2 e do número de partículas e gerações.

	(0,5;0,5)	**(1,5;1,5)**	**(1;1)**	**(2;2)**	**(0..2;0..2)**	
	ÁREA DA FRONTEIRA (MÉTRICA S)					**Média**
10 partículas e 10 gerações	48,12 (17,70)	47,71 (18,62)	48,03 (17,58)	47,56 (18,00)	48,51 (18,25)	**47,99**
10 partículas e 50 gerações	50,75 (18,96)	51,24 (18,88)	51,76 (19,23)	51,48 (18,65)	51,90 (18,89)	**51,43**
50 partículas e 10 gerações	55,50 (19,34)	54,23 (19,61)	54,93 (19,77)	54,10 (19,25)	54,81 (19,75)	**54,71**
10 partículas e 100 gerações	53,27 (19,24)	52,31 (18,92)	52,74 (19,37)	52,11 (18,78)	53,14 (18,83)	**52,71**
100 partículas e 10 gerações	57,18 (19,95)	56,52 (20,38)	56,42 (20,31)	55,97 (20,03)	56,79 (19,83)	**56,58**
Média	**52,96**	**52,4**	**52,78**	**52,24**	**53,03**	
	NÚMERO DE REGRAS					**Média**
10 partículas e 10 gerações	3,17 (1,13)	3,17 (1,34)	3,37 (1,30)	3,34 (1,36)	3,18 (1,38)	**3,25**
10 partículas e 50 gerações	4,39 (1,98)	4,07 (1,85)	5,26 (3,47)	4,30 (2,20)	4,14 (1,85)	**4,43**
50 partículas e 10 gerações	4,91 (2,26)	4,66 (2,03)	4,94 (2,39)	4,87 (2,27)	4,71 (2,18)	**4,82**
10 partículas e 100 gerações	4,68 (2,22)	4,47 (1,95)	5,02 (2,62)	4,90 (2,62)	4,73 (2,11)	**4,76**
100 partículas e 10 gerações	5,79 (3,01)	5,32 (2,67)	6,05 (3,16)	5,57 (2,77)	5,76 (3,06)	**5,70**
Média	**4,59**	**4,34**	**4,93**	**4,60**	**4,50**	

Um experimento foi realizado com relação ao número de regras encontradas por geração, para as dez bases de dados da Tabela 5.1. O experimento para cada configuração de parâmetros foi realizado trinta vezes, tomando-se a média para cada geração. Nove configurações de parâmetros foram testadas, combinando 10, 50 e 100 gerações com 10, 50 e 100 partículas. Os valores de w, ϕ_1 e ϕ_2 foram fixados em 1. A Tabela 5.2.2 apresenta as médias gerais para cada configuração de parâmetros. O desvio padrão aparece entre parênteses.

Tabela 5.2.2 — Análise do número de regras encontradas para gerações e partículas

	10 Gerações	**50 Gerações**	**100 Gerações**	**Média**
10 Partículas	15,39 (1,30)	13,17 (2,73)	11,66 (3,47)	**13,41**
50 Partículas	76,9 (17,49)	70,17 (25,13)	63,88 (27,81)	**70,32**
100 Partículas	146,58 (43,43)	132,63 (57,03)	122,33 (61,46)	**133,85**
Média	**79,62**	**71,99**	**65,96**	

Na tabela, nota-se que com o aumento do número de partículas, nos três casos, 10, 50 e 100 gerações, a média de regras encontradas aumentou. Na verdade, esse número de regras encontradas é a soma da quantidade encontrada para cada classe da base. Logo, o número máximo de regras que se pode obter a cada geração é de duas vezes o número de partículas, pois há duas classes em todas as bases.

Além disso, é possível notar também, como havíamos previsto, que com o aumento do número de gerações, a média, nos três casos, diminuiu, pois o algoritmo se aproxima da convergência para uma solução.

O experimento da análise de diversidade foi feito da mesma forma. A Tabela 5.2.3 apresenta o resultado desse experimento. Com o aumento do número de gerações, houve uma tendência ao aumento da diversidade, nas bases testadas. Isso pode ter acontecido devido à proximidade entre as partículas na nuvem inicial, que com o passar das gerações, vão explorando regiões do espaço distintas, aumentando a diversidade. Com poucas gerações, talvez não dê tempo para o algoritmo estabilizar a diversidade da nuvem. Outro aspecto é que com o aumento do número de partículas, a diversidade diminuiu como reflexo de maior chance de conflito entre as partículas, pois nossa medida de diversidade traduz que é mais fácil que uma nuvem contenha 10 partículas diferentes entre si do que 100.

Tabela 5.2.3 — Análise da Diversidade para as gerações e número de partículas

	10 Gerações	**50 Gerações**	**100 Gerações**	**Média**
10 Partículas	94,95 (2,75)	96,67 (2,57)	96,94 (2,60)	**96,19**
50 Partículas	89,15 (11,44)	92,33 (10,27)	92,90 (9,70)	**91,46**
100 Partículas	84,97 (15,57)	88,50 (14,96)	89,54 (14,07)	**87,67**
Média	**89,69**	**92,50**	**93,13**	

O número de partículas e gerações influencia diretamente o processo de geração das regras, porém sua influência no classificador é incerta e indireta. Por isso, a análise destes dois parâmetros, bem como a dos parâmetros da velocidade, foi feita com base nas medidas da fronteira, diversidade e número de regras encontradas, que estão diretamente ligadas à geração das regras. Além disso, estes parâmetros, excetuando-se os atributos inerentes ao tamanho das

bases, são os que estão intimamente ligados ao tempo de execução do algoritmo, conforme observado na Seção 4.9.

Na seção seguinte, é feita uma análise de aspectos que dizem respeito à aproximação da verdadeira fronteira de Pareto.

5.3 APROXIMAÇÃO DA FRONTEIRA DE PARETO

Um aspecto importante para se avaliar o desempenho do algoritmo de nuvem de partículas é comparar suas soluções não-dominadas com a verdadeira fronteira de Pareto. Isso nos permite verificar se o algoritmo multiobjetivo está encontrando soluções que se aproximam da fronteira de Pareto, sendo, portanto, soluções ótimas.

Para fazer essa verificação, foi gerada a verdadeira fronteira para as cinco menores bases de dados da Tabela 5.1, no que diz respeito ao número de regras possíveis: *haberman, new-thyroid, bupa, ecoli* e *pima.* O experimento não foi realizado com as demais bases de dados, pois encontrar a verdadeira fronteira de Pareto nessas bases é impraticável. As funções objetivo utilizadas para a aptidão das regras foram confiabilidade positiva e negativa (com a correção de Laplace). O experimento foi executado trinta vezes com 100 partículas em 50 gerações.

Para que pudéssemos ter a idéia de aproximação das regras obtidas pelo algoritmo com a fronteira de Pareto, calculamos a média e o desvio padrão das fronteiras obtidas pelo algoritmo e comparamos com a verdadeira Fronteira. A métrica utilizada para essa comparação foi a *Métrica S* (ver Apêndice B). A princípio, quanto maior o número de regras possíveis, maior a dificuldade de se obter a fronteira verdadeira na otimização. A Tabela 5.3.1 apresenta o resultado do desempenho das fronteiras.

Tabela 5.3.1 — Análise das Fronteiras de Pareto (%)

	Fronteira Verdadeira		Fronteira Algoritmo	
	Classe1	Classe2	Classe1	Classe2
bupa	45,43	61,20	44,31 (1,06)	58,43 (1,74)
haberman	60,19	43,56	60,19 (0,00)	43,56 (0,00)
new-thyroid	29,69	89,95	29,69 (0,00)	89,95 (0,00)
ecoli	31,06	78,08	31,05 (0,00)	74,31 (2,05)
pima	67,35	43,85	63,25 (4,16)	42,63 (1,12)
	46,74	**63,33**	**45,70**	**61,78**

Analisando a tabela, notamos que o algoritmo se aproxima bastante da fronteira nas cinco bases, igualando seu valor à verdadeira fronteira nas bases menores *haberman* e *new-thyroid*. Note que, à medida que o número de regras possíveis aumenta, maior é a diferença entre as fronteiras. No entanto, com um número maior de gerações ou de partículas, o resultado seria provavelmente melhor. Com relação ao número de regras, o algoritmo, pelo menos aparentemente, obteve um resultado um pouco pior. A Tabela 5.3.2 apresenta a relação do número de regras na fronteira para as duas classes. Note que nas bases maiores o número médio de regras fica bem abaixo, pelo menos para uma das classes. No entanto, como observamos anteriormente, a fronteira obtida se aproxima bem da verdadeira, quando se trata do desempenho global. Isso se deve ao fato de que vários pontos se repetem na fronteira verdadeira. Pontos repetidos não alteram o desempenho da fronteira, mas, obviamente, alteram o número de regras. Como exemplo, citamos a base *pima* que de 28 pontos para a primeira classe, apenas 14 pontos são distintos no espaço objetivo. Mesmo assim, um maior número de gerações e de partículas aumentaria, muito provavelmente, o número de regras.

