CELSO YOSHIKAZU ISHIDA

Explorando Abordagens Inovadoras para Geração de Classificadores

CURITIBA

Novembro 2008

CELSO YOSHIKAZU ISHIDA

Explorando Abordagens Inovadoras para Geração de Classificadores

Tese apresentada como requisito parcial à obtenção do grau de Doutor em Métodos Numéricos em Engenharia na área de concentração: Programação Matemática, pelo Programa de Pós-Graduação de Métodos Numéricos em Engenharia, Universidade Federal do Paraná.

Orientador: Aurora T. R. Pozo

CURITIBA

Novembro 2008

Termo de Aprovação

CELSO YOSHIKAZU ISHIDA

Explorando Abordagens Inovadoras para Geração de Classificadores

Tese aprovada como requisito parcial para obtenção do grau de Doutor em Métodos Numéricos em Engenharia na área de concentração: Programação Matemática, pelo Programa de Pós-Graduação em Métodos Numéricos em Engenharia, Universidade Federal do Paraná, pela seguinte banca examinadora:

Prof. Dra. Aurora T. R. Pozo
Programa de Pós-Graduação em Métodos Numéricos em Engenharia da UFPR

Prof. Dra. Elizabeth Ferreira Gouvêa Goldbarg
Programa de Pós-Graduação em Sistemas e Computação da UFRN

Prof. Dra. Maria Teresinha Arns Steiner
Programa de Pós-Graduação em Métodos Numéricos em Engenharia da UFPR

Dedicatória

Dedico a Deus, oditian, obatian, pai e mãe,

Pelo muito que Ele sempre fez e que permitiu que eu tivesse saúde, apoio da família e amigos para a construção deste trabalho. Dedico, também, a Yoshiharu e Fujie Ishida, Tsunetaro e Yatiko Funaki, e meus pais Keniti e Satiko Ishida; todos eles me ajudaram na minha formação como pessoa. Sem eles, não teria oportunidade de uma boa educação e o apoio necessário para que eu pudesse aproveitar as oportunidades que me foram concedidas.

Agradecimentos

À minha querida esposa Lucy, que amo, e que sempre me apoiou quando precisei. À minha irmã Erika e aos meus tios que me deram um grande suporte, até nas coisas básicas da vida, em especial aos tios Jaci Funaki, Eunice Funaki, Mario Funaki, Rosa Funaki e Akiko Ishida. Aos amigos, quase irmãos, que me apoiaram e torceram pelo meu sucesso. Aos professores, em especial, agradeço à amiga Aurora pela sua paciência e por ter me dado oportunidades nestes anos de estudo. Também agradeço à CAPES pelo apoio financeiro durante o doutorado e bolsa *sandwich* na França.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Siglas

DM	Data Mining
ML	Machine Learning
ES	Evolution Strategies
ROC	Receiver Operating Characteristic
GRASP	Greedy Randomized Adaptive Search Procedure
BES	Algoritmo Boosting com Evolution Strategies
VP	Verdadeiro Positivo
FN	Falso Negativo
FP	Falso Positivo
VN	Verdadeiro Negativo
IQ	Information Quantity
ROCCER	Algoritmo de seleção de regras baseado na curva ROC
ROGER	Algoritmo de aprendizado genético baseado na curva *ROC*
CI	Computational Intelligence
EC	Evolutionary Computation
GP	Genetic Programming
AUCIQ	Área abaixo da curva formada pelos intervalos não-dominados
BES	Algoritmo Boosting com Estratégia Evolucionária
AUC	Area Under the ROC Curve
PFE	Pareto Front Elite
PSO	Particle Swarm Optimization
MOPSO	Multiple Objective Particle Swarm Optimization
RCL	Restricted Candidate List
GPR-RL	Algoritmo de aprendizado de regra que utiliza *GRASP* e path-relinking

Resumo

Desde os anos 80, a área de aprendizado de máquina tem contribuído para a descoberta de conhecimento e, dentre as suas atribuições, mais especificamente, a tarefa de classificação tem sido utilizada para a formulação de modelos de auxílio para previsões. Neste sentido, o propósito desta tese é melhorar a indução de classificadores por meio da combinação de conjuntos não-ordenados de modelos com a utilização de critérios biobjetivos. Para atingir tal objetivo foram utilizadas duas abordagens. A primeira abordagem buscou verificar a hipótese de que a combinação das técnicas de *boosting* e estratégia evolucionária podem atingir o equilíbrio entre precisão e generalidade. Para validação da hipótese, foi proposta uma nova linguagem de representação, juntamente com novas medidas de avaliação. Experimentos foram realizados comparando a proposta com um algoritmo clássico e, embora apenas um caso tenha demonstrado um desempenho melhor, acredita-se que a linguagem de representação criada pode ser útil com outras estratégias, sendo apontados alguns caminhos a explorar para seu aperfeiçoamento. A segunda abordagem investigou a hipótese de que a criação de um conjunto de regras não-ordenadas segundo critérios biobjetivos pode maximizar a *AUC* (Área abaixo da curva ROC (*Receiver Operating Characteristic*)). Procurou-se identificar as medidas desejáveis para a criação de conjuntos que fossem os mais próximos da Fronteira de Pareto. Para tal, foi criado o algoritmo *Pareto Front Elite* (PFE) que, deterministicamente, gera as regras e faz a seleção de acordo com os critérios de sensitividade e especificidade. Os resultados dos experimentos demonstraram que é possível utilizar os critérios biobjetivos para a maximização da *AUC*. Outras investigações foram feitas com a mesma finalidade do algoritmo PFE, porém, com o objetivo maior de trabalhar com grandes bases de dados; os resultados foram dois algoritmos baseados em metaheurísticas para a criação num único passo do conjunto de regras não-ordenadas. O primeiro algoritmo, MOPSO, utilizou a nuvem de partículas com conceitos multiobjetivos; alguns experimentos foram realizados e os resultados de *AUC* foram semelhantes ou melhores do que a maioria dos trabalhos relacionados. A combinação entre *GRASP* e *path-relinking* resultou na construção do segundo algoritmo: *GRASP-PR Rule Learning*. Experimentos com o algoritmo foram realizados e os valores de *AUC* foram comparados, mostrando-se compatíveis com o algoritmo determinístico, além de serem melhores do que a maioria dos trabalhos relacionados. Outros experimentos foram conduzidos para a comparação dos conjuntos de aproximação utilizando como base o algoritmo com o melhor desempenho em termos de *AUC*; avaliações qualitativas e quantitativas multiobjetivas confirmaram que os conjuntos de aproximações da proposta são semelhantes às Fronteiras de Pareto.

Palavras-chave: Classificação; *Boosting*; Estratégia Evolucionária; *AUC*; *GRASP*.

Abstract

Since the 80's, the Machine Learning domain has been useful to knowledge discovery. More specifically, the classification task has been used to construct models for prevision. The goal of this thesis is to improve the induction of classifiers through the combination of non-ordered models using bi-objectives criteria. Two approaches were constructed for this proposes. The first one verified the hypothesis that boosting technique can be combined to evolutionary strategies to reach the balance of precision and generality. To validate this hypothesis, a new representation language and new evaluation measures were proposed. Experiments were done to compare the created algorithm with a classical one. Although, only one case demonstrated better performance, we believe the representation language can be useful with others strategies, for that, some options must be tested and some improvement were indicated. The second approach investigated the hypothesis that the creation of a non-ordered set of rules following bi-objectives criteria can maximize the *AUC* (Area Under the Receiver Operating Characteristic Curve). It was investigated the desirable measures for the creation of set of rules as close as possible to the Pareto Front. For that, Pareto Front Elite (PFE) algorithm was created to generate the rules with a deterministic mechanism which makes the rule selection using the sensitivity and specificity criteria. The experiment results show that it is possible to use bi-objective criteria to maximize the *AUC*. Other investigations were done with the same goal as PFE algorithm, however, the main goal was to deal with great data sets; the results were two meta-heuristic algorithms that create set of non-ordered rules in a single step. The first algorithm, MOPSO, used the particle swarm with multi objective criteria concepts. Some experiments with MOPSO were done and its *AUC* results were similar or better than the most part of the related works. The second algorithm was a result of GRASP combined with path-relinking. Experiments were done to evaluate *AUC* values and the results were close to the deterministic algorithm results, and, it was better than the most part of the related work. After choosing the algorithm with the best *AUC* performance to be the basis algorithm, other experiments where done to compare the approximation sets. Quality and Quantitative multiobjective measures confirm that most part of the approximation sets of this approach are similar to the Pareto Fronts.

Key-words: Classification; Boosting; ES; AUC; GRASP.

1 Introdução

No cenário econômico atual, as empresas têm grande necessidade de obter informações que realmente agreguem valor às decisões gerenciais (TURBAN; RAINER; POTTER, 2005). Estas informações, não-triviais, são importantes para a qualidade de seus produtos e até mesmo para a abertura de novos negócios. Neste sentido pode-se dizer que a análise de dados e a aplicação da informação e do conhecimento são necessidades crescentes para a sobrevivência de qualquer corporação (ISHIDA C. Y.; CAVALHEIRO, 1999).

A descoberta de conhecimento útil em grandes bancos de dados segue nesta trajetória e tem sido beneficiada, desde a década de 80, pelos algoritmos de mineração de dados (*Data Mining - DM*) e aprendizado de máquina (*Machine Learning - ML*). Esta descoberta tem aplicabilidade em áreas como: administração, biomedicina, bioinformática, projeto de equipamentos, finanças, medicina e outras (MITCHELL, 1997).

A classificação é um dos principais métodos utilizados para deduzir conhecimento novo e predizer eventos futuros (CHAN; STOLFO, 1993). Para tal, existem diferentes tarefas (*tasks*) de *DM*, como, por exemplo: descoberta de regras de associação, classificação, regressão, agrupamento (*clustering*), sumarização, dentre outros.

É possível encontrar analogias tanto na área de *ML*, como em *DM*, pois são duas áreas que se confundem, sendo que os algoritmos de *ML* são úteis para a tarefa de classificação em *DM*. A aprendizagem nos algoritmos de *ML* é feita em pequenos volumes de dados; em contraste a esta perspectiva, *DM* tem como objetivo trabalhar com grandes volumes de dados. Desta forma, é comum realizar o processo de classificação considerando pelo menos duas etapas. Na primeira, os classificadores são induzidos em diferentes conjuntos de dados utilizando *ML*. A segunda etapa combina classificadores numa abordagem integrada. Este trabalho enfoca a primeira etapa, que é a indução de classificadores em pequenos volumes de dados na tentativa de melhorar detalhes importantes no processo de classificação.

O resultado da tarefa de classificação é a formulação de um modelo, baseado em dados históricos. O modelo encontrado, também chamado de classificador, pode ser utilizado

para classificar um novo item em uma das classes pré-definidas. Para encontrar os modelos, os algoritmos de classificação têm como objetivo encontrar relações entre os atributos dos exemplos para a formulação dos modelos. O problema é que os dados históricos utilizados podem apresentar ruídos ou não representar corretamente a realidade, dificultando encontrar bons classificadores.

Com o objetivo de contribuir com a tarefa de classificação, esta tese vem trabalhar especificamente na melhoria da formulação de classificadores complexos apresentando duas novas abordagens.

A primeira abordagem visa à geração de classificadores segundo critérios biobjetivos conflitantes. Existem diversos critérios para a avaliação da qualidade dos modelos encontrados. Dentre eles, pode-se citar a precisão e a generalização que são dois critérios importantes e conflitantes; o equilíbrio entre estes critérios é o grande paradoxo da aprendizagem. Um algoritmo com ótima precisão é aquele que classifica os exemplos corretamente. Um algoritmo que tem muita precisão normalmente gera sobre-especialização (*overfitting*) no conjunto de exemplos, prejudicando a generalidade do modelo aprendido. A generalidade preocupa-se em fazer com que o modelo seja válido para o maior número de exemplos, ou seja, quanto mais exemplos são classificados corretamente pelo modelo, maior será a sua generalidade. Por outro lado, aprender com pouca precisão conduz a uma generalização de maneira que o modelo aprendido não é útil nem desejado pelo usuário (CORNUEJOLS; MICLET, 2003).

Obter um modelo que atenda aos dois critérios conflitantes é um trabalho árduo, não conseguido pelos algoritmos mais simples. Portanto, a primeira abordagem tem como hipótese a combinação do *boosting* e estratégia evolucionária para a criação do conjunto de classificadores que contemple os critérios de precisão e generalidade. *Boosting* (SCHAPIRE, 1990) (DRUCKER et al., 1994) (FREUND; SCHAPIRE, 1997) é uma técnica que busca obter a menor taxa de erro na classificação usando múltiplos preditores (DRUCKER, 1997). A técnica constrói modelos *fortes* a partir da combinação de outros modelos obtidos através de um algoritmo simples de classificação. No *boosting*, cada modelo é obtido sequencialmente com a utilização da dinâmica de pesos para a formação do conjunto solução.

A estratégia evolucionária (*Evolution Strategies - ES*) (RECHENBERG, 1973) é um dos paradigmas para a indução de classificadores na forma de uma função linear e que tem obtido resultados significantes em aplicações reais (SEBAG; AZE; LUCAS, 2003). Estratégia evolucionária é um algoritmo onde indivíduos (potenciais soluções) são codificados por um conjunto de variáveis de valores reais (KUSIAK, 2000). A ideia é imitar os princípios da evolução biológica em otimização de parâmetros. Segundo Schwefel (1981), “o método de evolução

orgânica representa uma estratégia ótima para a adaptação de seres vivos em seu ambiente... [e]... portanto seria conveniente utilizar os princípios da evolução biológica para a otimização de sistemas técnicos".

Outro aspecto desta tese foi trabalhar com mais um critério importante para a área de classificação: *AUC*. Uma desvantagem do critério de precisão é não ser um objetivo apropriado para muitos problemas reais (PROVOST; FAWCETT; KOHAVI, 1998). Portanto, muitos trabalhos têm preferido a análise *Receiver Operating Characteristic* (*ROC*). Na área de Aprendizado de Máquina, a curva *ROC* pode identificar visualmente o desempenho de um classificador (BRADLEY, 1997). Frequentemente, para o propósito de comparação de algoritmos de aprendizado, uma medida singular é preferida. Para tal, a área abaixo da curva *ROC* foi proposta para a construção e otimização de modelos de aprendizado (FERRI; FLACH; HERNANDEZ-ORALLO, 2002) (RAKOTOMAMONJY, 2004) (SEBAG; AZE; LUCAS, 2003). Quanto maior o valor da *AUC*, melhor é o conjunto de modelos. Com isso, a classificação pode ser considerada como um problema de otimização da medida *AUC*.

A segunda abordagem investiga a hipótese de que classificadores formados segundo critérios biobjetivos podem maximizar a *AUC*. Para esta abordagem foi utilizada a linguagem de representação dos modelos em forma de regras e o classificador é formado por um conjunto de regras não-ordenado. As regras são legíveis para os humanos (as regras são de fácil interpretação) e a ordem num conjunto não-ordenado de regras não influencia na classificação tornando o processo de avaliação menos oneroso para o desempenho do algoritmo.

Considerando que o classificador desejado seja um conjunto não-ordenado de regras, a maximização da *AUC* está diretamente relacionada à seleção correta das regras para cada conjunto de dados. A seleção é feita a partir das regras que podem ser geradas para cada conjunto de dados. Este conjunto de regras pode ser muito grande de modo a dificultar ou até mesmo inviabilizar a seleção do conjunto resultado. Por isso, o ideal para a formação do conjunto resultado é criar as regras com os critérios de avaliação mais desejáveis que favorecessem a maximização da *AUC*. Para tal, surge a hipótese de que um conjunto aproximado da Fronteira de Pareto de acordo com os critérios de sensitividade e especificidade possa contribuir para a maximização da *AUC*. A sensitividade é a precisão entre os positivos enquanto a especificidade é a precisão entre os negativos. A Fronteira procurada não é formada apenas com as melhores regras de acordo com a sensitividade **OU** especificidade, mas também, com as regras que possuem precisão média entre os positivos **E** entre os negativos, que, na nossa conjectura, são características importantes para a maximização da *AUC*.

A validação da hipótese pode ser feita com a existência de um algoritmo determinístico

que possa gerar a maior quantidade possível de regras para a formação do conjunto aproximado da Fronteira de Pareto (conjunto com todas as soluções não-dominadas). Uma vez comprovado quais são os critérios multiobjetivos que maximizam a *AUC* em pequenas bases de dados, é possível a criação de algoritmos que possam trabalhar com grandes bases de dados. Tais algoritmos podem utilizar metaheurísticas que otimizam o processo de criação e seleção das regras com a substituição da parte determinística. Nesta tese foram utilizadas duas metaheurísticas que têm obtido resultados interessantes em suas áreas. A primeira metaheurística, nuvem de partículas (KENNEDY; EBERHART, 1995), vem da área de computação natural que se baseia nos seres vivos para resolução de problemas de otimização; a segunda, *Greedy Randomized Adaptive Search Procedures* (FEO; RESENDE, 1995), vem da área de pesquisa operacional e tem apresentado resultados competitivos aos melhores algoritmos.

Nuvem de partículas é uma técnica de otimização estocástica baseada em manter a diversidade da população, inspirada pelo comportamento social de pássaros à procura de alimento ou no modelo de cardume de peixes (VESTERSTROEM; RIGET, 2002). Embora a nuvem de partículas já tenha sido utilizada em classificação (SOUSA; SILVA; NEVES, 2003) (SOUSA; SILVA; NEVES, 2004), poucos são os trabalhos que exploram as características multiobjetivas. Em sua dissertação de mestrado, Augusto (TORáCIO, 2008) criou o algoritmo *MOPSO*, um algoritmo de classificação multiobjetivo que apresentou bons resultados.

A metaheurística *Greedy Randomized Adaptive Search Procedures* (*GRASP*) é muito utilizada para resolução de problemas na área de otimização combinatória e que tem apresentado resultados significativos (FEO; RESENDE, 1995). *GRASP* é um processo iterativo com duas fases principais: uma fase de construção de soluções viáveis através de um procedimento guloso aleatório, e uma fase de busca por um mínimo local pertencente a uma dada vizinhança da solução construída na fase anterior. A fase de construção gulosa pode ser de grande valia para a geração das regras mais relevantes para a formação do conjunto aproximado da Fronteira.

1.1 Objetivos e Contribuições

O propósito desta tese é trabalhar com duas abordagens na área de classificação que lidam com critérios multiobjetivos para a geração de classificadores. A primeira abordagem buscou verificar a hipótese de que a combinação das técnicas de *boosting* e estratégia evolucionária podem atingir o equilíbrio entre precisão e generalidade. Para tal, uma extensão da linguagem de representação dos classificadores foi proposta. Essa nova representação exigiu uma nova medida para avaliar a qualidade dos classificadores em relação à precisão e a gener-

alidade. Foi criado um algoritmo para gerar os novos modelos de classificadores incorporando ideias de *boosting* de modo a considerar a aprendizagem com a utilização da dinâmica de pesos para cada exemplo. A dinâmica de pesos foi utilizada em conjunto com a *ES* para induzir um conjunto de classificadores num único processo. Os experimentos mostraram que durante a criação do conjunto a linguagem de representação conseguiu classificar muito bem os exemplos, porém a solução apresentou problemas de sobre-especialização (*overfitting*) quando testados em exemplos não vistos. Portanto, alguns caminhos a serem explorados são apontados para um futuro melhoramento.

A segunda abordagem investigou a hipótese de que a criação de um conjunto de regras não-ordenadas segundo critérios biobjetivos pode maximizar a *AUC*. Procurou-se identificar as medidas desejáveis para a criação de um conjunto de regras não-ordenadas que maximizem a *AUC*, esperando que os conjuntos encontrados fossem o mais próximo da Fronteira de Pareto. Baseado na hipótese de que os critérios de sensitividade e especificidade podem maximizar a *AUC*, três algoritmos multiobjetivos foram criados. O primeiro algoritmo, Pareto Front Elite (PFE) (ISHIDA; POZO, 2007b) (ISHIDA; POZO, 2007a), foi criado para validar a hipótese, comprovando que os critérios de sensitividade e especificidade encontram conjuntos com valores de *AUC* semelhantes aos melhores algoritmos existentes.

Validada a hipótese de que a sensitividade e especificidade podem maximizar a *AUC* e com o objetivo de lidar com grandes bases de dados, dois algoritmos metaheurísticos multiobjetivos foram criados. A ideia foi gerar num único passo as regras com as características mais favoráveis para a formação do conjunto de aproximação da fronteira com os critérios de sensitividade e especificidade. O primeiro algoritmo multiobjetivo, *MOPSO* (TORáCIO, 2008), utiliza a metaheurística nuvem de partículas (KENNEDY; EBERHART, 1995) com os mesmos objetivos do algoritmo Pareto Front Elite. Alguns experimentos foram realizados mostrando que o desempenho em termos de tempo computacional do algoritmo metaheurístico foi superior ao algoritmo determinístico e os resultados em termos de *AUC* foram satisfatórios, porém, com valores abaixo do algoritmo determinístico.

O segundo algoritmo metaheurístico multiobjetivo, *GRASP-PR Rule Learning*, utilizou o *GRASP* e o *path-relinking* para substituir a parte determinística (ISHIDA et al., 2008). Os experimentos mostram que o algoritmo obteve conjuntos de regras com ótimos valores da *AUC*. Outros experimentos também foram realizados e os resultados foram analisados através de metodologias multiobjetivas. A análise confirmou que na maioria das vezes os conjuntos de aproximação possuem características semelhantes às Fronteiras de Pareto.

1.2 Estrutura do Trabalho

Esta tese está organizada da seguinte maneira:

O capítulo 1 apresenta o tema e a descrição deste trabalho.

O capítulo 2 contextualiza o trabalho apresentando alguns conceitos iniciais da tarefa de classificação; após os conceitos, estão apresentados alguns trabalhos relacionados.

O capítulo 3 apresenta os conceitos de um problema multiobjetivo. São definidos: o problema multiobjetivo, Pareto-ótimo e conceitos de dominância. O capítulo é finalizado com a apresentação da metodologia de avaliação de desempenho para algoritmos multiobjetivos.

No capítulo 4, inicialmente, são apresentadas duas técnicas que serão combinadas: *boosting* e estratégia evolucionária. Ainda são discutidos: a criação de uma nova representação para a obtenção de conjunto de classificadores e um novo critério para a avaliação da precisão e generalidade. Na seção 4.3.7 é apresentado o algoritmo *boosting* com estratégia evolucionária (BES) para a obtenção do conjunto de classificadores com a nova representação e alguns experimentos realizados.

Os três algoritmos multiobjetivos estão no capítulo 5. Neste capítulo, discute-se a seleção de regras não-ordenadas com o objetivo de maximização da *AUC*. Primeiramente, é apresentado o algoritmo clássico, Apriori, para a geração de regras. Depois, o algoritmo Pareto Front Elite criado para a seleção de regras por critérios biobjetivos (sensitividade e especificidade), além de alguns experimentos de validação.

A seção 5.2 apresenta um dos algoritmos criados com uma metaheurística para a criação do conjunto não-ordenado de regras. Nesta seção apresenta-se o algoritmo básico da nuvem de partículas (*PSO*), seguido por uma explicação de como a nuvem de partículas pode ser utilizada para resolução de problemas multiobjetivos. Em 5.2.3, apresenta-se o algoritmo *MOPSO* criado para o aprendizado de regras. Na subseção 5.2.4 alguns experimentos conduzidos para a avaliação do *MOPSO* estão apresentados e analisados.

A seção 5.3 apresenta a utilização da metaheurística *GRASP* para a maximização da *AUC* seguindo os mesmos critérios biobjetivos do algoritmo PFE. Foram feitas algumas comparações com o algoritmo criado (*GRASP-PR Rule Learning* ou *GPR-RL*). A comparação com o algoritmo Pareto Front Elite avaliou a *AUC* e o número de regras gerado. O capítulo é finalizado com a comparação dos conjuntos de aproximação obtidos através da metodologia de avaliação de desempenho multiobjetivo. Os conjuntos do *GPR-RL* foram comparados com os resultados do PFE e *MOPSO*.

O capítulo 6 finaliza a tese com a avaliação das contribuições e sugestões de trabalhos futuros.

2 Classificação

Com muitas aplicações reais, a mineração de dados tem contribuído para a competitividade e sobrevivência das empresas (ISHIDA C. Y.; CAVALHEIRO, 1999). O termo mineração de dados ou *Data Mining* (*DM*) é utilizado para denotar a descoberta de padrões úteis em dados. *DM* consiste na aplicação de análise de dados e algoritmos de descoberta para produzir padrões ou modelos que sejam informações novas, úteis e interessantes (WONG; LEUNG, 2000). *DM* responde a questão de como melhor utilizar os dados históricos de um banco de dados para a descoberta de padrões gerais e para melhorar as decisões futuras (MITCHELL, 1999).

Segundo Mitchell (1997), o campo de aprendizado de máquina ou *Machine Learning* (*ML*) refere-se à construção de programas de computadores que melhoram automaticamente com a experiência. Um algoritmo de *ML* envolve a descoberta de um conceito através de exemplos e conhecimento prévio do domínio (MITCHELL, 1997).

Os algoritmos de *ML* têm provado o seu grande valor prático em diversas áreas. Eles são especialmente úteis em problemas de *DM* onde os grandes bancos de dados contêm padrões implícitos que podem ser descobertos automaticamente. Algumas observações são relevantes para compreender um pouco melhor *ML* e *DM*: os algoritmos de *ML* tentam aprender utilizando pequenos volumes de dados; em contraste a esta perspectiva, *DM* tem como objetivo trabalhar com grandes volumes de dados. Desta forma é comum realizar o processo de aprendizado considerando pelo menos duas etapas. Na primeira etapa os modelos são induzidos em diferentes conjuntos de dados utilizando *ML*. A segunda etapa combina estes modelos numa abordagem integrada. O *ML* enfoca a primeira etapa que é a indução de classificadores em pequenos volumes de dados (FREITAS; LAVINGTON, 1998).

Existem diversas estratégias que podem ser utilizadas por um sistema computacional para realizar o aprendizado e melhoria a partir de observações. Algumas destas estratégias são: aprendizado por hábito, aprendizado por analogia e aprendizado indutivo. A estratégia de aprendizado por hábito é a mais simples e inclui o aprendizado por memorização direta de descrições de um dado conceito. Já na estratégia de aprendizado por analogia e indução, o

aprendiz necessita dispender maior esforço para realizar o aprendizado (MITCHELL, 1997). A indução é uma forma de inferência que permite obter conclusões genéricas partindo de um conjunto particular de exemplos ou casos previamente observados. É caracterizada como o raciocínio que parte do específico para o geral, do particular para o universal, da parte para o todo. O aprendizado por indução é o objetivo de um paradigma do *ML* chamado aprendizado indutivo. Em outras palavras, a partir de um conjunto de exemplos com a informação de suas classes, a tarefa do aprendizado indutivo forma conceitos que descrevem relacionamentos entre os dados e prediz a classificação futura de instâncias que não foram classificadas. O aprendizado indutivo pode ser dividido em aprendizado supervisionado e aprendizado não-supervisionado, descritos a seguir.

No aprendizado não-supervisionado é fornecido ao sistema de aprendizado um conjunto de exemplos, no qual cada exemplo consiste somente de vetores, não incluindo a informação sobre a classe. O objetivo é construir um modelo que procure por padrões nos exemplos, formando agrupamentos (*clusters*) de exemplos com características similares.

O aprendizado supervisionado trabalha com conjuntos de dados cujos exemplos estão pré-classificados, i.e., estão rotulados como o atributo classe. O objetivo do aprendizado supervisionado é construir um modelo preditivo, que consiste em rotular novos exemplos que não possuem o atributo classe associado. Este tipo de aprendizado é descrito a seguir.

2.1 Aprendizado Supervisionado

No aprendizado supervisionado (*Supervised Learning*), um conjunto de exemplos é fornecido ao sistema de aprendizado $\xi = \{\varepsilon_1, \varepsilon_2, ..., \varepsilon_m\}$ (conjunto 2.1). Pode-se dizer que o exemplo ε_k é representado por uma tupla $(\mathbf{x_k}, y_k)$ (conjunto 2.1), onde $\mathbf{x_k}$ é um vetor de valores que representam os atributos do exemplo e y_k é o valor da classe desse exemplo. Os vetores $\mathbf{x_k}$ pertencem a um conjunto de vetores X o qual denominamos domínio, e os valores y_k fazem parte do conjunto de classes Y. Se cada exemplo tem n atributos então os vetores $\mathbf{x_k}$ têm dimensão n.

$$\xi = \{\varepsilon_k = (\mathbf{x_k}, y_k), \mathbf{x_k} \in X, y_k \in Y, k = 1, ..., m\} \tag{2.1}$$

Note que os elementos do domínio X podem ter várias representações. Por exemplo, os vetores $\mathbf{x_k}$ podem corresponder a pontos em um espaço n-dimensional e, mais especificamente, podem ser formados por valores reais (vetores em R^n). Estes valores representam as características gerais. Por exemplo, uma sequência de proteínas pode utilizar um vetor de valores

reais de dimensão 20 ($\Re^{20}$), para indicar a composição dos aminoácidos.

Neste tipo de aprendizado, o objetivo é induzir um mapeamento geral dos vetores $\mathbf{x}$ para os valores y, definido por $y_k = f(\mathbf{x_k})$. Portanto, o sistema de aprendizado deve construir um modelo de uma função desconhecida, f, que permita predizer valores y para exemplos previamente não vistos. Entretanto, o número de exemplos utilizados para a criação do modelo não é, na maioria dos casos, suficiente para caracterizar completamente esta função $f : X \rightarrow Y$. Na realidade, os sistemas de aprendizado são capazes de induzir uma função h que aproxima f, ou seja, $h(x) \approx f(x)$. Neste caso, h é chamada de hipótese sobre a função conceito f.

O atributo classe $y_k \in Y$ (conjunto 2.1) pode ser um atributo qualitativo que assume um conjunto de valores discretos, quando $Y = \{Y_1, Y_2, ..., Y_{nc}\}$, ou um atributo quantitativo que assume um conjunto de valores reais. No primeiro caso, o objetivo do aprendizado é encontrar uma função h que aproxima a função $f : X \rightarrow Y$. Neste caso, a hipótese h pode ser denominada classificador e a tarefa de aprendizado é denotada classificação. No caso do atributo classe quantitativo, o objetivo do aprendizado é encontrar uma função h que aproxima a função $f : X \rightarrow \Re$. Neste caso, a hipótese h é denominada regressor e a tarefa de aprendizado é denotada regressão.

É muito comum tratar os problemas de classificação com apenas duas classes, uma vez que os problemas envolvendo várias classes podem ser transformados em problemas binários. Escolhe-se a classe com o menor número de exemplos para a classe positiva e as outras classes são consideradas partes da classe negativa. Com apenas duas classes, as escolhas são estruturadas para predizer a ocorrência e a não ocorrência de um simples evento ou hipótese. As duas classes podem ter qualquer definição e usualmente são chamadas de positivo (+1) e negativo (-1).

Embora este trabalho aborde problemas com duas classes, outros problemas com três ou mais classes podem ser facilmente transformados em problemas com duas classes. Nestes casos, considera-se uma classe como positiva e todas outras classes formam a classe negativa.

2.2 Linguagem de Representação

Os classificadores podem ser apresentados de diversas formas. Algumas linguagens de representação podem ser (PRODROMIDIS; CHAN, 2000): árvores de decisão, regras, funções lineares, funções não-lineares, distribuição de probabilidades, etc. A seguir, um exemplo é apresentado e será utilizado para a definição das seguintes linguagens representações: função linear e regras.

Tabela 2.1: Banco de dados com informações dos clientes e livro de mangá japonês. São apresentados, respectivamente, o número do registro, o gênero do comprador, sua idade e a informação se o cliente comprou o livro ou não.

#	Gênero	Idade	Compra (objetivo)
1	Masculino	25	Sim
2	Masculino	21	Sim
3	Feminino	23	Sim
4	Feminino	24	Sim
5	Feminino	30	Não
6	Masculino	27	Não
7	Masculino	29	Não
8	Feminino	18	Não
9	Feminino	34	Não
10	Masculino	55	Não

Como exemplo da tarefa de classificação, pode-se citar o problema de predizer quais os consumidores que comprariam um livro sobre mangá japonês (história em quadrinhos). Suponha que uma editora possua em seu banco de dados informações de seus consumidores como sexo, idade e se o consumidor comprou ou não o livro em questão. Na Tabela 2.1 encontram-se 10 registros/tuplas de clientes diferentes, um em cada linha.

O objetivo do problema é obter informações sobre o atributo 'Compra' que pode ser chamado de atributo objetivo e os seus valores chamados de classes. As classes deste exemplo são duas: 'Sim' e 'Não'. A classe 'Sim' é a classe positiva (+1) que indica a ocorrência da compra e a classe 'Não' a classe negativa (-1). As demais informações (Gênero, Idade) necessárias para predizer sobre possíveis consumidores são chamadas de atributos de predição.

A classificação consiste em determinar quais valores dos atributos de predição estão associados a cada um dos valores do atributo objetivo. O resultado é um classificador que pode ser utilizado para predizer se qualquer novo comprador adquirirá ou não o livro. Na predição, as informações deste comprador não estão no banco de dados da companhia; consequentemente, o valor do atributo objetivo é desconhecido.

2.2.1 Classificador como Função Linear

Classificador pode ser expresso como uma função que classifica um item em uma das diversas classes pré-definidas (HAND, 1981), (WEISS; KULIKOWSKI, 1991). Assim, os sistemas classificadores têm como objetivo encontrar a melhor hipótese ou o melhor conjunto de hipóteses que classifique um conjunto de exemplos (itens).

Seja o conjunto dos exemplos definido segundo 2.1, onde $\mathbf{x_k}$ é um vetor de valores que

representam as características ou atributos do exemplo ε_k, e y_k é a classe do exemplo ε_k. Os vetores $\mathbf{x_k}$ pertencem ao domínio do problema. Lembrando que -1 refere-se à classe negativa e $+1$ refere-se à classe positiva.

Considerando um caso mais específico, onde o espaço considerado é o dos números reais, o classificador é uma função linear real ($f : X \rightarrow \Re$) com o conjunto $X \subset \Re^n$. Uma hipótese i classifica um exemplo através da função (2.3) e o objetivo da tarefa é aprender os pesos ($\mathbf{w_i}$ da função (2.2)). Para cada exemplo $\mathbf{x_k}$ é calculado um valor real ($g_i(\mathbf{x_k})$) através da combinação linear dos vetores $\mathbf{w_i}$ e $\mathbf{x_k}$ (indicado por $< \mathbf{w_i}, \mathbf{x_k} >$ na função (2.2)). Se este valor estiver acima do limiar (*threshold*) $t \in \Re$ o exemplo é classificado como positivo, pode-se dizer que a hipótese associa o exemplo à classe positiva. Caso contrário, o exemplo será classificado como negativo.

$$g_i(\mathbf{x_k}) =< \mathbf{w_i}, \mathbf{x_k} >= \sum_{l=1}^{n} w_{i.l} x_{k.l}, \forall \mathbf{w_i} \in \Re^n, \forall \mathbf{x_k} \in \Re^n, \mathbf{w_i} = (w_{i.1}, ..., w_{i.n}), \mathbf{x_k} = (x_{k.1}, ..., x_{k.n}) \tag{2.2}$$

$$h_i(\mathbf{x_k}) = \begin{cases} +1, & g_i(\mathbf{x_k}) > t \\ -1, & \textit{caso contrário} \end{cases} \tag{2.3}$$

No exemplo do livro sobre mangá japonês, é preciso que todos os atributos do vetor dos exemplos sejam numéricos. Para tal, para os dois valores possíveis do atributo 'Gênero': 'Feminino' e 'Masculino' associam-se, respectivamente, os valores 1 e 2. A Tabela 2.2 contém os vetores para cada registro da base de dados. Os vetores dos atributos ($\mathbf{x_i}$) são a entrada para um algoritmo de classificação que tem como objetivo buscar um modelo em forma de função linear que classifique corretamente os dados. Mais especificamente, procura-se pelo vetor de pesos $\mathbf{w}$ da função 2.2.

Um exemplo do resultado obtido pode ser o vetor de pesos $\mathbf{w} = (1,1)$ e o limiar $t = 28$. As funções utilizadas para a classificação são representadas pelas funções 2.4 e 2.5.

$$g_{fl}(\mathbf{x_k}) =< w, \mathbf{x_k} >, w = (1,1) \tag{2.4}$$

$$h_{fl}(\mathbf{x_k}) = \begin{cases} +1, & g_{fl}(\mathbf{x_k}) < 28 \\ -1, & \textit{caso contrário} \end{cases} \tag{2.5}$$

A hipótese e os exemplos podem ter a sua representação num plano cartesiano con-

Tabela 2.2: Vetor de cada exemplo da base de dados de compra do livro sobre mangá japonês (Tabela 2.1).

$\#i$	$x_i = (x_{i.1}, x_{i.2})$	y_i
1	$x_i = (2,25)$	+1
2	$x_2 = (2,21)$	+1
3	$x_3 = (1,23)$	+1
4	$x_4 = (1,24)$	+1
5	$x_5 = (1,30)$	-1
6	$x_6 = (2,27)$	-1
7	$x_7 = (2,29)$	-1
8	$x_8 = (1,18)$	-1
9	$x_9 = (1,34)$	-1
10	$x_{10} = (2,55)$	-1

forme a Figura 2.1. Na Figura, '+' representa um exemplo positivo e 'x' representa um exemplo negativo. A representação geométrica da hipótese é um hiperplano que divide o espaço em dois semi-espaços: um semi-espaço positivo ($h(x) = +1$) e outro negativo ($h(x) = -1$). A função do hiperplano é obtida igualando-se:

$$\begin{aligned} g_{fl}(\mathbf{x_k}) &= t \\ < w, \mathbf{x_k} > &= t \\ w_1 * x_1 + w_2 * x_2 &= t \\ 1 * x_1 + 1 * x_2 &= t \\ x_2 &= t - x_1 \\ y &= t - x \end{aligned}$$

Na Figura 2.1, a função do hiperplano ($y = t - x$) separa corretamente quase todos os exemplos dados, os positivos estão abaixo da reta e os negativos no grupo oposto. A exceção é um cliente negativo (#8) que foi classificado como positivo.

2.2.2 Classificador Expresso com Regras

O classificador também pode ser expresso como um conjunto de regras. As regras são usualmente utilizadas para expressar conhecimento e são facilmente compreendidas. Sua utilização é comum em sistemas especialistas em apoio à decisão (WONG; LEUNG, 2000).