Tabela 5.3.2 — Análise do número de regras das fronteiras

	Fronteira Verdadeira		Fronteira Algoritmo	
	Classe1	Classe2	Classe1	Classe2
bupa	6	13	5,63 (1,83)	7,4 (1,58)
haberman	6	2	6,00 (0,00)	2,00 (0,00)
new-thyroid	3	5	3,00 (0,00)	4,97 (0,17)
ecoli	12	18	10,87 (1,05)	6,47 (2,25)
pima	28	7	11,60 (2,63)	5,87 (0,88)
	11	**9**	**7,42**	**5,34**

Na seção seguinte, é feita uma análise de aspectos que dizem respeito ao classificador diretamente, pois modificam a forma de seleção das regras e não sua geração.

5.4 EXPERIMENTOS DOS LIMIARES DA FRONTEIRA

A fórmula da velocidade de PSO nos fornece uma boa forma de mutação da solução. Ao contrário da filosofia em que foi baseada, PSO não tende suas partículas a um mesmo ponto. Ela utiliza a posição do melhor global balanceando com a tendência de sua melhor posição. Ou seja, não tende ao melhor global e nem a sua melhor posição, mas procura outra solução entre essas duas. Evidentemente, não haveria razão para todas as partículas encontrarem a mesma solução. Seria um uso desnecessário da nuvem.

A idéia é que toda partícula poderia gerar uma nova regra distinta a cada geração. Dessa forma, para o aprendizado de regras, não só a diversidade da população é importante, para que não haja partículas na mesma posição, mas também a capacidade de exploração do espaço, para que seja possível encontrar novas regras. Porém, apesar da capacidade evolutiva, muitas regras ruins são geradas durante a otimização. Isso faz com que a Inteligência do algoritmo resida na seleção das melhores regras e não na sua geração.

O problema é que para cada base de dados diferente, é exigida uma inteligência de seleção diferente. Essa inteligência pode ser influenciada por vários fatores da base de dados, ainda não delineados. O número de atributos, número de exemplos, número total de regras possíveis e algumas outras estatísticas da base são exemplos de valores que poderiam influenciar na seleção de regras para o classificador.

Foi realizado um experimento que demonstra a alteração dos valores de AUC do classificador quando se altera o procedimento de seleção das regras. As regras são selecionadas baseadas nos conceitos de Pareto, porém, com o limiar E da equação 4.5.1. Assim, quanto maior o valor de desse limiar, maior o número de soluções não-dominadas e, portanto, maior o número de regras possíveis de

serem selecionadas. Com isso, o classificador cresce, trazendo mais regras com poder de voto. É de se imaginar que com maior número de regras, o classificador aumente sua precisão. Mas, isso nem sempre é válido. Chega o momento no qual a inserção de novas regras ao classificador reduz a precisão do mesmo, pois as regras inseridas já não são boas para a classificação. Logo, como nos outros experimentos, é de se esperar que cada base de dados possua seu próprio melhor limiar *E* da fronteira.

Os experimentos utilizaram, para cada base, 30 execuções de 100 partículas em 50 gerações, tomando-se a média, com validação cruzada em 10 partições. Os objetivos foram a confiabilidade positiva e negativa (com correção de Laplace). Os parâmetros $\phi_1 \; e \; \phi_2$ foram aleatórios para cada atualização da velocidade e a inércia foi fixada em $\omega = 1$. A AUC foi calculada com o método de classificação por voto de confiança com voto negativo. A Tabela 5.4.1 apresenta, para cada base, os valores de AUC e número de regras do classificador, com seus respectivos desvios padrões, dentre os valores limiares de 0, 0,01, 0,02, 0,03. Os valores entre parênteses representam o desvio padrão dentre as médias obtidas para as 10 partições. Note que quando $E = 0$, a seleção de regras equivale-se ao conceito de dominância de Pareto da equação 3.2.1. A seleção de regras para o melhor global continuou seguindo o padrão de limiar 0. Portanto, apenas a seleção das regras para o classificador foi alterada. Logo, a diversidade e a convergência da nuvem mantiveram os mesmos padrões apresentados na seção 5.1, alterando somente a classificação.

É possível perceber que o aumento do limiar, obviamente, aumentou consideravelmente o número de regras no classificador. A base *glass*, por exemplo, com o limiar 0, apresentou uma média de 10,49 regras em seu classificador, ao passo que, com o limiar 0,03, apresentou uma média de 267,35 regras. Esse aumento considerável não trouxe grandes ganhos de AUC para o classificador. Isso pode ser explicado pela característica e confiança das melhores regras. Com o aumento de regras de qualidade mediana no classificador, a qualidade da classificação fica prejudicada. Note que com o limiar 0,02, essa base obtém menos regras, porém uma maior média de AUC.

Tabela 5.4.1 — Resultados de AUC e número de regras para cada um dos limiares 0, 0,01, 0,02 e 0,03.

	0		0,01	
Bases	**AUC (%)**	**NUM REGRAS**	**AUC (%)**	**NUM REGRAS**
Breast	97,85 (1,22)	6,69 (0,37)	98,67 (1,09)	24,26 (1,60)
Bupa	70,03 (7,79)	13,36 (1,04)	69,18 (7,58)	19,44 (1,88)
Ecoli	82,58 (10,32)	14,88 (2,40)	90,17 (5,71)	41,73 (2,61)
German	73,06 (5,40)	19,87 (0,64)	73,85 (5,21)	63,67 (9,47)
Glass	72,13 (11,73)	10,49 (0,67)	76,06 (12,69)	50,57 (14,74)
Haberman	66,28 (6,86)	7,60 (2,12)	66,33 (7,75)	34,68 (21,36)
Heart	86,26 (6,69)	26,04 (1,97)	87,09 (6,71)	53,19 (2,76)
Ionosphere	84,11 (5,48)	7,65 (0,38)	85,53 (5,45)	10,98 (0,62)
New-thyroid	90,13 (10,31)	7,94 (0,04)	90,45 (11,97)	16,52 (1,15)
Pima	70,68 (4,94)	17,18 (1,65)	72,00 (3,95)	73,45 (9,50)
Média	**79,31**	**13,17**	**80,93**	**38,85**
	0,02		**0,03**	
Bases	**AUC (%)**	**NUM REGRAS**	**AUC (%)**	**NUM REGRAS**
Breast	98,85 (1,04)	34,74 (1,95)	98,99 (0,96)	45,20 (3,62)
Bupa	68,37 (8,43)	28,54 (4,00)	67,78 (8,90)	43,87 (7,05)
Ecoli	93,08 (5,50)	73,17 (8,70)	93,53 (4,42)	116,02 (47,01)
German	74,23 (5,34)	169,69 (18,36)	74,89 (5,24)	240,24 (14,90)
Glass	76,73 (13,73)	203,66 (47,59)	75,39 (14,96)	267,35 (17,34)
Haberman	64,79 (9,14)	71,57 (5,19)	63,46 (9,42)	142,67 (104,12)
Heart	87,54 (6,74)	75,68 (3,47)	87,66 (6,79)	98,52 (5,17)
Ionosphere	85,70 (5,31)	16,07 (0,76)	86,39 (5,11)	21,58 (0,99)
New-thyroid	92,79 (8,78)	23,57 (1,82)	94,69 (7,01)	29,87 (0,71)
Pima	73,51 (4,27)	204,55 (14,07)	74,10 (4,31)	242,25 (21,95)
Média	**81,56**	**90,12**	**81,69**	**124,76**

Isso pode ser visto com a base *bupa* que obteve seu melhor valor de AUC com o limiar 0, mantendo a média de 13,36 regras em seu classificador. Com a adição de novas regras ao classificador, seu valor de AUC decresceu. Com a base *pima*, o algoritmo obteve significante aumento de AUC, de 0 a 0,03.

Com esse experimento, pudemos observar que o comportamento do algoritmo é diferente para cada tipo de base de dados. Um bom trabalho futuro seria o de tentar associar esse limiar às características de cada base, como por exemplo: número de atributos, variância dos valores, quantidade de valores, número de exemplos, número de regras possíveis, etc. Com essa associação, talvez fosse possível utilizar, automaticamente, o melhor limiar de fronteira na otimização. Além disso, com outras funções-objetivo, ao invés da confiabilidade positiva e negativa, o resultado seria diferente.

5.5 COMPARAÇÃO COM OUTROS ALGORITMOS

Nesta seção, comparamos algumas abordagens de objetivos de nossa proposta com outros algoritmos da literatura. Nesse experimento, os parâmetros da velocidade $\omega, \phi_1 \, e \, \phi_2$ foram escolhidos aleatoriamente a cada atualização da partícula.

Como objetivos, consideramos quatro abordagens:

1. confiabilidade positiva e negativa (com correção de Laplace);
2. sensitividade com confiabilidade positiva (Laplace);
3. sensitividade e especificidade;
4. confiabilidade positiva (Laplace), confiabilidade negativa (Laplace), sensitividade e especificidade.