As regras são frequentemente representadas na forma de "IF-THEN" (em português "SE-ENTÃO") e o formato padrão está representado na Figura 2.2. Os atributos de predição (*predicting attributes*) formam a parte antecedente da regra, ou seja, o que se encontra entre o

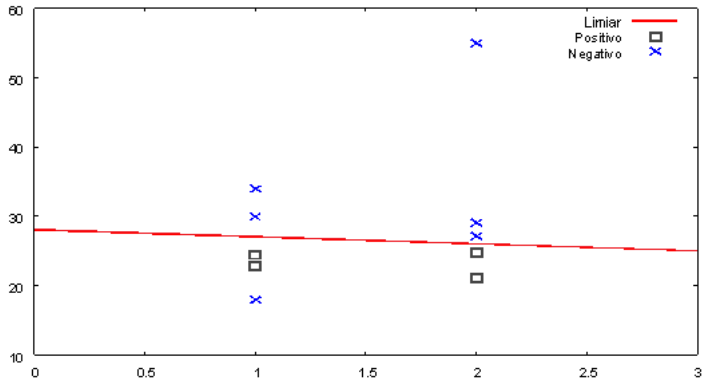


Figura 2.1: Exemplo de Hipótese Linear na Classificação de Exemplos

IF <antecedente> THEN <conseqüente>

Figura 2.2: Classificador em formato de regra.

string ‘IF’ e ‘THEN’; o atributo objetivo e a classe formam a parte consequente da regra, após o ‘THEN’. As regras são interpretadas como: “IF os atributos de predição da tupla satisfazem as condições do antecedente da regra, THEN a tupla tem a classe indicada no consequente da regra”. Em outras palavras, se todos os exemplos com atributos que satisfizerem a condição do antecedente são classificados para a classe da regra, pode-se dizer que estes exemplos são cobertos pela regra.

Existem diversos formatos de um atributo de predição. Diferentemente do classificador linear, o atributo pode ser descrito juntamente com um valor nominal ou um valor de um intervalo contínuo.

A partir dos dados do exemplo do livro sobre mangá japonês, Tabela 2.1, o algoritmo de classificação poderia extrair as três regras da Figura 2.3.

A primeira regra da Figura 2.3: “IF (Idade$>$ 18 and Idade$<=$ 25) THEN (Compra = “Sim”)”pode ser interpretada como: “SE a idade de cliente estiver entre 19 e 25 anos ENTÃO o cliente irá comprar”. Em outras palavras, os compradores em potencial do livro são os jovens adultos. Verificando a Tabela 2.1, esta regra classifica todos os exemplos positivos (1 a 4) da base corretamente.

```
IF (Idade>18 and Idade<=25) THEN (Compra = "Sim")
IF (Gênero="Masculino" and Idade>=27) THEN (Compra = "Não")
IF (Gênero="Feminino" and Idade>24) THEN (Compra = "Não")
```

Figura 2.3: Regras extraídas a partir da Tabela 2.1.

A segunda regra da Figura 2.3 pode ser interpretada como: “SE o cliente for homem com 27 anos ou mais ENTÃO o cliente não irá comprar”. Esta regra classifica os exemplos 6, 7 e 10 como negativos corretamente.

A terceira regra pode ser interpretada como: “SE o cliente for mulher com mais de 24 anos, ENTÃO não irá comprar”. Esta regra classifica os exemplos 5 e 9 como negativos corretamente. Note que a regra não cobre o exemplo negativo 8 que, apesar de ser mulher, possui menos de 24 anos.

2.3 Medidas de Classificação

Diversos critérios são utilizados para a avaliação da qualidade das hipóteses encontradas. A seguir, descreveremos algumas medidas importantes.

2.3.1 Precisão e Erro

A medida de precisão pode ser definida pela distância entre o valor predito e o valor real se for disponível (CORNUEJOLS; MICLET, 2003). Para calcular a medida de precisão de uma hipótese, primeiramente todos os exemplos são associados a uma classe, e esta classe predita é comparada com a classe real de cada exemplo. A divisão do valor total de exemplos classificados corretamente pelo número total de exemplos é a precisão (*accuracy*), conforme indica a definição 2.6.

$$precisão = \frac{\#classificacão\ correta}{\#exemplos} \tag{2.6}$$

Outro critério essencial para avaliação dos classificadores é o erro de classificação. Um erro ocorre quando o classificador classifica incorretamente um exemplo, ou seja, associa a uma classe que não é a sua classe padrão. Após a classificação de todos os exemplos, a taxa de erro é calculada pela divisão do total de erros pelo total de exemplos existentes, conforme a equação 2.7. Note que a precisão e a taxa de erro são complementares (2.8), ou seja, a precisão mais o erro é igual a 1 (representando 100%).

$$erro = \frac{\#classificacão\ incorreta}{\#exemplos} \tag{2.7}$$

$$precisão = 1 - erro \tag{2.8}$$

Tabela 2.3: Matriz de Confusão para Problemas Binários

	Classe Positiva (+)	Classe Negativa (-)	Total
Predição Positiva (+)	Verdadeiro Positivo (VP)	Falso Positivo (FP)	VP + FP = PP
Predição Negativa (-)	Falso Negativo (FN)	Verdadeiro Negativo (VN)	FN + VN = PN
Total	CP = VP + FN	CN =FP + VN	m=PP+PN=CP+CN

A taxa de erro sumariza o desempenho geral do classificador sem considerar os tipos de erros. Os erros podem ser separados em tipos de acordo com a classe original do exemplo. Estes tipos são considerados pois em problemas reais existe um custo diferente de classificar com erro os exemplos de uma classe específica. Por exemplo, o erro cometido em diagnosticar uma pessoa que tem uma doença como sendo uma pessoa saudável (conhecido como decisão falsa negativa) é considerado um erro muito mais sério do que o tipo de erro oposto que é diagnosticar alguém como doente quando, de fato, está saudável (conhecido como falso positivo). Devido à existência dos tipos de erros e da necessidade de se obter informações mais detalhadas existem outras medidas de precisão que podem ser descritas a partir da matriz de confusão.

2.3.2 Medidas a partir da Matriz de Confusão

A matriz de confusão mostra o número de classificações corretas em contraste com as classificações preditas para cada classe, enfatizando justamente os tipos de erro. A Tabela 2.3 ilustra a matriz de confusão para problemas de classificação para duas classes onde: VP (Verdadeiro Positivo) é o número de exemplos positivos classificados corretamente; FN (Falso Negativo) é o número de exemplos positivos classificados incorretamente como negativos; FP (Falso Positivo) = total de exemplos negativos classificados como positivos; VN (Verdadeiro Negativo) = número de negativos classificados corretamente; CP = total de exemplos da classe positiva; CN = total de exemplos da classe negativa; PP = total de exemplos preditos como positivos; PN = total de exemplos preditos como negativos e m é o total de exemplos.

Utilizando a nomenclatura da Tabela 2.3 tem-se uma nova representação para a precisão (equação 2.6) e o erro (equação 2.7), conforme as equações 2.9 e 2.10, respectivamente.

Para contornar os problemas da precisão, duas métricas para avaliação são comuns: sensitividade (*sensitivity*) e especificidade (*specificity*). Utilizando a Tabela 2.3, a sensitividade, ou completeza, ou taxa dos positivos corretamente classificados, indica a precisão entre as instâncias positivas; esta taxa é indicada pela equação 2.11. A especificidade ou taxa dos negativos verdadeiros indica a precisão entre os exemplos negativos, e é mostrada pela equação 2.12. A taxa de falsos positivos ou taxa de falso alarme é o complemento da especificidade (ou

seja, falso alarme=1-especificidade) descrita na equação 2.13.

$$precisão = \frac{VN+VP}{VN+VP+FN+FP} = \frac{VN+VP}{PP+PN} = \frac{VN+VP}{m} \tag{2.9}$$

$$erro = \frac{FN+FP}{VN+VP+FN+FP} = \frac{FN+FP}{PP+PN} = \frac{FN+FP}{m} \tag{2.10}$$

$$sensitividade = \frac{VP}{VP+FN} = \frac{VP}{CP} \tag{2.11}$$

$$especificidade = \frac{VN}{VN+FP} = \frac{VN}{CN} \tag{2.12}$$

$$falso\ alarme = \frac{FP}{VN+FP} = \frac{FP}{CN} \tag{2.13}$$

Nos sistemas de recuperação de informação, sistemas que organizam e recuperam informações não-estruturadas, as medidas de desempenho mais utilizadas são valor preditivo positivo (*recall*) (equação 2.14) e valor preditivo negativo (precision) (2.15). Estas medidas são indicadas para situações em que o VP é pequeno quando comparado com VN.

$$valor\ preditivo\ positivo = \frac{VP}{VP+FP} = \frac{VP}{PP} \tag{2.14}$$

$$valor\ preditivo\ negativo = \frac{VN}{VN+FN} = \frac{VN}{PN} \tag{2.15}$$

2.3.3 Area Under the Curve (*AUC*)

Como já mencionado, existem alguns problemas quando se utiliza apenas a precisão. O custo diferenciado para cada tipo de erro na vida real e o fato da precisão apresentar deficiências nos casos das distribuições de dados altamente desbalanceadas (por exemplo, quando existe 1% de positivos e 99% de negativos) são aspectos que devem ser considerados.

Existem duas medidas que caracterizam a hipótese: a taxa dos falsos positivos e a taxa dos falsos negativos. Como uma forma de expressar a qualidade de uma hipótese, surge a representação 2D conhecida como curva *ROC* (*Receiver Operating Characteristics*), originada do processamento de sinais e depois adotada na análise de dados médicos. Na área de *ML*, um

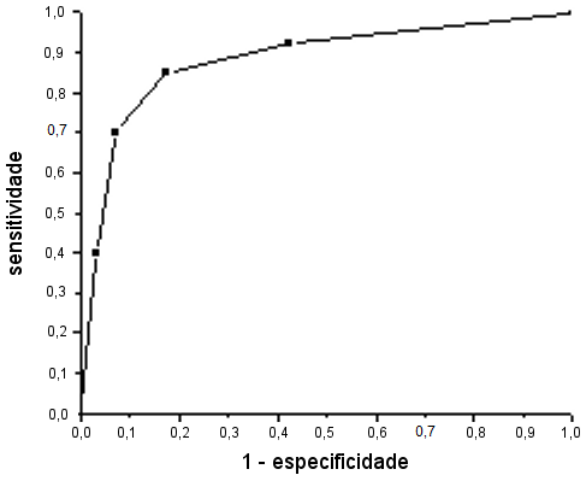


Figura 2.4: Exemplo de Curva *ROC*

dos primeiros trabalhos que utilizou a curva *ROC* para avaliação e comparação de algoritmos foi SPACKMAN (SPACKMAN, 1989). Recentemente esta medida começou a ser utilizada na indução de classificadores (FAWCETT, 2001).

As curvas *ROC* podem ser utilizadas para a avaliação do desempenho de algoritmos porque representam o compromisso entre a sensitividade no eixo y e falso alarme (1 - especificidade) no eixo x. A Figura 2.4 mostra um exemplo de alguns pontos formando a curva *ROC*.

Um ponto em um gráfico *ROC* domina outro se esse ponto estiver acima e a esquerda do outro, isto é, tem uma taxa mais alta de verdadeiros positivos e uma taxa mais baixa de falsos positivos. Um modelo de classificação h_1 domina outro modelo h_2 se todos os pontos da curva de h_2 estão abaixo da curva h_1. Caso isso não ocorra, o melhor modelo para uma faixa de valores pode ser derivado utilizando-se o método envoltório convexo (*convex hull*) (PROVOST; FAWCETT, 1997). Uma situação 'perfeita' ocorre quando a curva atinge o ponto $(0,1)$. E, supondo que uma curva seja formada por um classificador aleatório (que classifica os exemplos corretamente em apenas 50% dos casos), neste caso a curva é uma reta que passa pelos pontos $(0,0)$ e $(1,1)$.

Para induzir e comparar classificadores é preciso reduzir o desempenho *ROC* a um valor singular representando o desempenho esperado (SEBAG; AZE; LUCAS, 2003). Um método comum é calcular a área formada sob a curva *ROC* (*Area Under the Curve ROC*, *AUC*) (BRADLEY, 1997). Como a *AUC* é uma porção de uma área de uma unidade quadrada, o seu valor situa-se em 0 (zero) e 1 (um). O classificador 'perfeito' tem área igual a 1 (com a curva passando pelo ponto $(0,1)$) e o classificador aleatório tem área 0.5, portanto, não existe utilidade para o classificador com *AUC* menor do que 0.5. Para ilustrar a comparação de dois

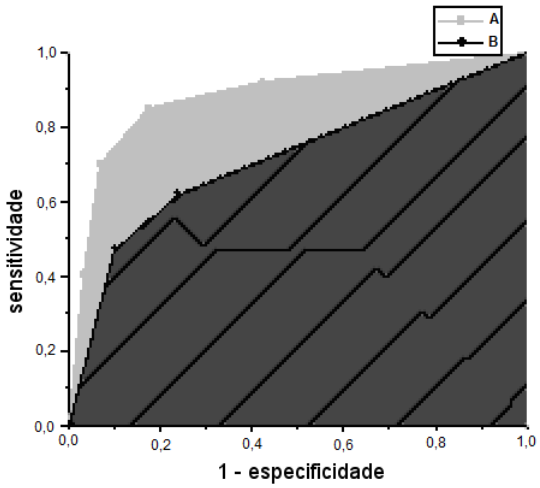


Figura 2.5: Exemplo de Comparação *AUC*

algoritmos, a Figura 2.5 mostra áreas sobre duas curvas *ROC*, curva *A* e *B*. O classificador *A* possui área maior e, portanto, desempenho médio melhor.

Um conjunto de regras pode ser utilizado como um simples classificador e, assim, obter pontos da curva *ROC*. Pode-se calcular o valor de *AUC* utilizando o processo de classificação por votação por pesos, baseado na confiança (FAWCETT, 2001). Neste processo, o conjunto de regras vota em cada exemplo. Cada regra utiliza sua confiança para associar um valor para os exemplos que o cobrem. A confiança de uma regra $A \rightarrow C$ indica o número de vezes que, quando A ocorre, C também ocorre. Uma regra positiva associa sua confiança aos exemplos com sinal positivo, enquanto as regras negativas associam aos exemplos com valor negativo. Depois que todas as regras votam, cada exemplo possui um valor associado que é a soma da confiança das regras positivas menos o somatório das regras negativas que o cobrem. Então, para cada instância de base de dados existe um *ranking* numérico com valores associados. Este *ranking* pode ser comparado com um limiar (*threshold*) para produzir um classificador binário. Se o *ranking* de uma determinada instância estiver acima do limiar, então o classificador produz um "sim", caso contrário, produz um "não". Cada valor de limiar gera um ponto diferente no plano *ROC*. Então, a variação do limiar de $-\infty$ para $+\infty$ produz uma curva no plano *ROC*, sendo possível calcular a área abaixo da curva (*AUC*) (PROVOST; FAWCETT, 2001). Este valor dá uma boa visualização do desempenho do classificador (BRADLEY, 1997). Numa classificação perfeita, os exemplos positivos têm os maiores valores agrupados no topo do *ranking* e, consequentemente, o classificador possui o maior valor de *AUC* (igual a um).

Para mais detalhes sobre a curva *ROC* e *AUC* aconselha-se a leitura de Fawcett (2003).

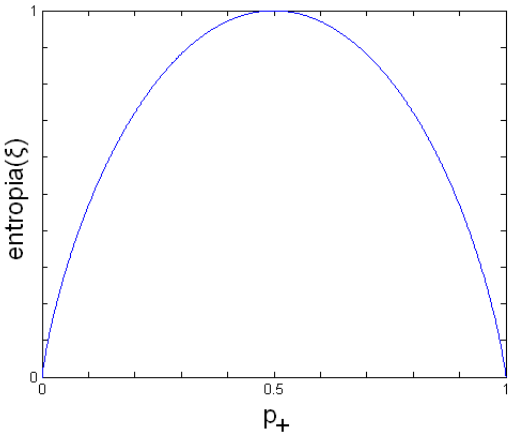


Figura 2.6: Valores Possíveis para Entropia

2.3.4 Outras medidas

Outra medida que pode ser citada é a entropia. Segundo Mitchell (1997), entropia é uma medida comumente utilizada em teoria da informação que expressa a pureza ou impureza de um conjunto arbitrário de exemplos, e também é chamada de quantidade de informação (*Information Quantity* - *IQ*). Dada uma coleção de exemplos ξ, contendo positivos e negativos, a entropia de ξ relativa a esta classificação binária é dada pela equação 2.16, onde p_+ é a proporção de exemplos positivos em ξ e p_- é a proporção de exemplos negativos em ξ, e ambos podem assumir valores reais entre 0 e 1. Em todos os cálculos considera-se que $0 * \log_2 0 = 0$.

$$entropia(\xi) = -p_+ \log_2 p_+ - p_- \log_2 p_- \tag{2.16}$$

A Figura 2.6 mostra os valores da entropia ($entropia(\xi) \in [0,1]$) de acordo com os valores de p_+. Note que a entropia é zero se todos os membros de ξ pertencerem à mesma classe, pois se todos forem exemplos positivos, então $p_+ = 1$, $p_- = 0$ e, consequentemente, o valor da entropia é igual a 0 (zero). O mesmo acontece quando todos os exemplos são negativos ($p_+ = 0$ e $p_- = 1$). Outra observação: quando o número de exemplos positivos é igual ao número de negativos, a entropia é igual a 1 (um). Se os números de exemplos não são iguais para as duas classes, o valor está entre 0 e 1.

Além da entropia, existem muitas outras medidas, tais como medidas de avaliação de regras (LAVRAC; FLACH; ZUPAN, 1999):

- Confiança Negativa é o correspondente à precisão, para os exemplos que não são cobertos por uma regra.

- Cobertura é uma medida do número (relativo) de exemplos cobertos pela regra.
- Suporte ou frequência é uma medida do número (relativo) de exemplos cobertos corretamente pela regra.
- Confiança negativa relativa é análoga à precisão relativa para os exemplos que não são cobertos pela regra.
- Precisão Ponderada Relativa (*Weighted Relative Accuracy*) indica a importância da regra em termos da diferença entre o número de verdadeiros positivos esperados com os observados ($\frac{PP}{m} * (\frac{VP}{PP} - \frac{CP}{m})$, ver Tabela 2.3).

2.4 Trabalhos relacionados

Nos últimos anos foi observada a proposição de vários trabalhos na área de classificação com objetivo de encontrar conjuntos de regras. Estes trabalhos têm destacado a grande importância de critérios diferentes ou complementares à precisão.

Para tal, alguns trabalhos utilizam a indução de regras de classificação e têm sido apresentados para a otimização da *AUC*. Um destes trabalhos, o algoritmo ROCCER (PRATI; FLACH, 2005) constrói uma envoltória convexa (*convex hull*) no espaço *ROC*. As regras são encontradas por um algoritmo de associação de regras e incorporadas ao ROCCER de acordo com sua contribuição à *AUC* e formam um conjunto de regras ordenadas. Porém, a inclusão de uma regra leva a uma reavaliação da contribuição das regras, o que causa um grande problema: a complexidade $O(n^2)$ (onde n é o número de regras geradas pelo algoritmo de associação de regras), no pior caso. Nos experimentos realizados para sua avaliação, o ROCCER obtém um conjunto de regras com valores *AUC* compatíveis com os melhores preditores probabilísticos tais como árvore de decisão de poda e, também, encontra poucas regras como resultado.

Alguns autores propõem a utilização do nível de confiança para seleção do conjunto de regras (GAMBERGER; LAVRAC, 2000). Nesta estratégia, os exemplos não cobertos aparecem como não classificados. Ferri, Flach e Hernandez-Orallo (2004) estendem a ideia criando novos classificadores para estes exemplos.

Trabalhos como Li, Han e Pei (2001), Liu, Hsu e Ma (1998), Yin e Han (2003), Prati e Flach (2005) e Batista et al. (2006) utilizam o algoritmo de regras de associação para gerar as regras. Devido ao grande número de regras geradas por estes algoritmos, Jovanoski e Lavrac (2001) apresentam o AprioriC onde as regras redundantes são removidas. Apesar das modificações, AprioriC ainda gera um grande conjunto de regras.

Uma das estratégias para a geração de regras é a extração a partir de árvore de decisão. O C4.5 (QUINLAN, 1993) utiliza o ganho de informação como uma medida para construir árvores de decisão e um passo de poda baseado na redução de erro. Além da versão com poda (C4.5) existe a versão sem a poda (C4.5NP). Na primeira versão do CN2 (CLARK; NIBLETT, 1989), uma lista de decisão é induzida usando a entropia como a heurística de busca. O algoritmo CN2 utiliza um mecanismo de votação para decidir qual a classificação final do exemplo (CLARK; NIBLETT, 1989). Ripper é um algoritmo de aprendizado de lista de decisão (COHEN, 1995). Tem características de poda baseado no erro e uma heurística baseada no princípio de MDL (Minimum Description Length) (LI; VITáNYI, 1997) para determinar quantas regras devem ser aprendidas. O Slipper (COHEN; SINGER, 1999) melhora o algoritmo Ripper, uma vez que utiliza a técnica de *weighted set-covering* que atribui peso a cada regra e o aumenta se classificar o exemplo corretamente.

Um algoritmo chamado ROGER (*ROC-based Genetic Learning*, algoritmo de aprendizado evolutivo baseado na curva *ROC*) foi criado para otimização do critério *AUC* (SEBAG; AZE; LUCAS, 2003). O trabalho é parte da solução de um problema prático da área médica com resultados significativos, segundo especialistas, para o diagnóstico da Arteriosclerose (SEBAG; AZÉ; LUCAS, 2003). O ROGER é baseado na Estratégia Evolucionária (BÄCK; SCHWEFEL, 1993). Assim como outros algoritmos evolucionários, a estratégia evolucionária é um algoritmo de busca estocástico baseado na abstração do processo de evolução de Darwin.

Recentemente houve o aumento do interesse em aplicar o conceito de Pareto-ótimo na área de aprendizado de máquina inspirado no sucesso da otimização evolucionária multiobjetiva. Estes trabalhos incluem seleção de atributos multiobjetivo, seleção de modelos multiobjetivo em treinamento de *perceptrons* multicamada, redes *radial-basis-function*, *support vector machines*, árvores de decisão e sistemas inteligentes (JIN, 2006). Contudo, poucos trabalhos lidam com algoritmos evolucionários multiobjetivos para aprendizado de regras. Algoritmos evolucionários mono-objetivos para aprendizado de regras podem resolver problemas lineares com certa facilidade. Já os algoritmos multiobjetivos possuem a característica de serem mais propícios para a resolução de problemas não-lineares.

A seguir, a apresentação dos conceitos de problemas multiobjetivos e alguns métodos para sua avaliação. Estes conceitos serão utilizados para a definição e avaliação das nossas propostas.

3 Problema Multiobjetivo

Os algoritmos de classificação podem encontrar conjuntos de modelos que maximizam o critério *AUC*. Se os resultados são obtidos por mais de um critério, estes algoritmos são tratados como algoritmos de otimização multiobjetivo. Este capítulo se inicia com a definição de problemas multiobjetivos e na seção 3.1 são apresentados os conceitos como Fronteira de Pareto e conjuntos de aproximação que descrevem os tipos dos resultados obtidos. Após, serão apresentados os conceitos de dominância de Pareto para a avaliação dos resultados. Na seção 3.2, métodos de avaliação, que concordam com os conceitos de dominância, são apresentados para a avaliação de algoritmos multiobjetivos.

Problemas de otimização com mais de uma função objetivo são chamados de problemas multiobjetivos. Um problema genérico de maximização multiobjetivo, sem nenhuma restrição, pode ser estabelecido como a maximização de:

$$f(x) = (f_1(x), ..., f_q(x)) \tag{3.1}$$

sujeito a $\mathbf{x} \in X$, onde $\mathbf{x}$ é um vetor de variáveis de decisão e X é um conjunto finito de soluções viáveis; $f_i(\mathbf{x}), i = 1, ..., q$ são as funções objetivos. Então, $f : X \rightarrow Z$ é uma função que atribui um valor objetivo $z = f(x) \in Z$ para cada solução $\mathbf{x} \in X$.

3.1 Pareto-ótimo

Em problemas multiobjetivos, os objetivos a serem otimizados estão usualmente em conflito entre si, o que significa que não existe uma solução única para tais casos, existem conjuntos de soluções eficientes no qual nenhuma é melhor que outra em todos os objetivos. Estes conjuntos fornecem soluções para a escolha de um decisor de acordo com sua preferência pessoal.

Um exemplo real a ser observado é a escolha de um produto eletrônico para a compra. A Figura 3.1 ilustra cinco produtos desejados. O objetivo é escolher um produto que não custe

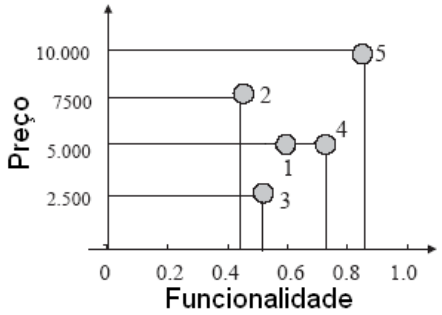


Figura 3.1: Exemplo de Problema multiobjetivo: Cinco escolhas de produção de eletrônicos de acordo com o seus preços e suas funcionalidades

muito e com o máximo de funcionalidades, ou seja, a escolha é um problema de maximização de funcionalidades (eixo x) e minimização do preço (eixo y). Quanto mais o ponto do produto estiver à direita e na parte inferior, mais vantajosa será a compra. Analisando as opções, pode-se descartar a solução 1, pois comparando as soluções 1 e 4, a solução 4 tem mais funcionalidades com o mesmo preço. Entre as soluções 2 e 4, escolhe-se a opção 4 por possuir mais funcionalidades por um menor preço. Já uma análise das opções 3, 4 e 5 não indica uma escolha com base no preço e funcionalidades. Por exemplo, embora a opção 4 tenha mais funcionalidades do que o produto 3, o preço é maior. Para o decisor seria necessário escolher entre pagar mais ou ter menos funcionalidades. Dizemos que a opção 3 não domina a opção 4 nem a opção 4 domina a 3. Também não existe uma opção que domine as opções 3, 4 ou 5, ou seja, nenhuma opção possui menor preço **E** mais funcionalidades do que elas. Portanto, as opções 3, 4 e 5 formam o conjunto das soluções não-dominadas e as opções 1 e 2 formam o conjunto de soluções dominadas.

Os conjuntos de soluções dominadas e não-dominadas possuem as propriedades:

- Qualquer solução que não pertença ao conjunto não-dominado deve ser dominada por, no mínimo, uma solução do conjunto não-dominado.
- Qualquer par de soluções do conjunto não-dominado deve ser não-dominado entre si.

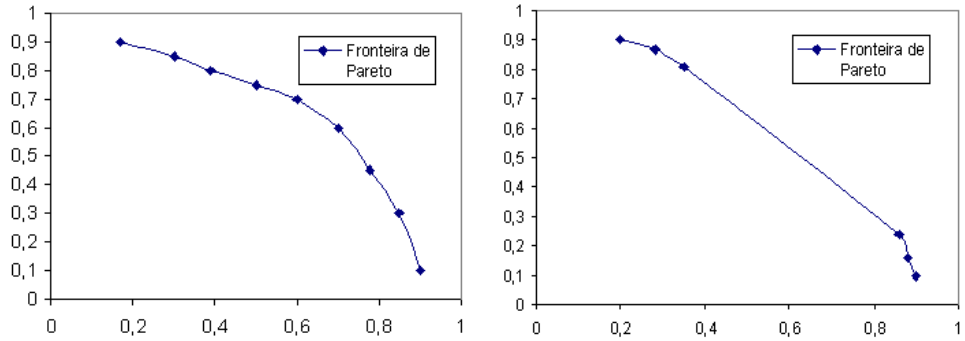
Um conjunto de vetores soluções não-dominados é chamado de *Pareto-ótimo* e o conjunto de todos os vetores objetivos não-dominados é chamado *Fronteira de Pareto*. O conjunto Pareto-ótimo é muito útil para problemas reais como, por exemplo, problemas de otimização em engenharia como maximização da utilização de recursos e minimização de custos, e provê informação valiosa sobre o problema em questão (KNOWLES; THIELE; ZITZLER, 2006). Quando a informação adicional sobre importância dos objetivos é desconhecida, todas as soluções Pareto-ótimas são igualmente importantes.

Para problemas multiobjetivos, Deb (2001) indica duas importantes metas para os algoritmos de otimização:

1. Encontrar um conjunto de soluções o mais próximo possível da Fronteira de Pareto.
2. Encontrar um conjunto de soluções com a maior diversidade possível no espaço objetivo.

A primeira meta é comum para qualquer processo de otimização; embora para alguns problemas esta meta seja de fácil resolução, encontrar a Fronteira de Pareto para problemas reais é uma tarefa difícil de ser alcançada pois a procura por uma única solução Pareto-ótimo para determinados tipos de problemas pode se tornar *NP-hard* (EHRGOTT, 2000). Então, o processo de otimização tem como foco encontrar o conjunto mais próximo possível do Pareto-ótimo, o chamado conjunto de aproximação.

Encontrar a maior diversidade entre os vetores objetivos é uma meta específica para o problema multiobjetivo. O exemplo de maximização de dois objetivos, Figura 3.2(a), mostra uma boa distribuição de soluções na Fronteira de Pareto, enquanto que na Figura 3.2(b) as soluções estão distribuídas apenas em duas regiões. É necessário assegurar a maior cobertura possível da fronteira, já que isso implica em ter um bom conjunto de soluções comprometidas com os objetivos desejados.



(a) Exemplo de boa distribuição (b) Concentração de soluções

Figura 3.2: Exemplo de distribuição da Fronteira de Pareto. A Figura (a) apresenta boa distribuição, a Figura (b) contém soluções concentradas em algumas regiões

Os algoritmos multiobjetivos podem ser avaliados pelos conjuntos de aproximação obtidos. Um exemplo de avaliação de dois algoritmos alg1 e alg2, que têm como objetivo a maximização de duas dimensões, está na Figura 3.3. O melhor ponto de maximização é o $(1, 1)$. A Figura mostra um exemplo dos conjuntos de aproximações gerados por dois algoritmos determinísticos de uma determinada base de dados. Nota-se que o alg1 possui todos os pontos da curva acima da curva do alg2, portanto, a curva do alg1 domina a do alg2. A área abaixo da

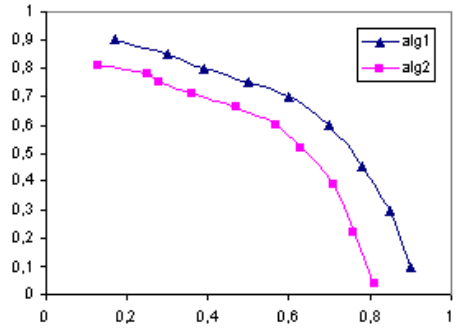


Figura 3.3: Comparação de Conjuntos de aproximação de dois algoritmos determinísticos (alg1 e alg2)

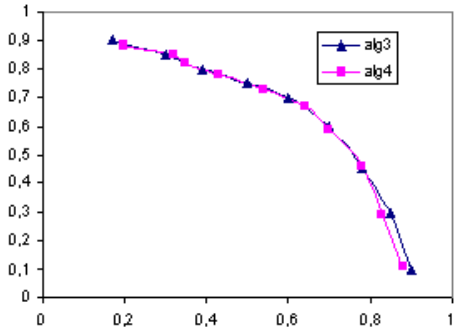


Figura 3.4: Comparação de Conjuntos de aproximação com muita semelhança de dois algoritmos determinísticos (alg3 e alg4)

curva de alg1 é maior, indicando que o algoritmo alg1 tem um desempenho melhor do que o alg2, segundo *AUC*.

A Figura 3.4 mostra outro exemplo de comparação de algoritmos. Nesta Figura não é possível uma fácil comparação visual dos conjuntos de aproximação. Mesmo com o aumento do zoom as distâncias entre os dois conjuntos não são visíveis. A comparação visual é muito subjetiva; outra opção é o cálculo de distância entre as curvas. Porém, mesmo este cálculo pode não identificar qual a melhor curva quando, por exemplo, as curvas estão parcialmente sobrepostas. Assim, uma metodologia de avaliação mais precisa é proposta com a utilização dos indicadores unários. Para tal, algumas definições sobre dominância são apresentadas.

Seja $\mathbf{z^1} = (z_1^1, ..., z_q^1)$ e $\mathbf{z^2} = (z_1^2, ..., z_q^2), \mathbf{z^1}, \mathbf{z^2} \in Z$ dois vetores objetivos.

- $z^1 \succ z^2$ (z^1 domina z^2) se z^1 não é pior do que z^2 em cada objetivo e é melhor em pelo menos um objetivo
- $z^1 \succ\succ z^2$ (z^1 domina estritamente z^2) se z^1 é melhor do que z^2 em todos os objetivos
- $z^1 \succcurlyeq z^2$ (z^1 domina fracamente z^2) se z^1 não é pior do que z^2 em qualquer objetivo

- $z^1 \mid\mid z^2$ (z^1 e z^2 são incomparáveis um com outro) se z^1 não domina z^2 e nem z^2 domina z^1
- $z^1 \sim z^2$ (z^1 e z^2 são indiferentes) se z^1 e z^2 são iguais em todos os objetivos.

Seja $A \subseteq Z$ um conjunto de vetores objetivos. A é dito um conjunto de aproximação se quaisquer dois elementos de A forem incomparáveis entre si. As relações de dominância podem ser estendidas para os conjuntos de aproximação. Dados dois conjuntos de aproximação A_1 e A_2, diz-se que A_1 domina A_2 $(A1 \succ A2)$ se todo vetor solução de A_2 é dominado por, pelo menos, um vetor objetivo de A_1.

Um conjunto de aproximação A_1 é dito *melhor do que* outro A_2, $(A_1 \triangleright A_2)$, se $A_1 \succcurlyeq A_2$ e $A_1 \neq A_2$. A_1 e A_2 são ditos *incomparáveis entre si* $(A_1 || A_2)$ se nem $A_1 \succcurlyeq A_2$ nem $A_2 \succcurlyeq A_1$, ou seja, são incomparáveis se as soluções não são dominadas fracamente entre si.

Os algoritmos determinísticos multiobjetivos têm associado exatamente um conjunto de aproximação para cada conjunto de dados. Por isso, a comparação de dois algoritmos determinísticos para resolver um problema de otimização envolve a comparação de dois únicos conjuntos de aproximação. Porém, um algoritmo que utiliza metaheurística é estocástico, ou seja, produz conjuntos diferentes a cada execução, o que dificulta a comparação. A seguir, apresenta-se uma metodologia de comparação de algoritmos multiobjetivos estocásticos.

3.2 Avaliação de Desempenho

Na otimização multiobjetivo, a avaliação e comparação de aproximações do conjunto Pareto-ótimo não é tão trivial, impossibilitando haver uma medida simples de comparação entre o conjunto aproximado em relação ao conjunto Pareto-ótimo. No início da criação dos algoritmos de otimização multiobjetivo estocásticos, utilizava-se a comparação visual; com o tempo, houve a necessidade de uma melhor definição qualitativa e quantitativa. Nesta seção, apresentam-se três métodos que podem ser utilizados para a comparação de algoritmos multiobjetivos estocásticos: rankeamento pela dominância, por indicador e função *attainment*. Primeiramente, é descrito o método para diferenciar qualitativamente os algoritmos, o chamado rankeamento por dominância; os demais serão descritos em seguida. A seção termina com uma discussão sobre os três métodos.

3.2.1 Rankeamento pela Dominância

O método de rankeamento pelo conceito de dominância, ou *dominance ranking*, propõe uma ordenação (*ranking*) para cada algoritmo com os conjuntos de aproximação de acordo com a relação de dominância de Pareto; estes *rankings* são comparados para verificar se existe alguma diferença estatística significativa entre eles. A seguir, o método de rankeamento pela dominância é detalhado.

Dados dois conjuntos $A = \{A_1, ..., A_k\}$ e $B = \{B_1, ..., B_j\}$ de dois algoritmos estocásticos, um *rank* é associado para cada conjunto de aproximação $Ci \in C$, $C = A \cup B$. O *rank* de C_i é o número de conjuntos de aproximação fracamente dominado e diferente de C_i mais um (definição 3.2). Então, quanto menor o *rank* do conjunto de aproximação C_i associado melhor é o conjunto.

$$rank(C_i) = 1 + \left|\{C_j \in C : C_j \triangleright C_i\}\right| \tag{3.2}$$

A equação 3.2 provê um valor inteiro que é associado para cada conjunto de aproximação de modo que cada algoritmo possui uma lista de valores chamado de *ranking*. Então, um teste estatístico pode ser utilizado para verificar se existe diferença significativa entre os conjuntos A e B. O teste verifica se os *ranks* associados para os conjuntos resultados de um algoritmo são melhores do que os *ranks* dos conjuntos de aproximação associados para outro. Pode ser utilizado o teste estatístico *U Mann-Whitney*, também chamado de teste *Mann-Whitney-Wilcoxon* ou teste *Wilcoxon rank-sum* (WILCOXON, 1945), que é um teste não-paramétrico utilizado para verificar a hipótese nula de que dois exemplares vêm da mesma população (CONOVER, 1971).

O método de rankeamento pela dominância concorda com os conceitos de dominância de Pareto fornecendo informações qualitativas relativas ao desempenho dos algoritmos em questão independentemente de qualquer preferência. Se os resultados obtidos com o teste de rankeamento pela dominância, entretanto, não suportam conclusões sobre as diferenças significativas entre os algoritmos investigados, então, novas avaliações devem ser executadas para que se possa diferenciar quantitativamente os conjuntos gerados. Por exemplo, a comparação pode utilizar os indicadores qualitativos ou função *attainment*, não existindo um consenso da comunidade sobre qual o melhor método ou mais indicado (KNOWLES; THIELE; ZITZLER, 2006).

3.2.2 Indicador Qualitativo

Os indicadores qualitativos podem ser aplicados para quantificar as diferenças potenciais de qualidade e para detectar diferenças que não podem ser reveladas pelo rankeamento pela dominância. O uso de indicadores é baseado em procedimentos de testes estatísticos e permite verificar se um algoritmo encontra conjuntos de aproximação de modo significantemente melhor do que outro algoritmo com respeito a um determinado indicador. Este método apresenta a ideia de utilizar um indicador para transformar os conjuntos de aproximação em valores reais que possam ser comparados estatisticamente e é realizado em duas etapas. Primeiramente um conjunto de aproximações é transformado em um valor real através da utilização de um indicador qualitativo; em seguida, os conjuntos de valores reais são comparados baseados num procedimento de teste estatístico padrão.

Seja $I(X)$ uma função que mapeia um conjunto de aproximação X num número real e A e B dois conjuntos de aproximação. A diferença entre os valores dos indicadores correspondentes $I(A)$ e $I(B)$ distinguem a qualidade de dois conjuntos. É essencial que sejam utilizados somente indicadores qualitativos que estejam de acordo com a dominância de Pareto (KNOWLES; THIELE; ZITZLER, 2006), ou seja, indicadores que têm a propriedade de que sempre que um conjunto de aproximação é preferido ao invés de outro de acordo com um indicador, então, o primeiro conjunto não deve ser dominado pelo segundo. Alguns indicadores unários que concordam com a dominância de Pareto são: hipervolume, epsilon aditivo e $R2$. O indicador unário é uma função $I(X)$ que mapeia um conjunto de aproximação X em um valor real. Seja A e B um par de conjuntos de aproximação, a diferença entre seus valores correspondentes dos indicadores $I(A)$ e $I(B)$ revela a diferença de qualidade entre os dois conjuntos.

O indicador de hipervolume $I_H(A)$ mede o hipervolume da porção do espaço das funções objetivos que é fracamente dominada por um conjunto de aproximação A (ZITZLER; THIELE, 1999). Para a utilização deste indicador, o espaço objetivo deve ser fechado (*bounded*); se não, um ponto de referência (dominado por todos os pontos) deve ser definido. Quando as medidas que estão sendo otimizadas são de maximização, o ponto de referência passa a ser o menor ponto possível. Por exemplo, se o objetivo é maximizar duas medidas que não aceitam valores negativos, então o ponto de referência pode ser o $(0,0)$. Também deve ser definido e utilizado um conjunto de referência R, o qual será descrito posteriormente, em geral, melhor do que o conjunto que está sendo analisado. Utiliza-se o teste de Mann-Whitney para verificar se a diferença entre o hipervolume R e o hipervolume de A é considerável. Quanto menor a diferença, melhor é a qualidade do conjunto de aproximação A. Uma característica do hipervolume é ser o único indicador unário capaz de detectar que um conjunto de aproximação A não é

melhor do que outro conjunto de aproximação B para todos os pares de soluções (KNOWLES; THIELE; ZITZLER, 2006).

O indicador unário epsilon aditivo $I_\varepsilon^1(A)$ fornece o mínimo fator ε pelo qual cada ponto em R pode ser diminuído tal que o resultado do conjunto de aproximação transformado é fracamente dominado por A (ZITZLER et al., 2003). Se o problema for de minimização, considera-se a adição do mínimo fator em vez de sua diminuição. Um conjunto de aproximação A é preferido a outro conjunto B, de acordo com o indicador aditivo unário epsilon, se $I_\varepsilon^1(A) < I_\varepsilon^1(B)$. Se o hipervolume e o epsilon aditivo unário indicarem preferências opostas para dois conjuntos de aproximação, então eles são incomparáveis.

O indicador $R2$ (HANSEN; JASZKIEWICZ, 1998), I_{R2}, é baseado no conjunto de funções utilitárias. Uma função utilitária é um mapeamento do conjunto Z de vetores objetivos de q-dimensões para um conjunto de números reais. Neste trabalho, a função utilizada pelo indicador I_{R2} é a função de Tchebycheff aumentada, dada pela equação 3.3, onde z_j^* é um ponto que domina fracamente o ponto z_j, $\xi = 0.01$, n o número de elementos do conjunto de aproximação e $\lambda_j \in \Delta$, o conjunto de vetores de pesos contendo (3.2) vetores dispersos normalizados.

$$u_\lambda = -\left(\max_{j \in 1..q} \lambda_j \left| z_j^* - z_j \right| + \xi \sum_{j=1}^{n} \left| z_j^* - z_j \right| \right) \tag{3.3}$$

O mesmo conjunto de referência é utilizado para os indicadores citados e, quando necessário, para os mesmos pontos de referência. O conjunto de referência é formado com todos os vetores não dominados de todos os conjuntos de aproximação gerados pelos algoritmos em questão.

Os pontos de referência (melhores e piores) de cada conjunto de exemplos são determinados através da análise dos pontos no conjunto de referência. Os piores e melhores valores de cada objetivo são calculados. O ponto de referência para o indicador de hipervolume pode ser formado pelos piores valores de cada objetivo menos 10 (maximização) ou mais 10 (minimização). O ponto de referência z^* do indicador $R2$ é obtido com os melhores valores de cada objetivo.

O teste estatístico Kruskal-Wallis (CONOVER, 1971) é utilizado para comparar os algoritmos baseando-se nos três indicadores de qualidade citados (hipervolume, epsilon aditivo e $R2$). O teste Kruskal-Wallis é uma extensão lógica do teste Mann-Whitney. É, também, um teste não paramétrico utilizado para comparar três ou mais conjuntos de exemplos, testando a hipótese nula de que todas as populações em análise possuem a mesma função de distribuição.

Em todos os indicadores utilizados, para a correta avaliação estatística, é necessário que a execução de $rank(C_i)$ seja independente para cada base de dados.

3.2.3 Função *Attainment*

Os resultados da execução dos algoritmos multiobjetivos estocásticos podem ser descritos por uma distribuição estatística. O método por função *attainment* modela as saídas do algoritmo multiobjetivo como uma função densidade de probabilidade no espaço objetivo. Ou seja, o método transforma os conjuntos de aproximação em funções de distribuição e depois faz a comparação estatística.

A função *attainment* é uma descrição da distribuição baseada na probabilidade de um vetor objetivo ser obtido pelo algoritmo. Seja Z um conjunto aleatório de vetores objetivos $\check{\mathbf{z}}^{\mathbf{j}}$ de cardinalidade $\check{m}$, como segue:

$$Z = \{\check{\mathbf{z}}^{\mathbf{j}} \in \Re^n, j = 1, ..., \check{m}\}, \tag{3.4}$$

onde n é o número de objetivos do problema. Seja z um vetor objetivo, a probabilidade de $\mathbf{z}$ ser obtido é a probabilidade de se obterem vetores que sejam dominados fracamente pelo vetor $\mathbf{z}$. A seguir, a definição da função $\alpha(.) : \Re^n \mapsto [0,1]$:

$$\begin{aligned} \alpha_Z(\mathbf{z}) &= P(\check{\mathbf{z}}^{\mathbf{1}} \preccurlyeq \mathbf{z} \vee \check{\mathbf{z}}^{\mathbf{2}} \preccurlyeq \mathbf{z} \vee ... \vee \check{\mathbf{z}}^{\check{\mathbf{m}}} \preccurlyeq \mathbf{z}) & (3.5) \\ &= P(Z \preccurlyeq \{\mathbf{z}\}) & (3.6) \end{aligned}$$

onde $P(.)$ é a função de densidade de probabilidade do algoritmo obter o vetor objetivo z numa simples rodada. A função *attainment* pode ser estimada a partir de exemplos de r execuções independentes do algoritmo através da função empírica *attainment* (EAF - *Empirical Attainment Function*) definida como:

$$\alpha_r(\mathbf{z}) = \frac{1}{r} \sum_{i=1}^{r} I(A^i \preccurlyeq \mathbf{z}), \tag{3.7}$$

onde A^i é o i-ésimo conjunto de aproximação (resultado de uma execução) do algoritmo e $I(.)$ é uma função indicadora que retorna 1 (um) se o argumento é verdadeiro e 0 (zero) se for falso. Em outras palavras, EAF fornece para o vetor objetivo a frequência relativa que o conjunto obtém no espaço objetivo com respeito às r execuções. A frequência é o número normalizado dos conjuntos de aproximação fracamente dominados pelo vetor. A saída de dois

algoritmos pode ser comparada por um teste estatístico sobre os resultados das duas EAFs. O teste estatístico não-paramétrico que pode ser aplicado é o Kolmogorov-Smirnov. Este teste, porém, não é capaz de determinar se todas as EAFs de um algoritmo estão acima de outro ou não; para descobrir tais diferenças, as EAFs podem ser visualizadas graficamente.

3.2.4 Discussão da Avaliação Multiobjetiva

O método de rankeamento pela dominância é utilizado para a comparação de algoritmos e é baseado na dominância de Pareto. O método é capaz de diferenciar qualitativamente conjuntos gerados por algoritmos diferentes e informar qual gera o melhor conjunto. Se o resultado da comparação indicar que um algoritmo é significantemente melhor do que outro então o mesmo é melhor consistentemente de acordo com a ordem de preferência dos conceitos de dominância. Ou seja, mesmo que outros métodos sejam utilizados para caracterizar alguma diferença entre a distribuição dos conjuntos de aproximação, os mesmos não são necessários, pois já se sabe qual algoritmo estocástico gera os melhores conjuntos. Por esta razão, recomenda-se este método como o primeiro passo de qualquer comparação (KNOWLES; THIELE; ZITZLER, 2006).

Se os resultados obtidos com o teste de rankeamento pela dominância, entretanto, não suportam conclusões sobre as diferenças significativas entre os algoritmos investigados, então novos experimentos devem ser executados. Por exemplo, a comparação pode utilizar os indicadores qualitativos unários que podem ser aplicados para quantificar as diferenças potenciais de qualidade e para detectar diferenças que não podem ser reveladas pelo rankeamento pela dominância. O uso de indicadores é baseado em procedimentos de testes estatísticos e permite verificar se um algoritmo encontra conjuntos de aproximação de forma significantemente melhor do que outro algoritmo com respeito a um determinado indicador.

Em contraste com o método de rankeamento pela dominância, o método com indicadores permite fazer suposições quantitativas sobre as diferenças de qualidade até mesmo para conjuntos de aproximação incomparáveis. Entretanto, esta vantagem tem um custo de generalidade: todo indicador qualitativo unário representa uma preferência específica. Consequentemente, toda afirmação do tipo ‘algoritmo A tem um desempenho melhor do que o algoritmo B’ precisa ser acompanhado do complemento ‘com respeito ao indicador qualitativo I’.

Quando o método com indicadores converte os conjuntos de aproximação, existe a perda de informação, pois ocorre a transformação de um conjunto multidimensional em um valor real (KNOWLES; THIELE; ZITZLER, 2006). Esta é uma desvantagem do método com indicadores qualitativos que pode afetar o uso dos indicadores de modo que as diferenças quali-

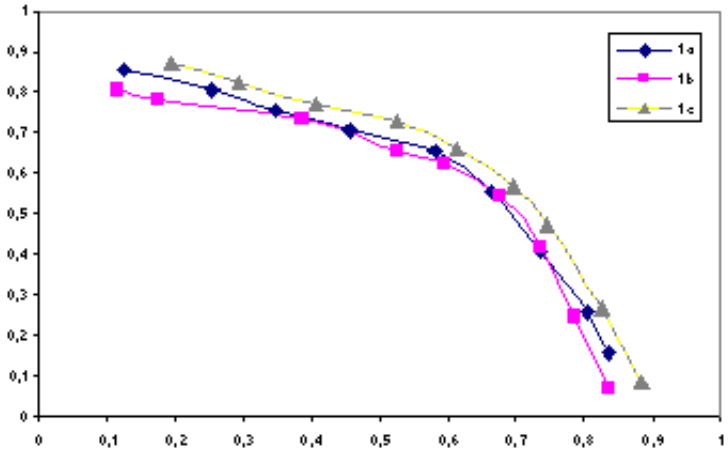


Figura 3.5: Curvas dos Conjuntos de aproximação do algoritmo 1

tativas podem não ser detectadas. No método por função *attainment*, os exemplos transformados são multidimensionais, o que diminui a perda da transformação.

Uma desvantagem do método por função *attainment* é ter pior desempenho computacional e ser aplicado apenas em casos com poucas funções objetivos (KNOWLES; THIELE; ZITZLER, 2006). Por estas razões, nesta tese, prefere-se a metodologia com indicadores para a comparação de algoritmos estocásticos multiobjetivos como complemento ao rankeamento pela dominância.

3.2.5 Exemplo de Avaliação Multiobjetiva

Nesta seção utiliza-se um exemplo para melhor visualização da metodologia de desempenho qualitativa. Cita-se um exemplo fictício de comparação de dois algoritmos estocásticos: algoritmo 1 e algoritmo 2. Os algoritmos têm como objetivo a maximização de dois objetivos, onde o melhor valor é $(1, 1)$. São comparados 3 conjuntos de aproximação obtidos de três execuções sem alterar os supostos parâmetros existentes nem os conjuntos de dados. Os dados dos seis conjuntos estão na Tabela A.1, em anexo. A contagem das soluções presentes nos conjuntos mostra que o algoritmo 2 possui menos elementos (21 pontos) que o algoritmo 1 (27 pontos). Nota-se a falta de alguns pontos próximos do valor $0,75$ do eixo x. Os três conjuntos de aproximação do algoritmo 1 estão representados na Figura 3.5 e os três conjuntos do algoritmo 2 estão na Figura 3.6. As curvas da Figura 3.5 do algoritmo 1 mostra certa diferença visual entre as três curvas, enquanto as curvas do algoritmo 2 mostram as curvas mais semelhantes entre si, o que pode indicar um sistema estocástico mais estável para a criação dos conjuntos de aproximação.

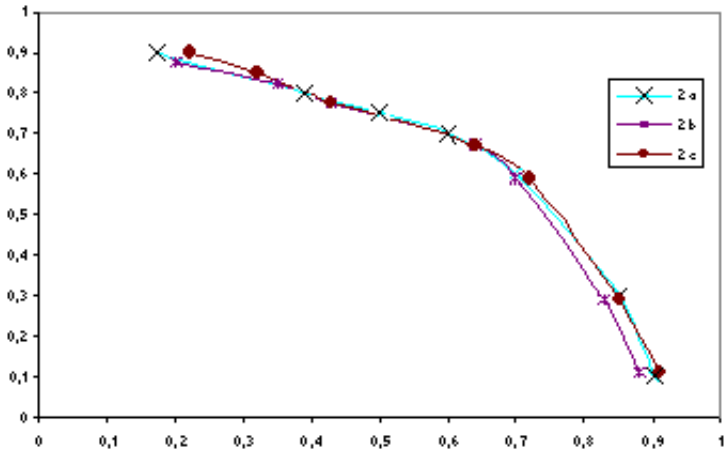


Figura 3.6: Curvas dos Conjuntos de aproximação do algoritmo 2

A Figura 3.7 contém as seis curvas dos conjuntos de aproximação para uma melhor comparação visual. Uma comparação visual entre as duas figuras indica uma suposta superioridade do algoritmo 2, pelo fato das curvas estarem mais próximas do ponto $(1,1)$. Contudo, existe uma proximidade entre duas curvas dos algoritmos (1c e 2b).

Como a comparação visual é muito subjetiva, a metodologia de avaliação pode desempenho propõe a utilização dos indicadores unários. Utilizou-se o *framework* PISA (BLEULER et al., 2003) para todas as avaliações deste exemplo. A avaliação começa com o rankeamento pela dominância. O *ranking* para os algoritmos obtidos foram: $(1,2,2)$ e $(4,4,4)$, respectivamente, para os algoritmos 1 e 2. A comparação do *ranking* utilizando o teste Wilcoxon rank-sum resultou num *p-value* igual a $0,1587$ para a hipótese nula ($H0$: algoritmo 2 diferente do algoritmo 1), indicando que os conjuntos de aproximação não possuem diferença significativa em relação ao rankeamento pela dominância. Com um resultado não conclusivo, os indicadores unários são avaliados.

Considerando o indicador hipervolume, a hipótese nula "algoritmo 2 melhor do que o algoritmo 1" teve o *p-value* igual a $0,0248$, o que significa que, por este indicador, o algoritmo 2 gera conjuntos de aproximação melhores.

Já com o indicador epsilon aditivo unário, o *p-value* foi de $0,1843$, mostrando que não há diferença significativa entre os conjuntos gerados segundo o mesmo.

Com o indicador $R2$, o *p-value* da hipótese $H0$ (algoritmo 2 gerar conjuntos estatisticamente diferentes do algoritmo 1) foi de $0,0248$, indicando a preferência do algoritmo 2 por gerar conjuntos estatisticamente diferentes de acordo com o indicador $R2$.

Embora o rankeamento pela dominância não diferencie os dois algoritmos, o algoritmo

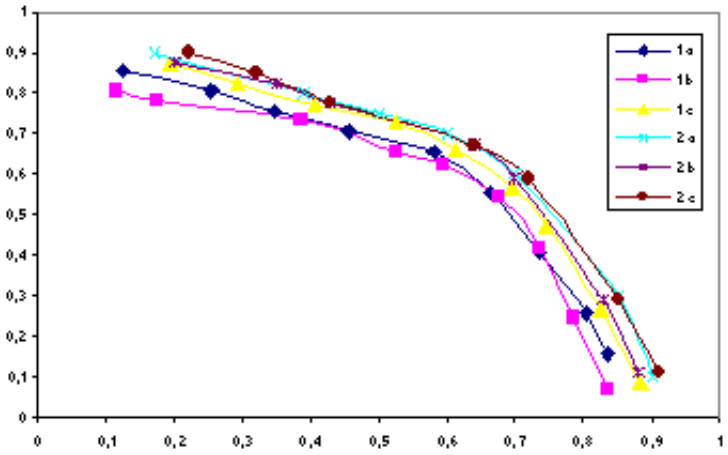


Figura 3.7: Todas as 6 curvas dos algoritmos 1 e 2

2 tem preferência segundo os indicadores unários de hipervolume e $R2$.

Nesta seção foram apresentados os problemas multiobjetivos, alguns de seus conceitos relacionados e alguns métodos de avaliação dos conjuntos resultados. Nos próximos capítulos serão apresentadas novas abordagens e os métodos de avaliação serão utilizados para a comparação dos conjuntos de aproximação obtidos pelos algoritmos multiobjetivos.

4 A Busca do Equilíbrio entre Precisão e Generalidade

A primeira investigação da tese busca o equilíbrio dos critérios de precisão e generalidade. Precisão e generalização são dois critérios importantes e contraditórios, formando um grande paradoxo da aprendizagem. Um algoritmo com ótima precisão (*accuracy*) é aquele que classifica os exemplos corretamente, tanto os positivos como os negativos. Um algoritmo que tem muita precisão leva a uma sobrecarga (*overfitting*), especialização em apenas um conjunto de exemplos, prejudicando a generalidade do modelo aprendido. Por outro lado, aprender com pouca precisão conduz a uma generalização de maneira que o modelo aprendido não é útil nem desejado pelo usuário (CORNUEJOLS; MICLET, 2003). A sobre-especialização (*overfitting*) acontece com a alta precisão sobre o conjunto de dados utilizado para treinamento enquanto seu desempenho nas bases de testes se torna muito baixo. A generalidade preocupa-se em fazer com que o modelo seja válido para o maior número de exemplos, ou seja, quanto mais exemplos são classificados pelo modelo, maior será a sua generalidade. O objetivo levantado neste capítulo é obter a precisão sem perder a generalidade, tendo como hipótese verificar se a combinação das técnicas de *boosting* e estratégia evolucionária podem atingir o equilíbrio entre os dois critérios.

Atingir os dois objetivos contraditórios ao mesmo tempo é um trabalho árduo e que não é obtido pelos algoritmos mais simples, sendo necessária a aplicação de técnicas mais elaboradas. Assim, para lidar com o problema, são apresentados *boosting* e Estratégia Evolucionária, que serão cuidadosamente combinados. Primeiramente apresenta-se a descrição dos conceitos introdutórios de ambos para então apresentarmos a solução proposta.

4.1 Meta-aprendizado

Meta-aprendizado (*Meta-learning*), também conhecido como processo de aprender meta-conhecimento sobre o conhecimento aprendido, pode ser visto como uma técnica geral para combinar o resultado de múltiplos algoritmos de aprendizado, cada qual aplicado ao conjunto de dados de treinamento (CHAN; STOLFO, 1993). O meta-aprendizado estuda como aumentar a eficiência dos sistemas de aprendizado através da experiência. O objetivo é com-

preender como o aprendizado por si só pode se tornar flexível de acordo com o domínio ou tarefa em estudo (VILALTA; DRISSI, 2002). Num cenário simplificado, podem-se citar diferentes estágios de meta-aprendizado (PRODROMIDIS; CHAN, 2000):

- Os classificadores bases são treinados a partir do conjunto inicial de treinamento (*base-level*).
- Predições são geradas a partir dos classificadores aprendidos em um conjunto separado de validação.
- Um conjunto meta de treinamento é construído a partir do conjunto de validação. As predições para validação dos classificadores são feitas sobre o mesmo conjunto de validação.
- O classificador final, chamado de meta-classificador (*meta-classifier*), é treinado a partir do conjunto meta de treinamento.

Mitchell, citado por Prodromidis e Chan (2000), afirma que o meta-aprendizado melhora o desempenho de predição pela combinação de diferentes sistemas de aprendizado cada qual tendo diferentes *bias* indutivos (por exemplo: representação, busca heurística, espaço de busca). Pela combinação dos conceitos aprendidos separadamente, é esperado que o meta-aprendizado apresente um alto nível de conhecimento aprendido que detalhe um grande banco de dados mais precisamente do que qualquer algoritmo individual. Além disso, meta-aprendizado constitui um método de aprendizado de máquina escalável, uma vez que pode ser generalizado a um meta-aprendizado hierárquico multinível (PRODROMIDIS; CHAN, 2000).

Muitos dos métodos existentes de aprendizado que encontram, avaliam e combinam um conjunto de classificadores podem ser considerados como votação por pesos entre as hipóteses. Além disso, muitos meta-algoritmos geram seus classificadores através da utilização dos mesmos algoritmos-base apenas com a variação dos mesmos conjuntos de dados (PRODROMIDIS; CHAN, 2000).

4.1.1 Meta-aprendizado para classificação

No caso do aprendizado induzido, meta-aprendizado significa aprender a partir dos classificadores produzidos pelo algoritmo de aprendizado e a predição (classe predita) destes classificadores sobre os dados de treinamento (CHAN; STOLFO, 1993). O foco maior está nos resultados dos algoritmos, mais do que nos próprios algoritmos. Os mesmos dados de

treinamento apresentados inicialmente para o algoritmo de aprendizado são também disponíveis para o meta-algoritmo (*meta-learner*) ou meta-aprendiz.

Segundo Sigletos et al. (2003), um conjunto de classificadores (*classifier ensemble*), consiste de um conjunto de q classificadores $C_1, ..., C_q$, chamados de classificadores-base (*base-level classifiers*). Um meta-classificador CML (*meta-classifier*) aprende como combinar as predições dos classificadores-base. Os classificadores-base são gerados aplicando q diferentes algoritmos de classificação sobre os conjuntos de exemplos com suas classes $\{(\mathbf{x_k}, y_k), k = 1, ..., m\}$. As predições individuais dos classificadores base sobre diferentes conjuntos de dados TM são utilizadas para treinar o meta-classificador CML. As predições dos classificadores bases sobre TM são então transformadas em um conjunto de nível meta de vetores de classificação. Em tempo de execução, CML combina as predições $PM(x) = P_i(x), i = 1, ..., q$ dos q classificadores sobre cada nova instância x, e decide sobre o valor final da classe $y(x)$. Em classificação, existe o interesse de agregar um conjunto de classificadores para melhorar uma medida de desempenho, por exemplo, a precisão. Embora a ideia de crescimento de árvores múltiplas (QUINLAN, 1987) não seja nova, a justificativa para tais métodos era geralmente empírica. Em contraste, duas técnicas muito conhecidas para a produção e utilização de diversos classificadores são aplicáveis a uma ampla variedade de sistemas de aprendizado e são baseadas na análise teórica do comportamento do classificador composto. Estas técnicas são *bagging* e *boosting*, descritas a seguir.

4.1.2 *Bagging* e *Boosting*

Bootstrap aggregating ou *bagging* (BREIMAN, 1996a) e *boosting* (SCHAPIRE, 1990) são técnicas para obter a menor taxa de erro na classificação usando múltiplos classificadores (DRUCKER, 1997). As duas técnicas constroem modelos fortes a partir da combinação de modelos fracos obtidos a partir de um algoritmo simples de classificação. Ambas manipulam os dados de treinamento para gerar diferentes classificadores. Enquanto *bagging* produz conjuntos de treinamento separados por *sampling* com reposição, *boosting* utiliza todas as instâncias a cada repetição e mantém um peso para cada instância do conjunto de treinamento que reflete sua importância; o ajuste de pesos faz o algoritmo de aprendizagem focar em diferentes instâncias e então encontrar diferentes classificadores. Nas duas técnicas, os múltiplos classificadores são combinados pela votação para formar o classificador composto. Em *bagging*, cada classificador componente tem o mesmo voto, enquanto *boosting* diferencia os votos por pesos para os classificadores componentes com base em sua precisão.

Assume-se um conjunto de m instâncias, cada instância pertencendo a uma classe

$(y_l, l = 1, .., nc)$, e um sistema de aprendizado que constrói um classificador a partir do conjunto de treinamento das instâncias. Como mencionado, *bagging* e *boosting* constroem múltiplos classificadores a partir das instâncias; o número R de repetições ou tentativas será fixado, embora Freund e Schapire (1997) tenham observado que este número pode ser determinado automaticamente pela validação cruzada (*cross-validation*) (KOHAVI, 1995). O classificador aprendido na tentativa r será representado por C_r. Para representar o classificador composto (*bagged* ou *boosted*) usa-se C^*. Para qualquer instância $\mathbf{x_k}, C_r(\mathbf{x_k})$ e $C^*(\mathbf{x_k})$ são as classes preditas por C_r e C^*, respectivamente.

Detalhes sobre Bagging

Para cada tentativa $r = 1, ..., R$, um conjunto de treinamento de tamanho m é criado por amostra com reposição a partir de instâncias originais. Este conjunto de treinamento (amostra *bootstrap*) é do mesmo tamanho que o conjunto de exemplos original, mas algumas instâncias podem não constar no conjunto e outros podem aparecer mais do que uma vez. O sistema de aprendizado gera um classificador C_r a partir do exemplo e o classificador final C^* é formado pela agregação de R classificadores criados destas tentativas. Para classificar uma instância $\mathbf{x_k}$, o voto para a classe j é registrado por cada classificador para o qual $C_r(\mathbf{x_k}) = j$ e $C^*(\mathbf{x_k})$ é então a classe com mais votos (o empate é resolvido arbitrariamente).

Breiman (1996a) introduziu o conceito de sistemas classificadores de comando correto (*order-correct*), como aqueles que, sobre muitos conjuntos de treinamento, tendem a predizer a classe correta de uma instância testada mais frequentemente do que qualquer outra classe. Um algoritmo *order-correct* pode produzir ótimos classificadores; Breiman mostra que classificadores agregados produzidos pelos algoritmos *order-correct* resultam em um classificador ótimo. Breiman (1996a) observa: "O elemento vital é a instabilidade do método de predição. Se a perturbação do conjunto de aprendizado produz mudanças significativas no preditor construído, o classificador resultante do algoritmo de *bagging* terá precisão".

Bauer (BAUER; KOHAVI, 1999) analisou o *bagging* e a amostra *bootstrap* chegando à conclusão de que existe uma pequena probabilidade do *bootstrap* conter apenas uma instância de cada exemplo. Esta perturbação do conjunto de treinamento faz com que o *bagging* construa classificadores diferentes se o algoritmo de indução é instável (ex: árvores de decisão, redes neurais) (BREIMAN, 1996b) e o desempenho pode melhorar se os classificadores são bons e não correlacionados. *Bagging* pode degradar levemente o desempenho de algoritmos estáveis (ex: *k-nearest neighbor*), porque conjuntos de treinamentos pequenos efetivamente são utilizados para treinar cada classificador (BREIMAN, 1996a).

Detalhes sobre Boosting

Boosting, um método de aprendizado de máquina, é baseado na observação de que encontrar muitas hipóteses fracas com um classificador pode ser mais fácil do que encontrar uma única hipótese com alta precisão. Segundo Schapire (2002), *boosting* refere-se a um método geral e provavelmente efetivo de produzir uma hipótese de predição muito precisa combinando hipóteses fracas e pouco precisas. A primeira proposta do algoritmo para problemas de classificação binária foi apresentada por Schapire (1990). Seguido por outros trabalhos como AdaBoost (FREUND; SCHAPIRE, 1999), *Coordinate Ascent Boosting* e *Approximate Coordinate Ascent Boosting* (RUDIN; SCHAPIRE; DAUBECHIES, 2004) que apresentaram bons resultados na área de classificação.

Para aplicar a abordagem de *boosting*, começa-se com um método ou algoritmo para encontrar as hipóteses fracas. O algoritmo de *boosting* chama repetidamente o algoritmo de aprendizado, cada vez alimentando-o com um subconjunto diferente dos exemplos de treinamento (mais precisamente, uma distribuição diferente dos exemplos ou a atribuição de um peso diferente para cada exemplo). Cada vez que é chamado, o algoritmo de aprendizado gera uma nova hipótese de predição fraca, e depois de muitas execuções, o algoritmo de *boosting* deve combinar essas hipóteses fracas dentro de uma única hipótese de predição que, espera-se ser mais precisa que qualquer outra gerada anteriormente (SCHAPIRE, 2002).

Para a execução do *boosting*, faz-se necessário um conjunto de exemplos de treinamento ξ (equação 2.1), onde cada instância x_i de um domínio ou um espaço de instância X corresponde a um rótulo y_i (classe), e também um conjunto de funções $h_1, ..., h_q$ de valores reais em X. Na literatura do *boosting*, esses h_i's são as chamadas hipóteses bases ou fracas (ZHANG et al., 2004).

Boosting objetiva rotular os x_i's utilizando uma combinação linear dessas hipóteses fracas. Em outras palavras, procura-se um vetor $\lambda = (\lambda_1, ..., \lambda_q)^T \in \Re^q$ tal que $f_\lambda(x_i) = \sum_{j=1}^{q} \lambda_i h_j(x_j)$ seja uma boa aproximação para a função básica de rótulos de classe. E o classificador forte ou final pode ser dado por $H(x) = sign(f(x))$, onde *sign* é o sinal. Contudo, em vez de usar f_λ diretamente como uma hipótese de classificação, normalmente postula-se que os y_i's são determinados por um modelo de probabilidade associado com $f_\lambda(x_i)$ (ZHANG et al., 2004). Por exemplo, Friedman, Hastie e Tibshirani (1998) sugerem que a probabilidade posterior de y é dada por uma função logística de $f_\lambda(x)$, como

$$\hat{P}(y|x) = \frac{1}{1 + e^{-y\sum_{j=1}^{m} \lambda_j h_j(x)}} \tag{4.1}$$

A seguir apresenta-se um dos primeiros algoritmos de *boosting* criado.

Algoritmo AdaBoost

O algoritmo AdaBoost, introduzido por Freund e Schapire em (FREUND; SCHAPIRE, 1996), ajusta um modelo logístico aditivo, o número de iterações é o número de funções usadas na representação aditiva, ou seja, o número de funções que serão utilizadas para aproximar a função estimada. O algoritmo gera em cada passo uma distribuição sobre as observações da amostra, dando um peso maior às observações classificadas incorretamente no passo anterior. Devido ao seu sucesso, em seguida, surgiram algumas variações do algoritmo denominadas AdaBoost.M1 e AdaBoost.M2 para problemas com múltiplas classes (FREUND; SCHAPIRE, 1997).

Assumindo duas classes $Y = \{-1, +1\}$, o algoritmo AdaBoost recebe como entrada um conjunto de treinamento ξ (equação 2.1). AdaBoost chama um dado algoritmo de aprendizado base ou fraco repetidamente em uma série de *ne* execuções. Uma das ideias principais é manter uma distribuição ou conjunto de pesos sobre o conjunto de treinamento. O peso dessa distribuição no exemplo de treinamento $\mathbf{x_k}$ na execução t é denotado $d^t(k), t = 1, ..., ne$, $k = 1, ..., m$. Inicialmente, todos os pesos são iguais, mas em cada execução, os pesos de exemplos classificados incorretamente são incrementados para que o algoritmo de aprendizado fraco seja forçado a focar nos exemplos difíceis do conjunto de treinamento (FREUND; SCHAPIRE, 1999).

O pseudocódigo do algoritmo é fornecido pelo Algoritmo 1 (FREUND; SCHAPIRE, 1999). Algumas variáveis são iniciadas nos passos 0 e 1.

O trabalho do algoritmo de aprendizado fraco (Passo 2.1 do Algoritmo 1) é encontrar uma hipótese fraca $h^t : X \rightarrow \{-1, +1\}$ apropriada para a distribuição d^t. Note que o passo 2 indica a execução do algoritmo fraco *ne* vezes.

Uma medida de qualidade da hipótese fraca é o erro e^t, calculado no passo 2.2 do Algoritmo 1. Este erro é medido com respeito à distribuição d^t, na qual o algoritmo de aprendizado fraco foi treinado.

Uma vez que a hipótese fraca h^t foi recebida, AdaBoost calcula o parâmetro α^t (Passo 2.3 do Algoritmo 1) que será utilizado como peso da hipótese fraca para a hipótese final. Intuitivamente, α^t mede a importância que é fornecida para h^t de acordo com o seu erro de classificação. Segundo Freund e Schapire (1999), observa-se que $\alpha^t \geq 0$ se $e^t \leq 1/2$ (assumindo não haver perda de generalidade), e que o valor de α^t aumenta quando o valor de e_t

diminui.

A distribuição d^t é a próxima atualização utilizando a regra mostrada no Passo 2.4 do Algoritmo 1. A atualização é baseada no modelo de probabilidade de perda exponencial 4.2. O efeito dessa regra é aumentar o peso de exemplos classificados de forma errada por h^t, e diminuir o peso dos exemplos classificados corretamente. Assim, o peso tende a se concentrar em exemplos difíceis de serem classificados (FREUND; SCHAPIRE, 1999).

$$L_t(x_i) = 1 - \exp\left(-\frac{|h^t(x_i) - y(x_i)|}{\max_{i=1\ldots m}|h^t(x_i) - y(x_i)|}\right) \tag{4.2}$$

A hipótese final H (cuja equação esta descrita no passo 3 do Algoritmo 1) define a classe dos exemplos. A classificação dos exemplos é determinada de acordo com o voto das *ne* hipóteses fracas (h^t) onde o parâmetro α^t é o peso do voto.

Algoritmo 1 Pseudocódigo do Adaboost para a classificação. Fonte: Tradução de (FREUND; SCHAPIRE, 1999).

0. Dados: $(x_1, y_1), ..., (x_m, y_m)$ onde $\mathbf{x_i} \in X, y_i \in Y = \{-1, +1\}$
1. Inicialize $d^1(k) = 1/m, k = 1, ..., m$
2. Para $t = 1, ..., ne$:
   2.1. Treine o aprendiz fraco utilizando a distribuição d^t
   2.2. Receba a hipótese fraca $h^t : X \to \{-1, +1\}$ com erro
   $e^t = Pr_{k\sim d^t}[h^t(\mathbf{x_k}) \neq y_k] = \sum_{k:h^t(\mathbf{x_k})\neq y_k} d^t(k)$
   2.3. Selecione $\alpha^t = \frac{1}{2} ln(\frac{1-e^t}{e^t})$
   2.4. Atualize:
   $d^{t+1}(k) = \frac{d^t(k)}{Z^t} \times \begin{cases} e^{-\alpha t} & if\ h^t(\mathbf{x_k}) = y_k \\ e^{\alpha t} & if\ h^t(\mathbf{x_k}) \neq y_k \end{cases}$
   $= \frac{d^t(k) exp(-\alpha^t y_k h^t(\mathbf{x_k}))}{Z^t}$
   onde Z^t é um fator de normalização (escolhido tal que d^t seja uma distribuição).
3 Forneça a hipótese final (onde sign é o sinal):
$H(x) = sign(\sum_{t=1}^{ne} \alpha^t h^t(x))$

A propriedade mais interessante do AdaBoost é a habilidade de reduzir o erro de treinamento (SCHAPIRE, 2002).

4.2 Computação Evolucionária

Um importante campo de pesquisa da computação é a inteligência computacional (*Computational Intelligence - CI*). Este campo inclui áreas como redes neurais (*Neural Networks*), sistemas nebulosos (*Fuzzy Logic*) e computação evolucionária (*Evolutionary Computation - EC*), os quais se originaram a partir do modelo de processamento de informação nos

sistemas naturais (BÄCK; FOGEL; MICHALEWICZ, 2000).

A *EC* abrange métodos de simulação frequentemente utilizados em computadores. Este ramo da *CI* propõe um paradigma para a solução de problemas inspirado na seleção natural de Darwin. Segundo a teoria da seleção natural de Darwin (DARWIN, 1859), na natureza sobrevivem os seres que têm a melhor capacidade de se adaptar às mudanças que ocorrem no meio ambiente. Esses indivíduos passarão suas características genéticas para seus descendentes e, ao final de muitas gerações, teremos uma população de indivíduos selecionados naturalmente. Em *EC*, os algoritmos são caracterizados por manipularem populações de indivíduos e contarem com variação aleatória e seleção. Quando contam com princípios evolucionários, os mesmos são chamados de algoritmos evolucionários (*evolutionary algorithms*). Estes algoritmos são classificados em algumas técnicas: algoritmos genéticos (*genetic algorithms*) (HOLLAND, 1992), programação genética (*genetic programming*) (KOZA, 1992), programação evolucionária (*evolutionary programming*) (FOGEL; OWENS; WALSH, 1966), estratégias evolucionárias (RECHENBERG, 1973) (SCHWEFEL, 1977) (BäCK; SCHWEFEL, 1993) e outros.

Os algoritmos destas técnicas possuem um ciclo de evolução básico, como mostra a Figura 4.1. Primeiramente é criada uma população de indivíduos (geração atual); depois uma estratégia de seleção escolhe os pais; os pais selecionados sofrem variação e mutação gerando uma nova geração; e finalmente a nova geração substitui a geração atual.

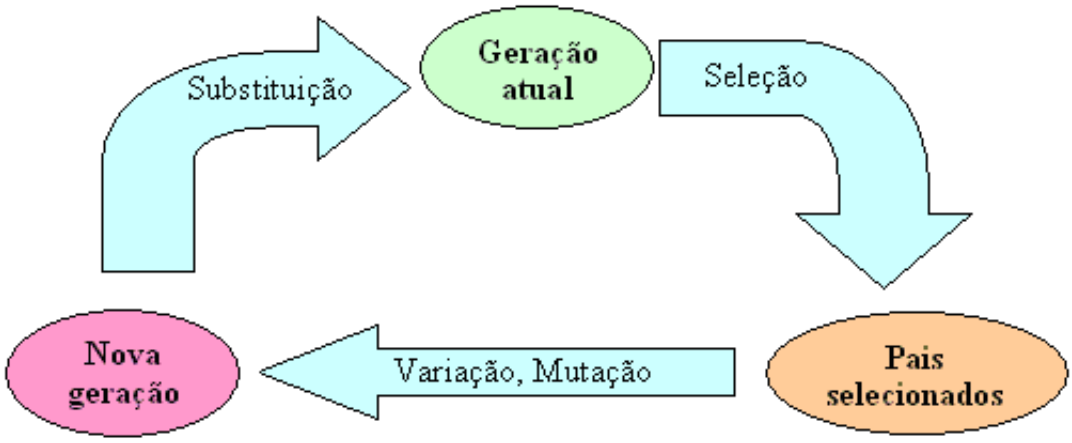


Figura 4.1: Ciclo Básico da Evolução. FONTE: Tradução de Dianati, Song e Treiber (2002)

Diferentes variações da computação evolucionária incorporam o mesmo ciclo básico com diferentes modelos de representações ou combinações específicas de variação, mutação, seleção e métodos de substituição. O ponto interessante na implementação é o balanço entre

duas operações opostas: de um lado, a operação de seleção pretende reduzir a diversidade da população (conjunto de soluções possíveis); de outro, os operadores de variação e mutação tentam incrementar a diversidade da população. A consequência é o direcionamento do algoritmo à convergência para uma solução global.

Originalmente, os algoritmos genéticos foram projetados como um sistema para adaptação mais do que um sistema de otimização. As características básicas são a forte ênfase em recombinação (*crossover*), o uso de um operador probabilístico de seleção e a interpretação da mutação como um operador auxiliar, executando um papel secundário no algoritmo. O algoritmo em sua forma original (o algoritmo genético canônico) representa as soluções por caracteres binários. Um número de variações dos algoritmos, incluindo os algoritmos genéticos de codificação real e inteira, também têm sido desenvolvidos para poder aplicar o algoritmo a outros espaços de busca diferentes dos binários (BÄCK; FOGEL; MICHALEWICZ, 2000).