Com a confiabilidade positiva e negativa, talvez tenhamos, no classificador, regras que cobrem poucos exemplos. No entanto, essas regras têm mais certeza em seu voto positivo ou negativo, visto que o voto correto é maximizado. Com especificidade e sensitividade, temos regras mais abrangentes, porém correndo o risco de obter regras com alto grau de falsos positivos ou negativos. A sensitividade com confiabilidade positiva dá ao classificador grandes chances de acerto positivo, porém ainda assim, os falsos positivos permanecem.

Para o cálculo da AUC, utilizamos o método de votação de confiança com voto negativo (VCVN). Com isso, quando uma regra não cobre um exemplo ela vota, tirando pontos da sua classe. A quantidade de pontos que ela tira é equivalente à sua confiabilidade negativa. Portanto, quando a indução das regras não procurou maximizar a confiabilidade negativa, talvez, o resultado seja indesejado. Por isso, com o par de objetivos de sensitividade e confiabilidade positiva, foi feita votação de confiança sem o voto negativo (VC). Com isso, alguns exemplos podem ficar sem classificação, se nenhuma regra do classificador os cobrir.

As Tabelas 5.5.1, 5.5.2, 5.5.3, 5.5.4 e 5.5.5 mostram os resultados de uma otimização com as 10 bases de dados da Tabela 5.1, executando-se 30 vezes o algoritmo, para, respectivamente, confiabilidade positiva e confiabilidade negativa

(Tabela 5.5.1), sensitividade e confiabilidade positiva (Tabela 5.5.2), sensitividade e especificidade (Tabela 5.5.3), tetraobjetivo com VC (Tabela 5.5.4) e tetraobjetivo com VCVN (Tabela 5.5.5).

Nas tabelas, é possível notar que os casos que utilizaram VCVN com o uso de confiabilidade positiva e confiabilidade negativa como objetivos obtiveram melhores resultados de AUC que as demais abordagens. Além disso, o número de regras do classificador foi menor nos experimentos que procuraram maximizar as confiabilidades positiva e negativa. No entanto, conforme pode ser visto na Tabela 5.5.6, o melhor suporte médio, ficou com as abordagens sensitividade com confiabilidade positiva, e sensitividade com especificidade, pois há uma maior cobertura das regras nesses casos. Por isso, para encontrar regras com alto suporte, a melhor escolha seria a segunda ou terceira abordagem.

Tabela 5.5.1 — Resultados de AUC e número de regras para cada um dos limiares. Confiabilidade positiva x confiabilidade negativa (VCVN)

	AUC	**NUM REGRAS**
breast	98,55 (1,10)	9,41 (0,92)
bupa	71,08 (7,52)	18,44 (1,78)
ecoli	83,59 (10,13)	24,71 (4,68)
german	74,57 (5,81)	44,65 (3,20)
glass	75,25 (13,96)	28,66 (13,33)
haberman	66,28 (6,86)	7,60 (2,12)
heart	86,19 (7,07)	32,47 (2,59)
ionosphere	90,28 (6,06)	12,50 (1,05)
new-thyroid	90,14 (10,31)	8,00 (0,00)
pima	70,26 (5,47)	23,64 (4,78)
	80,62	**21,01**

Tabela 5.5.2 — Resultados de AUC e número de regras para cada um dos limiares.
Sensitividade x confiabilidade positiva (VC)

	AUC	NUM REGRAS
breast	98,69 (1,06)	12,15 (1,09)
bupa	70,77 (7,02)	27,52 (2,61)
ecoli	77,97 (11,58)	25,54 (4,15)
german	74,51 (5,77)	64,22 (3,95)
glass	48,26 (28,75)	33,68 (13,68)
haberman	63,76 (11,63)	13,50 (2,55)
heart	86,92 (6,66)	43,16 (2,81)
ionosphere	90,16 (6,05)	15,01 (0,85)
new-thyroid	89,63 (10,39)	9,00 (0,01)
pima	72,49 (4,27)	32,20 (6,16)
	77,36	**27,60**

Tabela 5.5.3 — Resultados de AUC e número de regras para cada um dos limiares.
Sensitividade x especificidade (VCVN)

	AUC	NUM REGRAS
breast	98,58 (1,14)	11,96 (0,97)
bupa	68,16 (8,49)	39,08 (2,67)
ecoli	87,08 (6,89)	31,00 (5,06)
german	74,75 (5,79)	66,07 (1,61)
glass	60,67 (23,61)	61,91 (20,28)
haberman	61,42 (11,07)	21,51 (3,66)
heart	87,61 (6,37)	46,76 (2,74)
ionosphere	88,17 (6,35)	24,69 (1,61)
new-thyroid	90,00 (10,32)	9,00 (0,01)
pima	70,03 (5,52)	49,20 (5,87)
	78,65	**36,12**

Tabela 5.5.4 — Resultados de AUC e número de regras para cada um dos limiares. Sensitividade x confiabilidade positiva x confiabilidade negativa x especificidade (VC)

	AUC	NUM REGRAS
breast	98,68 (0,92)	9,31 (0,86)
bupa	71,29 (8,06)	18,59 (1,77)
ecoli	81,79 (10,26)	24,67 (4,64)
german	74,28 (5,79)	41,34 (1,91)
glass	65,13 (17,89)	38,59 (18,08)
haberman	66,14 (6,62)	7,60 (2,12)
heart	86,84 (6,96)	32,84 (3,15)
ionosphere	89,51 (6,36)	11,22 (1,04)
new-thyroid	89,63 (10,39)	8,00 (0,00)
pima	71,04 (5,23)	23,07 (4,15)
	79,43	**21,52**

Tabela 5.5.5 — Resultados de AUC e número de regras para cada um dos limiares. Sensitividade x confiabilidade positiva x confiabilidade negativa x especificidade (VCVN)

	AUC	NUM REGRAS
breast	98,56 (1,12)	9,32 (0,84)
bupa	71,04 (7,51)	18,58 (1,74)
ecoli	83,64 (10,27)	25,01 (4,55)
german	74,38 (5,71)	41,40 (2,54)
glass	75,22 (13,34)	39,17 (18,98)
haberman	66,28 (6,86)	7,60 (2,12)
heart	86,65 (7,03)	32,82 (3,12)
ionosphere	89,84 (6,08)	11,29 (1,01)
new-thyroid	90,14 (10,31)	8,00 (0,00)
pima	70,31 (5,45)	23,25 (4,52)
	80,60	**21,64**

Tabela 5.5.6 — Suporte Médio dos classificadores obtidos pelas abordagens de objetivos.

Conf pos. e conf neg.	Conf. pos. e sensitividade	Sensitividade e especificidade	Conf pos, conf neg, sensitividade e especificidade
18,76 (11,67)	**22,49 (10,91)**	**22,45 (10,93)**	**18,47 (11,74)**

Considerando a primeira abordagem (confiabilidade positiva e negativa com correção de Laplace) a que obteve melhores médias de AUC e número de regras dentre as abordagens testadas, utilizamos esta abordagem para comparação com outros algoritmos.

Para comparar nossa técnica com os demais algoritmos de aprendizado de regras da literatura, realizamos um experimento de análise de AUC e de número de regras. Com esses experimentos, comparamos com os resultados de outros

algoritmos obtidos do trabalho de Prati e Flash [18]. Para isso, utilizamos a mesma metodologia e as mesmas 16 bases de dados, que nos foram gentilmente cedidas pelos autores e são apresentadas na Tabela 5.5.7. As comparações entre os algoritmos foram feitas em nível de confiança de 95%. Os parâmetros de quantidade de partículas e número de gerações foram definidos para, respectivamente, 450 e 100. As execuções foram repetidas 100 vezes tomando-se média e desvios padrões, apresentados entre parênteses nas tabelas. As bases *kr-vs-kp, letter-a e satimage,* foram aprendidas com 450 partículas em 50 gerações e foram executadas apenas 30 vezes devido ao tempo computacional. No entanto, a normalidade de seus resultados de AUC foi verificada, com a utilização do programa estatístico *R* [61], com seu teste *shapiro.test*.

Tabela 5.5.7 — Bases de dados da UCI de [18].

índice	Base	Num. atributos	num. exemplos	Classe maj. (%)	Num. regras possíveis
1	breast	10	683	65,00	1.800.000.000
2	bupa	7	345	57,98	77.760
3	ecoli	8	336	89,58	1.458.000
4	german	21	1.000	70,00	39.907.676.160.000
5	glass	10	214	92,07	10.206.000
6	haberman	4	306	73,53	1.000
7	heart	14	270	55,55	192.893.400
8	ionosphere	34	351	64,10	$2{,}60 \times 10^{32}$
9	new-thyroid	6	215	96,06	26.400
10	pima	9	768	65,10	6.667.920
11	flag	29	194	91,24	386.614.654.382.880.000
12	nursery	9	12960	97,45	230.400
13	kr-vs-kp	37	3196	52,22	400.252.360.791.997.656
14	letter-a	17	20000	96,06	42.443.058.438.000.000
15	satimage	37	6435	90,27	$2{,}27 \times 10^{36}$
16	vehicle	19	846	76,48	59.019.018.854.400.000

Os algoritmos utilizados para comparação foram:

- Roccer – É descrito em [18]. Utiliza a curva ROC para selecionar regras de um conjunto gerado pelo Apriori. Com isso, tem-se a certeza de que somente as regras que aumentarão a taxa de verdadeiros positivos ou diminuirão a de falsos positivos, proporcionando um ganho ao classificador, serão inseridas.