A programação genética (*Genetic Programming - GP*) foi criada com o objetivo de produzir programas automaticamente. Foi introduzida por John Koza (KOZA, 1992), que se baseou na teoria de Darwin e na ideia dos algoritmos genéticos (AG) apresentada por John Holland (HOLLAND, 1992). *GP* também pode ser vista como uma técnica de busca, aplicada sobre o universo de todos os programas de computador que podem ser gerados em uma dada linguagem. No lugar de uma população de seres vivos, em *GP* tem-se uma população de programas de computador, representados sob a forma de genes. Através da recombinação desses genes, o objetivo do algoritmo de *GP* é chegar ao programa de computador que melhor resolve um determinado problema. Para isso, parte-se de uma população inicial aleatória e, geração após geração, os operadores genéticos são aplicados para simular o processo evolutivo.

4.2.1 Estratégia Evolucionária

Jorge Muniz Barreto apresenta um breve histórico a respeito das Estratégias Evolucionárias (BARRETO, 1999): “Em 1963, dois estudantes da Universidade Técnica de Berlim (TUB), Ingo Rechenberg e Hans-Paul Schwefel, iniciaram uma estreita colaboração no túnel de vento da universidade. Eles estavam estudando a determinação de formas ótimas de corpos sujeitos a um fluxo de ar. Este tipo de estudo se caracteriza por ausência quase total de métodos algorítmicos que levem a uma solução e é normalmente abordada através de extensas experimentações, guiada principalmente pela experiência dos pesquisadores. Possuidores de um metódico caráter germânico, eles tiveram a ideia de buscar uma metodologia mais ordenada para resolver o problema, e se lançaram em técnicas de programação não-linear (método do gradiente), mas suas tentativas fracassaram. Foi então que um dos dois estudantes, Rechenberg,

mais tarde professor de Biônica e Engenharia Evolucionista, teve a ideia de tentar modificações aleatórias das soluções, tal como ocorre nas mutações nos seres vivos. Ele deu ao método o nome de Estratégia Evolucionária. Foi então que um terceiro estudante, Peter Biernet, uniu-se aos dois primeiros e iniciou a construção de um sistema para fazer automaticamente o trabalho usando inspiração biológica e implementando os operadores de mutação e seleção".

Estratégia evolucionária (*Evolution Strategies*, *ES*) (RECHENBERG, 1973) (SCHWEFEL, 1977) é um algoritmo onde indivíduos (soluções potenciais) são codificados por um conjunto de variáveis de valores reais, o "genoma" (KUSIAK, 2000). Nas palavras de Schwefel (1981), "o método de evolução orgânica representa uma estratégia ótima para a adaptação de seres vivos em seu ambiente... [e]... portanto seria conveniente utilizar os princípios da evolução biológica para a otimização de sistemas técnicos". *ES* foi aplicado primeiramente para problemas de otimização experimental com parâmetros mutáveis continuamente, mas não de forma completa (BÄCK; HOFFMEISTER, 1991). Uma das primeiras aplicações numéricas foi executada por Hartmann (HARTMANN, 1974) e a primeira tentativa para estender esta estratégia para resolver problemas de otimização de parâmetros binária ou discreta foi feita em (SCHWEFEL, 1975).

Um *ES* é caracterizado pelo tamanho da população, o número de descendentes produzidos em cada geração e se a nova população é selecionada de pais e descendentes ou somente de descendentes (KUSIAK, 2000). Sua notação pode ser visualizada na Figura 4.2 (VRIESMANN, 2005). A seguir, as estratégias de *ES* serão descritas e possibilitarão melhor entendimento da notação utilizada nesta figura.

Existem três esquemas de *ES* diferentes: a primeira é uma estratégia envolvendo apenas dois membros, o simples mecanismo de seleção é caracterizado pelo termo: seleção $(1+1)$; a segunda estratégia é baseada na noção de população, a chamada *ES* de vários membros ou, estratégia $(\mu+1)$, com $\mu > 1$ indivíduos pais; a última estratégia de *ES* é a $(\mu+\lambda)$, indicando que μ indivíduos pais criam λ filhos pela recombinação e mutação, e os melhores μ indivíduos descendentes são selecionados, de modo determinístico, para substituir os pais.

Estratégia $(1+1)$-ES

O primeiro algoritmo de *ES* proposto utilizava um esquema mutação-seleção conhecido como *two-membered ES* ou $(1+1)$-ES (DIANATI; SONG; TREIBER, 2002). Segundo (BäCK; SCHWEFEL, 1993), o $(1+1)$-ES trabalhava com um indivíduo, o qual criava um descendente por meio de mutação, sendo que o melhor entre os dois era selecionado de modo determinístico para integrar a próxima geração. A mutação sofrida obedecia à distribuição nor-

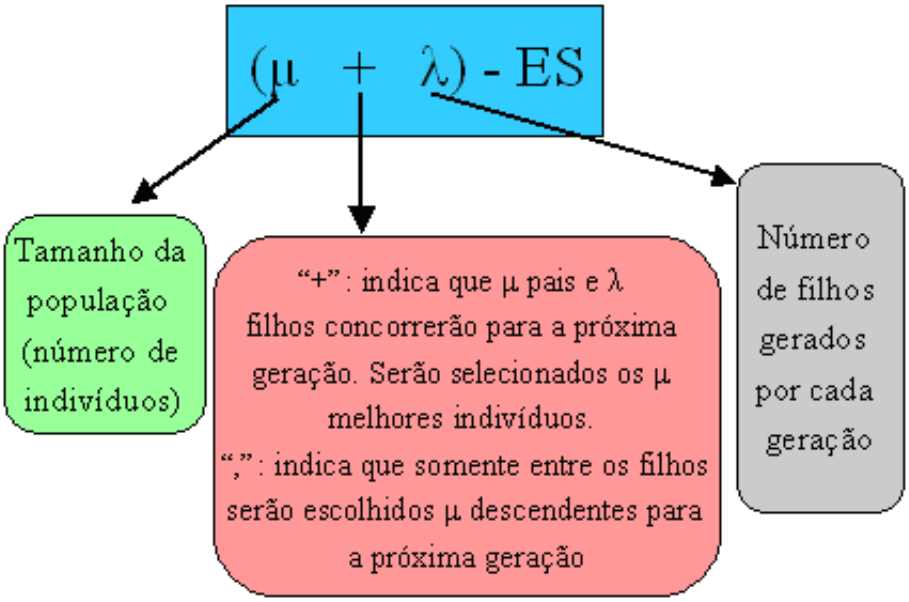


Figura 4.2: Notação de *ES*

mal, ou seja, pequenas alterações têm maior probabilidade de ocorrer do que grandes alterações.

Para obter a taxa de mutação ótima desse esquema, Rechenberg calculou as taxas de convergência de duas funções modelo e seus desvios-padrão ótimos para mutações de sucesso (DIANATI; SONG; TREIBER, 2002). Disso, ele postulou sua regra de sucesso $1/5$ (RECHENBERG, 1973): "A proporção de mutações de sucesso para todas as mutações deve ser $1/5$. Se for maior que $1/5$, incremente a variância; se for menor, diminua a variância de mutação".

Esta *ES* não trabalha com população de indivíduos (BäCK; SCHWEFEL, 1993). A 'população' consiste de um indivíduo pai (um vetor real) e um descendente, criado pela adição de números aleatórios de distribuição normal. Um algoritmo simples pode ser visto de maneira mais detalhada no algoritmo 2. Note a necessidade do parâmetro σ (passo 2) indicando a variância necessária para a mutação.

Algoritmo 2 Algoritmo $(1+1)$-ES (DIANATI; SONG; TREIBER, 2002).

1. Escolha um único vetor pai $\mathbf{x}$ que contém n parâmetros ($\mathbf{x}$ é um vetor n-dimensional). Cada parâmetro é escolhido por um processo aleatório e satisfaz às restrições do problema.
2. Para o passo t, **crie um novo descendente por meio da mutação**. Para obter a mutação nesse método, adicione um vetor aleatório do tamanho de $\mathbf{x}$ com distribuição normal (média zero e desvio-padrão):
   $\mathbf{x}' = \mathbf{x} + N(0, \sigma)$
   Se for o caso, aplique a regra de sucesso $1/5$ sobre a variância.
3. **Compare as soluções x e x′**. Escolha **o melhor** membro para a próxima geração.
4. Repita os passos 2 e 3 até encontrar uma solução satisfatória, ou até que tenha se esgotado o tempo computacional (ou número de gerações).

Estratégia $(\mu + 1)$-ES

O procedimento anterior foi aperfeiçoado, adicionando a noção de população. Rechenberg propôs o *multimembered ES* onde $\mu > 1$ pais participam na geração de 1 (um) descendente, sendo criada uma nova estratégia: $(\mu + 1)$-ES. Nesse método, todos os pais têm as mesmas probabilidades de casamento e, como o *two-membered ES*, de todos os pais e o descendente, o membro de menor ajuste da população é eliminado em cada geração (DIANATI; SONG; TREIBER, 2002). Os demais indivíduos permanecem para a próxima geração.

A $(\mu + 1)$-ES não é muito utilizada, contudo, esta estratégia direcionou para avanços propostos por Schwefel em 1975 (SCHWEFEL, 1975) (SCHWEFEL, 1977) (SCHWEFEL, 1981) para habilitar a auto-adaptação de parâmetros como o desvio-padrão das mutações. Os algoritmos $(\mu + 1)$-ES, segundo (DIANATI; SONG; TREIBER, 2002), implementaram auto-adaptação submetendo os parâmetros de evolução (desvio-padrão de mutações) no processo de criação de novos indivíduos. A extensão desta estratégia foi a criação do parâmetro para indicar o número de descendentes; a estratégia resultante será descrita a seguir.

A motivação de estender a $(\mu + 1)$-ES para a (μ, λ)-ES e $(\mu + \lambda)$-ES tem dois motivos (SCHWEFEL, 1977) (SCHWEFEL, 1981): fazer o uso da computação paralela e permitir a auto-adaptação de parâmetros estratégicos, como o desvio-padrão σ das mutações. Ao invés de mudar o σ de maneira determinística externa ao algoritmo, Schwefel viu o *sigma* como uma parte da informação genética de um indivíduo. Com isso, o desvio padrão passa a ser um assunto para a recombinação e mutação. Espera-se que os indivíduos que melhor ajustem os parâmetros estratégicos sejam os que tenham um desempenho melhor. Logo, a seleção favoreceria então estes indivíduos que, cedo ou tarde, dominarão a população.

Auto-adaptação de Parâmetros Estratégicos

A auto-adaptação de parâmetros estratégicos é uma das características mais importantes para o sucesso de Estratégias Evolucionárias (BÄCK; FOGEL; MICHALEWICZ, 2000), pois utiliza princípios evolucionários para a busca no espaço de variáveis e no de parâmetros estratégicos.

O termo "parâmetros estratégicos" refere-se a parâmetros que controlam o processo evolucionário de busca, tais como taxas de mutação, variâncias de mutação e probabilidades de recombinação. Normalmente, os parâmetros estratégicos são auto-adaptados ao nível dos indivíduos e incorporados à sua representação juntamente com o conjunto de variáveis, ou seja, o indivíduo é o conjunto de variáveis e o conjunto de parâmetros estratégicos (BÄCK; FOGEL;

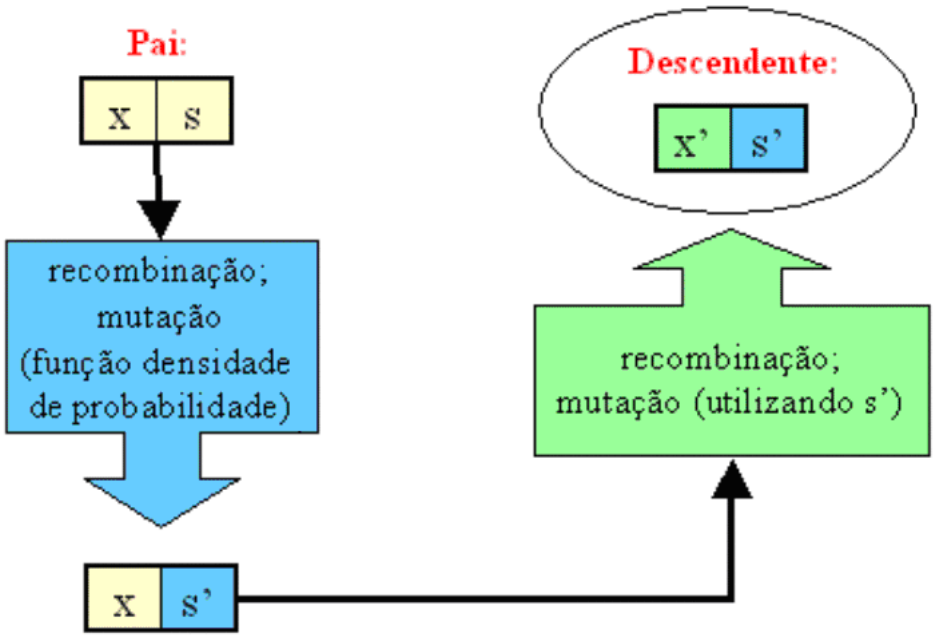


Figura 4.3: Criação de um descendente em *ES*

MICHALEWICZ, 2000).

Para um indivíduo $a = (\mathbf{x}, \mathbf{s})$ consistindo de um vetor de variáveis $\mathbf{x}$ e um conjunto de parâmetros de estratégia $\mathbf{s}$, a mutação é feita de acordo com alguma função de densidade de probabilidade. Na mutação, o vetor de parâmetros estratégicos $\mathbf{s}$ produz $\mathbf{s}'$, e os novos parâmetros de estratégia são utilizados para a mutação do vetor de variáveis $\mathbf{x}$ em $\mathbf{x}'$ (BÄCK; FOGEL; MICHALEWICZ, 2000). Esse processo é ilustrado na Figura 4.3 (VRIESMANN, 2005).

Detalhes sobre a auto-adaptação e mutação para *ES* podem ser encontrados em BÄCK, FOGEL e MICHALEWICZ (2000).

Estratégias (μ, λ)-ES e $(\mu + \lambda)$-ES

O $(\mu + \lambda)$-ES especifica que μ pais produzem λ descendentes $(\mu < \lambda)$ pela recombinação, mutação e seleção. Os descendentes competem com seus pais na seleção dos μ melhores indivíduos para a próxima geração. Assim, os pais sobrevivem até que sejam substituídos por descendentes melhores, podendo até ser possível que indivíduos bem adaptados sobrevivam por várias gerações. Esta característica traz algumas deficiências para a estratégia (BÄCK; HOFFMEISTER; SCHWEFEL, 1991):

- Em problemas com o ótimo variando no tempo, esta estratégia pode ficar em ótimos locais

Tabela 4.1: Operadores de Recombinação

Operador de Recombinação	Descrição
Sem recombinação	Selecione aleatoriamente um pai e deixe $x''_t = x_t$.
Discreto	Selecione aleatoriamente dois pais a e b
	E deixe $x'_t = x_{t,a}$ ou $x'_t = x_{t,b}$ com probabilidade igual.
Intermediário	Selecione randomicamente dois pais a e b
	E deixe $x'_t = 1/2(x_{t,a} + x_{t,b})$.
Discreto global	Selecione um novo par de a e b pais para cada parâmetro x_t.
	Faça: $x'_t = (x_{at,1}$ ou $x_{bt,2})$ com probabilidade igual.
Intermediário global	Selecione um novo par de a_t e b_t pais para cada parâmetro x_t.
	Faça: $xt' = 1/2(x_{at,1} + x_{bt,2})$.

se o conjunto de parâmetros internos se tornarem incapacitados para 'pular' para um novo campo de possíveis melhoras.

- O mesmo acontece se a medida de *fitness* (função objetivo) ou o ajuste das variáveis objetivo estão sujeitos a ruídos (i.e., parâmetros experimentais).

Para evitar estes efeitos, Schwefel investigou as propriedades onde somente os descendentes participam da seleção, limitando o tempo de vida dos indivíduos para apenas uma geração. O tempo de vida permite o 'esquecimento' de parâmetros internos inapropriados. Isto pode resultar em fases curtas de recessão (sem alteração significativa do *fitness* durante algumas gerações), mas evita fases longas de estagnação devido aos parâmetros estratégicos mal adaptados (SCHWEFEL, 1987).

Esta estratégia (μ, λ)-ES e $(\mu + \lambda)$-ES são diferenciadas pelo mecanismo de seleção (BACK; RUDOLPH; SCHWEFEL,). A estratégia $(\mu + \lambda)$-ES forma a próxima população a partir dos melhores pais e descendentes, enquanto na estratégia (μ, λ)-ES os melhores indivíduos são selecionados somente dos descendentes.

Algoritmo Geral de *ES*

Um algoritmo geral de (μ, λ)-ES e $(\mu + \lambda)$-ES é expresso no algoritmo 3, adaptação de (DIANATI; SONG; TREIBER, 2002). Pode-se observar que as funções principais de cada passo estão enfatizadas em negrito. Existem 5 (cinco) tipos de recombinação realizadas no passo 2, citados e descritos na Tabela 4.1 (aconselha-se a consulta de Dianati, Song e Treiber (2002) para eventuais dúvidas). Note que na criação da população descendente da geração t (Passo 2 - algoritmo 3), o operador de mutação é aplicado tanto para os parâmetros quanto para as variâncias.

Para concluir, são citadas, a seguir, alterações importantes desde a formulação das ideias básicas sobre *ES* (BÄCK; HOFFMEISTER; SCHWEFEL, 1991):

Algoritmo 3 Algoritmo Geral de ES.

1. Escolher μ vetores pai que contém n parâmetros $\mathbf{x_t} = (\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n})$. Cada parâmetro é escolhido por um processo aleatório e satisfaz as restrições do problema.
2. **Criar λ novos descendentes** por meio de recombinação de μ pais e de mutação (como no passo 2 do algoritmo 2)

Criação da população descendente da geração t com o seguinte algoritmo:
População corrente: P^t
Descendente intermediário: $\mathbf{x'_t} = (\mathbf{x'_1}, \mathbf{x'_2}, ..., \mathbf{x'_m})$
Operador de mutação: M
Meta-controle de tamanho de passo: $\Delta\sigma$
Operador de recombinação: R
$(\mathbf{x'_t}, \sigma') = R(P^t)$
$(\mathbf{x''_t}, \sigma'') = M[(\mathbf{x'_t}, ')]$
$\sigma'' = \sigma' . \exp(N(0, \Delta\sigma))$
$\mathbf{x''_t} = \mathbf{x'_t} + N(0, \sigma')$

3. **Escolha μ soluções mais adaptadas** para a próxima geração:
Se $(\mu + \lambda)$-ES: Selecione a próxima geração da população $(\mu + \lambda)$ de todos os pais e descendentes.
Se (μ, λ)-ES: Selecione a próxima geração da população λ de descendentes.
4. Repetir os passos 2 e 3 até encontrar uma solução satisfatória, ou até que tenha se esgotado o tempo computacional (ou número de gerações).

- Cálculos sobre a taxa de convergência analítica, dos quais resultaram no desenvolvimento da regra de sucesso 1/5 (RECHENBERG, 1973).
- A introdução do conceito de população ao invés de apenas um único indivíduo, que permitiu o processo de recombinação sexuada.
- Processo de auto-aprendizado dos parâmetros estratégicos pela incorporação destes parâmetros dentro do conjunto de variáveis herdados geneticamente.
- A estratégia (μ, λ)-ES, que introduziu o princípio de esquecimento (importante para ambientes de constante mudança), bem como uma medida contra a sobre-adaptação (especialmente a dos parâmetros estratégicos).

4.3 Abordagem Proposta

Nesta seção, apresenta-se a primeira abordagem da tese que tem como hipótese verificar se a combinação das técnicas de *boosting* e estratégia evolucionária podem atingir o equilíbrio entre os critérios precisão e generalidade.

A representação de uma hipótese define como os exemplos são selecionados, o que

influencia no resultado da tarefa de classificação. Portanto, uma análise da representação se faz necessária para a obtenção de bons resultados em termos de precisão e generalidade e, assim, ser possível o equilíbrio entre os critérios. O *ranking* dos exemplos gerados por uma hipótese linear é analisado e apresentada uma justificativa para a criação de uma nova linguagem de representação na seção 4.3.1.

Visando ao equilíbrio entre os critérios contraditórios, a nova representação utiliza a ideia de intervalos para agrupar os exemplos de uma mesma classe e é apresentada na seção 4.3.3. Duas medidas de avaliação dos intervalos - entropia (*IQ*) e tamanho - são apresentadas na seção 4.3.4. O *IQ* indica a pureza do conjunto que, por sua vez, esta relacionada à precisão e o tamanho diz respeito à generalidade da hipótese. Construindo um conjunto com os melhores intervalos dentro dos critérios de *IQ* e tamanho, tem-se um conjunto de intervalos não-dominados. Os intervalos não-dominados que formam uma hipótese são discutidos na seção 4.3.5. Para avaliar a hipótese, foi criada uma medida que calcula a área abaixo da curva formada pelos intervalos não-dominados (*AUCIQ*). Esta medida considera a qualidade dos intervalos que a formam e é descrita na seção 4.3.6.

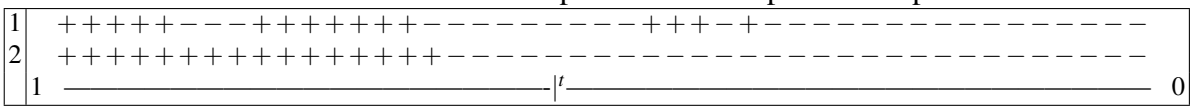
Com a nova linguagem de representação e suas medidas de avaliação criadas, um novo algoritmo também se faz necessário para que a proposta pudesse ser avaliada. Este algoritmo não deveria considerar apenas a criação isolada de hipóteses e suas combinações, mas também a evolução de um conjunto de classificadores. Assim, o algoritmo foi baseado na combinação de boosting e estratégia evolucionária (*ES*). Os trabalhos de boosting isolado mostram a combinação de classificadores; já a nossa abordagem, busca a evolução de conjuntos de classificadores. Pelos bons resultados em termos de *AUC* (SEBAG; AZE; LUCAS, 2003), acredita-se nas chances da somatória de *ES* e *boosting* para evoluir com sucesso tais conjuntos. A abordagem é finalizada com os experimentos na seção 4.3.8 e a discussão na seção 4.3.9.

4.3.1 Justificativa

Considerando problemas de classificação binária, os exemplos definidos pela equação 2.1 terão como elementos do conjunto Y as classes: positivo e negativo, $Y = \{-1, +1\}$. E, se for considerado que os atributos dos exemplos estão quantificados em valores reais, o conjunto X será um subconjunto dos vetores reais n-dimensional. Será considerada a função linear como a linguagem de representação das hipóteses.

Dada uma hipótese z_i, são calculados os valores $h_i(\mathbf{x_k})$ (equação 2.3) para todos os exemplos $\mathbf{x_k} \in X$. Uma hipótese ideal, com relação à ordenação, é aquela que coloca todos os exemplos positivos antes dos negativos, ou vice-versa. Em outras palavras, a melhor hipótese

Tabela 4.2: Classe dos Exemplos Ordenados por uma Hipótese.



divide os exemplos em dois grupos distintos. A Tabela 4.2 contém a ordem das classes dos exemplos de duas hipóteses diferentes. Na primeira linha, observam-se 47 exemplos e suas classes (+=positivo e −=negativo) ordenados pelo valor calculado pela equação 2.2. Note que na 2^a linha da tabela, os exemplos são ordenados por uma hipótese ideal com relação à ordenação. Na última linha, existe uma hipótese cuja linguagem de representação é uma função linear clássica (equação 2.3). Nesta linha da tabela existe o segmento de reta horizontal indicando valores de 0 a 1 e uma linha vertical indicando o limiar t. A hipótese classifica corretamente o conjunto de exemplos da linha 2, diferentemente da primeira linha. Note que mesmo alterando o limiar t, os exemplos da 1^a linha não podem ser classificados corretamente, devido a uma limitação da linguagem de representação. Por isso, propõe-se uma pequena modificação na forma de representação dos classificadores descrita a seguir.

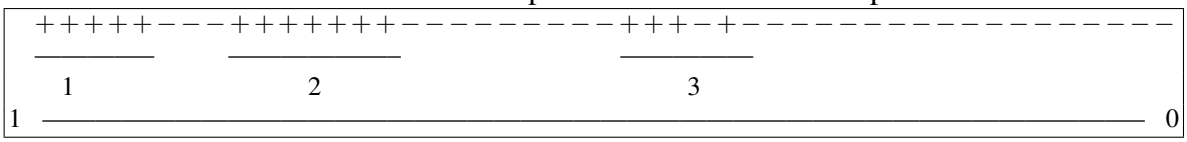
4.3.2 Representação das hipóteses

A representação dos modelos foi estendida com a ideia dos intervalos para a hipótese e com a ordenação dos exemplos baseados em seus valores. O intervalo j de uma hipótese i é representado por l_{ij} e definido pela equação 4.3; o conjunto de ni intervalos da hipótese i é representado por L_i, equação 4.5. O classificador passa a ser associado a um vetor de pesos (vetor $\mathbf{w_i} \in \Re^n$ da função $g_i(\mathbf{x_k})$ definido pela equação 2.2) e um conjunto de intervalos. A equação 4.6 indica um classificador C_i, formado pelo vetor de pesos $\mathbf{w_i}$ e um conjunto L_i de ni intervalos associados. Já a equação 4.7 indica um classificador, vetor de pesos $\mathbf{w_i}$, e um único intervalo $l_i j$.

Seja $z : x \rightarrow \{-1, +1\}$ uma hipótese que classifica um exemplo através da equação 4.4; para cada exemplo $\mathbf{x_k}$ é calculado um valor real (dado por $g_i(\mathbf{x_k})$ definido pela equação 2.2), se este valor pertence ao intervalo l_{ij}, o exemplo é classificado como positivo e pode-se dizer que a hipótese associa o exemplo à classe positiva; caso contrário, o exemplo será classificado como negativo.

$$l_{ij} = [a_{ij}, b_{ij}], \quad a_{ij} \in \Re, \quad 0 \leq a_{ij} < 1, \quad b_{ij} \in \Re, \quad 0 < b_{ij} \leq 1, \quad a_{ij} < b_{ij} \tag{4.3}$$

Tabela 4.3: Intervalos de uma Hipótese e Classes dos Exemplos Ordenados.



$$z(x) = \begin{cases} +1, & g_i(\mathbf{x_k}) \in l_{ij} \\ -1, & caso\ contrário \end{cases} \tag{4.4}$$

$$L_i = ([a_{i.1}, b_{i.1}], ..., [a_{i.ni}, b_{i.ni}]) \tag{4.5}$$

$$C_i = (\mathbf{w_i}, L_i) = ((w_{i.1}, ..., w_{i.n}), ([a_{i.1}, b_{i.1}], ..., [a_{i.ni}, b_{i.ni}])) \tag{4.6}$$

$$C_{ij} = (\mathbf{w_i}, l_{ij}), \quad l_{ij} \in L_i, \quad \mathbf{w_i} = (w_{i.1}, ..., w_{i.n}), \quad l_{ij} = [a_{ij}, b_{ij}] \tag{4.7}$$

4.3.3 Intervalos de uma Hipótese

A ideia inicial é a extensão da representação das hipóteses. Observando a possibilidade de ordenação dos exemplos, foi criada a ideia de intervalo que contém exemplos. A ideia é criar intervalos para as hipóteses que definam um conjunto de exemplos de uma mesma classe. Na Tabela 4.3 observam-se três possíveis intervalos representados por segmentos de retas e que estão abaixo de todas as classes dos exemplos ordenados pelo valor de $g_i(\mathbf{x_k})$.

O intervalo j de uma hipótese z_i é dado pela equação 4.3 e o conjunto de todos os intervalos da hipótese por 4.8. Pode-se gerar um ou vários intervalos para cada hipótese. A ideia é que cada hipótese tenha no máximo ni intervalos, sendo $1 \leq ni \leq m$. Cada hipótese deve ter obrigatoriamente um intervalo associado e não mais do que o número de exemplos m. Se existirem mais intervalos que o número de exemplos, ou os intervalos passam a ser redundantes (intervalos contidos em outros intervalos), ou não cobrem nenhum exemplo.

$$L_i = \{l_{ij} = [a_{ij}, b_{ij}]_{j=1,...,ni} | a_{ij} \in \Re, 0 \leq a_{ij} < 1, b_{ij} \in \Re, 0 < b_{ij} \leq 1, a_{ij} < b_{ij}\} \tag{4.8}$$

Os intervalos criam a ideia de exemplos cobertos. A equação 4.9 indica se o exemplo é coberto pelo classificador. Os intervalos de um classificador cobre um exemplo ε_k (equação

2.1) se o valor de $g_i(\mathbf{x_k})$ (equação 2.2) estiver entre algum intervalo pertencente ao conjunto de intervalos da hipótese L_i (equação 4.8). O conjunto de todos os exemplos que são cobertos por L_i da hipótese i é representado pela definição 4.10.

$$cob(e_k, C_i) = \begin{cases} 1, & C_i = (\mathbf{w_i}, L_i), \exists [a_{ij}, b_{ij}] \in L_i, a_{ij} \leq < \mathbf{w_i}, \mathbf{x_k} > \leq b_{ij} \\ 0, & caso\ contrário \end{cases} \tag{4.9}$$

$$\xi_i = \{\varepsilon_k = (\mathbf{x_k}, y_k) | \quad cob(\varepsilon_k, C_i) = 1\} \tag{4.10}$$

A indicação se um exemplo é coberto, saída $+1$, por um intervalo j de uma hipótese i é dado pela equação 4.11. Todos os exemplos que são cobertos pelo intervalo j da hipótese i são representados pela equação 4.12. O conjunto dos exemplos positivos do intervalo j é definido na equação 4.13, e o conjunto dos negativos é definido pela expressão 4.14.

$$cob(e_k, C_{ij}) = \begin{cases} 1, & C_{ij} = (\mathbf{w_i}, L_{ij}), \quad l_{ij} = [a_{ij}, b_{ij}] \in L_i,\ a_{ij} \leq < \mathbf{w_i}, \mathbf{x_k} > \leq b_{ij} \\ 0, & caso\ contrário \end{cases} \tag{4.11}$$

$$\xi_{ij} = \{\varepsilon_k = (\mathbf{x_k}, y_k) / cob(\varepsilon_k, C_{ij}) = 1\} \tag{4.12}$$

$$\xi_{ij}^{+} = \{(\mathbf{x_k}, y_k) \in \xi_{ij} / y_k = 1\} \tag{4.13}$$

$$\xi_{ij}^{-} = \{(\mathbf{x_k}, y_k) \in \xi_{ij} / y_k = -1\} \tag{4.14}$$

Um intervalo pode ser definido como intervalo positivo ou intervalo negativo. O intervalo positivo é aquele que possui mais exemplos positivos do que negativos ($|\xi_{ij}^{+}| > |\xi_{ij}^{-}|$); o negativo, aquele que possui mais exemplos negativos do que positivos ($|\xi_{ij}^{+}| \leq |\xi_{ij}^{-}|$). A equação 4.15 indica o conjunto dos intervalos positivos de uma hipótese i, e o conjunto dos intervalos negativos da mesma hipótese é representado pela equação 4.16.

$$L_i^{+} = \{l_{ij} / |\xi_{ij}^{+}| > |\xi_{ij}^{-}|\} \tag{4.15}$$

$$L_i^- = \{l_{ij} / |\xi_{ij}^+| > |\xi_{ij}^-|\} \tag{4.16}$$

4.3.4 Qualidade dos intervalos

Dada a definição anterior, podem ser definidas algumas medidas de eficiência dos intervalos: *IQ* e tamanho do intervalo. Definida com base na equação 2.16 da seção 2.3.4, *IQ* é a entropia ou pureza do conjunto de exemplos.

Medida *IQ*

Com a modificação da representação das hipóteses, faz-se necessária a criação de novas medidas para a avaliação dos intervalos. A proporção de exemplos positivos cobertos pelo intervalo j da hipótese i (equação 4.17) é uma destas medidas. Na equação 4.18 encontra-se outra medida: proporção de exemplos negativos cobertos pelo intervalo j da hipótese i. Estas duas proporções são complementares, ou seja, $p_{ij}^+ + p_{ij}^- = 1$. Considerando os exemplos ordenados por uma hipótese e os três intervalos (1, 2 e 3) na Tabela 4.3, observa-se que o intervalo 3 cobre quatro (4) exemplos positivos e dois (2) exemplos negativos. O total de exemplos cobertos ($\left|\xi_{ij}\right|$) deste intervalo é seis (6), a proporção de exemplos positivos (p_{ij}^+) é $0,67 = \frac{4}{6}$ e a proporção de exemplos negativos (p_{ij}^-) é $0,33 = \frac{2}{6}$.

A medida de precisão de um intervalo j da hipótese i (equação 4.19) depende se o l_{ij} é um intervalo positivo ($l_{ij} \in L_i^+$) ou um intervalo negativo ($l_{ij} \in L_i^-$). Se o intervalo for positivo, a precisão é calculada pela divisão do número de exemplos positivos cobertos pelo total de exemplos cobertos (equação 4.17). Se o intervalo for negativo, a precisão é a divisão do número de exemplos negativos pelo total de exemplos cobertos (equação 4.18). No exemplo descrito na Tabela 4.3, o intervalo 3 é considerado positivo pois o total de exemplos positivos cobertos (4) pelo intervalo é maior que o total de exemplos negativos cobertos (2). Portanto, a precisão do intervalo do exemplo é $p_{ij} = 0,67 = p_{ij}^+$.

$$p_{ij}^+ = |\xi_{ij}^+| / |\xi_{ij}| \tag{4.17}$$

$$p_{ij}^- = |\xi_{ij}^-| / |\xi_{ij}| \tag{4.18}$$

$$p_{ij} = \begin{cases} |\xi_{ij}^{+}|/ & |\xi_{ij}|,\ l_{ij} \in L_i^{+} \\ |\xi_{ij}^{-}|/ & |\xi_{ij}|,\ \textit{caso contrário} \end{cases} \tag{4.19}$$

A pureza de um intervalo é indicada pela quantidade de informação. A equação 2.16 está redefinida na equação 4.20 em termos de *IQ* e precisão (p_{ij}). Note que se pode utilizar p_{ij}^{+} ou p_{ij}^{-} no lugar de p_{ij}. Deve ser considerado que $0 * \log_2 0$ seja igual a 0 (zero). O *IQ* de um intervalo é um valor entre 0 (zero) e 1 (um). O *IQ* tem valor zero quando todos os exemplos cobertos pelo intervalo pertencerem à mesma classe e é igual a 1 (um) quando o número de exemplos positivos é igual ao número de negativos. Ou seja, os melhores intervalos, ou os intervalos mais puros (com maior número de exemplos da mesma classe), são aqueles cujo *IQ* está próximo de zero. Considerando novamente o exemplo do intervalo 3 da Tabela 4.3, o *IQ* associado ao intervalo 3 é $0,9183 = -0,67 * \log_2 0.67 - (1-0,67) * \log_2 (1-0,67)$, ou seja, não é um bom intervalo segundo sua pureza.

$$IQ(l_{ij}) = -p_{ij} \log_2 p_{ij} - (1-p_{ij}) \log_2 (1-p_{ij}) \tag{4.20}$$

Tamanho do Intervalo

A generalidade dos intervalos é calculada pelo número de exemplos cobertos pelo intervalo (ξ_{ij}). Este número está num intervalo de zero até o número total de exemplos. O interessante é a presença de intervalos não-pequenos, porém não é interessante se ter intervalos muito grandes, cujo tamanho ultrapassa o número total de exemplos positivos ou negativos, pela simples razão de que, se o tamanho ultrapassar estes totais, o *IQ* do intervalo passa a ser prejudicado. Nos exemplos de intervalos da Tabela 4.3 (intervalos 1, 2 e 3), os tamanhos dos intervalos são, respectivamente, 5 (cinco), 7 (sete) e 6 (seis).

4.3.5 Intervalos não-dominados

O conceito de intervalos não-dominados será posteriormente útil para a definição de qualidade das hipóteses. Uma hipótese i possui vários intervalos associados (L_i da equação 4.8). Muitos intervalos podem ser redundantes ou de qualidade inferior a outros. Por exemplo, se existirem dois intervalos l'_{ij} e l''_{ij} de mesmo tamanho, o primeiro com $IQ(l'_{ij}) = 0,4$ e o segundo com $IQ(l''_{ij}) = 0$, o segundo será preferível ao primeiro ou, em outras palavras, o segundo domina o primeiro. Em outro caso, onde $IQ(l'_{ij}) = IQ(l''_{ij})$, se os tamanhos forem diferentes a dominância é para o intervalo de maior tamanho.

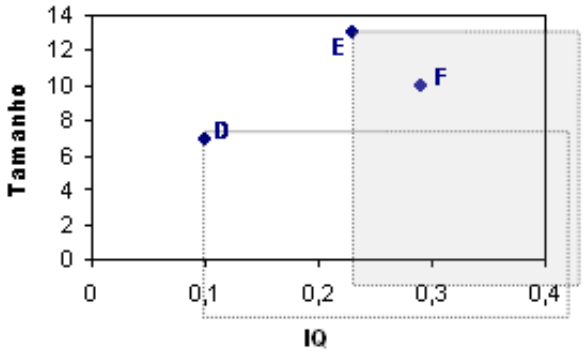


Figura 4.4: Exemplo de Dominância

No gráfico da Figura 4.4 é possível visualizar a área de dominância de um intervalo. É associada a cada intervalo uma tupla (valor do *IQ* e o tamanho), que, representado num gráfico *IQ* X Tamanho, é representado por um ponto. Na Figura 4.4 observam-se três pontos: D, E, F, e duas áreas de dominância (uma para o ponto D e outra para o ponto E). A área de dominância do ponto E (área sombreada) isola os intervalos que são dominados pelo ponto E (ou que o ponto E domina). Pode-se observar que o ponto F é dominado pelo ponto E, uma vez que F se encontra dentro da área de E e possui tamanho menor e *IQ* maior. Em contrapartida, não existe dominância entre os pontos D e E. Observe que os pontos estão contidos apenas em sua área de dominância.

Na Figura 4.5 pode-se observar alguns pontos críticos. O ponto A tem um bom valor para o *IQ*, contudo, o seu intervalo associado não é interessante, pois o seu tamanho é zero. O ponto C possui um bom tamanho, mas a sua puridade é a pior possível. O ideal para o intervalo se encontra em B, que representa um intervalo mais puro e com o maior tamanho possível. Quanto maior a proximidade com o ponto B, melhor será o intervalo.

Dado um conjunto de todos os intervalos (L_i) da hipótese z_i, pode-se separar estes intervalos em dois grupos: o grupo dos intervalos dominantes (ou não-dominados) e o grupo dos intervalos dominados. A equação 4.21 indica o conjunto dos intervalos não-dominados da hipótese *i*; o conjunto dos intervalos dominados é formado pelos intervalos que estão contidos L_i e não pertencem à L_i^*.

Após esta separação de intervalos, os intervalos não-dominados e o vetor de pesos ($\mathbf{w_i}$) fazem uma tupla definida por 4.22, que é o classificador com os *nd* intervalos dominantes.

$$L_i^* = \{l_{ij} \in L_i / \sim \exists l_{ip} \in L_i : (IQ(l_{ip}) < IQ(l_{ij})) \& (|\xi_{ip}| > |\xi_{ij}|)\} \tag{4.21}$$

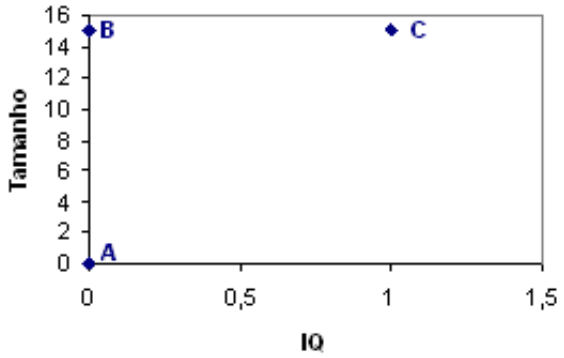


Figura 4.5: Exemplo de Dominância 2

$$C_i^* = (\mathbf{w_i}, L_i^*) = ((w_1, ..., w_n), ([a_{i.1}, b_{i.1}], ..., [a_{i.nd}, b_{i.nd}])) \tag{4.22}$$

4.3.6 Qualidades das Hipóteses

Outra consequência da criação do conceito dos intervalos é a necessidade de uma medida de qualidade do classificador. Como visto anteriormente, quanto maior o tamanho dos intervalos e quanto menor o *IQ*, melhor será o intervalo em questão. Para unir os dois, uma nova medida foi criada para indicar a generalidade e a precisão dos intervalos. Esta medida foi inspirada em *AUC* para indicar a nova medida de qualidade das hipóteses.

A partir das medidas apresentadas para os intervalos, pode-se obter o gráfico dos intervalos não-dominados (Figura 4.7). Na Figura 4.6, que contém 50 intervalos de uma hipótese representados por pontos (*IQ*, Tamanho), observa-se a existência de pontos que não são interessantes. Se um gráfico for construído com os pontos de intervalos não-dominados, será obtido o gráfico da Figura 4.7. Estes pontos formam a fronteira de Pareto e são os intervalos relevantes para a hipótese em questão.

Se os pontos não-dominados da Figura 4.7 forem ligados por semi-retas pode-se calcular a área que está abaixo desta curva, conforme a Figura 4.8. A área abaixo da curva formada pelos intervalos não-dominados (*AUCIQ*) pode ser utilizada como um indicador de qualidade do classificador. Pode-se observar que, quanto maior a área, melhor será o conjunto dos classificadores, uma vez que os pontos não-dominados estão mais próximos do ponto crítico B (Figura 4.5).

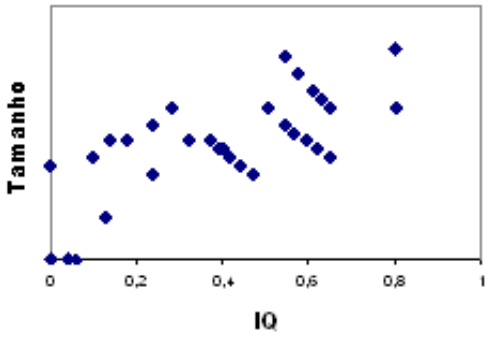


Figura 4.6: Intervalos de uma Hipótese

4.3.7 Algoritmo Proposto

Para conseguir um grupo de regras com características específicas, no caso com alta precisão e com generalidade, foi utilizada uma nova representação de hipóteses baseada em intervalos, na qual cada classificador é representado por uma tupla de pesos $\mathbf{w_i}$ e conjunto de intervalos L_i (equação 4.6). Para evoluir os classificadores foi criado o algoritmo BES (BES), código no algoritmo 4, para juntar as estratégias de *ES* e *boosting* para lidar eficientemente com a nova representação. Como o AdaBoost, um algoritmo de *boosting*, o algoritmo possui uma estratégia de pesos para os exemplos. Utilizando a *ES*, os pesos são utilizados como um indicador para a evolução conjunta dos classificadores.

O algoritmo 4 recebe como entrada dois parâmetros: um conjunto de exemplos de treinamento (ξ_e) e a porcentagem utilizada para a criação de intervalos (*pni*). Cada exemplo de treinamento tem associado um vetor $\mathbf{x} \in \Re^m$ e uma classe y.

O passo inicial do algoritmo 4 é estabelecer o peso inicial de cada exemplo ($d(\mathbf{x_k}) = 1/|\xi_e|$), onde $|\xi_e|$ é o número de exemplos do conjunto de treinamento, e a criação do conjunto

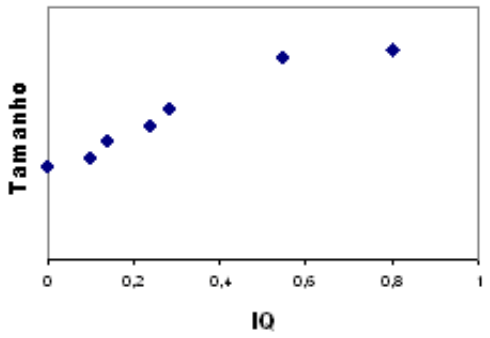


Figura 4.7: Intervalos não-dominados de uma Hipótese

Algoritmo 4 Algoritmo BES

Entrada:
ξ_{tr} - conjunto de treinamento
pni - porcentagem para criação de intervalos

Saída:
HI_v - Conjunto de hipóteses e intervalos

Passos Gerais:

0 Configurações iniciais
- Iniciar $d^0(\mathbf{x_k}) = 1/|\xi_{tr}|$, como sendo os **pesos dos exemplos**,
onde *m* é o número de exemplos para o conjunto de treinamento
- **Criar o conjunto de pesos das hipóteses** ($\mathbf{w_i}$ da função 2.2)

1 - **Criar conjunto de** *ni* **intervalos para cada hipótese** (L_i, equação 4.5)
$pni * |\xi_{tr}|$ tentativas para a criação dos intervalos
O conjunto de hipóteses e o conjunto dos intervalos formam o conjunto *HI*

2 Criar conjunto das hipóteses e intervalos não-dominados (HI^*)
- Para cada hipótese **selecionar os intervalos dominantes ou não-dominados**

3 Para cada exemplo $\mathbf{x_k}$ faça: (**votação**)
- Selecione de HI^*, as hipóteses e intervalos que contenham $\mathbf{x_k}$ para formar o conjunto $HI_{\mathbf{x_k}}$
- Achar a hipótese com o maior *AUCIQ*
- Somar $d^t(\mathbf{x_k})$ para as hipóteses que estão contidas em HI_{xk} = votos do classificador

4 Separar as hipóteses e intervalos que foram votados HI_v

5 Calcular a precisão de HI_v. Para cada $h_i \in HI_v$:
- Calcular a classe para cada $\mathbf{x_k} \in \xi_{tr}$, baseado no diagnóstico
. Calcular o diagnóstico do exemplo $diag(\mathbf{x_k})$
. Calcular a classe de $\mathbf{x_k}$
- Calcular a precisão de HI_v

6 Fazer a evolução das hipóteses usando $(\mu + \lambda)$-ES
- Fazer a mutação e/ou recombinação
- Selecionar a melhor hipótese, de acordo com o *fitness*, a partir dos pais e filhos para fazerem parte da próxima geração
Obs: *Fitness* da hipótese = soma dos votos recebidos
- Se o conjunto de hipótese votados pode ser a solução, então a solução final será este conjunto

7 Atualizar o vetor de pesos (**d**) de acordo com os exemplos que não foram bem classificados
$d^{t+1}(k) = \frac{d^t(k) exp(-\alpha^t y_k h^t(\mathbf{x_k}))}{Z^t}$

8 Voltar ao passo 1

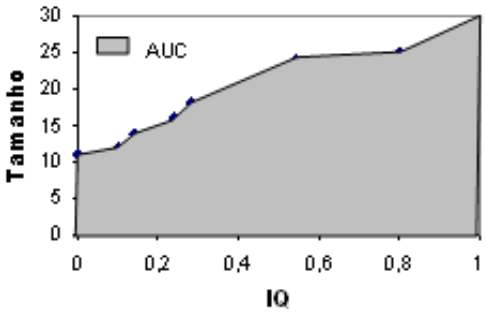


Figura 4.8: *AUC* (*IQ* X Tamanho)

inicial de hipóteses (indivíduos). O número de hipóteses criadas varia de acordo com o número de exemplos de uma base de dados. Quanto maior o número de exemplos, maior o número de hipóteses. O número de hipóteses é cerca de 10% do número de exemplos. A criação de uma hipótese *i* envolve a criação de um vetor de pesos ($\mathbf{w_i} \in \Re^m$) com valores aleatórios de 0 (zero) a 1 (um).

A iteração principal envolve a criação de intervalos (passo 1), a seleção dos intervalos não-dominados (passo 2) que serão votados pelos exemplos (passo 3), seleção (passo 4), avaliação (passo 5) das hipóteses votadas, evolução das hipóteses utilizando *ES* (passo 6) e a atualização de pesos segundo *boosting* (passo 7).

No passo 1, existem $pni * |\xi_e|$ tentativas para a criação do conjunto de *ni* intervalos (L_i, equação 4.5) para cada hipótese *i*. Ou seja, o número de tentativas nunca ultrapassará o número de exemplos. Cada hipótese associa um valor $g_i(\mathbf{x_k})$ a cada exemplo, fornecendo a cada conjunto uma ordem (*ranking*). De acordo com os exemplos agrupados desta ordem, são criados intervalos de modo que possam cobrir corretamente os exemplos positivos ou negativos, como na Tabela 4.3, na qual existem três intervalos ($ni = 3$). A criação dos intervalos é feita de modo determinístico para cobrir o maior número de exemplos. Ou seja, como cada hipótese ordena os exemplos de uma maneira específica, os intervalos são criados de maneira que melhor cubram os exemplos positivos. Se forem criados intervalos positivos, a hipótese será da classe positiva; caso contrário, em alguns casos a classe será negativa. O conjunto de todas as hipóteses e seus intervalos forma o conjunto *HI*.

No passo 2 são selecionados apenas os intervalos não-dominados de cada hipótese (HI^*). A dominância dos intervalos é avaliada com base nos valores de *IQ* e tamanho, garantindo que sejam selecionados apenas os melhores em termos de precisão e generalidade.

Depois, cada exemplo ($\mathbf{x_k}$) vota utilizando o seu peso ($d^t(\mathbf{x_k})$) na melhor hipótese

(passo 3). Para a votação, primeiro são selecionados as hipóteses e os intervalos não-dominados que cubram o exemplo $\mathbf{x_k}$ e que sejam da mesma classe para formação do conjunto $HI_{\mathbf{x_k}}$. Deste conjunto, seleciona-se a hipótese com o maior *AUCIQ*. No final do passo 3, a hipótese recebe como votação a soma dos pesos dos exemplos que a escolheram como a melhor hipótese.

O passo 4 seleciona apenas as hipóteses que foram 'votadas' (escolhidas por pelo menos um exemplo no passo 3), associando-as ao conjunto HI_v.

No passo 5, os votos recebidos pela hipótese no passo 3 são utilizados para o cálculo do diagnóstico ($diag(\mathbf{x_k})$). O diagnóstico do exemplo $\mathbf{x_k}$ é a soma dos votos das hipóteses com intervalos positivos menos a soma dos votos dos negativos. Se o diagnóstico é negativo, então o exemplo é classificado como negativo; caso contrário, o exemplo é classificado como positivo. Após todos os exemplos serem classificados pela hipótese h_i, a precisão de h_i é calculada.

A utilização da $(\mu + \lambda)$-ES para a evolução das hipóteses é feita no passo 6. A variável de avaliação (*fitness*) é o *AUCIQ*. O tamanho da população inicial é definido inicialmente com 10. Para cada indivíduo será gerado, através de mutação, λ indivíduos filhos, sobrevivendo o melhor indivíduo para a próxima geração.

Após a evolução do conjunto de classificadores, serão atualizados os pesos dos exemplos de acordo com sua classificação. A atualização é feita segundo passo 2.4 do algoritmo Adaboost (algoritmo 1). A ideia é diminuir o peso para os exemplos classificados corretamente e manter ou aumentar os pesos para os exemplos classificados incorretamente.

A seguir são apresentados experimentos com o algoritmo proposto.

4.3.8 Experimentos

Todos os experimentos conduzidos nesta avaliação foram feitos com conjunto de dados do UCI Machine Learning Repository (NEWMAN; BLAKE; MERZ, 1998) obtidos em junho de 2006. A Tabela 4.4 apresenta a descrição dos conjuntos de dados utilizados nos experimentos com o algoritmo BES. Os conjuntos de dados foram escolhidos pela facilidade de acesso aos resultados dos trabalhos relacionados. A Tabela contém o número da base de dados (coluna '#'), a descrição do conjunto de dados (coluna 'Conjunto'), o número de atributos (coluna 'At'), número de exemplos (coluna 'Exe') e o número de classes (coluna 'Cl') dos 5 (cinco) conjuntos de dados. A última coluna contém a porcentagem de positivos da base de dados (coluna '%(+)'). A dimensão de cada espaço de busca é dada por 'At' + 1 (classe). Os experimentos foram executados utilizando validação cruzada para 10 partições (*10-fold cross-validation* (WITTEN; FRANK, 1999)). A estratégia validação cruzada divide a base de exemplos em

Tabela 4.4: Descrição dos conjuntos de dados para teste do Boosting com Estratégia Evolucionária.

#	Conjunto	At	Exe	Cl	%(+)
1	german	21	1000	2	30,00
2	glass	10	214	6	7,81
3	heart	14	270	2	44,44
4	ionosphere	34	351	2	35,87
5	kr-vs-kp	37	3196	2	47,77

Tabela 4.5: Parâmetros utilizados para execução da ES.

Parâmetro	Valor
Boosting	10
Fitness	*AUCIQ*
Tamanho da População	10
Seleção	Torneio (2)
Porcentagem de Descendentes	700%
Estratégia de troca de descendentes	(+) Plus
Número mínimo de gerações	100
Número máximo de gerações	300
Número de geração melhora	200
Probabilidade de Mutação	1
Recombinação das variáveis objeto	discreto
Tipo de recombinação ES	global
Parâmetros de estratégia de recombinação de mutação	intermediate

10 (dez) conjuntos disjuntos de mesmo tamanho e com a mesma distribuição em cada classe. Destes, 9 (nove) são utilizados como a base de treinamento para construção dos classificadores e o décimo conjunto como base de teste para avaliação destes classificadores. O experimento é pareado, isto é, para todos os algoritmos foram utilizados os mesmos conjuntos de treinamento e de teste.

O algoritmo BES foi executado dez vezes para cada base de dados. Para cada uma delas o algoritmo apresenta como resultado o conjunto das hipóteses e intervalos (HI_v). Os parâmetros utilizados para o $(\mu + \lambda)$-ES escolhido para a evolução das hipóteses estão descritos na Tabela 4.5. É importante observar que vários testes empíricos foram feitos com outros parâmetros e os escolhidos foram os melhores e que são semelhantes aos utilizados num trabalho que utiliza ES para a otimização da *AUC* num problema real (SEBAG; AZÉ; LUCAS, 2003). A diferença foi o número mínimo e máximo de gerações de cada execução que, no conjunto com outros parâmetros, resultou em melhoria da precisão. A variável de avaliação para a evolução, *fitness* utilizado, é o *AUCIQ*. O tamanho da população inicial é de 10 indivíduos. Para cada indivíduo pai serão gerados, através de mutação, 7 indivíduos filhos, sobrevivendo o

Tabela 4.6: Valores médios de Precisão do algoritmo BES e C4.5 para cada base de dados com seus respectivos desvios-padrão.

#	BES	C4.5
1	69,40(4,12)	71,9(1,4)
2	91,60(1,93)	67,3(2,4)
3	67,41(22,07)	81,2(1,3)
4	86,35(5,73)	89,9(1,5)
5	83,29(8,60)	99,4(0,1)

melhor indivíduo para a próxima geração.

A comparação foi feita com o algoritmo C4.5 o qual foi escolhido por ser um algoritmo clássico de aprendizado de regras. A análise do algoritmo será feita de acordo com a precisão utilizando o teste Wilcoxon (WILCOXON, 1945) com 95% de nível de confiança.

Os resultados do algoritmo BES estão na primeira coluna da Tabela 4.6. A primeira coluna indica o número do conjunto de dados. A segunda coluna contém precisão e desvio-padrão entre parênteses. Na terceira coluna esta a precisão do algoritmo C4.5 (FRANK; WITTEN, 1998).

As cores cinza indicam os valores estatisticamente diferentes. A célula em cinza escuro indica que o resultado do algoritmo BES é estatisticamente melhor do que o algoritmo C4.5. Já as células em cinza claro indicam valores que são estatisticamente inferiores aos das precisões do C4.5. Em apenas um conjunto de dados (#2) o algoritmo BES é estatisticamente melhor. Nos outros quatro conjuntos de dados, o algoritmo BES teve um desempenho pior.

Analisando o conjunto de dados que o algoritmo BES teve um bom desempenho (#2), verificou-se a precisão entre os exemplos positivos (sensitividade) e a precisão entre os negativos (especificidade). Infelizmente, nota-se que a precisão entre os negativos chega a quase 100% e entre os positivos é praticamente zero. Ou seja, neste conjunto de dados todos os exemplos são classificados como negativos.

Para uma análise mais aprofundada do desempenho do algoritmo BES, a precisão nas bases de treinamento está na Tabela 4.7. Os resultados mostram a presença de *overfitting*, ou seja, há uma alta precisão na base de treinamento e um desempenho ruim na base de testes.

4.3.9 Discussão

Nos experimentos realizados com o algoritmo BES, os intervalos cobriram os exemplos na base de treinamento com precisão. Contudo, na base de testes, os resultados não foram

Tabela 4.7: Valores médios de Precisão das bases de treinamento do algoritmo BES e seus desvios-padrão.

#	Precisão
1	91,61(1,90)
2	92,63(0,59)
3	94,32(1,86)
4	97,97(0,98)
5	99,33(0,18)

satisfatórios. Por exemplo, em uma das bases, a maioria dos exemplos foi classificada como negativa.

Embora o algoritmo não apresente flexibilidade com as possibilidades de parâmetros, é importante observar que foram analisadas várias variações do algoritmo. Por exemplo, para a criação dos intervalos foram avaliadas as criações aleatória, determinística e considerados apenas intervalos positivos. Outras tentativas também foram feitas para a atualização de pesos. O resultado apresentado utiliza a mesma atualização de peso do Adaboost, porém, foram avaliadas diferentes atualizações de outros algoritmos como o *smooth margin* (RUDIN; SCHAPIRE; DAUBECHIES, 2004) e atualizações baseadas na *AUCIQ* das hipóteses que cobrem o exemplo.

A experiência mostra que a estratégia de pesos com estratégia evolucionária não foi eficiente para encontrar um conjunto de hipóteses baseado em intervalos. Entretanto, não é possível afirmar que o modelo apresentado não possa ser eficiente com a utilização de outras estratégias.

O equilíbrio entre precisão e generalidade continua a ser um desafio. Contudo, a ideia de uma nova linguagem de representação mostrou-se interessante pelo fato de, teoricamente, ser possível trabalhar com situações em que uma simples representação linear não é capaz de cobrir corretamente os exemplos. A linguagem de representação por intervalos poderia ser utilizada por outras estratégias em busca de bons classificadores.

Como uma forma de evitar o *overfitting*, é necessária uma maior investigação sobre a criação dos intervalos. Poder-se-ia diminuir a precisão de cada intervalo, desconsiderando exemplos considerados ruídos, ou investigar quais hipóteses possuem boa precisão. Porém, quando combinadas com outras hipóteses, prejudicam o conjunto final e precisam ser desconsideradas.

5 Critérios Biobjetivos para Maximização da *AUC*

A segunda abordagem desta tese investiga a seguinte questão: a criação de um conjunto de aproximação de regras não-ordenadas segundo critérios biobjetivos pode maximizar a *AUC*?

Na área de Aprendizado de Máquina, a curva *ROC* pode identificar visualmente o desempenho de um classificador (BRADLEY, 1997). Frequentemente, para o propósito de comparação de algoritmos de aprendizado, uma medida singular é preferida. Para tal, a área abaixo da curva *ROC* (*Area Under the ROC Curve* (AUC)) foi proposta para a construção e otimização de modelos de aprendizado (FERRI; FLACH; HERNANDEZ-ORALLO, 2002) (RAKOTOMAMONJY, 2004) (SEBAG; AZE; LUCAS, 2003). Quanto maior o valor da *AUC*, melhor é o conjunto de modelos. Consequentemente, a classificação pode ser considerada como um problema de otimização da medida *AUC*.

Considerando que o classificador desejado seja um conjunto de regras não-ordenadas, a maximização da *AUC* está diretamente relacionada à seleção correta das regras para cada conjunto de dados. A seleção é feita a partir das regras que podem ser geradas para cada conjunto de dados. O ideal para a formação do conjunto de regras resultado é criar apenas as regras com os critérios de avaliação mais desejáveis que favorecessem a maximização da *AUC*, diminuindo o esforço da criação desnecessária. Para a maximização, prefere-se que os exemplos positivos recebam mais votos das regras positivas do que das negativas. Ou seja, são preferidas regras com alta precisão entre os positivos (alta sensitividade) e entre os negativos (alta especificidade). Para tal, surge a hipótese de que um conjunto aproximado da fronteira de Pareto de acordo com os critérios de sensitividade e especificidade possa contribuir para a maximização da *AUC*. A fronteira procurada não é formada apenas com as melhores regras de acordo com a sensitividade **OU** especificidade, mas também, as regras que possuem precisão média entre os positivos **E**, também, entre os negativos, o que, na nossa conjectura, são características importantes para a maximização da *AUC*. A validação da questão é feita com a existência de um algoritmo determinístico que possa gerar a maior quantidade possível de regras para a formação do conjunto aproximado da fronteira de Pareto.

Após a validação da hipótese, é necessária a criação de abordagens que possam trabalhar com grandes bases de dados. A ideia é a utilização de metaheurísticas para gerar o conjunto de regras num único passo. Dentre as muitas metaheurísticas existentes, duas têm obtido bons resultados em suas áreas: nuvem de partículas (KENNEDY; EBERHART, 1995) e *Greedy Randomized Adaptive Search Procedures* (FEO; RESENDE, 1995).

Nuvem de partículas é uma técnica de otimização estocástica baseada em manter a diversidade da população, inspirada pelo comportamento social de pássaros à procura de alimento ou no modelo de cardume de peixes (VESTERSTROEM; RIGET, 2002). Embora a nuvem de partículas já tenha sido utilizada em classificação (SOUSA; SILVA; NEVES, 2003) (SOUSA; SILVA; NEVES, 2004), não foram exploradas as características multiobjetivas.

Greedy Randomized Adaptive Search Procedures (*GRASP*) é uma metaheurística muito utilizada e com bons resultados para problemas de otimização combinatória (FEO; RESENDE, 1995). *GRASP* é um processo iterativo com duas fases principais: uma fase de construção de soluções viáveis através de um procedimento guloso aleatório, e uma fase de busca por um mínimo local pertencente a uma dada vizinhança da solução construída na fase anterior. A fase de construção gulosa pode ser de grande valia para a geração das regras mais relevantes para a formação do conjunto aproximado da fronteira.

A seguir, na seção 5.1, é apresentada uma proposta para verificar a hipótese de que a sensitividade e a especificidade podem maximizar a *AUC*. Depois, serão apresentadas as duas propostas que utilizam metaheurísticas para a geração do conjunto de regras não-ordenado num único passo que maximize a *AUC* em grandes bases de dados. Na seção 5.2 apresenta-se a abordagem que utiliza a nuvem de partículas e, na seção 5.3, apresenta-se a utilização do *GRASP* para uma nova abordagem.

5.1 Sensitividade e Especificidade Maximizam a *AUC*?

Para validar a hipótese de que a seleção de regras de acordo com os critérios de sensitividade e especificidade maximizam a *AUC*, será criado especificamente um algoritmo determinístico. As regras serão criadas por um algoritmo de regras de associação e, para a formação dos conjuntos de aproximação, a seleção das regras será feita de acordo com os critérios biobjetivos citados. A seguir, prossegue-se com a definição de algoritmos de regras de associação, a explanação de um algoritmo clássico para a geração de regras, nossa solução criada e os experimentos para a validação da ideia.

5.1.1 Regras de Associação para Classificação

O progresso na tecnologia de código de barra permitiu as organizações de varejo colecionarem e armazenarem grandes quantidades de dados de vendas, os chamados *basket data* (AGRAWAL; IMIELINSKI; SWAMI, 1993). Com a grande quantidade de dados surgiu a necessidade de se obter as informações mais relevantes. Neste contexto, o algoritmo de regras de associação, ou *association rules*, foi introduzido por Agrawal, Imielinski e Swami (1993) e rapidamente se tornaram muito populares em função de sua aplicabilidade a problemas reais. O objetivo destes algoritmos é encontrar regras que possam descrever a base de dados. Para encontrar as regras de associação, os algoritmos deste tipo realizam uma busca global por regras que satisfazem algumas restrições como suporte e confiança mínima (BATISTA et al., 2006).

As regras geradas pelos algoritmos de associação podem ser utilizadas para a predição na chamada classificação associativa (LIU; HSU; MA, 1998). A classificação associativa utiliza os algoritmos de regras de associação para gerar regras com somente um consequente (atributo classe). Após a geração das regras, um método agrupa as regras para formação do classificador que melhor se adapte ao conjunto de dados como, por exemplo, um classificador com alta precisão.

Para calcular a *AUC* do conjunto com a utilização do processo de classificação por votação por pesos baseado na confiança, cada regra vota nos exemplos cobertos por ela. As regras positivas votam nos exemplos com valores positivos, já as regras negativas utilizam valores negativos. Por isso, para a correta classificação, a preferência é que os exemplos positivos recebam mais votos das regras positivas do que das negativas. Ou seja, são preferidas regras com alta precisão entre os positivos (sensitividade) e entre os negativos (especificidade). Sendo assim, espera-se que um conjunto não-ordenado de regras forme, de acordo com os critérios de sensitividade e especificidade, uma fronteira de Pareto que maximize a *AUC*.

A seguir, será apresentado o algoritmo clássico utilizado para a geração de regras (Apriori) que será utilizado para a geração das regras na nossa abordagem.

5.1.2 Algoritmo Apriori

Um dos primeiros e mais conhecidos algoritmos de regras de associação é o Apriori (AGRAWAL; SRIKANT, 1994). O algoritmo Apriori gera todas as regras possíveis para a base de dados que atendam aos critérios de suporte e confiança. A idéia é criar as regras baseadas no conjunto de itens frequentes, denominados *itemsets* frequentes (L_k). O Apriori procura por conhecimento num grande conjunto de dados (*basket data*). Num *basket data*, uma

transação consiste num conjunto de atributos chamado de itens. Cada item pode assumir o valor verdadeiro ou falso, se as características que estes itens representam estiverem presentes ou ausentes, respectivamente, nesta transação.

O algoritmo possui duas partes principais: gerar candidatos e eliminar os itens não frequentes, e a extração de regras de associação.

O primeiro passo encontra todas as combinações de itens com suporte maior que o suporte mínimo especificado pelo usuário, chamados de *itemsets* frequentes. Inicia-se com a contagem de ocorrências dos itens para determinar os *itemsets* frequentes de tamanho unitário (1-*itemsets* frequentes). Os passos posteriores, k, consistem de duas fases. Primeiro, os *itemsets* frequentes L_{k-1}, encontrados no passo anterior $(k-1)$ são utilizados para gerar os conjuntos de itens potencialmente frequentes, os *itemsets* candidatos (C_k). O procedimento para geração de candidatos é descrito no parágrafo seguinte. Na sequência, é realizada uma nova busca no banco de dados, contando-se o suporte de cada candidato em C_k.

A geração dos *itemsets* candidatos toma como argumento L_{k-1}, o conjunto de todos $(k-1)$-*itemsets* frequentes. Para tal, utiliza-se uma função AprioriGen que retorna um super-conjunto de todos os *k-itemsets* frequentes. A intuição por trás desse procedimento é que se, um *itemset X* tem suporte mínimo, todos os seus subconjuntos também o terão (AGRAWAL; SHAFER, 1996). A função, em um primeiro estágio, une L_{k-1} com L_k. No estágio seguinte, são eliminados os *itemsets* $c_k \in L_{k-1}$, desde que um dado $(k-1)$-subset de c_k não pertença a L_{k-1}.

Na última fase do algoritmo, a partir dos *itemsets* frequentes $l \in L_k$, são geradas todas as regras com confiança mínima maior que a confiança mínima especificada pelo usuário. Para cada subconjunto $a \in l$ é gerada uma regra $(l-a) \rightarrow a$, se a confiança da regra ($\frac{suporte(l)}{suporte(a)}$) é maior ou igual à mínima confiança especificada.

5.1.3 Algoritmo Pareto Front Elite (PFE)

O algoritmo Pareto Front Elite (PFE) (ISHIDA; POZO, 2007b) (ISHIDA; POZO, 2007a) tem como objetivo encontrar um conjunto de regras não-ordenadas para maximizar *AUC*. O algoritmo seleciona as melhores regras geradas a partir de um algoritmo de associação de regras (por exemplo: Apriori). As melhores regras, ou regras Elite, passam por uma seleção para a formação da Fronteira de Pareto.

Os passos gerais, detalhados no Algoritmo 5, são: geração das regras Elite (conjunto $RegrasElite$) através da execução do algoritmo de associação de regras e seleção da Fronteira

de Pareto (*paretoFront*). Estes passos são executados duas vezes, uma para cada *head* (que indica a classe), ou seja, uma para selecionar um conjunto de regras para exemplos positivos e outro conjunto para exemplos negativos. Se houver mais de duas classes, a classe com menos exemplos será a classe positiva e os exemplos de outras classes serão considerados da classe negativa.

Um dos grandes problemas de um algoritmo de associação de regras é a geração de um grande número de regras. Isso, geralmente, pode ser contornado com o estabelecimento dos parâmetros mínimos de suporte e confiança pelo usuário e a execução do algoritmo criando apenas as regras com suporte e confiança maiores do que os mínimos. Quanto maiores os mínimos escolhidos para a execução do Apriori, menor será o número de regras geradas. As regras geradas pelo Apriori são chamadas de regras Elites (*RegrasElite*).

A partir das regras Elite geradas, cria-se a Fronteira de Pareto (*paretoFront*) com as regras não-dominadas. Ou seja, uma regra fará parte do conjunto *paretoFront* se, e somente se, não existir nenhuma outra regra com valores maiores de Sensitividade e Especificidade.

Algoritmo 5 Pareto Front Elite

Entrada:
head - Cabeçalhos das regras (positivo / negativo)

Saída:
paretoFront - Fronteira de Pareto

Passos Gerais:

RegrasElite = Apriori(*head*, *SuporteMinimo*, *ConfiancaMinima*)

$$paretoFront = \{rule_j \in RegrasElite \;/\; \neg\exists rule_l, \\ Sensitividade(rule_l) > Sensitividade(rule_j) \\ \text{and } Especificidade(rule_l) > Especificidade(rule_j)\}$$

Dois experimentos serão feitos com o algoritmo PFE para a validação da hipótese de que a seleção de regras de acordo com os critérios de sensitividade e especificidade possa maximizar a *AUC*. Para tal, na subseção 5.1.4 os resultados *AUC* de alguns trabalhos relacionados são apresentados. Na seção 5.1.5 apresentam-se três grupos de experimentos realizados com a variação dos parâmetros mínimos do algoritmo PFE. Para cada grupo, os valores de *AUC* são comparados com os resultados dos trabalhos relacionados. A seção é finalizada com uma pequena discussão dos experimentos e dos resultados obtidos.

Tabela 5.1: Descrição dos conjuntos de dados.

#	Conjunto	At	Exe	Cl	#	Conjunto	At	Exe	Cl
1	breast	10	683	2	9	ionosphere	34	351	2
2	bupa	7	345	2	10	kr-vs-kp	37	3196	2
3	ecoli	8	336	8	11	lettera	17	20000	26
4	flag	28	194	6	12	new-thyroid	6	215	3
5	german	21	1000	2	13	nursery	9	12960	5
6	glass	10	214	6	14	pima	9	768	2
7	haberman	3	306	2	15	satimage	37	6435	6
8	heart	14	270	2	16	vehicle	19	846	4

5.1.4 Resultados dos Trabalhos Relacionados

Todos os experimentos conduzidos nesta subseção foram feitos com conjuntos de dados do UCI Machine Learning Repository da Universidade da Califórnia (NEWMAN; BLAKE; MERZ, 1998) obtidos em junho de 2006 e os resultados apresentados no artigo que apresenta o algoritmo ROCCER (PRATI; FLACH, 2005). As bases de dados escolhidas são usualmente utilizadas como *benchmark* para validação dos algoritmos (NEWMAN; BLAKE; MERZ, 1998). Foram escolhidos bases de dados do repositório de dados da UCI com diferentes graus de desbalanceamento entre as classes. Como o desbalanceamento pode ser um problema na indução de modelos para a classificação de exemplos (PRATI, 2006), os conjuntos escolhidos formam um bom conjunto para validação de algoritmos de classificação. Os exemplos com valores desconhecidos nos atributos foram retirados pois o algoritmo Apriori, utilizado pelo ROCCER para a geração das regras de associação no experimento, não trata destes tipos de atributos. A Tabela 5.1 apresenta a descrição dos conjuntos de dados utilizados nos experimentos. A Tabela contém o número da base de dados (coluna '#'), a descrição do conjunto de dados (coluna 'Conjunto'), o número de atributos (coluna 'At'), número de exemplos (coluna 'Exe') e o número de classes (coluna 'Cl') dos 16 conjuntos de dados. A dimensão de cada espaço de procura é dada por 'At' + 1 (classe). Os experimentos foram executados utilizando estratégia de validação cruzada para 10 partições (*10-fold cross-validation* (WITTEN; FRANK, 1999)) e para todos os algoritmos foram utilizados os mesmos conjuntos de treinamento e de teste.

Nos experimentos realizados do PFE, as regras obtidas foram utilizadas como classificadores, com a utilização do processo de classificação de votação por pesos baseados na confiança (FAWCETT, 2001). Os resultados dos valores de *AUC* foram estimados utilizando a regra trapezoidal. E, em todas as comparações de *AUC* foi utilizado o teste T - *student* (GOSSET, 1908) (GOSSET; PEARSON; WISHART, 1942) com 95% de nível de confiança. Não

Tabela 5.2: Valores médios de *AUC* dos trabalhos relacionados (PRATI; FLACH, 2005). Os números entre parênteses indicam os desvios-padrão.

#	ROCCER	C4.5	C4.5NP	CN2	CN2OR	Ripper	Slipper
1	98,63(1,88)	97,76(1,51)	98,39(1,3)	99,26(0,81)	99,13(0,92)	98,72(1,38)	99,24(0,57)
2	65,3(7,93)	62,14(9,91)	57,44(11,92)	62,74(8,85)	62,21(8,11)	69,1(7,78)	59,84(6,44)
3	90,31(11,56)	50(0)	90,06(7,75)	90,17(6,9)	85,15(11,38)	61,86(25,49)	74,78(15,94)
4	61,83(24,14)	50(0)	68,68(17,22)	53,22(24,12)	42,78(24,43)	45,28(14,93)	52,35(7,44)
5	72,08(6,02)	71,43(5,89)	67,71(4,12)	75,25(5,38)	70,9(4,7)	64,02(13,62)	71,32(6,2)
6	79,45(12,98)	50(0)	81,5(12,65)	73,74(15,4)	79,64(13,24)	49,75(0,79)	50(2,36)
7	66,41(11,54)	55,84(6,14)	64,33(13,58)	59,83(9,87)	59,28(10,13)	57,45(3,85)	50,4(11,14)
8	85,78(8,43)	84,81(6,57)	81,11(7,91)	83,61(6,89)	82,25(6,59)	84,89(7,68)	84,03(6,36)
9	94,18(4,49)	86,09(9,97)	90,91(6,03)	96,23(2,97)	92,18(7,54)	92,06(5,94)	93,95(6,82)
10	99,35(0,36)	99,85(0,2)	99,86(0,2)	99,85(0,16)	99,91(0,17)	99,85(0,21)	99,91(0,09)
11	96,08(0,52)	95,49(1,96)	99,33(0,46)	99,34(0,28)	99,44(0,63)	97,27(1,86)	98,82(0,44)
12	98,4(1,7)	87,85(10,43)	97,5(3,39)	99,14(1,19)	98,43(2,58)	94,95(9,94)	99,12(1,25)
13	97,85(0,44)	99,42(0,14)	99,74(0,13)	100(0)	99,99(0,01)	99,43(0,26)	94,4(1,59)
14	70,68(5,09)	72,07(4,42)	72,6(6,5)	70,96(4,62)	71,97(5,44)	68,07(9,46)	70,02(5,97)
15	89,39(2,38)	90,15(1,7)	91,31(1,32)	91,48(1,45)	91,48(0,9)	86,83(3,94)	89,06(1,98)
16	96,42(1,47)	94,76(3)	96,99(1,44)	97,38(2,05)	96,49(2,41)	95,01(2,22)	93,99(3,13)
Média	85,13	77,98	84,84	84,51	83,20	79,03	80,08

foi utilizado um teste não-paramétrico pela falta de todos os dados dos trabalhos relacionados. Seguiremos com a apresentação dos resultados dos trabalhos relacionados para então prosseguir com a comparação com o algoritmo Pareto Front Elite.

Resultados de trabalhos relacionados

Um dos trabalhos relacionados é o ROCCER, um algoritmo de seleção de regras baseado na análise *ROC* que seleciona regras a partir de um grande conjunto de regras através da manutenção da envoltória convexa no espaço *ROC* (PRATI; FLACH, 2005). Resultados experimentais demonstram que o ROCCER tem valores *AUC* compatíveis com os melhores preditores probabilísticos (PRATI; FLACH, 2005). ROCCER é comparado com C4.5 (QUINLAN, 1993), CN2 (CLARK; NIBLETT, 1989), Ripper (COHEN, 1995) e Slipper (COHEN; SINGER, 1999). O algoritmo ROCCER foi executado com parâmetro mínimo de confiança de 50% e suporte mínimo igual a $1/3$ da classe minoritária. A Tabela 5.2 mostra os resultados apresentados no artigo (PRATI; FLACH, 2005).

A Tabela 5.2 mostra que o algoritmo ROCCER obtém a maior média de *AUC* comparado com os outros algoritmos. Por isso, este algoritmo foi utilizado como base de comparação com os algoritmos criados. As células com as cores cinza claro ou escuro mostram, respectivamente, valores que são estatisticamente melhores ou piores do que o ROCCER. Na maioria

das bases de dados, não existe diferença entre ROCCER e outros algoritmos. O algoritmo ROCCER tem desempenho melhor do que os algoritmos C4.5, Ripper e Slipper em quatro conjuntos de dados e pior desempenho em apenas um conjunto de dados. A comparação do ROCCER contra C4.5 sem *pruning*, CN2 e CN2OR mostra duas perdas e quatorze resultados similares.

Na maioria das bases de dados, o ROCCER, obteve resultados semelhantes (bases #1, #2, #4, #8, #10, #14 e #16, que representa quase metade de todas as bases) ou resultados melhores em comparação com alguns algoritmos (bases #3, #5, #6, #7 #9, #12 #15). Apenas em duas bases de dados (#11 e #13) o desempenho do ROCCER não foi favorável.

Estes resultados ressaltam a eficiência do algoritmo ROCCER diante dos outros algoritmos. Por isso, os próximos experimentos serão baseados, principalmente, nos resultados do ROCCER.