- CN2 – É um típico algoritmo do tipo separar-para-conquistar. É descrito em [17]. Ele induz uma lista de decisão usando a medida de entropia como função heurística de busca. Em [14], foi modificado para utilizar a correção de erro de Laplace como heurística.
- CN2-Não Ordenado – Versão não ordenada do CN2, descrita em [14].
- C4.5 – Algoritmo dividir-para-conquistar proposto em [19]. Utiliza o ganho de informação para produzir uma árvore de decisão e um passo de poda baseado na redução do erro.
- C4.5 sem poda de árvore;
- Ripper – aprende uma lista de árvores de decisão [62]. Possui uma heurística baseada em MDL para determinar quantas regras devem ser aprendidas.
- Slipper – melhora o algoritmo Ripper utilizando a técnica de cobertura do conjunto com pesos [63].

A Tabela 5.5.8 mostra os valores de AUC obtidos por nosso algoritmo em contraste com os valores dos outros algoritmos, com relação às bases da Tabela 5.5.7. Números em vermelho significam perdas significantes de nosso algoritmo (MOPSO) contra outro. Números em azul significam ganhos significantes do MOPSO.

É possível notar que nosso algoritmo é competitivo com todos os outros algoritmos apresentados. Ele, no entanto, apresenta resultados piores com as bases maiores: *nursery, kr-vs-kp, letter-a, satimage e vehicle*, que possuem maior número de exemplos. Isso se deve possivelmente à falta de cobertura nessas bases, pois a fronteira encontrada não é suficiente para abarcar todos os exemplos de teste e possivelmente o número de partículas foi insuficiente. Se fosse definido um critério de parada que se relacionasse à cobertura da base, talvez os resultados fossem melhores. Além disso, um número maior de partículas ou de gerações poderia, também, melhorar os resultados. Há também a possibilidade de utilização de outros parâmetros da velocidade, contudo,

acreditamos que o melhor efeito talvez se dê com o aumento do limiar do grupo classificador, o que faz com que o número de regras selecionadas aumente.

Tabela 5.5.8 — Resultados de AUC (%) do nosso algoritmo em relação a outros.

bases	MOPSO	Roccer	C4.5	C4.5 SP
breast	98,56 (1,07)	98,63 (1,88)	97,76 (1,51)	98,39 (1,30)
bupa	71,09 (7,16)	65,30 (7,93)	62,14 (9,91)	57,44 (11,92)
ecoli	83,60 (9,69)	90,31 (11,56)	50,00 (0,00)	90,06 (7,75)
german	74,59 (5,56)	72,08 (6,02)	71,63 (5,89)	67,71 (4,12)
glass	75,11 (13,67)	79,45 (12,98)	50,00 (0,00)	81,50 (12,65)
haberman	66,28 (6,51)	66,41 (11,54)	55,84 (6,14)	64,33 (13,58)
heart	86,21 (6,76)	85,78 (8,43)	84,81 (6,57)	81,11 (7,91)
ionosphere	90,38 (6,67)	94,18 (4,49)	86,09 (9,97)	90,91 (6,03)
new-thyroid	90,14 (9,78)	98,40 (1,70)	87,85 (10,43)	97,50 (3,39)
pima	70,26 (5,24)	70,68 (5,09)	72,07 (4,42)	72,60 (6,50)
flag	71,65 (21,41)	61,83 (24,14)	50,00 (0,00)	68,68 (17,22)
nursery	91,57 (2,04)	97,85 (0,44)	99,42 (0,14)	99,74 (0,13)
kr-vs-kp	95,61 (2,01)	99,35 (0,36)	99,85 (0,20)	99,86 (0,20)
letter-a	88,78 (2,93)	96,08 (0,52)	95,49 (1,96)	99,33 (0,46)
satimage	84,52 (3,40)	89,39 (2,38)	90,15 (1,70)	91,31 (1,32)
vehicle	89,16 (4,73)	96,42 (1,47)	94,76 (3,00)	96,99 (1,44)
Média	**82,97**	**85,13**	**77,98**	**84,84**
bases	**CN2 NO**	**CN2**	**Ripper**	**Slipper**
breast	99,26 (0,81)	99,13 (0,12)	98,72 (1,38)	99,24 (0,57)
bupa	62,74 (8,85)	62,21 (8,11)	69,10 (7,78)	59,84 (6,44)
ecoli	90,17 (6,90)	85,15 (11,38)	61,86 (25,49)	74,78 (15,94)
german	75,25 (5,38)	70,90 (4,70)	64,02 (13,62)	71,32 (6,20)
glass	73,74 (15,40)	79,64 (13,24)	49,75 (0,79)	50,00 (2,36)
haberman	59,83 (9,87)	59,28 (10,13)	57,45 (3,85)	50,40 (11,14)
heart	83,61 (6,89)	82,25 (6,59)	84,89 (7,68)	84,03 (6,36)
ionosphere	96,23 (2,97)	92,18 (7,54)	92,06 (5,94)	93,95 (6,82)
new-thyroid	99,14 (1,19)	98,43 (2,58)	94,95 (9,94)	99,12 (1,25)
pima	70,96 (4,62)	71,97 (5,44)	68,07 (9,46)	70,02 (5,97)
flag	53,22 (24,12)	42,78 (24,43)	45,28 (14,93)	52,35 (7,44)
nursery	100,00 (0,00)	99,99 (0,01)	99,43 (0,26)	94,40 (1,59)
kr-vs-kp	99,85 (0,16)	99,91 (0,17)	99,85 (0,21)	99,91 (0,09)
letter-a	99,34 (0,28)	99,44 (0,63)	97,27 (1,86)	98,82 (0,44)
satimage	91,48 (1,45)	91,48 (0,90)	86,83 (3,94)	89,06 (1,98)
vehicle	97,38 (2,05)	96,49 (2,41)	95,01 (2,22)	93,99 (3,13)
Média	**84,51**	**83,20**	**79,03**	**80,08**

A Tabela 5.5.9 apresenta a comparação de nosso método com os outros algoritmos com relação ao número de regras, ou tamanho, do classificador. O MOPSO com os objetivos da confiabilidade positiva e negativa gerou classificadores de bom tamanho, menores que os valores obtidos pelos outros

algoritmos, com exceção do *Ripper*. Ainda assim, a menor média é da base *haberman*, com *7,60,* o que mostra que o algoritmo não gera classificadores muito pequenos, a ponto de possuírem apenas uma regra, o que, dependendo dos objetivos da classificação, poderia ser muito ruim. De 112 comparações possíveis, o MOPSO ganhou significativamente 65, e obteve maior média em 75 casos. Analisando a tabela em conjunto com a Tabela 5.5.8, é possível notar que nas bases em que ele apresentou baixa AUC em relação aos demais, o tamanho do classificador foi, em geral, bem menor, justificando o resultado.

Por isso, a fim de mostrar que o algoritmo pode obter resultados de AUC para as bases *kr-vs-kp, lettera, nursery, vehicle* e *satimage* comparáveis aos demais, mostramos os resultados de AUC com limiar *E* do classificador 0,01. O algoritmo apresentou a média de 97,45 (0,91) para a base *kr-vs-kp,* 99,14 (0,37) para *nursery,* 96,24 (0,99) para *letter-a,* 90,26 (1,99) para *satimage* e 92,96 (2,96) para *vehicle*. Isso elevou a média geral do algoritmo para 84,62, alcançando os demais algoritmos nestas bases. No entanto, o número de regras no classificador aumentou consideravelmente: 667,67 (76,79) para a base *kr-vs-kp,* 1758,03 (246,98) para *nursery,* 516,56 (159,28) para *let*ter-a, 160,00 (20,11) para *satimage* e 93,46 (9,75) para *vehicle*. Isso elevou a média geral do número de regras para 215,69, levado, principalmente, pela média de *nursery*.

Como uma forma de mostrar que nosso algoritmo gera um conjunto de boa classificação, mas que, ao mesmo tempo, possui regras importantes e que podem ser aproveitadas de forma isolada, a Tabela 5.5.10 mostra a média geral do suporte e da precisão relativa ponderada (WRAcc) de todas as regras obtidas de todas as execuções do algoritmo com o limiar zero. A precisão relativa ponderada mede a importância da regra em termos da diferença entre o número de verdadeiros positivos esperados com os observados, variando de 0 a 0,25 [18]. O suporte varia de 0 a 1 e mede a cobertura relativa de cada regra.