5.1.5 Parâmetros do Pareto Front Elite

Como visto anteriormente, o Pareto Front Elite utiliza um algoritmo de associação de regras (Apriori) na primeira fase e que pode gerar um grande número de regras. Este problema é geralmente resolvido com a utilização de parâmetros mínimos de suporte e confiança; assim, o algoritmo apenas cria regras com suporte e confiança maiores do que o estabelecido pelo usuário.

Os experimentos desta subseção foram conduzidos para verificar a influência dos parâmetros de suporte mínimo e confiança mínimo sobre o algoritmo Pareto Front Elite. Foram utilizados parâmetros que geram pequenos ou grandes conjuntos. O objetivo principal é diminuir os valores dos parâmetros para aumentar os valores de *AUC*.

Antes da execução do algoritmo PFE os mínimos são calculados para cada base de dados com base em todas as regras de um atributo. Seja $MeanSupport$ a média de suporte destas regras e $StdDevSupport$ o desvio-padrão. E, seja $MeanConfidence$ a média dos valores de confiança e $StdDevConfidence$ o desvio-padrão correspondente. Primeiramente, atribui-se o valor $MinSupport_1$ (equação 5.1) como o parâmetro de suporte mínimo e $MinConfidence_1$ (equação 5.2) para o parâmetro mínimo de confiança. Estes são os maiores valores utilizados para estes parâmetros. Para o segundo conjunto de valores são utilizados valores menores: $MinSupport_2$ (equação 5.3) e $MinConfidence_2$ (equação 5.4). O objetivo é o aumento do número de regras geradas, mantendo bons valores de suporte e confiança, com a expectativa de aumentar os valores de *AUC*. O terceiro conjunto de parâmetros: $MinSupport_3$ (equação 5.5) e $MinConfidence_3$ (equação 5.6) são atribuídos com valor zero, ou seja, executar o algo-

Tabela 5.3: *AUC* Médio do algoritmo Pareto Front Elite e os números entre parênteses indicam os desvios-padrão. As colunas são, respectivamente, o número do conjunto de dados, *AUC* para os valores de parâmetros mais altos (equações 5.1 e 5.2), *AUC* para os parâmetros com valores das equações 5.3 e 5.4, os resultados para todas as regras (equações 5.5 e 5.6). O símbolo (-) indica que o conjunto de exemplo não foi executado. Célula em cinza escuro indica, segundo teste T, valor estatisticamente inferior em comparação com os valores da segunda coluna, em cinza claro são os valores estatisticamente inferiores.

#	**PFE(5.1)(5.2)**	**PFE(5.3)(5.4)**	**PFE(5.5)(5.6)**
1	99.33(0.95)	99.08(0.22)	99.39(0.88)
2	63.83(10.29)	68.08(0.54)	65.40(8.77)
3	91.55(4.78)	90.87(1.59)	92.22(6.29)
4	69.07(18.38)	70.83(1.33)	(-)
5	73.51(6.49)	74.95(0.29)	75.86(6.71)
6	77.07(14.36)	82.24(0.76)	80.86(13.48)
7	66.75(6.03)	58.54(0.80)	59.65(11.65)
8	88.56(7.00)	88.77(0.27)	88.61(6.38)
9	95.85(3.12)	96.93(0.50)	(-)
10	97.71(0.86)	(-)	(-)
11	95.46(1.41)	(-)	(-)
12	97.41(4.00)	98.55(0.31)	97.75(3.06)
13	98.82(0.43)	99.80(0.03)	100.00(0.00)
14	70.41(3.87)	71.27(0.27)	72.01(4.94)
16	93.72(2.80)	97.96(0.21)	(-)

ritmo com a geração de todas as regras. Todos os valores de PFE são apresentados na Tabela 5.3. Foram utilizados a mesma metodologia e o mesmo conjunto de exemplos de (PRATI; FLACH, 2005). Os conjuntos de exemplos com (-) não foram executados devido ao grande número de regras possíveis e limitação de nossos recursos computacionais (memória e/ou espaço em disco). A seguir, a análise para cada grupo de parâmetros. Todas as comparações feitas dois a dois utilizaram o teste T (GOSSET, 1908) (GOSSET; PEARSON; WISHART, 1942) com nível de significância de 5%.

$$MinSupport_1 = MeanSupport + StdDevSupport \tag{5.1}$$

$$MinConfidence_1 = MeanConfidence + StdDevConfidence \tag{5.2}$$

$$MinSupport_2 = MeanSupport/2 \tag{5.3}$$

$$MinConfidence_2 = MeanConfidence/2 \tag{5.4}$$

$$MinSupport_3 = 0 \tag{5.5}$$

$$MinConfidence_3 = 0 \tag{5.6}$$

PFE executado com valores de parâmetros mais altos.

O primeiro grupo de parâmetros utilizados para a execução do algoritmo Pareto Front Elite foi o $MinSupport_1$ para suporte mínimo (equação 5.1) e $MinConfidence_1$ para confiança mínima (equação 5.2). A primeira coluna da Tabela 5.3 contém o número do conjunto de exemplos. A segunda coluna possui *AUC* médio e o desvio-padrão (entre parênteses) para cada conjunto de exemplo. Numa análise dos valores de *AUC* da segunda coluna como base da comparação estatística, o Pareto Front Elite tem bom desempenho com relação ao *AUC* na maior parte dos conjuntos de dados. Os valores analisados também estão presentes na Tabela C.1 para uma melhor comparação. Uma comparação com os algoritmos relacionados, resumo encontrado na Tabela C.3 e Figura C.1, mostra que o algoritmo Pareto Front Elite obtém desempenho similar ao algoritmo ROCCER. PFE possui um valor melhor do que o obtido pelo algoritmo ROCCER (conjunto de exemplos #16), e é pior em apenas no conjunto de exemplos #13. Os valores de *AUC* do PFE apresentam melhores desempenhos do que os algoritmos C4.5, Ripper e Slipper, onde o PFE tem, respectivamente, melhores valores de *AUC* em 5, 4 e 4 conjuntos de exemplos. Contra os algoritmos C4.5NP e CN2OR existe um balanço. Entretanto, o algoritmo CN2 tem um melhor desempenho contra o PFE.

Uma análise dos conjuntos de exemplos (resumo na Tabela C.2) mostra que, na maior parte, o algoritmo PFE tem valores similares a outros algoritmos. Apenas entre os conjuntos de exemplos #11(lettera), #13 (nursery) e #16 (vehicle) os valores do PFE têm resultados inferiores na maior parte dos algoritmos.

PFE com os valores de parâmetros intermediários.

O segundo conjunto de valores para os parâmetros mínimos implica no aumento do número de regras geradas em relação ao primeiro grupo. São atribuídos os valores $MinSupport_2$ (equação 5.3) e $MinConfidence_2$ (equação 5.4) para os parâmetros mínimos. A terceira coluna da Tabela 5.3 possui os valores *AUC* de resultado e são a base para a comparação estatística. Para facilitar a visualização dos dados, os resultados foram organizados na Tabela C.4 e as diferenças estatísticas foram destacadas. Na análise por base de dados (resumo na Tabela C.5), foi observado que o algoritmo Pareto Front Elite mantém os bons valores de *AUC* para a maior parte dos conjuntos de exemplos. Foram verificados valores estatisticamente melhores ou similares para a maior parte dos conjuntos de exemplos. A exceção é para o conjunto de exemplo #16, onde existe uma melhora significativa. O primeiro grupo de parâmetros resultava em valores de *AUC* piores do que quatro algoritmos. Agora, com o segundo, os valores de *AUC* são similares. As exceções são para os algoritmos CN2 e CN2OR onde o algoritmo PFE não gera

Tabela 5.4: Suporte Médio e Precisão Ponderada Relativa (*Weighted Relative Accuracy*) do algoritmo PFE e trabalhos relacionados.

Algoritmo	**Suporte (%)**	**WRAcc**
PF Elite (Eq. 5.3;Eq. 5.4)	13.19 (4.15)	0.0261 (0.020)
PF Elite (Eq. 5.1;Eq. 5.2)	15.26 (5.20)	0.0282 (0.022)
ROCCER	13.67 (13.89)	0.0355 (0.018)
C4.5	3.73 (6.01)	0.0094 (0.013)
C4.5NP	1.19 (1.06)	0.003 (0.003)
CN2	3.9 (2.52)	0.011 (0.009)
CN2OR	3.1 (2.18)	0.0085 (0.007)
Ripper	5.96 (5.34)	0.0184 (0.012)
Slipper	1.92 (1.58)	0.006 (0.006)

bons valores.

Os valores de parâmetros $MinSupport_1$ (equação 5.1) e $MinConfidence_1$ (equação 5.2) são utilizados no primeiro grupo e conseguem maximizar os valores de *AUC* (segunda coluna da Tabela 5.3). Diminuindo os valores para $MinSupport_2$ e $MinConfidence_2$, o algoritmo Pareto Front Elite, ainda, seleciona conjuntos de regras com altos valores de *AUC*. Valores menores para os parâmetros implicam na melhora de *AUC* para o conjunto de exemplo #16. Em nenhum dos casos os parâmetros com menores valores, $MinSupport_2$ e $MinConfidence_2$, são estatisticamente piores do que os resultados com os maiores valores para os parâmetros ($MinSupport_1$ e $MinConfidence_1$).

Para obter bons resultados, o algoritmo Pareto Front Elite gera um grande conjunto de regras em comparação com outros algoritmos. Os bons resultados devem-se à qualidade das regras nos conjuntos de aproximação. A terceira linha da Tabela 5.4 apresenta o suporte médio das regras com os maiores valores para os parâmetros de suporte e confiança mínimo. As regras de todos os conjuntos de aproximação para estes valores de parâmetros apresentam média de 15.26. Com exceção do algoritmo ROCCER, todos os outros algoritmos têm valor médio de suporte significantemente menor. Analisando os dois grupos de resultados do algoritmo Pareto Front Elite (segundo e terceira linhas), é esperada uma menor média para o grupo de parâmetros $MinSupport_2$ (equação 5.3) e $MinConfidence_2$ (equação 5.4). Porém, até com os valores menores de parâmetros, o resultado é estatisticamente similar ao algoritmo ROCCER e melhor do que outros algoritmos.

A coluna WRAcc da Tabela 5.4 possui a medida que indica a importância da regra: precisão ponderada relativa (*Weighted Relative Accuracy*). A média para esta medida não apresenta diferença estatística entre os dois resultados do algoritmo Pareto Front Elite, resultados dos algoritmos ROCCER e o algoritmo Ripper, sendo que outros algoritmos apresentam valores

estatisticamente piores.

Análise de PFE sem parâmetros

O próximo experimento tem como objetivo compreender melhor o efeito dos parâmetros de suporte e confiança mínimos para o algoritmo Pareto Front Elite. Neste experimento, é atribuído o valor zero para os parâmetros mínimos (equações 5.5 e 5.6), ou seja, todas as regras serão criadas. Com estes parâmetros, é possível gerar todas as regras para a Fronteira de Pareto de acordo com os critérios de sensitividade e especificidade.

O experimento foi executado em dez conjuntos de exemplos, todos os que tiveram sua execução possível apesar das limitações de nossos recursos computacionais. A quarta coluna da Tabela 5.3 mostra os resultados sem valores para os parâmetros mínimos. Como esperado, o número de regras criadas é maior, porém com maiores valores de *AUC*. Apesar de valores melhores, somente para o conjunto de exemplo #13 o valor de *AUC* é estatisticamente diferente.

Analisando o conjunto de exemplo #7(haberman), pode ser observado o *overfitting* (sobre especialização). O maior valor para os parâmetros mínimos gera melhor valor de *AUC* (66.75, segunda coluna da Tabela 5.3). A diminuição dos parâmetros aumenta o número de regras geradas e o resultado é a diminuição do *AUC* de 61.92 para 59.65, respectivamente, o resultado para os menores valores dos parâmetros mínimos (equações 5.3 e 5.4) e o resultado sem os parâmetros mínimos.

O ponto falho do algoritmo Pareto Front Elite é a execução nos conjuntos de exemplos com muitos atributos. Mesmo com valores para os parâmetros mínimos, o algoritmo Apriori não é capaz de gerar regras para os conjuntos de exemplos com os maiores números de atributos (conjuntos de exemplos #4, #9, #10, #11, #15 e #16).

5.1.6 Discussão

O algoritmo ROCCER é o algoritmo que apresenta um bom desempenho de *AUC* entre os trabalhos relacionados. Utilizando o ROCCER para comparação com o algoritmo Pareto Front Elite, na maioria dos resultados da *AUC* foi verificada certa semelhança entre os resultados. O algoritmo PFE foi executado com três grupos de valores diferentes para os parâmetros mínimos. O primeiro grupo gerou menos regras que outros grupos e com valores de *AUC* semelhantes ao algoritmo ROCCER. Já no segundo grupo, os valores de parâmetros mínimos eram menores do que do primeiro grupo. Os resultados dos dois grupos foram semelhantes, porém, com maior número de regras para o segundo caso.

O terceiro grupo de parâmetros executou o PFE com todas as regras possíveis, sendo possível gerar a Fronteira de Pareto para os critérios de sensitividade e especificidade. Apesar do aumento do número de regras no conjunto resultado, apenas em um único conjunto de dados o valor de *AUC* é maior. Em outros conjuntos de dados os resultados foram semelhantes aos resultados de outros grupos. Isso enfatiza que resultados semelhantes de *AUC* podem ser obtidos sem a necessidade de aumentar o número de regras no conjunto final.

Os resultados dos três grupos de resultados são comparáveis ao ROCCER que, apesar de não utilizar critérios biobjetivos, possui a maior média de *AUC* entre os algoritmos relacionados considerando todas as bases de dados. A obtenção dos bons valores de *AUC* confirma a hipótese de que a maximização da *AUC* pode ser alcançada através de conjuntos de regras não-dominadas selecionadas de acordo com os critérios de sensitividade e especificidade.

No último grupo de valores, sem os parâmetros mínimos, não foi possível executar o PFE nos conjuntos de dados que contêm os maiores números de atributos, enfatizando a necessidade de substituição do algoritmo determinístico como gerador de regras. Para tal, outros algoritmos baseados em metodologias heurísticas são propostos; o primeiro é apresentado na próxima seção.

5.2 Metaheurística Biobjetiva para Maximização da *AUC*

Nesta seção verifica-se a seguinte hipótese: assim como o algoritmo Pareto Front Elite, será possível obter regras e gerar o conjunto de aproximação segundo os critérios de sensitividade e especificidade utilizando um algoritmo metaheurístico que maximize a *AUC*? Para validar a hipótese, utiliza-se a metaheurística nuvem de partículas sem a utilização do algoritmo determinístico. A ideia é manter os mesmos objetivos do PFE com a criação de um algoritmo biobjetivo para a obtenção de conjuntos não-ordenados de regras com resultados muito próximos em termos de *AUC*.

Otimização de nuvens de partículas (*Particle Swarm Optimization*, *PSO*) é uma técnica de otimização estocástica baseada em população e inspirada pelo comportamento social de movimento dos pássaros (*bird flocking*) ou processo de educação dos peixes (*fish schooling*) (KENNEDY; EBERHART, 1995).

PSO pertence à categoria de Inteligência de Enxames (*Swarm Intelligence*), além de possuir influência da Vida Artificial (VESTERSTROEM; RIGET, 2002) e Computação Evolucionária. Na computação evolucionária, *PSO* compartilha semelhanças com os Algoritmos Genéticos (AG). O sistema é inicializado com a população de soluções e busca pelo ótimo

através da atualização das gerações. Entretanto, diferentemente de AG, *PSO* não possui operadores evolucionários como *crossover* e mutação (*mutation*). *PSO* tem sido aplicado para resolver problemas de classificação como, por exemplo, o algoritmo *PSO*/ACO1 que suporta nativamente dados nominais (HOLDEN; FREITAS, 2005).

A seguir é descrito o algoritmo básico *PSO*.

5.2.1 Algoritmo Básico de *Particle Swarm Optimization*

Em *PSO*, existe um conjunto de partículas, chamado de nuvem, que possui possíveis soluções para o problema. Estas partículas se movem através de um espaço n-dimensional baseado nas melhores posições de seus vizinhos e em suas próprias melhores posições. A cada geração, a posição e a velocidade das partículas são atualizadas, considerando a melhor posição obtida pela partícula e a melhor posição obtida por todas as partículas da nuvem. As melhores partículas são encontradas baseadas na função de *fitness*, que é a função objetivo do problema. Cada partícula p, numa iteração t, possui uma posição em R^n, $\mathbf{x}(\mathbf{t})$, e uma velocidade de deslocamento neste espaço, $\mathbf{v}(\mathbf{t})$. Cada partícula também possui uma pequena memória que contém suas melhores posições já alcançadas, $pbest(t)$, e a melhor posição, $\mathbf{gbest}(\mathbf{t})$, já passada por partículas as quais p conhece, ou seja, o melhor $p(t)$ de todas as partículas próximas da vizinha de $p(N(p))$. É importante observar que $\mathbf{x}(\mathbf{t})$, $\mathbf{v}(\mathbf{t})$, $\mathbf{pbest}(\mathbf{t})$ e $\mathbf{gbest}(\mathbf{t})$ são vetores n-dimensionais.

O algoritmo trabalha apenas como a descrição a seguir. A partícula é inicializada para o tempo $t = 0$, espalhando as partículas aleatoriamente no espaço. Depois disso, inicia-se o processo iterativo. A posição e velocidade, na próxima iteração, são calculadas pelas equações 5.7 e 5.8.

$$\overrightarrow{v}(t+1) = \varpi * \overrightarrow{v}(t) + \phi_1 * (\overrightarrow{pbest}(t) - \overrightarrow{x}(t)) + \phi_2 * (\overrightarrow{gbest}(t) - \overrightarrow{x}(t)) \tag{5.7}$$

$$\overrightarrow{x}(t+1) = \overrightarrow{x}(t) + \overrightarrow{v}(t+1) \tag{5.8}$$

onde ϕ_1 e ϕ_2 , na equação 5.7, são coeficientes que determinam a influência das melhores partículas ($\mathbf{pbest}(\mathbf{t})$) e a partícula global ($\mathbf{gbest}(\mathbf{t})$) sobre a velocidade da partícula, respectivamente. O coeficiente ϖ é a inércia da partícula, ou seja, quanto à velocidade anterior afeta a velocidade corrente. Após as velocidades e as posições de todas as partículas terem sido atual-

izadas, $\mathbf{pbest(t+1)}$ e $\mathbf{gbest(t+1)}$ são calculadas e mantidas para a próxima iteração ou até a execução ser finalizada.

Com algumas variações, o algoritmo apresentado nesta subseção tem sido aplicado com sucesso em áreas como redes neurais, apresentando resultados em menor tempo e com a mesma eficiência (KENNEDY; EBERHART, 1995) (EBERHART; SHI, 1998). A seguir, apresenta-se *PSO* para problemas multiobjetivos.

5.2.2 *PSO* para Problemas Multiobjetivos

Um algoritmo de nuvem de partícula para a solução de problemas multiobjetivos foi apresentada por Coello e lechuga (2002) e discutida em (ISHIDA et al., 2009). Em *Multiple Objective Particle Swarm Optimization* (*MOPSO*), em contraste ao *PSO*, existem muitas funções de *fitness*. Nesse caso, trata-se de um problema de otimização multiobjetivo, onde todos os objetivos devem ser otimizados. Diferentemente do *PSO*, o *MOPSO* não possui o melhor global, mas possui um repositório global com soluções não-dominadas para que cada partícula deposite suas experiências de movimentação a cada ciclo. Uma solução é incluída somente se alguma experiência da partícula não for dominada por nenhuma solução do repositório. O repositório é utilizado pelas partículas para que escolham um líder a ser seguido chamado de guia. Cada partícula não-dominada possui uma região do espaço objetivo reservada para a escolha do líder. Com isso, toda partícula cujos objetivos estão dentro do espaço seguirão a partícula não-dominada líder da região.

Existem muitas formas para fazer esta escolha (COELLO; LECHUGA, 2002). Por exemplo, o método da distância-sigma (MOSTAGHIM; TEICH, 2003), que divide o espaço de busca entre linhas da posição das partículas para a origem. Para tal, para cada partícula do repositório e da nuvem é calculado o vetor sigma:

$$\sigma = \begin{pmatrix} f_1(x)^2 - f_2(x)^2 \\ f_2(x)^2 - f_3(x)^2 \\ \vdots \\ f_k(x)^2 - f_1(x)^2 \end{pmatrix} / (f_1(x)^2 + f_2(x)^2 + \cdots + f_k(x)^2) \tag{5.9}$$

onde k é o número de objetivos do problema; f_i é a i-ésima função de *fitness* para i de 1 a k. Neste caso, a partícula guia é a partícula do repositório que detém o vetor sigma com a menor distância euclidiana do vetor sigma da partícula da nuvem.

Após a escolha da partícula guia, e a cada geração, a velocidade das partículas é atual-

izada pela equação 5.10.

$$\overrightarrow{v}(t+1) = \varpi * \overrightarrow{v}(t) + \phi_1 * (\overrightarrow{pbest}(t) - \overrightarrow{x}(t)) + \phi_2 * (\overrightarrow{R_h}(t) - \overrightarrow{x}(t)) \tag{5.10}$$

onde R_h é uma posição da partícula do repositório, escolhido como um guia.

5.2.3 *MOPSO* para Aprendizado de Regras

O algoritmo de aprendizado de regras utilizando o *MOPSO*, criado e publicado por Torácio (2008), utiliza a abordagem *Michigan* para a representação de uma partícula, onde cada partícula da nuvem representa uma regra de classificação simples. Uma regra é representada pela posição da partícula no espaço de busca e, se a base de dados tem n atributos, o espaço de busca é n-dimensional. Embora o algoritmo apresentado nesta seção tenha sido criado por outro autor, é importante observar que a contribuição do autor desta tese foi apresentar a ideia para a criação do algoritmo através de critérios biobjetivos de sensitividade e especificidade.

Cada atributo, além dos valores que aparecem na base, pode aceitar um valor '?'. Isto significa que nesta regra o atributo não importa para a classificação. Então, por exemplo, para uma base com os atributos: (Céu, Temperatura, Umidade, Vento, Joga), sendo que o último é o atributo classe, uma partícula com a posição (sol,?,?,sim,sim) é equivalente a regra: IF Céu=sol AND Vento=sim THEN Joga=sim.

A codificação dos valores dos atributos no conjunto de dados é feita por números inteiros. Por exemplo, tendo a base e a posição da partícula mencionada, cada um destes valores é trocado por um número inteiro correspondente, sol=2, ?=0, ?=0, sim=1, sim=2, respectivamente. Então, o vetor: (2,0,0,1,2) representa sua posição no espaço N^5. A representação numérica possibilita a utilização, sem modificação, das fórmulas originais de velocidade e posição do *MOPSO*.

Para cada base de dados, informações dos atributos e valores possíveis são retirados diretamente da base de treinamento por um processo automático. O processo automático identifica os valores de cada atributo e cria uma lista com os valores possíveis. O algoritmo começa pela criação da nuvem para cada classe e iniciação das partículas. Ou seja, para um problema de duas classes e, se considerarmos uma otimização de 50 partículas, o algoritmo cria duas nuvens com 50 partículas e possui dois grupos dos melhores globais e dois grupos classificadores, um para cada classe.

O pseudo-código está apresentado no Algoritmo 6. No início do algoritmo, as partículas são espalhadas aleatoriamente no espaço de busca discreto (passo 1). Isto é feito por roletas onde os valores mais frequentes do banco de dados têm melhores possibilidades de serem escolhidos. A probabilidade do valor genérico de cada atributo ('?') é calculada com uma função de um número de valores possíveis para o atributo. Quanto maior o número de valores, maior é a probabilidade do valor genérico na roleta. Esta restrição evita regras ou partículas muito específicas, gerando, por exemplo, regras para cobrir poucos exemplos. Depois disso, as partículas são avaliadas para todos os objetivos (passo 2). No nosso caso, foram utilizadas a sensitividade e a especificidade.

Através dos critérios da dominância de Pareto, as partículas são colocadas com as soluções não-dominadas no repositório global (passo 3). As soluções que já estejam no repositório, mas dominadas por outra solução, devem ser excluídas. Analisando o repositório, as regras mais específicas são removidas e apenas as mais genéricas são mantidas (passo 4). A partir deste ponto, divide-se o espaço de busca entre cada partícula do repositório (passo 5).

No processo iterativo, as partículas da nuvem devem escolher uma partícula não-dominada do repositório como a melhor da vizinhança (neste trabalho foi considerada a distância sigma) e que será a partícula guia. Então, a velocidade e a posição das partículas no espaço são atualizadas através, respectivamente, das equações 5.10 e 5.8. Depois disso, novamente, as partículas são avaliadas e o espaço do objetivo entre as partículas do repositório é dividido, reiniciando o laço (passo 6). O laço termina quando o número de gerações estabelecido pelo usuário é alcançado. A suspeita e contribuição desta tese para este trabalho é que, depois de algumas gerações, as melhores regras de classificação para a classe, de acordo com os objetivos, estarão no repositório (passo 7).

Um dos critérios de parada do algoritmo é o número limite de gerações. O problema deste critério é ser escolhido um limite baixo de modo a parar o processo quando a convergência está longe, ou de forma contrária, quando o algoritmo já convergiu muito antes do limite ser alcançado. Para evitar este problema, um critério relativo à convergência foi criado. Neste caso, o algoritmo para se não obtiver certa quantidade de regras novas durante um período de gerações seguidas (definido como parâmetro).

5.2.4 Experimentos e Análise do *MOPSO*

Algoritmo Multiple Objective Particle Swarm Optimization (*MOPSO*) foi executado 50 vezes com 500 partículas até no máximo 50 gerações. O algoritmo *MOPSO* é primeiramente comparado com ROCCER (PRATI; FLACH, 2005), C4.5 (QUINLAN, 1993), CN2 (CLARK;

Algoritmo 6 Algoritmo de aprendizado de regras com *MOPSO*.

1. Para cada partícula i, faça:
   a. Inicialize $\overrightarrow{x_i}$ com uma solução aleatória do problema
   b. Inicialize $\overrightarrow{v_i}$ com uma velocidade aleatória.
   c. Inicialize $\overrightarrow{pbest_i} = \overrightarrow{x_i}$
2. Avaliar partículas utilizando função de *fitness*.
3. Encontrar soluções não-dominadas, armazenando-as no repositório.
4. Filtrando o repositório, mantendo as regras mais genéricas.
5. Dividir o espaço de busca entre as soluções do repositório.
6. Enquanto não atingir o critério de parada:
   a. Para cada partícula i da nuvem, faça:
      1) $\varpi = random[0,0.8]; \phi_1 = random[0,4]; \phi_2 = random[0,4];$
      2) $\overrightarrow{v}(t+1) = (\varpi * \overrightarrow{v}(t) + \phi_1 * (\overrightarrow{pbest}(t) - \overrightarrow{x}(t)) + \phi_2 * (\overrightarrow{R_h}(t) - \overrightarrow{x}(t))) mod \overrightarrow{N_i}$
      Nota: $\overrightarrow{N_i}$ é um vetor do número de valores possíveis para cada atributo do banco de dados. Isto restringe a partícula dentro do espaço de busca.
      3) $\overrightarrow{x}(t+1) = (\overrightarrow{x}(t) + \overrightarrow{v}(t+1)) mod \overrightarrow{N_i}$
      4) Avaliar partículas. A partícula terá um valor para cada objetivo do problema: Sensitividade e Especificidade
      5) Atualizar $\overrightarrow{pbest}(t)$
   b. Atualizar o repositório com as partículas não-dominadas.
   c. Dividir o espaço de busca, encontrando $\overrightarrow{R_h}(t)$ das partículas.
7. Retornar o Repositório

NIBLETT, 1989), Ripper (COHEN, 1995) e Slipper (COHEN; SINGER, 1999). A Tabela 5.5 mostra os valores de *AUC* médios do algoritmo *MOPSO* e a Tabela 5.2 os valores dos trabalhos relacionados. Estes resultados também podem ser encontrados na Tabela C.10 com os valores estatisticamente diferentes pelo teste T (GOSSET, 1908) (GOSSET; PEARSON; WISHART, 1942) destacados. Para uma melhor visualização do desempenho do *MOPSO*, a Tabela 5.6 traz um resumo das comparações com outros algoritmos. O algoritmo *MOPSO* tem desempenho similar a C4.5 e Ripper. Considerando C4.5, o *MOPSO* tem valores de *AUC* melhores em cinco conjuntos de dados (#1, #3, #4, #6 e #7) e em seis conjuntos de dados (#2, #5, #8, #9, #12 e #14) não há diferença entre os valores. Comparado com Ripper, o *MOPSO* também tem valores de *AUC* melhores em cinco conjuntos de dados (#3 até #7), porém, em quatro conjuntos de dados (#1, #2, #8 e #14) não há diferença entre os valores. Já na comparação com o algoritmo Slipper, este possui sete valores melhores (conjuntos de dados #1, #9, #10, #11, #13, #15 e #16) e o *MOPSO* é melhor em quatro conjuntos de dados (#2, #4, #6 e #7), além de valores similares nos conjuntos de dados #3, #5, #8, #12 e #14.

Nos conjuntos de dados #4 e #8, o algoritmo *MOPSO* apresenta bons valores de *AUC* comparado com outros algoritmos. O *MOPSO* é melhor do que quatro algoritmos (C4.5, CN2OR, Ripper e Slipper) para o conjunto #4 e é similar a três algoritmos. No conjunto de

Tabela 5.5: Os *AUC* médios do algoritmo *MOPSO* e seus desvios-padrão indicados entre parênteses.

#	MOPSO
1	98.75(0.16)
2	66.15(1.10)
3	82.69(2.22)
4	63.26(4.37)
5	73.39(0.83)
6	65.00(4.19)
7	59.96(1.23)
8	87.48(0.60)
9	81.19(2.84)
10	95.78(0.53)
11	93.80(0.72)
12	87.62(0.43)
13	85.91(0.61)
14	71.67(0.83)
15	65.39(9.11)
16	88.83(1.02)
Média	79.18

Tabela 5.6: Total de bases de dados que o algoritmo *MOPSO* é, segundo teste T, superior (segunda linha), similar (terceira linha) ou inferior (quarta linha) aos resultados dos trabalhos relacionados. São considerados os resultados do *MOPSO* (Tabela 5.5) e os resultados dos trabalhos relacionados (Tabela 5.2 e Tabela C.10). A contagem é feita com base nas comparações da Tabela C.10.

Comparação	**ROCCER**	**C4.5**	**C4.5NP**	**CN2**	**CN2OR**	**Ripper**	**Slipper**	**Total(%)**
Superior	0	5	3	1	2	5	4	20(0,18)
Similar	6	6	4	6	6	4	5	37(0,33)
Inferior	10	5	9	9	8	7	7	55(0,49)

dados #8, *MOPSO* tem valores semelhantes a quatro algoritmos (ROCCER, C4.5, Ripper e Slipper) e é melhor do que três algoritmos (C4.5NP, CN2 e CN2OR). No conjunto de dados #7, o algoritmo *MOPSO* também é melhor do que três algoritmos (C4.5, Ripper e Slipper), porém, é pior do que o algoritmo ROCCER e similar a outros três algoritmos (C4.5NP, Ripper e Slipper). O *MOPSO* tem valores de *AUC* semelhantes a outros algoritmos nos conjuntos de dados #1, #3, #5, #6 e #14. Em sete conjuntos de dados (#9, #10, #11, #12, #13, #15 e #16), *MOPSO* apresenta valores ruins de *AUC*. A análise do total das comparações (última coluna da Tabela 5.6) indica que em quase metade das comparações (51%) o algoritmo *MOPSO* é igual ou melhor do que outros algoritmos.

Tabela 5.7: Comparação dos *AUC* Médio do algoritmo *MOPSO* (segunda coluna da Tabela 5.5) com os resultados do algoritmo Pareto Front Elite obtidos com os maiores valores para os parâmetros mínimos (terceira coluna, equações 5.1 5.2), valores intermediários (equações 5.3 5.4) e sem valores (equações 5.5 e 5.6) (apresentados na quarta coluna da Tabela 5.3). A segunda coluna (resultados do *MOPSO*) é a base para a comparação com outros algoritmos utilizando o teste T. Células em cinza escuro indicam valores estatisticamente inferiores dos resultados do *MOPSO* e células em cinza claro indicam valores estatisticamente superiores.

#	MOPSO	PFE (5.1)(5.2)	PFE (5.3)(5.4)	PFE (5.5)(5.6)
1	98,75	99,33	99,08	99,39
2	66,15	63,83	68,08	65,40
3	82,69	91,55	90,87	92,22
4	63,26	69,07	70,83	(-)
5	73,39	73,51	74,95	75,86
6	65,00	77,07	82,24	80,86
7	59,96	66,75	58,54	59,65
8	87,48	88,56	88,77	88,61
9	81,19	95,85	96,93	(-)
10	95,78	97,71	(-)	(-)
11	93,80	95,46	(-)	(-)
12	87,62	97,41	98,55	97,75
13	85,91	98,82	99,80	100,00
14	71,67	70,41	71,27	72,01
16	88,83	93,72	97,96	(-)

MOPSO X PFE

Pela existência de bons resultados do PFE, comparamos os resultados dos algoritmos Pareto Front Elite e *MOPSO*. A segunda coluna da Tabela 5.5 contém os valores de *AUC* do algoritmo *MOPSO*. Os valores médios de *AUC* do algoritmo PFE estão na Tabela 5.3 separados em três conjuntos de parâmetros. Os resultados de *AUC* para o algoritmo PFE com os maiores valores para os parâmetros mínimos (equações 5.1 e 5.2) estão na segunda coluna e os resultados com os parâmetros $MinSupport_2$ (equação 5.3) e $MinConfidence_2$ (equação 5.4) estão apresentados na terceira coluna da Tabela 5.3. Os resultados do PFE com todas as regras, ou seja, sem valores dos parâmetros mínimos, estão na quarta coluna da Tabela 5.3. Todos os resultados citados também podem ser encontrados na Tabela 5.7 que possui em destaque os valores estatisticamente diferentes comparados com os resultados do algoritmo *MOPSO*.

A comparação do algoritmo *MOPSO* com o PFE com os maiores valores de parâmetros mínimos ($MinSupport_1$ e $MinConfidence_1$) mostra valores de *AUC* similares em cinco conjuntos de dados (#2, #4, #5, #8 e #14) e valores piores para o *MOPSO* nos outros dez conjuntos de dados. Na comparação com os resultados dos parâmetros $MinSupport_2$ e $MinConfidence_2$,

o algoritmo *MOPSO* obtém resultados ruins na maioria dos conjuntos de dados. O algoritmo *MOPSO* apresenta os piores valores em 11 dos 13 conjuntos de dados. Porém, apresenta o melhor resultado em um conjunto de dados (#7) e desempenho similar ao PFE no conjunto #14.

A comparação do algoritmo de nuvem de partículas com o algoritmo Pareto Front Elite sem valores para os parâmetros mínimos (quarta coluna da Tabela 5.3) mostra que o algoritmo *MOPSO* obtém valores de *AUC* similares nos conjuntos de dados #2, #5, #7, #8 e #14, ou seja, em metade dos dez conjuntos comparados. Em outros conjuntos de dados (#1, #3, #6, #12 e #13) o algoritmo *MOPSO* possui pior desempenho na comparação com o PFE.

O algoritmo PFE sem os parâmetros mínimos pode ser considerado como um algoritmo determinístico que gera todas as regras possíveis para o conjunto de dados. A vantagem do algoritmo *MOPSO* é não gerar todas as regras de modo exaustivo; os resultados da metaheurística utilizada mostrando valores de *AUC* com desempenho semelhante em metade dos conjuntos de dados faz do *MOPSO* um algoritmo promissor. Quando o algoritmo PFE cria apenas as regras com suporte e confiança maiores do que os parâmetros mínimos, o desempenho do PFE se torna muito melhor do que o *MOPSO*, indicando a necessidade de melhoria do algoritmo *MOPSO* e reforçando os bons valores de resultados do PFE.

5.3 *GRASP-PR Rule Learning*

Nesta seção apresenta-se uma nova abordagem criada para geração de um conjunto de regras não-ordenadas de acordo com os critérios de sensitividade e especificidade em grandes bases de dados. Assim como o algoritmo PFE, o objetivo é maximizar a *AUC* utilizando critérios biobjetivos. Porém, em vez do algoritmo determinístico será utilizada a metaheurística *GRASP* para gerar o conjunto não-ordenado de regras e o *path-relinking* para melhorar os resultados obtidos. Para tanto, a próxima seção apresenta um resumo sobre *GRASP* e a seção 5.3.2 discute sobre o *path-relinking*. O algoritmo proposto está na seção 5.3.3 e alguns experimentos e comparações com os trabalhos relacionados estão na seção 5.3.4.

5.3.1 *GRASP*

Greedy Randomized Adaptive Search Procedure (*GRASP*) (FEO; RESENDE, 1995) é um algoritmo metaheurístico criado para otimização combinatória muito estudado e difundido, principalmente, pela sua simplicidade. *GRASP* é um processo *multi-start* em que cada iteração consiste em duas fases: a construção da solução inicial e a busca local. A fase de construção cria

uma solução factível através de um processo guloso (obtém o melhor benefício a cada passo) e aleatório. A partir destas soluções iniciais, a busca local explora a vizinhança até um mínimo local. Ao final de todas as iterações existe o conjunto de vários ótimos locais.

A cada iteração da fase de construção, a solução inicial é construída de forma incremental utilizando o conjunto de elementos candidatos. Um elemento candidato é uma parte que pode ser incorporada à solução parcial sem destruir a sua factibilidade. Todos os elementos candidatos são ordenados utilizando uma função que mede o benefício de cada elemento para a solução. Os *r%* melhores elementos são selecionados para formarem a *Restricted Candidate List* (*RCL*), onde *r* é um parâmetro. O algoritmo *GRASP* escolhe aleatoriamente um elemento da *RCL* para ser adicionado à solução. Deste modo, diferentes soluções são obtidas a cada iteração, sem necessariamente comprometer o poder do componente guloso adaptativo do método (FEO; RESENDE, 1995). Ao final, os elementos candidatos são atualizados e as funções novamente avaliadas.

As soluções geradas pela construção gulosa aleatória não são necessariamente ótimas, até mesmo com respeito aos seus vizinhos (FEO; RESENDE, 1995), e sua fase de busca local geralmente melhora as soluções construídas explorando a vizinhança.

A busca local é usada com frequência para resolver problemas de otimização combinatória e serve principalmente como estratégia de apoio a outros algoritmos como o *GRASP* e a busca tabu. Isto se deve ao seu conceito de estratégia iterativa de melhoria que consiste de, dado um ponto ou uma solução inicial x, analisar soluções próximas a x denominada vizinhança de x, $V(x)$, tendo como meta uma solução de melhor valor. Se uma solução melhor for encontrada, então segue a busca pela vizinhança desta solução de melhor valor. A busca vai seguindo até que não haja uma solução de melhor valor, quando é então encontrado o ótimo local. Por exemplo, se a busca local tem como objetivo encontrar o mínimo local, a análise da vizinhança será em busca de uma solução de menor valor em relação à x. Não encontrando uma solução de menor valor, então tal solução será chamada de mínimo local. Outro critério de parada da busca local que pode ser utilizado é um número limitado de iterações. O algoritmo 7 ilustra os passos básicos de busca por um mínimo local, avaliado por uma função $f(x)$.