Tabela 5.5.9 — Resultado comparativo do número de regras no classificador.

bases	MOPSO	Roccer	C4.5	C4.5 SP
breast	9,43 (0,95)	48,40 (2,32)	37,80 (12,62)	104,20 (9,58)
bupa	18,48 (1,73)	3,90 (0,99)	15,00 (10,53)	143,30 (13,12)
ecoli	24,50 (4,80)	6,70 (0,82)	0,00 (0,00)	62,00 (10,46)
german	44,30 (4,64)	23,70 (6,75)	78,20 (18,50)	388,60 (19,07)
glass	27,72 (12,89)	2,40 (0,52)	0,00 (0,00)	32,10 (5,99)
haberman	7,60 (2,01)	0,80 (0,42)	5,60 (5,72)	71,20 (9,39)
heart	32,23 (3,22)	68,20 (4,42)	13,20 (4,49)	97,40 (15,04)
ionosphere	12,15 (2,47)	67,10 (5,38)	23,40 (3,98)	128,40 (14,10)
new-thyroid	8,00 (0,00)	8,50 (0,53)	4,00 (0,00)	35,40 (4,20)
pima	23,68 (5,74)	4,00 (0,82)	49,40 (20,27)	347,40 (10,00)
flag	47,17 (18,22)	1,70 (2,26)	0,00 (0,00)	35,70 (7,41)
nursery	13,99 (1,47)	18,90 (1,66)	114,60 (3,10)	227,10 (3,57)
kr-vs-kp	60,25 (24,88)	43,60 (7,97)	29,20 (2,15)	37,40 (3,50)
letter-a	12,15 (1,93)	78,40 (3,06)	183,80 (15,22)	700,40 (31,76)
satimage	18,15 (2,70)	143,90 (51,28)	531,10 (84,63)	1767,2 (111,93)
vehicle	10,81 (1,72)	48,40 (4,48)	89,50 (12,12)	188,90 (9,42)
Média	**23,16**	**35,54**	**73,43**	**272,92**
bases	**CN2NO**	**CN2**	**Ripper**	**Slipper**
breast	32,80 (2,74)	24,30 (2,71)	16,90 (1,10)	36,70 (10,78)
bupa	100,70 (3,77)	83,60 (5,87)	7,50 (1,84)	29,30 (8,67)
ecoli	35,30 (2,83)	33,60 (3,13)	4,90 (4,33)	15,20 (6,73)
german	143,90 (6,98)	106,80 (4,37)	8,90 (3,25)	37,40 (12,42)
glass	21,60 (1,26)	21,90 (3,18)	0,20 (0,42)	2,00 (3,43)
haberman	75,90 (3,90)	57,00 (5,98)	3,50 (0,97)	11,80 (13,21)
heart	42,80 (2,49)	36,10 (1,79)	7,70 (1,25)	27,20 (8,73)
ionosphere	36,90 (2,28)	22,90 (1,29)	19,60 (4,43)	38,80 (10,43)
new-thyroid	18,70 (0,67)	15,30 (0,82)	10,90 (3,21)	20,70 (5,48)
pima	169,70 (10,08)	168,40 (7,90)	8,70 (2,21)	30,50 (10,82)
flag	20,20 (1,62)	13,40 (2,12)	1,30 (0,48)	1,60 (3,53)
nursery	112,40 (2,27)	17,60 (0,52)	44,10 (7,32)	57,60 (2,37)
kr-vs-kp	30,10 (1,85)	28,10 (1,73)	27,30 (1,70)	61,20 (9,87)
letter-a	126,30 (3,56)	120,20 (2,70)	71,00 (6,58)	115,20 (6,16)
satimage	158,80 (6,09)	199,80 (6,99)	32,00 (4,37)	52,30 (13,90)
vehicle	49,80 (3,29)	41,20 (2,82)	23,70 (4,06)	75,70 (12,11)
Média	**73,49**	**61,89**	**18,33**	**38,33**

Na Tabela 5.5.10, é possível observar que o MOPSO, com a abordagem analisada, apresentou o maior suporte médio e precisão relativa ponderada de todos os algoritmos analisados. Possivelmente, conforme já verificado, o valor do suporte seria maior com a utilização de objetivos que preservam a cobertura das regras. Dessa forma, há grande vantagem do algoritmo na legibilidade e facilidade de compreensão de suas regras em relação aos outros algoritmos analisados.

Tabela 5.5.10 — Suporte Médio e Precisão Relativa Ponderada das regras obtidas.

	Suporte (%)	WRAcc
MOPSO	20,80 (8,75)	0,0487 (0,038)
Roccer	13,67 (13,89)	0,0355 (0,018)
C4.5	3,73 (6,01)	0,0094 (0,013)
C4.5 SP	1,19 (1,06)	0,0030 (0,003)
CN2NO	3,90 (2,52)	0,0110 (0,009)
CN2	3,10 (2,18)	0,0085 (0,007)
Ripper	5,96 (5,34)	0,0184 (0,012)
Slipper	1,92 (1,58)	0,0114 (0,014)

Isso vem mostrar que o algoritmo se adapta às necessidades do problema, com a manipulação dos parâmetros, obtendo ainda regras boas com alta importância e suporte, sendo possível assim, sua utilização para a mineração de dados, por exemplo, bem como uma melhor leitura e entendimento do classificador. Essa é uma grande vantagem desta abordagem.

Outro aspecto a analisar é o tempo de computação das otimizações. O tempo de execução do MOPSO é influenciado, de forma simplificada, pelo tamanho da base em número de exemplos e atributos (assim como todos os outros algoritmos) e também pelo número de partículas e gerações. Com isso, a base que necessitou de maior tempo de computação (em um Intel Celeron 1.3 GHz, 512 RAM) com 100 gerações e 450 partículas foi a *nursery* com a média de 570 segundos por partição, seguida da *german* com a média de 100 segundos por partição, o que é um valor bom. Das execuções com 50 gerações e 450 partículas a base mais custosa foi a *letter-a* com a média de 800 segundos por partição, seguida da *satimage* com a média de 550 segundos por partição. Esses valores foram quase os mesmos com o limiar de seleção do classificador em 0,01.

6 CONCLUSÕES

Neste trabalho, foi apresentado um método de aprendizado de regras de classificação que utiliza a técnica de nuvem de partículas multiobjetivo (MOPSO) para encontrar uma fronteira de regras que são então selecionadas para integrar um classificador não-ordenado.

Os objetivos do algoritmo são: encontrar um classificador sem retirar exemplos da base, de modo que as regras tenham sentido isoladamente e sejam boas para a classificação e posterior entendimento; que seja possível escolher objetivos diversos, para obter regras com características diversas, como por exemplo, precisão, sensitividade e novidade.

Para isso, descrevemos os principais conceitos de aprendizado de regras, bem como alguns problemas de outros algoritmos para a obtenção de nossos objetivos. Foi descrita a técnica de nuvem de partículas e sua variante MOPSO. Vimos como os parâmetros da velocidade podem influenciar na otimização e convergência da nuvem. Em seguida, no Capítulo 4, apresentamos todos os aspectos de nosso método, desde a criação e representação das regras até a complexidade do algoritmo.

No Capítulo 5, pudemos perceber quantos experimentos podem ser feitos para o entendimento do comportamento do algoritmo. Ainda assim, muitos outros possíveis não foram efetuados e podem se constituir em trabalhos futuros. Realizamos experimentos com os parâmetros da velocidade, analisando sua influência na obtenção da fronteira e na exploração do espaço. Ainda assim, outras configurações de parâmetros poderiam ser testadas como a que varia as prioridades continuamente durante as gerações, começando, por exemplo, valorizando ϕ_1, passando, com o tempo, a valorizar ϕ_2. Esta talvez seja uma boa escolha de parâmetros de velocidade. Vimos também uma influência do número de gerações e de partículas nessa exploração e obtenção da fronteira, observando que o aumento do número de partículas foi melhor que o aumento do número de gerações, para a obtenção da fronteira.

Nesse capítulo ainda, verificamos os resultados do algoritmo com o limiar da fronteira do classificador maior que zero e como isso pode afetar a AUC e o tamanho do classificador. Esse limiar pode ser bem empregado nos casos em que a fronteira de Pareto é muito pequena. Logo, nesse caso, é interessante aumentar a fronteira, deixando regras que estão próximas da fronteira fazerem parte dela. No entanto, chega o ponto em que a inclusão de novas regras no classificador traz um pior desempenho de AUC. Um estudo de como encontrar esse ponto pode vir a ser alvo de um trabalho futuro.

Em seguida, comparamos algumas abordagens de objetivos e de classificação. Percebeu-se que a utilização da votação de confiança com voto negativo, aliada à otimização da confiabilidade positiva e negativa, trouxe melhores resultados de AUC e número de regras.

Na comparação com outros algoritmos da literatura, percebemos que nosso algoritmo é competitivo e que, em grande parte das bases testadas, o valor de AUC e de número de regras foi bom. Nos piores resultados, vimos que com um maior limiar de seleção para o grupo classificador, mencionado anteriormente, o resultado seria melhor, alcançando os outros algoritmos. Além disso, as regras do classificador são regras de alto suporte e importância, o que era um dos objetivos da análise, e isso ainda é obtido em um baixo tempo computacional.

O algoritmo apresentado traz interessantes contribuições à área de aprendizado de regras, pois mostra uma nova abordagem do tema, uma vez que a seleção de suas regras é feita de modo multiobjetivo, sem a depreciação das regras subseqüentes. É interessante perceber, nesse caso, como a sobreposição das regras dá maior poder ao voto, sendo um ponto positivo para a classificação. Além disso, o voto negativo, que introduzimos no método de votação por confiança, representa uma efetiva contribuição no processo de classificação de classificadores não ordenados, pois obriga o voto de todas as regras para todas as instâncias de entrada.

Uma extensão natural deste trabalho seria a definição de um melhor critério de parada que pudesse evitar a baixa cobertura nas bases com alto número de exemplos, conforme proposto na Seção 4.7. Outra extensão natural seria a de

trabalhar com outros tipos de atributos não-categóricos. Ainda, conforme sugerido na seção 5.4, poderia ser definida uma relação que pudesse estabelecer automaticamente o limiar de seleção das regras para o classificador *E*, e que trouxesse uma melhor relação AUC/tamanho do classificador, analisando, possivelmente, características de cada base de dados. É interessante também um estudo mais aprofundado do comportamento dos parâmetros de nuvens de partículas, bem como a utilização de outros objetivos na otimização, como novidade e outras medidas de interesse.

Além disso, seria ainda interessante o emprego do algoritmo em uma aplicação prática de mineração de dados, visto que suas regras possuem alto suporte e precisão relativa ponderada. Outro aspecto a considerar é o estudo de métodos de paralelização do algoritmo, a fim de reduzir o tempo computacional, podendo definir um alto número de partículas e gerações, para que assim o algoritmo aumente sua aproximação da verdadeira Fronteira de Pareto.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] MONARD, M. C.; BARANAUSKAS, J. A. Indução de Regras e Árvores de Decisão. In: REZENDE, S. (Ed.) **Sistemas Inteligentes:** Fundamentos e Aplicações. Barueri: Editora Manole, 2003. 525 p.

[2] NICOLA, U. **Antologia ilustrada de Filosofia: Das origens à idade moderna.** São Paulo: Editora Globo, 2006. 480 p.

[3] MITCHELL, T. M. **Machine Learning**. [S.l.]: WCB McGraw-Hill, 1997. 421 p.

[4] NILSSON, N.J. **Introduction to Machine Learning: An early draft of a proposed textbook.** Standford: Standford University, dez. 1996. 209 p.

[5] RUSSEL, S.; NORVIG, P. **Artificial Intelligence: A modern approach**. Nova Jérsei: Prentice Hall, 2003. 946 p.

[6] PARETO, V. **Manuel D'Économie Politique.** Paris: Marcel Giard, 1927.

[7] BAÑOS, R.; GIL, C.; PAECHTER, B.; ORTEGA, J. Parallelization of Population-based Multi-Objective Metaheuristics: An Empirical Study. **In: Applied Mathematical Modelling**, v.30, n.7, jul. 2006. p. 578-592.

[8] HOLLAND, J. H. **Adaptation in Natural and Artificial Systems**. Ann Arbor: University of Michigan Press, 1975. 211 p.

[9] KENNEDY, J.; EBERHART, R.C. Particle Swarm Optimization. In: **Proceedings of the 1995 IEEE International Conference on Neural Networks.** IEEE Press, 1995. p. 1492-1948.

[10] DORIGO, M. **Optimization, Learning and Natural Algorithms**. 1992. Tese (doutorado). – Politécnico de Milão, Milão, 1992.

[11] SOUSA, T. F.; SILVA, A. P.; SILVA, A. F. Particle Swarm Based Data Mining Algorithms for Classification Tasks. **Parallel Computing**. 30, p. 767-783, Elsevier B. V., jan. 2004. p. 767-783.

[12] SOUSA, T. F.; SILVA, A. P.; SILVA, A. F. Particle Swarm Data Miner. **In: Lecture Notes in Computer Science**, **v. 2902: Progress in Artificial Intelligence. Proceedings of the 11th Portuguese Conference on Artificial Intelligence.** 2003. p. 43-53.

[13] FAWCETT, T. Using Rule Sets to Maximize ROC Performance. In: **IEEE International Conference on Data Mining (ICDM-01),** 2001, p. 131-138.

[14] CLARK, P.; BOSWELL, R. Rule Induction with CN2: Some Recent Improvements. **Machine Learning — Proceedings of the European Conference.** Berlim: Springer-Verlag, 1991. p. 151-163.

[15] PHAM, T. H.; CLEMENTE, J. C.; SATOU, K.; HO, T. B. Computational Discovery of transcriptional regulatory rules. **In: Bioinformatics**, v. 21, n. 1, set. 2005. p. 101-107.

[16] PRATI, C. R.; FLASH, P. A. Roccer: A ROC convex hull rule learning algorithm. In: **ECML/PKDD 04 Workshop on Advances in Inductive Rule Learning.** Pisa (Itália): [s.n], 2004. p. 144-153.

[17] NIBLETT, T.; CLARK, P. The CN2 induction algorithm. **Machine Learning.** v. 3, n. 4, mar. 1989. p. 261-283.

[18] PRATI, C. R.; FLASH, P. A. ROCCER: An Algorithm for Rule Learning Based on ROC Analysis. In: **Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI'05).** IJCAI, ago. 2005. p. 823-828.

[19] QUINLAN, J. R. **C4.5 programs for Machine Learning.** San Mateo (EUA): Morgan Kaufmann, 1993.

[20] LIU, B.; HSU,W.; MA, Y. Integrating classification and association rule mining. **In: Proceedings of the 4th Int. Conf. on Knowledge Discovery and Data Mining**. Nova Iorque, 1998. p. 80-86.

[21] BATISTA, G. E. A. P. A.; PRATI, R. C.; MONARD, M. C.; GIUSTI, R; MILARÉ, C. R. Classificação Associativa Utilizando Seleção e Construção de Regras: um Estudo Comparativo. **In: Encontro Nacional de Inteligência Artificial (ENIA)**, 2007, Rio de Janeiro. Anais do Congresso da Sociedade Brasileira de Computação, 2007. p. 1321-1330.

[22] STEVENS, S. S. Mathematics, measurement, and psychophysics. In: S.S. Stevens (Ed.), **Handbook of experimental psychology**. New York: John Wiley, 1951. p. 1-49.

[23] STEVENS, S. S. On the theory of scales of measurement. **Science**, **103**, 1946. p. 677-680.

[24] AGRAWAL, R.; MANNILA, H.; SRIKANT, R.; TOIVONEN, H.; VERKAMO, A. I. Fast Discovery of Association Rules. **In: U.M.Fayyad, G. Piatetsky- Shapiro, P. Smyth and R. Uthurusamy, eds., Advances in Knowledge Discovery and Data Mining. AAAI/MIT**, 1996. p. 307-328.

[25] LAVRAC, N.; FLACH, P. A.; KASEK, B.; TODOROVSKI, L. Rule induction for subgroup discovery with CN2-SD. **In: ECML/PKDD'02 workshop on Integration and Collaboration Aspects of Data Mining**, Decision Support and Meta-Learning. M. Bohanec, B. Kasek, N. Lavrac, D. Mladenic, (eds.), ago. 2002. p. 77–87.

[26] PROVOST, F.; FAWCETT, T. The case against accuracy estimation for comparing induction algorithms. **In: Proceedings of the 15th International Conference on Machine Learning**. São Francisco: Morgan Kaufmann, 1998. p. 445-453.

[27] WESTIN, L. K. **Receiver operating characteristics (ROC) analysis:** Evaluating discriminance effects among decision support systems. Umeå (Suécia): Umeå University, 2001. 28 p.

[28] BRADLEY, A. P. The use of the area under the ROC curve in the evaluation of machine learning algorithms. **Pattern Recognition**. 30(7):1145-1159, 1997.

[29] HANLEY, J. A.; MCNEIL, B. J. The meaning and use of area under a receiver operating characteristic (ROC) curve. **Radiology**. n. 143, 1982. p. 29-36.

[30] PROVOST, F.; FAWCETT, T. Robust classification for imprecise environments. **Machine Learning**, v. 42, n. 3, mar. 2001. p. 203-231.

[31] JIN, Y. (ed.) **Multi-Objective Machine Learning**. Berlim: Springer, 2006.

[32] ISHIBUCHI, H.; NOJIMA, Y. Accuracy-Complexity Tradeoff Analysis by Multiobjective Rule Selection. **Proceedings of ICDM 2005 Workshop on Computational Intelligence in Data Mining,** 2005. p. 39-48.

[33] IGLESIA, B.; REYNOLDS, A.; RAYWARD-SMITH, V. J. Developments on a Multi-Objective Metaheuristic (MOMH) Algorithm for Finding Interesting Sets of Classification Rules. **Lecture Notes in Computer Science, v. 3410.**

Evolutionary Multi-Criterion Optimization - EMO 2005. Berlim: Springer, 2005. p. 826-840.

[34] ISHIBUCHI, H.; KUWAJIMA, I.; NOJIMA, Y. Multiobjective association rule mining. **Proceedings of PPSN Workshop on Multiobjective Problem Solving from Nature.** Reiquejavique. Set. 2006. 12 p.