Algoritmo 7 ProcuraLocal(x)

Enquanto x não é um mínimo local faça
  Encontre $x' \in V(x)$ tal que $f(x') < f(x)$ e $f(x') < f(x''), \forall x'' \in V(x)$;
  $x \leftarrow x'$
Fim enquanto

Retorna x

Existe um grande número de variações para a busca local do *GRASP*; a vizinhança também pode ser implementada utilizando-se a estratégia de *best-improvement* (melhor melhora) ou *first-improvement* (primeira melhora) (FEO; RESENDE, 1995). Na estratégia de melhor busca, todos os vizinhos são avaliados e o melhor vizinho substitui a solução corrente. Na estratégia de primeira melhora, a solução corrente é substituída pelo primeiro vizinho que melhore a função de custo. Na prática, observou-se que ambas as estratégias levam à mesma solução final, mas a primeira melhora com menor tempo computacional. Também se observou a convergência prematura para um mínimo local não global com a estratégia de melhor melhora (FEO; RESENDE, 1995).

Um problema da busca local é a dependência da solução inicial para a obtenção do resultado. Uma estratégia para amenizar o problema é a escolha de mais de um ponto de partida para executar a busca local. Assim, é escolhida a melhor solução dentre as encontradas, obtendo-se, em geral, uma solução de melhor qualidade do que se tivesse partido de uma única solução. O *GRASP*, com sua estratégia de múltiplos inícios, irá gerar vários pontos de partida e executar a busca local a partir de todos. Por fim, a solução que tiver menor valor será escolhida.

A eficiência da busca local depende de vários aspectos, tais como a solução inicial, a rápida avaliação da função de custos da vizinhança, a estrutura da vizinhança e técnica de procura da vizinhança (FEO; RESENDE, 1995).

5.3.2 *Path-relinking*

Para melhorar as soluções encontradas pelo *GRASP*, o *path-relinking* foi proposto como uma estratégia de busca que explora trajetórias conectando soluções Elite obtidas pela busca Tabu (GLOVER, 1996). O uso do *path-relinking* com o *GRASP* foi primeiramente proposto por Laguna e Marti (LAGUNA; MARTI, 1999), seguido por diversas extensões, melhoras e aplicações bem sucedidas (AIEX et al., 2000) (CANUTO; RESENDE; RIBEIRO, 2001) (RESENDE; RIBEIRO, 2003) (RIBEIRO; UCHOA; WERNECK, 2002). Dadas duas soluções, os elementos em comum são mantidos e o espaço de soluções é percorrido por estes elementos com o objetivo de achar uma solução melhor.

O *path-relinking* pode ser visto como uma estratégia que procura incorporar atributos de soluções de alta qualidade, pelo favorecimento destes atributos nos movimentos selecionados (RESENDE; RIBEIRO, 2002). O *path-relinking* pode ser aplicado como uma estratégia de intensificação ou como um passo de pós-otimização para todos os pares das melhores soluções resultantes da busca local (RESENDE; RIBEIRO, 2002).

Duas estratégias básicas são:

- Path-relinking é aplicada como uma estratégia de intensificação para cada ótimo local obtido na fase de busca local.
- Path-relinking é aplicada como um passo de pós-otimização do *GRASP* para todos os pares das soluções encontradas.

Para a estratégia de intensificação é necessário que o *GRASP* armazene o conjunto das melhores soluções num conjunto temporário. O conjunto temporário começa vazio e cada ótimo local encontrado na busca local é candidato a ser inserido dentro do conjunto. A solução é inserida se for suficientemente diferente (diferença simétrica) de todas outras soluções que já se encontram no conjunto temporário. Se o conjunto temporário já possuir o número máximo de soluções e o candidato for melhor do que o pior, então o candidato é inserido e o pior é retirado do conjunto; caso contrário, o candidato é simplesmente inserido.

Em detalhes, a cada iteração a estratégia de intensificação seleciona a solução $\mathbf{x_1}$ resultante da fase de busca local e outra solução arbritária $\mathbf{x_2}$ escolhida aleatoriamente do conjunto temporário formando o par de vetores $(\mathbf{x1}, \mathbf{x2})$. O algoritmo calcula a diferença simétrica $\Delta(\mathbf{x_1}, \mathbf{x_2})$ entre $\mathbf{x_1}$ e $\mathbf{x_2}$ para realizar movimentos de uma solução inicial x_1 até alcançar outra solução $\mathbf{x_2}$ (solução guia). Começando da solução inicial, o melhor movimento de $\Delta(\mathbf{x_1}, \mathbf{x_2})$ ainda não realizado é aplicado para a solução corrente, até que a solução guia seja atingida. A melhor solução encontrada ao longo da trajetória é também considerada como um candidato para inserção no conjunto temporário.

A estratégia de *path-relinking* de pós-otimização funciona como uma melhoria das soluções encontradas no *GRASP*. Após a execução do *GRASP* existe o conjunto de todos os ótimos locais. Deste conjunto, cada par de soluções é aplicada a estratégia de *path-relinking*. Assim como a estratégia anterior, a ideia é caminhar por todos os intermediários de uma solução inicial até a solução guia.

A estratégia de *path-relinking* aplicada a todo ótimo local parece ser mais efetiva do que simplesmente utilizar como um passo de pós-otimização. Contudo, esta inclusão aumenta consideravelmente o número de operações e o desempenho do algoritmo. Além das duas estratégias básicas, diversas outras combinações foram implementadas, tais como (RESENDE; RIBEIRO, 2002):

- Aplicar o *path-relinking* em algumas iterações do *GRASP* escolhidas aleatoriamente.

- Explorar duas trajetórias, utilizando primeiramente $\mathbf{x_1}$ como solução inicial e depois inicializar com $\mathbf{x_2}$.
- Explorar apenas uma trajetória iniciando de $\mathbf{x_1}$ (solução resultado da busca local) ou $\mathbf{x_2}$ (solução escolhida aleatoriamente do conjunto temporário).
- Percorrer apenas uma parte da trajetória (*truncated path-relinking*).

Todas as alternativas envolvem conflitos entre tempo computacional e qualidade das soluções. Ribeiro, Uchoa e Werneck (2002) observam que explorar duas trajetórias entre dois pontos leva duas vezes mais o tempo necessário para explorar uma única trajetória, contudo, com poucas melhorias na qualidade das soluções. Os pesquisadores também observaram que melhores soluções são encontradas quando o *path-relinking* começa com o melhor entre os dois pontos. Como a vizinhança da solução inicial é mais cuidadosamente explorada do que a solução guia, logo, começando do melhor entre as duas soluções, maiores serão as chances de explorar as soluções mais promissoras. Pela mesma razão, as melhores soluções são usualmente encontradas mais próximas da solução inicial do que da solução guia, o que permite abortar a trajetória antes que a solução guia seja atingida.

5.3.3 Algoritmo *GRASP-PR Rule Learning*

A estratégia *GRASP* gera soluções novas com certas características: exploração (*exploration*), e a busca por sua vizinhança (*exploitation*). A exploração busca diversificação através da escolha aleatória dos itens da lista *RCL*; ao invés de sempre optar pela melhor solução, às vezes boas soluções são escolhidas para obter informações que possam levar para o ótimo global. A busca pela vizinhança trabalha com a intensificação de modo a reforçar os resultados já obtidos. Aproveitando o potencial mencionado, a intenção é criar um novo algoritmo que utilize o método *GRASP* para gerar regras apropriadas para a formação do conjunto de aproximação que maximize a *AUC*.

O código do novo algoritmo, chamado de *GRASP Path Relinking Rule Learning ou GRASP-PR Rule Learning* (GPR-RL), está no algoritmo 8. O algoritmo *GRASP-PR Rule Learning* possui o mesmo objetivo do Pareto Front Elite que é maximizar o critério *AUC* pela tentativa da construção da fronteira de Pareto de acordo com a sensitividade e a especificidade. O algoritmo obtém o conjunto de regras não-ordenadas em duas partes principais: *GRASP* e *path-relinking*. A primeira parte, *GRASP*, possui uma iteração principal com duas fases:

1. Construção das regras Elite (*ConstructGreedyRandomizedSolution*)

Algoritmo 8 Algoritmo *GRASP-PR Rule Learning*

```
InputInstance()
procedure grasp()
  boolean VPbias = true // true=VP rate como bias; false=TN rate como bias
  for i = to maxLoop
    Solution = ConstructGreedyRandomizedSolution( VPbias, percentageElements)
    localBest = LocalSearch(Solution, VPbias)
    UpdateSolution( localBest, EliteRules)
    VPbias = not VPbias
  endfor
  EliteRules = filterGeneric( EliteRules)
  return( EliteRules)
end grasp
non_dom = InsertParetoFront ( EliteRules )
procedure path-relinking( EliteRules,non_dom )
  foreach rule1 ∈ EliteRules
    foreach rule2 ∈ EliteRules
      path (rule1, rule2, non_dom )
    endforeach
  endforeach
return(non_dom)

procedure path(rule1, rule2, non_dom)
  intermediate = rule1
  while intermediate ≠ rule2
    foreach attribute ∈ rule2
      intermediate.add( attribute)
      evaluate( intermediate)
      non_dom = InsertParetoFront( intermediate )
    endforeach
  endwhile
return(Pareto)
```

2. Busca local (*LocalSearch*)

O resultado destas duas fases é a criação de uma lista de regras chamada regras Elite (*EliteRules*) que será melhorada com a segunda parte do algoritmo que utiliza o *path-relinking*. O *path-relinking* faz uma exploração pelas soluções intermediárias entre cada par de regras Elite tentando assegurar uma boa cobertura da Fronteira de Pareto.

A fase da construção das regras Elites é baseada na lista *RCL*. Um elemento candidato da *RCL* é um par atributo-valor (atributo e um valor possível para o mesmo). Para cada par possível, o algoritmo calcula a taxa de VP (Verdadeiro Positivo) e de VN (Verdadeiro Negativo) e escolhe *n%* dos melhores candidatos de acordo com taxa de VP, ou de acordo com a

taxa de VN, uma taxa de cada vez. Uma regra é inicializada na fase de construção com um par de atributo-valor escolhido aleatoriamente da *RCL* com probabilidade uniforme. Depois disso, todos os possíveis candidatos são avaliados e *RCL* é construída novamente. A matriz de confusão é recalculada para cada elemento candidato. Então, o algoritmo escolhe outro par de *RCL* aleatoriamente. Se o par escolhido melhorar a taxa de VP ou de VN, o par é incluído na regra em construção. Caso contrário, o par é removido da *RCL*. A construção termina quando *RCL* está vazia.

Cada regra é melhorada com o procedimento da busca local. O algoritmo seleciona aleatoriamente um par a ser trocado na solução corrente a cada iteração. Se a troca melhorar a taxa de VP (ou a taxa de VN), então, a nova regra substitui a solução corrente e o processo continua até não haver mais melhora. No final desta fase, as regras são inseridas no arquivo das soluções não-dominadas. Antes da inserção, o procedimento ($filterGeneric$) procura por regras mais genéricas ou específicas no arquivo. Uma regra é mais genérica do que outra regra se tem menos restrições de atributo e a mesma matriz de confusão. Somente as regras mais genéricas são mantidas no arquivo das regras não-dominadas.

A segunda parte do algoritmo aplica o *path-relinking* entre as regras Elite e, também, inicializa o arquivo das regras não-dominadas (*InsertParetoFront*). Para cada duas regras, o *path-relinking* cria as regras intermediárias entre elas (*path*). Considerando uma regra inicial e uma final, a ideia é substituir um par de atributo-valor da regra inicial com um par atributo-valor da regra final. Esta substituição é feita iterativamente até que a regra intermediária seja igual à regra final. A cada iteração, a regra intermediária é avaliada e se a regra não é dominada por nenhuma regra do arquivo Elite, ela é incluída no arquivo. A função remove soluções do arquivo que são dominadas pela nova regra. O arquivo das soluções não-dominadas é construído selecionando todas as regras não-dominadas de acordo com a sensitividade e a especificidade.

5.3.4 Experimentos do *GRASP-PR Rule Learning*

Nesta seção, apresentamos os experimentos realizados com o algoritmo *GRASP-PR Rule Learning* e a comparação dos resultados de *AUC* com os trabalhos relacionados. Os experimentos com o *GRASP-PR Rule Learning* foram feitos com a mesma metodologia utilizada para os experimentos do PFE e *MOPSO* das seções anteriores. Para tal, foram utilizadas as mesmas bases de dados pareadas da UCI Machine Learning Repository (NEWMAN; BLAKE; MERZ, 1998) (Tabela 5.1). As colunas da Tabela, em ordem, são: o número da base de dados (coluna '#'), a descrição do conjunto de dados (coluna 'Conjunto'), o número de atributos (coluna 'At'), número de exemplos (coluna 'Exe') e o número de classes. Os experimentos

Tabela 5.8: *AUC* médio e desvio-padrão (entre parênteses) para o algoritmo *GRASP-PR Rule Learning* com parâmetro de número máximo de soluções construídas atribuídos para 100 (segunda coluna), 200 (terceira coluna) e 300 (quarta coluna). A cor cinza indica valor estatisticamente melhor, segundo teste de Wilcoxon, do que o correspondente para o parâmetro $maxLoop = 100$. Os *p-value*'s estão da Tabela B.2.

#	**100 regras**	**200 regras**	**300 regras**
1	98.75(0.41)	99.04(0.25)	99.07(0.20)
2	67.99(1.12)	68.07(0.64)	68.05(0.67)
3	90.27(1.58)	90.33(1.51)	90.52(1.63)
4	69.15(2.92)	70.66(1.76)	70.72(1.29)
5	74.30(0.60)	74.67(0.46)	74.89(0.34)
6	82.29(2.14)	82.13(1.31)	82.28(0.99)
7	59.27(1.73)	59.32(1.05)	58.60(0.84)
8	88.08(0.55)	88.59(0.33)	88.79(0.36)
9	94.43(1.17)	96.13(0.76)	96.77(0.45)
10	98.50(0.15)	98.91(0.09)	99.04(0.07)
11	95.89(0.39)	96.67(0.26)	96.87(0.27)
12	98.11(0.50)	98.35(0.39)	98.50(0.32)
13	98.90(0.17)	99.48(0.08)	99.70(0.04)
14	71.18(1.06)	71.02(0.65)	71.29(0.38)
15	89.95(0.45)	90.70(0.27)	90.89(0.32)
16	94.24(0.95)	96.83(0.51)	97.66(0.31)
Média	85.71	86.31	86.48

foram executados utilizando estratégia de validação cruzada para 10 partições (*10-fold cross-validation* (WITTEN; FRANK, 1999)) e para todos os algoritmos foram utilizados os mesmos conjuntos de treinamento e de teste.

O algoritmo *GRASP-PR Rule Learning* foi executado com a lista *RCL* contendo 50% dos melhores pares de acordo com a taxa de VP (ou VN). Três valores são utilizados para o parâmetro *maxLoop* que controla o número de soluções geradas na fase de construção: 100 (50 regras positivas e 50 regras negativas), 200 (100 regras positivas e 100 regras negativas) e 300 (150 regras positivas e 150 regras negativas). Para cada valor de *maxloop*, 50 rodadas independentes são executadas para cada conjunto de exemplo.

O *AUC* médio e os desvios-padrão para cada conjunto de exemplos são apresentados na Tabela 5.8. A primeira coluna contém o número da base de dado (o mesmo da Tabela 5.1), a segunda coluna tem os valores médios de *AUC* para parâmetro $maxLoop = 100$, a terceira coluna possui os valores para o $maxLoop = 200$ e a quarta coluna possui os valores para o $maxLoop = 300$.

Na Tabela 5.8, as cores de fundo indicam diferenças estatísticas entre os resultados segundo o teste de Wilcoxon (WILCOXON, 1945) com 95% de nível de confiança. Os valores

da segunda coluna da Tabela 5.8 (100 regras) são a base para a comparação. O cinza escuro indica valores estatisticamente melhores do que o resultado com $maxLoop = 100$. As células sem cores indicam valores sem diferença entre os valores bases. É possível notar a melhora dos valores *AUC* para o algoritmo *GPR-RL* com o aumento do parâmetro $maxLoop$. A comparação entre os resultados para o parâmetro $maxLoop$ com valor 100 e 200 indicam uma melhora em 11 das 16 bases de exemplos. Se o parâmetro é aumentado para 300, os resultados também melhoram em 11 conjuntos de dados. Comparando-se os resultados com o parâmetro $maxLoop$ com valor 200 e 300 (Tabela C.12) é possível observar uma melhora em quatro bases de dados.

A seguir, são apresentadas as análises feitas nas 16 bases de dados (5.1) entre os algoritmos relacionados comparados com cada um dos três grupos de resultados executados com os valores 100, 200 e 300 para o parâmetro $maxLoop$. Por falta de todos os valores dos algoritmos relacionados necessários para a devida comparação com o teste de Wilcoxon, a comparação do *GRASP-PR Rule Learning* (Tabela 5.8) com os algoritmos relacionados (Tabela 5.2) foi feita utilizando o teste T (GOSSET, 1908) (GOSSET; PEARSON; WISHART, 1942).

GPR-RL maxLoop=100 X trabalhos relacionados

O primeiro conjunto de resultados utilizado como base de comparação foi o dos valores de *AUC* obtidos pelo *GPR-RL* com o parâmetro $maxLoop = 100$ (segunda coluna da Tabela 5.8). Todas as diferenças estatísticas das comparações podem ser verificadas visualmente na Tabela C.13. Para facilitar a comparações, a Tabela 5.9 traz um resumo das comparações por base de dados. A comparação indicou bons resultados para o *GPR-RL* em 6 bases de dados. Nas bases #2, #4, #6, #8, #3 e #5, o algoritmo *GPR-RL* apresentou desempenho estatisticamente superior, respectivamente, a 5, 5, 4, 4, 3 e 3 algoritmos relacionados. Na base de dados #14 o resultado foi semelhante a todos os algoritmos relacionados. Porém, nas bases #15, #16, #11, #13 e #10 os valores de *AUC* foram inferiores a, respectivamente, 3, 4, 5, 5 e 4 algoritmos relacionados. Destas bases, o pior desempenho foi para a base de dados #10, na qual o algoritmo *GPR-RL* apresentou o pior resultado de todos os algoritmos.

Na Tabela 5.10 contém o resumo das comparações por algoritmo. O *GRASP-PR Rule Learning* teve os melhores desempenhos contra os algoritmos C4.5 e Slipper apresentando valores melhores em sete bases de dados. Os algoritmos C4.5 e Slipper obtiveram valores maiores em, respectivamente, duas e quatro bases de dados. O *GPR-RL* teve desempenho semelhante aos algoritmos C4.5NP, CN2OR e Ripper com valores maiores em, respectivamente, quatro, quatro e cinco bases de dados e com valores piores em, respectivamente, cinco, cinco e três bases de dados. O algoritmo CN2 teve melhor desempenho em 8 bases de dados, porém, em

Tabela 5.9: Total de valores de *AUC* do algoritmo *GPR-RL* com $maxLoop = 100$ (Tabela 5.8) que é superior (segunda coluna), similar (terceira coluna) ou inferior (quarta coluna) aos valores dos trabalhos relacionados (Tabela 5.2). A contagem é feita com base nas comparações destacadas visualmente na Tabela C.13.

#	Superior	Similar	Inferior
1	1	4	2
2	5	2	0
3	3	4	0
4	5	2	0
5	3	4	0
6	4	3	0
7	1	5	1
8	4	3	0
9	2	4	1
10	0	0	7
11	0	2	5
12	1	4	2
13	2	0	5
14	0	7	0
15	1	3	3
16	0	3	4
Total	32	50	30

Tabela 5.10: Total de bases de dados que o algoritmo *GPR-RL* com $maxLoop = 100$ (Tabela 5.8) é superior (segunda linha), similar (terceira linha) ou inferior (quarta linha) comparado com os resultados dos trabalhos relacionados (Tabela 5.2). A contagem é feita com base nas comparações destacadas visualmente na Tabela C.13.

Comparação	ROCCER	C4.5	C4.5NP	CN2	CN2OR	Ripper	Slipper	Total(%)
Superior	1	7	4	4	4	5	7	32(0,29)
Similar	12	7	7	4	7	8	5	50(0,45)
Inferior	3	2	5	8	5	3	4	30(0,27)

quatro bases de dados o algoritmo *GPR-RL* teve melhores resultados. O algoritmo *GPR-RL* teve desempenho similar ao ROCCER, apresentando valor superior em uma base de dados, inferior em 3 bases e semelhantes nas outras 12 bases.

Numa análise geral de todas as comparações (112 comparações em 16 bases de dados e sete algoritmos) sumarizadas na última linha da Tabela 5.10, o algoritmo *GPR-RL* teve valores estatisticamente superiores em 32 (29%) bases de dados, valores inferiores em 30 (27%) bases de dados e nas 50 (45%) demais bases de dados não houve diferença significativa. Ou seja, em quase 70% das comparações o *GRASP-PR Rule Learning* obteve valores iguais ou superiores aos algoritmos relacionados

GPR-RL maxLoop=200 X trabalhos relacionados

Comparando os valores de *AUC* do *GPR-RL* executado com o parâmetro $maxLoop = 200$ (terceira coluna da Tabela 5.8) com outros algoritmos, os resultados também são mais favoráveis para o algoritmo *GPR-RL*. As diferenças estatísticas podem ser observadas visualmente na Tabela C.14 e estão resumidas nas Tabelas 5.11 e 5.12. Analisando as bases de dados (Tabelas 5.11 e C.14), os melhores resultados estão nas bases #2, #4 e #8 onde o *GPR-RL* foi melhor do que cinco algoritmos. Na base #2 e #8 o *GPR-RL* foi melhor do que os algoritmos C4.5, C4.5NP, CN2, CN2OR e Slipper e na base #4 foi melhor do que os algoritmos C4.5, CN2, CN2OR, Ripper e Slipper. Nas bases #5 o algoritmo *GPR-RL* tem valores melhores do que quatro outros algoritmos (C4.5, C4.5NP, CN2OR, Ripper). Nas bases #3, #6 e #16 o *GPR-RL* foi melhor do que os algoritmos C4.5, Ripper e Slipper. Na base #9 também foi melhor do que três outros algoritmos: C4.5, C4.5NP e Ripper. O algoritmo *GPR-RL* tem valores similares aos outros algoritmos nas bases de exemplos #7, #12 e #14. Apenas nas bases #10, #11 e #13 o desempenho do algoritmo *GPR-RL* foi pior do que outros algoritmos; destas, na base #10 o algoritmo teve o pior desempenho. Na base #11 o algoritmo *GPR-RL* foi melhor do que os algoritmos ROCCER e C4.5, porém com valores menores em comparação com os algoritmos C4.5NP, CN2, CN2OR e Slipper. Na base #13 o *GPR-RL* foi melhor do que o ROCCER e Slipper, porém inferior aos algoritmos C4.5NP, CN2 e CN2OR.

Comparando os resultados entre os algoritmos (Tabela 5.12), nota-se que o *GPR-RL* teve a maioria dos valores melhores ou semelhantes. Os melhores resultados foram contra os algoritmos C4.5, Ripper e Slipper. Comparando com o algoritmo C4.5, o algoritmo *GPR-RL* teve desempenho melhor em doze (75%) das 16 bases de dados e apenas um resultado inferior. Contra o algoritmo Ripper, o algoritmo *GPR-RL* foi melhor do que sete (44%) bases de dados e foi inferior em uma base de dados. O algoritmo Slipper teve desempenho inferior em nove (56%) bases de dados e desempenho superior em três (19%). Comparando-se com o algoritmo ROCCER, o *GPR-RL* teve melhores resultados em duas bases de dados e em outras duas bases de dados os valores foram inferiores; nas outras 12 (75%) bases de dados os resultados foram semelhantes. Contra os algoritmos C4.5NP, CN2 e CN2OR, o algoritmo *GPR-RL* obteve resultados semelhantes em boa parte das bases; os resultados foram semelhantes em, respectivamente, 9(56%), 9(56%) e 8(50%) bases de dados; além de obter melhores valores em, respectivamente, quatro, três e quatro bases de dados.

Numa análise geral de todas as 112 comparações (última coluna da Tabela 5.12), o algoritmo *GPR-RL* teve valores estatisticamente superiores em 41 (37%) bases de dados, valores inferiores em apenas 18 (16%) bases de dados e nas 53 (47%) demais bases de dados não houve

Tabela 5.11: Total de valores de *AUC* do algoritmo *GPR-RL* com $maxLoop = 200$ (Tabela 5.8) que é superior (segunda coluna), similar (terceira coluna) ou inferior (quarta coluna) aos valores dos trabalhos relacionados (Tabela 5.2). A contagem é feita com base nas comparações destacadas visualmente na Tabela C.14.

#	**Superior**	**Similar**	**Inferior**
1	1	6	0
2	5	2	0
3	3	4	0
4	5	2	0
5	4	3	0
6	3	4	0
7	2	4	1
8	5	2	0
9	3	4	0
10	0	0	7
11	2	1	4
12	1	4	2
13	2	2	3
14	0	7	0
15	2	4	1
16	3	4	0
Total	41	53	18

Tabela 5.12: Total de bases de dados que o algoritmo *GPR-RL* com $maxLoop = 200$ (Tabela 5.8) é superior (segunda linha), similar (terceira linha) ou inferior (quarta linha) comparado com os resultados dos trabalhos relacionados (Tabela 5.2). A contagem é feita com base nas comparações destacadas visualmente na Tabela C.14.

Comparação	**ROCCER**	**C4.5**	**C4.5NP**	**CN2**	**CN2OR**	**Ripper**	**Slipper**	**Total(%)**
Superior	2	12	4	3	4	7	9	41(0,37)
Similar	12	3	9	9	8	8	4	53(0,47)
Inferior	2	1	3	4	4	1	3	18(0,16)

Tabela 5.13: Total de valores de *AUC* do algoritmo *GPR-RL* com $maxLoop = 300$ (Tabela 5.8) que é superior (segunda coluna), similar (terceira coluna) ou inferior (quarta coluna) aos valores dos trabalhos relacionados (Tabela 5.2). A contagem é feita com base nas comparações destacadas visualmente na Tabela C.15.

#	Superior	Similar	Inferior
1	1	6	0
2	5	2	0
3	3	4	0
4	5	2	0
5	4	3	0
6	4	3	0
7	2	4	1
8	5	2	0
9	5	2	0
10	0	0	7
11	2	1	4
12	1	6	0
13	4	1	2
14	0	7	0
15	3	3	1
16	4	3	0
Total	48	49	15

diferença significativa. Nota-se que o aumento do valor parâmetro *maxLoop* de 100 para 200, provocou uma diminuição nos resultados produzidos pelo *GPR-RL* que são estatisticamente inferiores a outros algoritmos. Esta diminuição foi de 27% para 16%.

GPR-RL maxLoop=300 X trabalhos relacionados

Comparando os valores de *AUC* do *GPR-RL* executado com o maior valor (300) para o parâmetro *maxLoop* (quarta coluna da Tabela 5.8) os dados são ainda mais favoráveis para o algoritmo. As diferenças estatísticas podem ser observadas visualmente na Tabela C.15 e estão resumidas nas Tabelas 5.13 e 5.14. Analisando as bases de dados, os melhores resultados são nas bases #2, #4, #8, #9, onde o *GPR-RL* foi melhor do que 5 algoritmos (verificar Tabela C.15). Nas bases #5, #6, #13 e #16 o algoritmo *GPR-RL* tem valores melhores do que quatro outros algoritmos; destas bases, apenas na base #13 o algoritmo *GPR-RL* teve valores menores do que 2 outros algoritmos. Nas bases #3 e #15 o algoritmo ainda obtém melhores resultados do que 3 outros algoritmos. O algoritmo *GPR-RL* tem valores similares aos outros algoritmos nas bases de exemplos #7, #12 e #14. Apenas nas bases #10 e #11 o desempenho do algoritmo *GPR-RL* foi pior do que, respectivamente, sete e quatro algoritmos.

Fazendo uma comparação dos resultados entre os algoritmos, nota-se que o *GPR-RL*

Tabela 5.14: Total de bases de dados que o algoritmo *GPR-RL* com $maxLoop = 300$ (Tabela 5.8) é superior (segunda linha), similar (terceira linha) ou inferior (quarta linha) comparado com os resultados dos trabalhos relacionados (Tabela 5.2). A contagem é feita com base nas comparações destacadas visualmente na Tabela C.15.

Comparação	**ROCCER**	**C4.5**	**C4.5NP**	**CN2**	**CN2OR**	**Ripper**	**Slipper**	**Total(%)**
Superior	5	13	4	4	5	8	9	48(0,43)
Similar	9	2	10	9	7	7	5	49(0,44)
Inferior	2	1	2	3	4	1	2	15(0,13)

teve valores melhores ou iguais na maioria das bases de dados. Os melhores resultados foram contra os algoritmos C4.5, Slipper, Ripper e ROCCER. Comparando com o algoritmo C4.5, o algoritmo *GPR-RL* teve desempenho melhor em 13 das 16 bases de dados e com apenas um resultado inferior. Contra o algoritmo Ripper, o algoritmo *GPR-RL* foi melhor do que 9 bases de dados e foi inferior em duas bases de dados. O algoritmo Slipper teve desempenho inferior em oito bases de dados e foi superior em uma base de dados. Comparando com o algoritmo ROCCER, o *GPR-RL* teve melhores resultados em 5 bases de dados e em 2 bases de dados os valores foram inferiores. Contra os algoritmos C4.5NP, CN2 e CN2OR, o algoritmo *GPR-RL* ainda obteve resultados semelhantes na maioria das bases, respectivamente, em 10, 9 e sete bases de dados; além de obter melhores valores em, respectivamente, quatro, quatro e cinco bases de dados.

Numa análise geral de todas as 112 comparações, o algoritmo *GPR-RL* teve valores estatisticamente superiores em 48 (43%) bases de dados, valores inferiores em apenas 15 (13%) bases de dados e nas 49 (44%) demais bases de dados não houve diferença significativa. Nota-se que o aumento do parâmetro *maxLoop*, melhora os resultados de *AUC* de modo que a porcentagem de comparações que o algoritmo *GPR-RL* passa de 29% com $maxLoop = 100$ para 49% com $maxLoop = 300$.

5.4 Comparação do algoritmo *GRASP-PR Rule Learning* com PFE e *MOPSO*

Nesta seção, os resultados do algoritmo *GRASP-PR Rule Learning* são comparados com os resultados do Pareto Front Elite. Optou-se pelo algoritmo PFE pelo fato de possuir os mesmos objetivos do *GPR-RL*, pelos bons resultados de *AUC* apresentados entre os trabalhos relacionados e a possibilidade da obtenção das Fronteiras de Pareto. Primeiramente os valores de *AUC* dos dois algoritmos são comparados; após, segue-se uma análise do total de regras geradas na seção 5.4.1. Os conjuntos de aproximação do *GPR-RL* são comparados com a Fronteira

do PFE na seção 5.4.2. Esta seção é finalizada com uma análise multiobjetiva dos conjuntos de aproximação do *GPR-RL* com outro algoritmo metaheurístico multiobjetivo *MOPSO*.

Os resultados do Pareto Front Elite utilizados para a comparação serão os executados com parâmetros mínimos atribuídos com valor zero, quarta coluna da Tabela 5.3. Esta execução pode ser considerada como um método "todas as regras", ou seja, o algoritmo gera todas as regras possíveis para o conjunto de exemplos (busca exaustiva). As regras desta execução com certeza formam a fronteira de Pareto de acordo com sensitividade e especificidade. A execução sem os parâmetros mínimos não é, obviamente, a melhor opção para grandes conjuntos de exemplos; entretanto, as fronteiras de Pareto geradas pelo PFE sem os parâmetros mínimos servem como referência para outros algoritmos.

Os valores de *AUC* médio e os desvios-padrão do *GPR-RL* considerados para as comparações estão na Tabela 5.8. A segunda coluna desta tabela contém os valores para parâmetro $maxLoop = 100$, a terceira coluna possui os valores para o $maxLoop = 200$ e a quarta coluna possui os valores para o $maxLoop = 300$.

As comparações foram feitas com a utilização do teste Wilcoxon com 95% de nível de confiança. Em todas as comparações feitas entre o algoritmo PFE e *GPR-RL* não foi encontrada nenhuma diferença em todas as bases de dados avaliadas. Em outras palavras, o algoritmo metaheurístico é capaz de obter valores de *AUC* semelhantes ao algoritmo PFE com todas as regras.

Uma análise feita na seção 5.3.4 mostrou que existe uma melhora entre os resultados do *GPR-RL* com o aumento do parâmetro *maxLoop* que aumenta o número de regras geradas. Esta melhora é observada em bases de exemplos nas quais não foi possível a execução do algoritmo PFE sem os parâmetros mínimos (#4, #9, #10, #11, #15 e #16). Estas bases são as que possuem um maior número de atributos e mais exemplos. Isto confirma que é possível que o *GPR-RL* gere os conjuntos para grandes bases de dados, bastando variar o parâmetro do número de regras geradas (*maxLoop*).

5.4.1 Regras Geradas

A próxima análise leva em conta a eficiência do algoritmo em gerar poucas ou muitas regras para a seleção da fronteira. Com isso, pode-se responder se o algoritmo *GRASP-PR Rule Learning* gera ou não um grande número de regras. A Tabela 5.15 apresenta o número médio de regras geradas por cada algoritmo e entre parênteses o desvio-padrão correspondente a cada média. Para a comparação foi utilizado o algoritmo Pareto Front Elite com os parâmetros

Tabela 5.15: Total de regras geradas tendo como referência o algoritmo PFE. As colunas contêm, respectivamente, o número da base de exemplo, média de regras geradas para o PFE sem parâmetros mínimos, média de tentativas para *GPR-RL* com maxLoop=100, proporção entre maxLoop=100 e PFE, média de tentativas do *GPR-RL* com maxLoop=200, a proporção maxLoop=200 e PFE, média de tentativas para o *GPR-RL* com maxLoop=300 e sua proporção baseado no PFE.

#	PFE	*GPR-RL* 100	100/PFE	*GPR-RL* 200	200/PFE	*GPR-RL* 300	300/PFE
1	131532.5(839.82)	11926.21(348.83)	0.09	35615.29(705.43)	0.27	68003.63(1110.36)	0.52
2	5805.6(47.38)	7964.80(224.27)	1.37	20631.36(547.69)	3.55	35767.82(889.58)	6.16
3	14647.2(97.97)	7808.03(268.54)	0.53	21855.60(593.85)	1.49	40805.98(1018.84)	2.79
5	863287.5(3015.42)	18673.76(356.95)	0.02	57885.83(1145.45)	0.07	112472.26(1840.79)	0.13
6	32907.8(534.07)	11746.56(340.33)	0.36	36389.92(1010.06)	1.11	71860.22(1848.74)	2.18
7	409(6.06)	3966.66(119.32)	9.70	9026.90(245.25)	22.07	14693.90(349.07)	35.93
8	1167010.4(7769.39)	13244.88(353.18)	0.01	37514.63(882.15)	0.03	68318.42(1427.47)	0.06
12	1213.8(11.26)	5303.58(206.91)	4.37	13162.39(569.90)	10.84	22679.77(771.76)	18.68
13	124047.6(44.63)	6947.60(205.25)	0.06	18424.71(492.23)	0.15	33564.02(785.29)	0.27
14	63518.9(396.48)	10103.53(230.69)	0.16	27635.40(574.68)	0.44	48966.56(931.01)	0.77
Tot	2467899.20	107789.13	0.04	305777.43	0.12	517132.58	0.22

mínimos atribuídos com o valor zero. O total do algoritmo PFE é o número de todas as regras geradas pelo algoritmo de associação para cada conjunto de dados. O total das regras para o *GPR-RL* é a soma das regras criadas na fase de construção, mais o total das tentativas na fase da procura local e o número de regras intermediárias criadas na fase do *path-relinking*.

A execução do algoritmo Pareto Front Elite com mínimo de suporte e confiança atribuídos com zero gera todas as possíveis regras para cada *fold* de cada conjunto de exemplo. A primeira coluna da Tabela 5.15 possui o número do conjunto de exemplo. A segunda coluna tem o número médio de regras para cada conjunto de exemplos gerado pelo PFE. A terceira e quarta colunas possuem informações sobre as regras geradas do algoritmo *GRASP-PR Rule Learning* com o parâmetro $maxLoop = 100$. A terceira coluna tem o total de regras geradas pelo algoritmo e a quarta coluna possui a proporção do total baseado no resultado do algoritmo PFE. A quinta coluna tem o total de regras geradas do algoritmo *GPR-RL* com o parâmetro $maxLoop = 200$. A razão deste total pelo resultado do PFE está na sexta coluna. A sétima coluna tem o total de regras geradas pelo parâmetro $maxLoop = 300$ e a oitava coluna a sua proporção em relação ao resultado do PFE.

As cores de fundo da Tabela 5.15 indicam se a execução do algoritmo *GRASP-PR Rule Learning* tem desempenho estatisticamente inferior ou superior ao algoritmo PFE em relação ao número de regras geradas. As células em cinza escuro indicam as execuções que geram menos regras que o algoritmo PFE. A cor cinza claro indica os resultados com mais regras do que o algoritmo Pareto Front Elite. Em cinco conjuntos de exemplos (#1, #5, #8, #13 e #14), o algoritmo metaheurístico é muito mais eficiente do que os resultados utilizados como base. Nestes conjuntos, houve o aumento dos valores de *AUC*, porém, mesmo com o aumento do

número de tentativas da iteração principal do *GPR-RL* (aumento do parâmetro *maxLoop*), o número de regras geradas não chegou a ser superior ao PFE. Estes mesmos conjuntos de dados são os que mais possuem atributos entre os conjuntos comparados. Quanto maior o número de atributos, maior o total de regras possíveis, que é o espaço de procura. O melhor desempenho é a execução do parâmetro $maxLoop = 100$ para o conjunto de exemplos #8 que gera apenas 1% de todas as regras geradas pelo algoritmo PFE. Além do mais, o *AUC* para o algoritmo *GPR-RL* é similar ao resultado do PFE. Para conjuntos de exemplos com poucos atributos (#2, #3, #7 e #12), o algoritmo *GRASP-PR Rule Learning* gera mais regras que o algoritmo PFE. Como o número de regras geradas pelo *GPR-RL* pode ser controlado, o problema é contornado com a diminuição do parâmetro *maxLoop* para conjuntos com poucos atributos.

Analisando a última linha da Tabela 5.15 é possível verificar como o *GPR-RL* é muito mais eficiente para a geração de regras. Observa-se que o algoritmo *GPR-RL* com o parâmetro $maxLoop = 100$ gera aproximadamente 4% do total de regras gerados pelo PFE sem parâmetros. Mesmo com o parâmetro $maxLoop = 300$, que gera os maiores conjuntos de regras, o algoritmo *GRASP-PR Rule Learning* gera apenas 22% do total de regras possível.