[35] PILA, A. D.; GIUSTI, R.; PRATI, R. C.; MONARD, M. C. A Multi-Objective Evolutionary Algorithm to Build Knowledge Classification Rules with Specific Properties. In: **Sixth International Conference on Hybrid Intelligent Systems (HIS'06)**, 2006. p. 41.

[36] IGLESIA, B.; PHILPOTT, M. S.; BAGNALL, A. J.; RAYWARD-SMITH, V. J. Data Mining Rules using Multi-Objective Evolutionary Algorithms. **Proceedings of 2003 Congress on Evolutionary Computation**. 2003. p. 1552-1559.

[37] VESTERSTRØM, J. S.; RIGET, J. **Particle Swarms: Extensions for improved local, multi-modal, dynamic search in numerical optimization. 2002.** Dissertação (mestrado). Faculty of Science, Aarhus Universitet, Aarhus (Dinamarca), 2002. 203 p.

[38] KENNEDY, J.; EBERHART, R.C. **Swarm Intelligence.** [S.l]: Academic Press, 2001. 512 p.

[39] SOUZA, G. R. **Uma abordagem por nuvem de partículas para problemas de otimização combinatória.** Dissertação mestrado. Natal: Universidade Federal do Rio Grande do Norte, fev. 2006. 75 p.

[40] EBERHART, R. C.; SHI, Y. Evolving artificial neural networks. **Proceedings 1998 Int'l Conf. on Neural Networks and Brain**. Pequim, P.R.C., p. 5-13.

[41] ESMIN, A. A. A.; AOKI, A. R.; LAMBERT-TORRES, G. Particle swarm optimization for fuzzy membership functions optimization. In: **IEEE International Conference on Systems, Man and Cybernetics 2002**. Tunisia, 2002. p. 108-113.

[42] EBERHART, R. C.; HU, X. Human tremor analysis using particle swarm optimization. **Proceedings of the Congreess on Evolutionary Computation 1999,** Washington, DC, p. 1927-1930.

[43] PAPACOSTANTIS, E.; ENGELBRECHT, A. P.; FRANKEN, N. Coevolving Probabilistic Game Playing Agents using Particle Swarm Optimization Algorithms. **In: Proceedings of the IEEE on Evolutionary Computation in Games Symposium**, 2005. p. 195-202.

[44] YOSHIDA, H.; KAWATA, K.; FUKUYAMA, S.; NAKANISHI, Y. A Particle Swarm Optimization for reactive Power and Voltage Control considering voltage stability. **In: Proceedings of the international Conference on Intelligent System Application to Power System**, 1999. p. 117-121.

[45] YOSHIDA, H.; KAWATA, K.; FUKUYAMA, S.; NAKANISHI, Y. A Particle Swarm Optimization for reactive Power and Voltage Control considering voltage security assessment. **In: Proceedings of the IEEE International Conference on Systems Man, and Cybernatics**, v.6, 1999. p. 497-502.

[46] TILLET, J.C.; RAO, R.; SAHIN, C. K. F.; RAO, T. M. Cluster-head Identification in Ad-hoc Sensor Networks using Particle Swarm optimization. **In: Proceedings of 2002 IEEE International Conference on personal Wireless Comunications**. 2002. p. 201-205.

[47] SHI, Y. EBERHART, R. C. A modified particle swarm optimizer. **In: IEEE International Conference on Evolutionary Computation.** Anchorage, Alaska, maio 1998. p. 4-9.

[48] CLERC, M.; KENNEDY, J. The Particle Swarm – Explosion, Stability, and convergence in a Multi-dimensional Complex Space. **In: IEEE Transactions on Evolutionary Computation**, v. 6., n. 1, fev. 2002. p. 58-73.

[49] LØVBJERG, M.; RASMUSSEN, T. K.; KRINK, T. Hybrid Particle Swarm Optimiser with Breeding and Subpopulations. **Proceedings of the third Genetic and Evolutionary Computation Conference (GECCO-2001).** 2001.

[50] LØVBJERG, M.; KRINK, T. Extending Particle Swarms Optimisers with Self-Organized Criticality. **In: Proceedings of the Fourth Congress on Evolutionary Computation (CEC-2002).** 2002.

[51] EBERHART, R. C.; SHI, Y. Comparing inertia weights and constriction factors in particle swarm optimization. **Proceedings Congress on Evolutionary Computation 2000 (CEC-2000),** San Diego, EUA, 2000. p. 84-88.

[52] FAN, H.Y. SHI, Y. Study of Vmax of the particle swarm optimization algorithm. **Proceedings of the Workshop on Paticle Swarm Optimization.** Indianapolis, Purdue School of Engineering and Technology, IUPUI.

[53] SHI, Y.; EBERHART, R. C. Parameter selection in particle swarm optimization. **In: Evolutionary Programming VII: Proc EP98,** Nova Iorque: Springer Verlag, p. 591-600.

[54] KENNEDY, J.; MENDES, R. Neighborhood Topologies in Fully-informed and best-of-neighborhood Particle Swarms. In: **Proceedings of the IEEE International Workshop on Soft Computing in Industrial Applications**. jun. 2003. p. 45-50.

[55] FIELDSEND, J.E. **Multi-Objective particle Swarm optimization methods.** [S.l.]: [s.n.], mar. 2004

[56] COELLO, C. A. C.; LECHUGA, M. S. MOPSO: A proposal for Multiple Objective Particle Swarm Optimization. In: IEEE Wold Congress on Computational Intelligence, 2002. **Proceedings of the 2002 Congress on Evolutionary Computation.** Havaí: IEEE Press, maio 2002. p. 1051-1056.

[57] MOSTAGHIM, S.; TEICH, J. Strategies for finding good local guides in multi-objective particle swarm optimization (mopso). **In: IEEE 2003 Swarm Intelligence Symposium**, 2003. p. 26-33.

[58] FREITAS, A. A. **Data Mining and Knowledge Discovery with Evolutionary Algorithms.** Berlim: Spinger-Verlag, 2002. 264 p.

[59] ASUNCION, A.; NEWMAN, D. J. (2007). **UCI Machine Learning Repository.** Disponível em: <http://www.ics.uci.edu/mlearn/MLRepository.html>. Irvine: Universidade da Califórnia.

[60] ZITZLER, E. **Evolutionary algorithms for multiobjective optimization: methods and applications.** PhD Thesis Swiss Federal Institute of Technology. Zurique, Suíça.

[61] **THE R PROJETCT FOR STATISTICAL COMPUTING.** Disponível em: <http://www.r-project.org> Acesso em: 5 fev. 2008.

[62] COHEN, W. W. Fast effective rule induction. **In: ICML**, 1995. p. 115-123.

[63] COHEN, W. W.; SINGER, Y. A simple, fast, and effective rule learner. In: **Proceedings of the 11th Conference on Innovative Applications of Artificial Intelligence**, Menlo Park, Cal. AAAI/MIT Press. 1999. p. 335-342.

APÊNDICE A – OBTENÇÃO DA ÁREA ABAIXO DA CURVA ROC (AUC)

Este capítulo mostra um exemplo do cálculo da pontuação de um classificador com três classes e de sua medida de AUC. A Tabela A.1 traz um exemplo de pontuações obtidas por um método de classificação por votação, para uma base de três classes. A coluna *classe base* representa a verdadeira classe do exemplo, presente na base de treinamento. As colunas *votos C1, votos C2* e *votos C3* mostram o número de votos obtidos, para cada classe em um classificador com 20 regras. Nesse caso, cada voto vale 1 ponto e não há voto negativo. As colunas *limiar C1, limiar C2* e *limiar C3* representam a pontuação resultante de cada exemplo para cada classe da base, estruturada na forma de positivo ou negativo.

Tabela A.1 — Pontuação de cada classe para o exemplo

índice	votos C1	votos C2	votos C3	Limiar C1	Limiar C2	Limiar C3	Classe Base
1	10	7	3	0	-6	-14	C1
2	7	7	6	-6	-6	-8	C2
3	0	9	11	-20	-2	2	C2
4	5	3	12	-10	-14	4	C3
5	13	3	4	6	-14	-12	C1

A pontuação de uma classe é calculada por exemplo da base, sendo o resultado do número de votos obtidos pela classe menos a soma dos votos obtidos pelas outras classes. Neste exemplo, é criada a curva ROC da classe C1. Portanto, somente a coluna *Limiar C1* será considerada. Se o limiar do classificador for menor ou igual ao limiar de C1, então o exemplo é classificado como positivo. Do contrário, é negativo.

Para criar a curva, definimos limiares para o classificador, começando por aquele que o faz classificar todos os exemplos como positivos. Calculam-se então a sensitividade e especificidade do classificador, que nesse caso é 1 e 0, respectivamente. Em seguida, elevamos o limiar para um valor mais alto, e calculamos as métricas novamente. Esse processo continua até o limiar que classifica todos os exemplos como negativos para a classe C1.

Logo, no exemplo da Tabela A.1, começaríamos pelo limiar -20, o que faria com que todos os exemplos fossem classificados como positivos. Ao subirmos o

limiar do classificador, o exemplo número 3 passaria a ser classificado como negativo, enquanto os outros se manteriam como positivos. Nesse caso, a sensitividade é 1, pois todos os exemplos C1 são classificados como positivos, e a especificidade é 1/3, pois um exemplo não C1 é classificado como negativo. Em seguida, o próximo limiar a causar alteração na pontuação seria o de valor maior que -10. Nesse caso, os exemplos 3 e 4 seriam classificados como negativos. A sensitividade seria 1 e a especificidade seria 2/3. O próximo limiar a causar alteração na pontuação seria maior que -6, o que faria com que os exemplos 2, 3 e 4 fossem negativos. Nesse caso, a sensitividade continua 1 e a especificidade passa também a 1. O próximo exemplo a ser negativo é o primeiro, sendo, nesse caso, a sensitividade ½ e a especificidade igual a 1. Depois, com o limiar maior que 6, o classificador classificaria todos os exemplos como negativos em relação a C1, e assim, a sensitividade seria 0 e a especificidade, 1.

Assim, os pontos que formam a curva ROC do classificador para a classe C1 são $\left\{(1,0),\left(1,\frac{1}{3}\right),\left(1,\frac{2}{3}\right),(1,1),\left(\frac{1}{2},1\right),(0,1)\right\}$. A Figura A.1 mostra o gráfico ROC para o classificador da classe C1 do exemplo. Note, no gráfico, que a AUC do classificador para essa classe é 1, pois é a área de todo o retângulo formado abaixo da curva. Nesse gráfico, o valor da abscissa é *1 – especificidade*, que é a taxa de falso positivo.

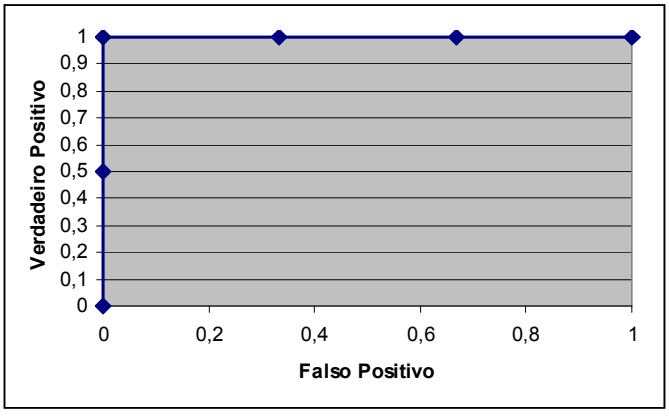

Figura A.1 – Curva ROC da classe C1.

APÊNDICE B – DESEMPENHO DA FRONTEIRA COM A MÉTRICA S

Este capítulo mostra como se dá o cálculo de desempenho da fronteira para fins de comparação por meio da Métrica S [60].

A métrica S calcula o hiper-volume ou área de uma região multidimensional compreendida entre um conjunto de pontos e um ponto de referência. Um bom ponto de referência Z_{ref} seria o pior valor de cada objetivo. Dessa forma, (1,1) para um problema de minimização biobjetivo.

Ordenando os pontos do primeiro objetivo em ordem decrescente, podemos calcular a métrica S para um problema biobjetivo, pela seguinte expressão:

$$\sum_{i\in 1..N} \left| z_1^i - z_1^{ref} \right| \cdot \left| z_2^i - z_2^{i-1} \right|,$$

onde z_2^0 é z_2^{ref} e N é o número de pontos da fronteira.

Com essa métrica, é possível avaliar a qualidade geral de um conjunto não-dominado. É uma métrica não cardinal, pois não é baseada no número de pontos da fronteira [39]. A Figura B.1 apresenta uma fronteira e sua equivalente hiper-área da métrica S.

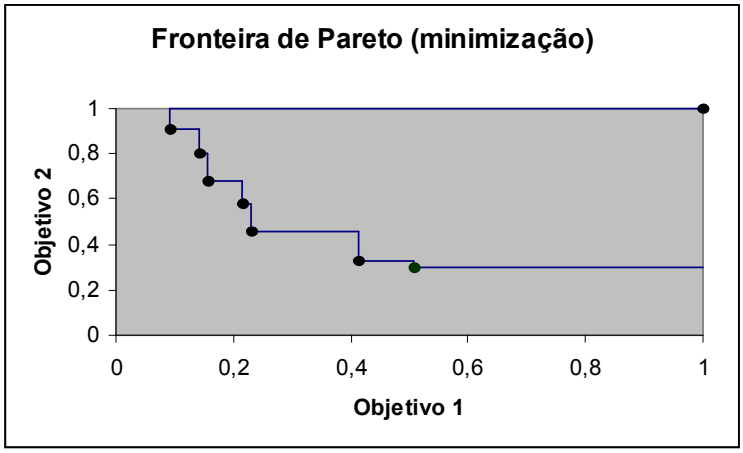

Figura B.1 – Desempenho de uma fronteira exemplo.
A área acima da linha é o valor da métrica S.

APÊNDICE C – VOTAÇÃO POR CONFIANÇA COM VOTO NEGATIVO

Este capítulo descreve a forma de classificação por confiança que utiliza o voto negativo. Essa forma se baseia na idéia de que as regras, além de, às vezes, terem certeza positivamente, representada por sua confiabilidade positiva, podem ter certeza negativamente, representada por sua confiabilidade negativa. Ou seja, as regras têm condições de determinar, com um grau de certeza, se a instância a ser classificada não pertence a sua classe. Nesse caso, a pontuação da sua classe é diminuída proporcionalmente a essa certeza.

Se uma regra tem alta confiabilidade negativa, então na maioria das vezes que esta regra não cobre um exemplo, é muito provável que o exemplo não pertença à classe da regra. Portanto, temos de diminuir sua pontuação com relação às outras classes. Isso deve funcionar de forma satisfatória quando se tem sobreposição de regras, pois assim há várias opiniões na votação.

A Tabela C.1 apresenta um exemplo de um conjunto de regras classificadoras de três classes e os exemplos de uma base de teste, acompanhados por uma lista das regras que os cobrem e outra, das que não cobrem.

Tabela C.1 – Regras e exemplos de teste

Regra	**Confiabilidade +**	**Confiabilidade -**	**Classe Predita**
R1	0,90	0,70	C1
R2	0,85	0,75	C1
R3	0,85	0,75	C2
R4	0,85	0,75	C3
R5	0,70	0,80	C1
R6	0,70	0,80	C2
R7	0,65	0,85	C3
R8	0,60	0,90	C2
Ex.	**Coberto por**	**Não coberto por**	**CLASSE**
1	R1, R2, R7, R8	R3, R4, R5, R6	C1
2	R3, R5, R6, R8	R1, R2, R4, R7	C2
3	R3, R6, R8	R1, R2, R4, R5, R7	C2
4	R4, R7, R8	R1, R2, R3, R5, R6	C3
5	R1, R2, R6	R3, R4, R5, R7, R8	C1

Neste exemplo, demonstramos como é feita a obtenção dos limiares de classificação para cada classe, que são então utilizados para a obtenção da AUC, conforme é descrito no Apêndice A. Regras que cobrem o exemplo somam a confiabilidade positiva às suas classes. Regras que não cobrem subtraem sua confiabilidade negativa de suas classes. Dessa forma, a pontuação para o primeiro exemplo é calculada da seguinte maneira:

R1 cobre, soma-se 0,90 à classe C1 (C1=0,90; C2=0; C3=0).

R2 cobre, soma-se 0,85 à classe C1 (C1=1,75; C2=0; C3=0).

R7 cobre, soma-se 0,65 à classe C3 (C1=1,75; C2=0; C3=0,65).

R8 cobre, soma-se 0,60 à classe C2 (C1=1,75; C2=0,60; C3=0,65).

R3 não cobre, subtrai-se 0,75 de C2 (C1=1,75; C2=-0,15; C3=0,65).

R4 não cobre, subtrai-se 0,75 de C3 (C1=1,75; C2=-0,15; C3=-0,10).

R5 não cobre, subtrai-se 0,80 de C1 (C1=0,95; C2=-0,15; C3=-0,10).

R6 não cobre, subtrai-se 0,80 de C2 (C1=0,95; C2=-0,95; C3=0,65).

Com isso, tem-se a pontuação de cada classe para o primeiro exemplo de teste. A Tabela C.2 apresenta a pontuação para os demais exemplos, já calculados os limiares das classes.

Tabela C.2 – Pontuação de cada classe (VCVN)

	Ponto C1	**Ponto C2**	**Ponto C3**	**Limiar C1**	**Limiar C2**	**Limiar C3**	**Classe Base**
1	0,95	-0,95	0,65	1,25	-2,55	0,65	C1
2	-0,75	2,15	-1,60	-1,30	4,50	-3,00	C2
3	-2,25	2,15	-1,60	-2,80	6,00	-1,50	C2
4	-2,25	-0,95	1,50	-2,80	-0,20	4,70	C3
5	0,95	-0,95	-1,60	3,50	-0,30	-1,60	C1

Considerando o limiar de C1, note que com o limiar do classificador definido como 1, por exemplo, o classificador acertaria todos os exemplos de teste para a classe C1, pois consideraria os exemplos 1 e 5 como C1 e os demais não C1.