5.4.2 Análise dos Conjuntos de Aproximação

Após os bons valores de *AUC* comparado com os algoritmos relacionados, é feita uma avaliação mais detalhada dos resultados obtidos pelo algoritmo *GRASP-PR Rule Learning*. Para tal, foi necessária a escolha de um algoritmo como base de comparação. Os algoritmos PFE, *MOPSO* e *GPR-RL* são abordagens com o mesmo objetivo de encontrar um conjunto não-ordenado de regras que maximize *AUC* (ISHIDA et al., 2009). Entre PFE e *MOPSO*, o primeiro possui resultados melhores do que os apresentados pelo segundo, conforme seção 5.2.4. A seguir, os resultados do PFE foram utilizados como base de comparação para a avaliação dos conjuntos de aproximação. Os melhores valores do algoritmo PFE foram os obtidos sem a indicação dos parâmetros mínimos, ou seja, com todas as regras possíveis em cada base de dados. Depois, os conjuntos de aproximação dos algoritmos multiobjetivos heurísticos *GPR-RL* e *MOPSO* serão comparados.

O Pareto-ótimo do algoritmo PFE é comparado com 50 conjuntos de aproximação do algoritmo *GRASP-PR Rule Learning* obtidos de 50 diferentes execuções. Os conjuntos comparados foram obtidos com a mesma metodologia de execução e com as mesmas bases de dados pareadas; são os mesmos conjuntos cujos valores de *AUC* foram apresentados nas seções 5.1.5 e 5.3.4. Os conjuntos de aproximação são analisados utilizando rankeamento pela dominância (KNOWLES; THIELE; ZITZLER, 2005) e os indicadores de qualidade utilizando *framework*

PISA (BLEULER et al., 2003). Em cada conjunto de exemplos existem as 10 partições (*folds*) separadas pela validação cruzada (*cross-validation*). Cada comparação inclui resultados obtidos de um *fold* de *cross-validation*. Então, para cada exemplo de dados existem 10 comparações entre os algoritmos.

Na comparação de rankeamento pelo conceito de dominância, cada conjunto de aproximação recebe um *rank* baseado na relação de dominância através da contagem dos números da fronteira que o domina. Os resultados para todos os conjuntos de dados, incluindo todos os *folds*, indicam que nenhum dos algoritmos, nem PFE ou *GPR-RL*, gera melhores curvas de aproximação de acordo com o *dominance ranking* com um nível de significância de 5% utilizando o indicador epsilon unário.

Como os resultados com o rankeamento pelo conceito de dominância não são conclusivos, três indicadores de qualidade são utilizados nesta ordem: hipervolume (ZITZLER; THIELE, 1999), epsilon aditivo (ZITZLER et al., 2003) e $R2$ (HANSEN; JASZKIEWICZ, 1998). Considerando que o objetivo é maximizar os critérios de sensitividade e especificidade, o ponto de referência utilizado para calcular o hipervolume e epsilon é $(0,0)$ e o ponto de referência para o indicador $R2$ é $(1,1)$.

A seguir, os conjuntos de aproximação do algoritmo Pareto Front Elite sem os parâmetros mínimos e o algoritmo *GRASP-PR Rule Learning* com o parâmetro $maxLoop = 100$ são testados. Depois, a versão do algoritmo PFE é comparada com os resultados do algoritmo *GPR-RL* com o parâmetro $maxLoop = 200$, seguido da comparação com os resultados do parâmetro $maxLoop = 300$. Os indicadores foram calculados com o auxílio do *framework* PISA (BLEULER et al., 2003). E para a comparação dos algoritmos foi utilizado o teste estatístico Wilcoxon para validar a hipótese $H0$: algoritmos em questão serem diferentes; se o *p-value* for menor do que $0,05$ então os algoritmos são estatisticamente diferentes segundo o indicado utilizado.

PFE com todas as regras X *GPR-RL* maxLoop=100

Nesta seção são comparados os conjuntos de aproximação do algoritmo Pareto Front Elite sem parâmetros mínimos com o algoritmo *GRASP-PR Rule Learning* executado com o parâmetro $maxLoop = 100$. A hipótese $H0$: algoritmo PFE sem parâmetros ser diferente do *GPR-RL* para a base de dados e partição em questão. O valor do *p-value* de cada avaliação é encontrado nas Tabelas na seção D.1; os valores para o indicador de hipervolume estão nas Tabelas D.1 e D.2, os valores para o indicador epsilon estão nas Tabelas D.3 e D.4 e os valores para o indicador $R2$ estão nas Tabelas D.5 e D.6.

Tabela 5.16: Número de vezes, de acordo com cada indicador (hipervolume, epsilon e *R*2), que o algoritmo PFE sem parâmetros mínimos tem melhores conjuntos de aproximação com regras positivas e negativas do que o algoritmo *GRASP-PR Rule Learning* com o parâmetro $maxLoop = 100$.

	Regras Positivas			Regras Negativas		
#	hipervolume	epsilon	*R*2	hipervolume	epsilon	*R*2
1	0	0	0	0	0	0
2	0	1	1	9	10	9
3	4	4	3	0	0	0
5	10	10	7	10	10	1
6	1	2	0	9	10	0
7	0	0	0	1	0	0
8	9	10	0	10	10	8
12	0	0	0	0	0	0
13	8	9	0	0	0	0
14	10	10	5	8	9	1
Total(%)	42(0.42)	46(0.46)	16(0.16)	47(0.47)	49(0.49)	19(0.19)

O resumo dos resultados para as regras positivas e negativas analisados com os indicadores hipervolume, epsilon e *R*2 estão na Tabela 5.16. Os valores da primeira, segunda e terceira coluna da Tabela 5.16 mostram o número de vezes que o PFE obtém resultados significantemente melhores do que o *GPR-RL* com o parâmetro $maxLoop = 100$, de acordo com cada indicador. Considerando estes indicadores, os resultados mostram que o algoritmo *GRASP-PR Rule Learning* gera fronteiras de Pareto comparáveis ao algoritmo Pareto Front Elite em 5 dos 10 conjuntos de exemplos (#1, #2, #6, #7 e #12). Entretanto, existem 4 conjuntos de exemplos (#5, #8, #13 e #14) nos quais o algoritmo metaheurístico não gera boas fronteiras.

A quinta, sexta e sétima coluna da Tabela 5.16 mostram o número de vezes, entre as regras negativas, em que o PFE é preferido ao *GRASP-PR Rule Learning* de acordo com os indicadores de qualidade. Os resultados mostram que o algoritmo *GPR-RL* apresenta entre as regras negativas desempenho similar ao que há entre as regras positivas. De acordo com os 3(três) indicadores, *GPR-RL* apresenta resultados com qualidade equivalente ao PFE em cinco conjuntos de exemplos (#1, #3, #7, #12 e #13).

A última linha da Tabela 5.16 mostra a porcentagem de vezes que o algoritmo PFE obtém fronteiras de Pareto melhores do que o algoritmo *GPR-RL* de acordo com cada indicador de qualidade. O complemento das porcentagens desta última linha representa o número de vezes que o algoritmo *GPR-RL* obtém conjuntos de aproximação com alta qualidade. Os resultados dos indicadores hipervolume e epsilon mostram que, em média, em mais de 50% das execuções, o algoritmo heurístico apresenta desempenho similar ao PFE. A análise semelhante do indicador *R*2 indica que 80% dos resultados do *GPR-RL* são semelhantes à fronteira de pareto do PFE.

Tabela 5.17: Número de vezes que o algoritmo PFE sem parâmetros possui melhores conjuntos de aproximação de regras positivas e negativas do que o algoritmo *GPR-RL* com parâmetro $maxLoop = 200$ de acordo com cada indicador (hipervolume, epsilon e $R2$).

	Regras Positivas			Regras Negativas		
#	hipervolume	epsilon	$R2$	hipervolume	epsilon	$R2$
1	0	0	0	0	0	0
2	0	0	0	8	9	8
3	3	3	3	0	0	0
5	9	3	6	10	10	1
6	1	1	0	6	9	0
7	0	0	0	1	0	0
8	6	6	0	10	10	5
12	0	0	0	0	0	0
13	0	0	0	0	0	0
14	10	10	4	1	3	1
Total(%)	29(0.29)	23(0.23)	13(0.13)	36(0.36)	41(0.41)	15(0.15)

PFE com todas as regras X maxLoop=200

Nesta seção verifica-se o aumento do número das regras para o algoritmo *GRASP-PR Rule Learning*. Os conjuntos de aproximação do algoritmo Pareto Front Elite sem parâmetros mínimos são comparados com os conjuntos do algoritmo *GRASP-PR Rule Learning* executado com o parâmetro $maxLoop = 200$. A hipótese $H0$: algoritmo PFE sem parâmetros ser diferente do *GPR-RL* para a base de dados e partição em questão. O valor do *p-value* de cada avaliação é encontrado nas Tabelas na seção D.2; os valores para o indicador de hipervolume estão nas Tabelas D.7 e D.8, os valores para o indicador epsilon estão nas Tabelas D.9 e D.10 e os valores para o indicador $R2$ estão nas Tabelas D.11 e D.12.

O resumo dos resultados dos indicadores para regras positivas estão apresentadas na segunda, terceira e quarta colunas da Tabela 5.17. Estas colunas mostram o número de vezes que o algoritmo PFE apresenta melhores fronteiras do que o *GRASP-PR Rule Learning* com o parâmetro $maxLoop = 200$ de acordo com indicadores hipervolume, epsilon e R2, respectivamente. Considerando estes indicadores, os resultados mostram que o algoritmo *GRASP-PR Rule Learning* gera conjuntos de aproximação equivalentes do algoritmo Pareto Front Elite em seis dos dez conjuntos de exemplos. Entretanto, em três conjuntos de exemplos (#5, #8 e #14), o algoritmo metaheurístico não gera boas fronteiras em comparação com PFE.

Nas quinta, sexta e sétima colunas da Tabela 5.17 é exposto o número de vezes que os conjuntos de aproximação gerados pelo PFE são melhores do que o algoritmo *GRASP-PR Rule Learning* para as regras negativas de acordo com os indicadores hipervolume, epsilon e $R2$, respectivamente. De acordo com estes indicadores, o *GPR-RL* obtém resultados similares

em seis conjuntos de exemplos.

A última linha da Tabela 5.17 mostra o total e a porcentagem de vezes que o PFE tem melhores fronteiras de Pareto para cada indicador. O complemento desta porcentagem mostra a porcentagem de vezes que o *GPR-RL* obtém conjuntos similares na maior parte das comparações. Entre as regras positivas, mais de 70% dos conjuntos de aproximação gerados pelo algoritmo *GRASP-PR Rule Learning* são similares ao PFE de acordo com hipervolume e epsilon. Entre as regras negativas a porcentagem foi a menor, mas ainda próxima a 65%. Para o indicador $R2$, 80% das fronteiras são similares.

PFE com todas as regras X maxLoop=300

Nesta seção são comparados os conjuntos de aproximação do algoritmo Pareto Front Elite sem parâmetros mínimos com o algoritmo *GRASP-PR Rule Learning* executado com o parâmetro $maxLoop = 300$. A hipótese $H0$: algoritmo PFE sem parâmetros ser diferente do *GPR-RL* para a base de dados e partição em questão. O valor do *p-value* de cada avaliação é encontrado nas Tabelas na seção D.3; os valores para o indicador de hipervolume estão nas Tabelas D.13 e D.14, os valores para o indicador epsilon estão nas Tabelas D.15 e D.16 e os valores para o indicador $R2$ estão nas Tabelas D.17 e D.18.

A Tabela 5.18 possui o resumo dos resultados dos indicadores, a primeira coluna é o número do conjunto de dados. A segunda, terceira e quarta colunas mostram o total de conjuntos de aproximação de regras positivas do PFE que são preferidos de acordo com, respectivamente, os indicadores de hipervolume, epsilon aditivo e $R2$. Considerando estes indicadores, os resultados mostram que o algoritmo *GRASP-PR Rule Learning* gera conjuntos de aproximações semelhantes ao algoritmo Pareto Front Elite em seis dos dez conjuntos de dados (#1, #2, #6, #7, #12 e #13). Entretanto, existem três conjuntos de dados (#5, #8 e #14) nos quais o algoritmo metaheurístico não gera bons resultados em comparação ao algoritmo PFE. Os resultados do conjunto de exemplos #8 são similares de acordo com os indicadores de hipervolume e $R2$, porém, o epsilon aditivo indica preferência para o PFE nos mesmos conjuntos.

A quinta, sexta e sétima colunas contêm o total de conjuntos de aproximação do PFE para regras negativas. Entre as regras negativas, o desempenho do algoritmo *GPR-RL* foi similar. De acordo com estes indicadores, *GPR-RL* obtém resultados similares em cinco conjuntos de dados (#1, #3, #7, #12 e #13).

A última linha da Tabela 5.18 mostra o total e a porcentagem de preferência para as fronteiras do algoritmo PFE para cada indicador. Os complementos destas porcentagens

Tabela 5.18: Número de vezes que o algoritmo PFE sem parâmetros possui melhores conjuntos de aproximação de regras positivas e negativas do que o algoritmo *GPR-RL* com parâmetro $maxLoop = 300$ de acordo com cada indicador (hipervolume, epsilon e $R2$).

	Regras Positivas			Regras Negativas		
#	hipervolume	epsilon	$R2$	hipervolume	epsilon	$R2$
1	0	0	0	0	0	0
2	0	0	0	7	7	7
3	3	3	3	0	0	0
5	5	1	6	7	7	0
6	1	1	0	9	9	0
7	0	0	0	1	0	0
8	0	5	0	7	10	2
12	0	0	0	0	0	0
13	0	0	0	0	0	0
14	7	10	4	1	1	1
Total(%)	16(0.16)	20(0.2)	13(0.13)	32(0.32)	34(0.34)	10(0.10)

mostram que o *GPR-RL* obteve conjuntos de aproximação similares na maioria das comparações. Entre as regras positivas, mais de 80% dos conjuntos de aproximação gerados pelo algoritmo *GRASP-PR Rule Learning* são similares ao PFE, de acordo com o hipervolume e epsilon. Entre as regras negativas, o desempenho de acordo com o hipervolume e epsilon aditivo foi menor, porém, com valores acima de 65%. De acordo com o indicador $R2$, quase 90% dos conjuntos de aproximação positivo e negativo são similares.

Comparações Multiobjetivas do *GRASP-PR Rule Learning*

São apresentados alguns experimentos de avaliação do algoritmo *GRASP-PR Rule Learning* com base no algoritmo multiobjetivo *MOPSO*. Os conjuntos de aproximação do *MOPSO* são comparados com os conjuntos do algoritmo *GRASP-PR Rule Learning* com *RCL* igual a 50% com máximo de 200 regras construídas. Os experimentos realizados com os dois algoritmos heurísticos, *MOPSO* e *GPR-RL*, foram apresentados nas seções, respectivamente, 5.2.4 e 5.4 e os valores de *AUC* foram apresentados nas Tabelas, respectivamente, 5.5 e 5.8.

Primeiramente os conjuntos de aproximação de ambos os algoritmos são comparados com o indicador rankeamento pelo conceito de dominância (*dominance ranking*) (KNOWLES; THIELE; ZITZLER, 2005) utilizando o *framework* PISA (BLEULER et al., 2003). Cada fronteira de Pareto recebe um *rank* baseado na relação de dominância através da contagem do total de fronteiras que o domina. O teste *one-tailed Mann-Whitney rank sum* (KNOWLES; THIELE; ZITZLER, 2005) é aplicado para verificar a diferença estatística entre o *ranking* dos dois algoritmos.

Os resultados para a maior parte dos conjuntos de exemplos, incluindo todos os *folds*, indicam que nenhum dos algoritmos, nem *GPR-RL* nem *MOPSO*, gera melhor conjunto de aproximação de acordo com o rankeamento pelo conceito de dominância com nível de significância de 5%.

Como os resultados do rankeamento pelo conceito de dominância não são conclusivos, utilizamos os indicadores qualitativos unários. Os resultados para os indicadores hipervolume, epsilon aditivo e $R2$ são apresentados na Tabela 5.19. A primeira coluna tem o número do conjunto de exemplo. A segunda e terceira colunas têm informação sobre indicador hipervolume para regras positivas. A segunda coluna indica o número de vezes que o *GRASP-PR Rule Learning* é preferido ao algoritmo *MOPSO* pelo indicador de hipervolume e a terceira coluna tem o número de vezes de preferência para o algoritmo *MOPSO*; o mesmo vale para a oitava e nona colunas, porém, para regras negativas. As quantidades de comparações que um algoritmo é preferido pelo outro, segundo o indicador epsilon aditivo, estão na quarta e quita colunas; na quarta coluna para o algoritmo *GPR-RL* e, a quinta, para o algoritmo *MOPSO*. Para as regras negativas, as preferências para os algoritmos *GPR-RL* e *MOPSO* estão, respectivamente, nas décima e décima primeira colunas. As sextas e sétimas colunas têm o número de vezes de preferência para as regras positivas segundo o indicador $R2$ e as duas últimas colunas para as regras negativas.

Entre as regras positivas e de acordo com os três indicadores, o *GPR-RL* é preferido ao algoritmo *MOPSO* em todos os *folds* dos conjuntos de exemplos #4, #6, #9, #11 e #15. Nos conjuntos de exemplos #2, #3, #5, #8, #10, #14 e #16 o *GPR-RL* é ainda preferido na maior parte dos *folds*. Somente para os conjuntos de exemplos #1, #7, #13 e #14 o *GPR-RL* não é preferido pelos três indicadores.

Entre as regras negativas, o desempenho dos conjuntos de aproximação do *MOPSO* é melhor, mas ainda não superam o *GPR-RL*. A maior parte dos conjuntos de aproximação do *GPR-RL* nos conjuntos de exemplos #1, #4, #5, #9, #10, #11 e #15 é preferida de acordo com os três indicadores. Nos conjuntos de exemplos #6 e #14, é possível notar a presença de conjuntos de aproximação incomparáveis.

O total de vezes que cada algoritmo é preferido de acordo com cada indicador está na última linha da Tabela 5.19. É possível observar que o algoritmo *GRASP-PR Rule Learning* é preferido na maioria das comparações. Entre as regras positivas, aproximadamente 81% das fronteiras do *GPR-RL* é preferido pelo indicador de hipervolume, 82% de preferência de acordo com o indicador epsilon aditivo e 75% pelo indicador $R2$. Até mesmo entre as regras negativas é possível observar a grande diferença entre os algoritmos.

Tabela 5.19: Resultados dos Indicadores nas comparações entre as fronteiras do algoritmo *GPR-RL* executado com o parâmetro $maxLoop = 200$ e algoritmo *MOPSO*. A primeira coluna possui o número do conjunto de dados. As outras colunas mostram o número de vezes que um algoritmo é preferido baseado no indicador correspondente à coluna. O valor 'G' representa o algoritmo *GPR-RL* e a letra 'M' representa o algoritmo *MOPSO*.

	Regras Positivas						Regras Negativas					
	hipervolume		epsilon		*R*2		hipervolume		epsilon		*R*2	
#	G	M	G	M	G	M	G	M	G	M	G	M
1	3	6	2	7	3	6	10	0	9	0	7	3
2	9	0	9	0	9	0	0	10	0	8	0	10
3	10	0	10	0	9	1	2	4	4	2	4	2
4	10	0	10	0	10	0	9	0	9	0	7	1
5	10	0	10	0	7	1	10	0	10	0	3	0
6	10	0	10	0	10	0	10	0	10	0	3	7
7	4	6	5	5	4	6	6	4	3	7	8	2
8	10	0	10	0	8	0	7	0	6	0	1	3
9	10	0	10	0	10	0	10	0	10	0	10	0
10	10	0	10	0	8	2	10	0	10	0	10	0
11	10	0	10	0	10	0	10	0	10	0	8	0
12	3	7	5	5	5	5	6	4	6	4	1	9
13	0	9	0	9	2	5	5	5	6	4	4	6
14	7	0	7	0	4	1	9	0	9	0	1	6
15	10	0	10	0	10	0	10	0	10	0	10	0
16	9	0	9	0	7	0	0	0	0	0	0	0
Total	125	28	127	26	116	27	114	27	112	25	77	49
%	0.8117	0.1854	0.8247	0.1745	0.7532	0.1800	0.7403	0.1788	0.7273	0.1678	0.5000	0.3267

Os experimentos conduzidos entre os algoritmos multiobjetivos mostram que a maioria dos resultados aprendidos pelo *GPR-RL* é preferida ao *MOPSO* pelos indicadores de qualidade. Os indicadores mostram que o algoritmo *GRASP-PR Rule Learning* tem um melhor desempenho do que o algoritmo *MOPSO* na maior parte dos conjuntos de exemplos. Entretanto, considerando o tempo de processamento, o algoritmo *GRASP-PR Rule Learning* gasta quase 200 horas para as 50 execuções de todos os conjuntos de exemplos, contra aproximadamente 100 horas gastas pelo algoritmo *MOPSO*.

5.4.3 Discussão

A ideia principal do algoritmo *GRASP-PR Rule Learning* (*GPR-RL*) foi gerar, num único passo (gera as regras e forma o conjunto ao mesmo tempo), um conjunto não-ordenado de regras de acordo com a sensitividade e a especificidade de modo a maximizar a *AUC*. A primeira análise comparou os valores de *AUC* obtidos. A comparação com o algoritmo Pareto Front Elite executado com todas as regras indicou semelhança entre os dois algoritmos, com exceção de um conjunto em que o PFE teve desempenho melhor. Uma comparação entre os algoritmos relacionados também indicou que o algoritmo *GPR-RL* gerou bons resultados. Também foi feita uma análise do parâmetro de número de tentativas para a construção de regras (parâmetro *maxLoop*). Com o aumento do total de tentativas verificou-se a melhoria da *AUC* em diversos conjuntos de dados.

Uma análise do número de regras indicou que o algoritmo metaheurístico *GPR-RL* gera um menor número de regras do que o PFE em diversos conjuntos, o que implica num menor tempo de processamento e de memória utilizada (em algumas bases de dados, o PFE chegou a demorar sete dias para um único *fold* até ser abortado por falta de memória). O custo de geração das regras é menor, principalmente, nos maiores conjuntos de exemplos. Já nos conjuntos menores, o algoritmo PFE teve uma menor geração do número de regras.

Através de uma metodologia de avaliação multiobjetiva, foram feitas comparações qualitativas dos conjuntos de aproximação obtidos do *GPR-RL* com os conjuntos obtidos pelo PFE com todas as regras. A comparação segundo o rankeamento pelo conceito de dominância não mostrou diferença entre os algoritmos, o que motivou prosseguir a comparação com a utilização dos indicadores hipervolume, epsilon aditivo e $R2$. Os resultados dos indicadores do *GPR-RL* com o menor valor para o parâmetro *maxLoop* indicaram proximidade ao algoritmo PFE em, pelo menos, 50% das comparações. O aumento do parâmetro *maxLoop*, que implica no aumento do número de regras na fronteira, resultou numa melhora segundo os indicadores. A melhora foi observada com o aumento do total de conjuntos de aproximação semelhantes à

fronteira do PFE.

Finalizou-se com as comparações do algoritmo *GRASP-PR Rule Learning* contra o algoritmo relacionado multiobjetivo *MOPSO*. A análise segundo rankeamento pelo conceito de dominância mostrou diferenças apenas em poucos conjuntos. Assim, seguiu-se utilizando os indicadores, comprovando que a maioria dos conjuntos de aproximação do *GPR-RL* é preferida aos conjuntos do *MOPSO*. Contudo, em termos de tempo computacional, o algoritmo *MOPSO* teve um desempenho melhor do que o *GPR-RL*.

As avaliações desta seção não consideraram apenas os valores de *AUC*, mas o número de regras gerado e a cobertura dos conjuntos de aproximação. Estas avaliações confirmaram que é possível, num único passo, criar um conjunto de regras não-ordenado com a utilização de uma metaheurística com os mesmos resultados que o algoritmo determinístico Pareto Front Elite.

6 Conclusões

O objetivo deste trabalho consiste na investigação de novas abordagens para a tarefa de classificação. Procurou-se melhorar o processo para gerar classificadores com bons critérios de qualidade.

A primeira investigação envolveu o equilíbrio dos critérios de precisão e generalidade dos classificadores. Foram propostas uma nova representação de hipótese baseada em intervalos e uma nova medida de avaliação que contemplasse a precisão e a generalidade. Para validar a proposta foi criado um algoritmo que combina as técnicas de *boosting* e estratégia evolucionária para a evolução dos novos modelos de classificação. Os resultados não foram satisfatórios, confirmando que avaliar a precisão e a generalidade não é uma tarefa trivial. A nova linguagem de representação por intervalos apresentou boa cobertura dos exemplos, porém, a estratégia proposta não pode evitar o *overfitting*.

A segunda abordagem investigou a hipótese de que um conjunto não-ordenado de regras formado segundo critérios biobjetivos poderia maximizar a *AUC*. Trabalhou-se com a suposição de que os critérios de sensitividade e especificidade são importantes para a formação de um conjunto de regras não-ordenadas que possa maximizar a *AUC*. Para validar esta hipótese, foi criado um algoritmo de seleção de regras (*rule subset selection*) chamado de Pareto Front Elite (PFE) (apresentado nos artigos (ISHIDA; POZO, 2007b) e (ISHIDA; POZO, 2007a)). O algoritmo constrói Fronteiras de Pareto usando os critérios de sensitividade e especificidade para selecionar as regras a partir de um grande conjunto de regras. A avaliação do PFE foi feita comparando-o com algoritmos conhecidos como ROCCER, C4.5 e CN2 e em 16 conjuntos de dados do UCI Machine Learning Repository (NEWMAN; BLAKE; MERZ, 1998) usualmente utilizados para *benchmark* para comparação de algoritmos. Na avaliação foram considerados três grupos de valores para os parâmetros mínimos de suporte e confiança. O primeiro grupo estabeleceu os maiores valores para os parâmetros gerando o menor número de regras; os resultados foram semelhantes a um dos melhores trabalhos relacionados, o ROCCER. O segundo grupo de experimentos considerou um valor menor para os parâmetros mínimos que gerou conjuntos com valores de *AUC* semelhantes, porém, com um maior número de regras. O terceiro

grupo de experimentos executou o PFE sem os parâmetros mínimos, sendo possível gerar a Fronteira de Pareto com os critérios de sensitividade e especificidade. Os resultados do terceiro foram semelhantes aos dos outros dois grupos de resultados, enfatizando que resultados semelhantes podem ser obtidos sem a necessidade do aumento do número de regras no conjunto final. Outra avaliação foi feita com os valores de suporte de todas as regras dos conjuntos, confirmando que os conjuntos obtidos possuem regras com valores de suporte acima de todos outros algoritmos. A qualidade das regras que compõem o conjunto resultado foi importante para a maximização da *AUC*, assim, validando a hipótese de que os critérios de sensitividade e especificidade são eficazes para a obtenção de conjuntos de regras não-ordenadas que maximizem a *AUC*.

Uma vez comprovada a hipótese da utilização dos critérios biobjetivos para a maximização da *AUC*, novas abordagens com a utilização de metaheurísticas foram propostas para criar, num único passo (gerar as regras e formar o conjunto ao mesmo tempo), os conjuntos de regras não-ordenadas para grandes bases de dados. Um dos algoritmos criados foi o *MOPSO* que utiliza a metaheurística nuvem de partículas (desenvolvido por (TORáCIO, 2008)). A contribuição neste trabalho foi a idéia para a criação do algoritmo. Os resultados em termos de *AUC* foram semelhantes aos resultados do PFE em metade das bases de dados, confirmando que é possível substituir a parte determinística e criar um algoritmo biobjetivo para a maximização da *AUC*. Os resultados foram satisfatórios, contudo, passíveis de serem melhorados.

Outra nova abordagem utilizou a estratégia *Greedy Randomized Adaptive Search Procedure* (*GRASP*) com *path-relinking* para a criação de regras e seleção da construção da Fronteira de Pareto. A ideia principal do algoritmo *GRASP-PR Rule Learning* (*GPR-RL*) (publicado em (ISHIDA et al., 2008)) foi gerar o conjunto de regras com as características mais apropriadas para a maximização da *AUC*. A estratégia *GRASP* gera os conjuntos não-ordenados e a estratégia de *path-relinking* melhora estes conjuntos. Foram realizados experimentos com três valores para o parâmetro *maxLoop*: 100, 200 e 300; estes resultados foram analisados e comparados com os trabalhos relacionados. Após estes experimentos, foram feitas comparações com o algoritmo PFE avaliando os valores de *AUC*, o número de regras gerado e os conjuntos de aproximação gerados. A última comparação do *GPR-RL* avaliou os conjuntos de aproximação com os resultados obtidos pelo *MOPSO*. Na comparação com os três grupos de resultados do algoritmo *GRASP-PR Rule Learning*, identificaram-se melhoras significativas com o aumento do número de tentativas de geração de regras. Em comparação com os trabalhos relacionados, o desempenho do algoritmo *GPR-RL* foi bastante satisfatório, gerando conjuntos melhores ou iguais na maioria das bases de dados. Na comparação dos valores de *AUC* do *GPR-RL* com o algoritmo Pareto Front Elite, não houve nenhuma diferença estatística em todas as bases de da-

dos. Contudo, o algoritmo metaheurístico gera um menor número de regras, o que implica num menor tempo de processamento e de memória. A metodologia de avaliação de desempenho multiobjetivo, que inclui a verificação do *dominance ranking* e a utilização de indicadores de qualidade, confirmou a qualidade dos conjuntos formados pelos critérios biobjetivos do algoritmo *GPR-RL*. Na maioria dos casos, os conjuntos de aproximação continham características semelhantes às Fronteiras de Pareto segundo os indicadores qualitativos de hipervolume, epsilon e $R2$. Todos os experimentos com o algoritmo metaheurístico *GPR-RL* indicam que é possível, na maioria dos casos, criar regras considerando os critérios de sensitividade e especificidade com bons valores de *AUC* e com a mesma cobertura das regras não-dominadas do algoritmo PFE (avaliadas através da análise dos indicadores de qualidade multiobjetivos).

A comparação multiobjetiva entre o *GRASP-PR Rule Learning* e o algoritmo multiobjetivo *MOPSO* indicou resultados favoráveis ao primeiro para a maioria das comparações (ISHIDA et al., 2009). Porém, o desempenho em relação ao tempo computacional do *MOPSO* foi superior ao *GPR-RL*. Mesmo com a geração de um menor número de regras comparado com o algoritmo determinístico PFE, o algoritmo *GPR-RL* deve passar por uma melhoria com a criação de uma versão paralela.

A tese contribuiu com alguns aspectos da classificação, que podem ser resumidos em:

- Apresentação de um novo algoritmo de classificação com o objetivo de atingir a precisão e a generalidade.
- Criação de uma extensão da representação de hipótese.
- Geração de uma nova medida de qualidade de precisão e generalidade.
- Utilização da ES para a indução de um conjunto de classificadores.
- Proposição de um algoritmo determinístico para a seleção de um conjunto de regras de acordo com critérios biobjetivos de sensitividade e especificidade com valores de *AUC* comparados aos melhores trabalhos relacionados.
- Identificação de que os critérios biobjetivos de sensitividade e especificidade maximizam os valores de *AUC* e, assim, contribuindo para a criação de algoritmos como o *MOPSO*.
- Idéia para a criação do algoritmo *MOPSO*.
- Criação de um algoritmo metaheurístico biobjetivo, capaz de gerar bons resultados de *AUC*, sendo que, de acordo com o método de avaliação multiobjetivo, os conjuntos de

aproximação obtidos são semelhantes à Fronteira de Pareto do algoritmo determinístico PFE segundo os indicadores qualitativos unários.

- Utilização do *GRASP* e *path-relinking* para a classificação.

6.1 Sugestões para Trabalhos Futuros

Equilibrar os critérios de precisão e generalidade continua a ser um desafio para a classificação. Novas ideias devem ser exploradas para que o equilíbrio seja obtido. Pelo fato da linguagem de representação por intervalos apresentar bons resultados na base de treinamento, porém com *overfitting*, novas abordagens podem ser sugeridas utilizando a mesma linguagem de representação. Aconselha-se a utilização de outros métodos num processo mais sistemático e com diminuição do número de parâmetros utilizados. O processo mais sistemático possibilitaria a validação da nova linguagem de representação e a diminuição dos parâmetros faria com que houvesse menor variação dos resultados.

Como proposta de melhoria do Pareto Front Elite, sugere-se a execução em novos conjuntos de dados para melhor avaliar a influência dos parâmetros de suporte e confiança na construção do classificador. O algoritmo pode ser utilizado para a avaliação de outros critérios multiobjetivos que possam ser combinados para se obter um conjunto de regras menor, porém, capaz de conseguir a mesma cobertura da Fronteira de Pareto. A estrutura do PFE favorece testes empíricos com outros critérios que possibilitem a maximização da *AUC*.

Apesar dos bons resultados de *AUC* e da boa cobertura obtidos pelo *GRASP-PR Rule Learning*, o mesmo ainda pode ser melhorado. Um estudo pode ser feito para verificar a influência da lista *RCL*, estratégia de procura local e da vizinhança. A lista *RCL* pode influenciar na construção dos resultados iniciais, experimentos podem ser realizados para verificar o impacto da variação do tamanho da lista *RCL* na qualidade dos conjuntos de aproximação gerados. Aqui, foi testada apenas uma versão da procura local, mas outras estratégias de busca local poderiam ser testadas para que o *path-relinking* não seja necessário. Diversos testes poderiam ser feitos para que uma estratégia de *path-relinking* mais eficiente possa diminuir a quantidade de operações da pós-operação. Uma versão paralela do algoritmo pode ser uma solução para lidar com grandes conjuntos de dados encontrados em problemas reais. Uma versão distribuída poderia ser feita com várias instâncias executadas com diferentes listas *RCL*, cada lista com quantidades diferentes de atributos. Nesta versão, existiria uma instância principal que agruparia as regras geradas por outras para a formação do conjunto não-ordenado de regras.

Para os dois algoritmos é sugerido como trabalho futuro um processo de pós-otimização

para reduzir o número de regras. A ideia é submeter o conjunto de regras obtido após a execução do algoritmo em questão e reduzir o número de regras do conjunto sem perder a cobertura da Fronteira de Pareto.

Referências Bibliográficas

AGRAWAL, R.; IMIELINSKI, T.; SWAMI, A. Mining association rules between sets of items in large databases. In: *19 ACM SIGMOD Conf. on the Management of Data, Washington,DC*. [S.l.: s.n.], 1993.

AGRAWAL, R.; SHAFER, J. C. Parallel mining of association rules. *IEEE Transactions on Knowledge and Data Engineering*, v. 8, n. 6, p. 962–969, dez. 1996.

AGRAWAL, R.; SRIKANT, R. Fast algorithms for mining association rules. In: BOCCA, J. B.; JARKE, M.; ZANIOLO, C. (Ed.). *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*. [S.l.]: Morgan Kaufmann, 1994. p. 487–499. ISBN 1-55860-153-8.

AIEX, R. M. et al. *GRASP with path-relinking for the three-index assignment problem*. [S.l.], 2000.

BÄCK, T.; FOGEL, D. B.; MICHALEWICZ, Z. *Evolutionary computation 2 : advanced algorithms and operators*. [S.l.]: Institute of Physics Publishing, 2000.

BÄCK, T.; HOFFMEISTER, F. Extended selection mechanisms in genetic algorithms. In: BELEW, R. K.; BOOKER, L. B. (Ed.). *Proceedings of the Fourth International Conference on Genetic Algorithms (ICGA'91)*. San Mateo, California: Morgan Kaufmann Publishers, 1991. p. 92–99.

BÄCK, T.; HOFFMEISTER, F.; SCHWEFEL, H.-P. A survey of evolution strategies. In: BELEW, R. K.; BOOKER, L. B. (Ed.). *Proceedings of the Fourth International Conference on Genetic Algorithms (ICGA'91)*. San Mateo, California: Morgan Kaufmann Publishers, 1991. p. 2–9.

BACK, T.; RUDOLPH, G.; SCHWEFEL, H. *Evolutionary programming and evolution strategies: Similarities and differences*. Disponível em: <citeseer.ist.psu.edu/3317.html>.

BÄCK, T.; SCHWEFEL, H.-P. An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, v. 1, n. 1, p. 1–23, 1993.

BäCK, T.; SCHWEFEL, H.-P. An overview of evolutionary algorithms for parameter optimization. *Evol. Comput.*, MIT Press, Cambridge, MA, USA, v. 1, n. 1, p. 1–23, 1993. ISSN 1063-6560.

BARRETO, J. M. *Inteligencia Artificial no Limiar do Seculo XXI: abordagem hibrida, simbolica, conexionista e evolucionaria*. 2. ed. Florianopolis, SC: [s.n.], 1999.

BATISTA, G. et al. A comparison of methods for rule subset selection applied to associative classification. *Inteligencia Artificial. Revista Iberoamericana de IA*, n. 32, p. 29–35, 2006.

BAUER, E.; KOHAVI, R. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, v. 36, n. 1,2, 1999.

BLEULER, S. et al. PISA — a platform and programming language independent interface for search algorithms. In: FONSECA, C. M. et al. (Ed.). *Evolutionary Multi-Criterion Optimization (EMO 2003)*. Berlin: Springer, 2003. (Lecture Notes in Computer Science), p. 494 – 508.

BRADLEY, A. P. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, v. 30, n. 7, p. 1145–1159, jul. 1997.

BREIMAN, L. Bagging predictors. *Machine Learning*, v. 24, n. 2, p. 123–140, 1996.

BREIMAN, L. Heuristics of instability and stabilization in model selection. *Annals of Statistics*, v. 24, p. 2350–2383, 1996.

CANUTO, S. A.; RESENDE, M. G. C.; RIBEIRO, C. C. Local search with perturbations for the prize-collecting steiner tree problem in graphs. *NETWORKS: Networks: An International Journal*, v. 38, 2001.

CHAN, P. K.; STOLFO, S. J. Experiments in multistrategy learning by meta-learning. In: *Proceedings of the second international conference on information and knowledge management*. Washington, DC: [s.n.], 1993. p. 314–323.

CLARK, P.; NIBLETT, T. The CN2 induction algorithm. *Machine Learning*, Kluwer Academic Publishers, Boston, v. 3, p. 261–283, 1989.

COELLO, C.; LECHUGA, M. Mopso: A proposal for multiple objective particle swarm optimization. In: *IEEE Wold Congress on Computational Intelligence*. [S.l.]: IEEE Press, 2002. p. 1051–1056.

COHEN, W. W. Fast effective rule induction. In: *ICML*. [S.l.: s.n.], 1995. p. 115–123.

COHEN, W. W.; SINGER, Y. A simple, fast, and effective rule learner. In: *Proceedings of the 6th National Conference on Artificial Intelligence (AAAI-99); Proceedings of the 11th Conference on Innovative Applications of Artificial Intelligence*. Menlo Park, Cal.: AAAI/MIT Press, 1999. p. 335–342.

CONOVER, W. J. *Practical nonparametric statistics*. [S.l.]: Wiley, 1971.

CORNUEJOLS, A.; MICLET, L. *Apprentissage artificiel - Concepts et algorithmes*. Paris: Eyrolles, 2003.

DARWIN, C. *On the origin of species by means of natural selection or the preservation of favored races in the struggle for life*. [S.l.]: Murray: London, 1859.

DEB, K. *Multi-Objective Optimization using Evolutionary Algorithms*. Chichester, UK: John Wiley & Sons, 2001. ISBN 0-471-87339-X.

DIANATI, M.; SONG, I.; TREIBER, M. An introduction to genetic algorithms and evolution strategies. 2002.

DRUCKER, H. Improving regressors using boosting techniques. In: *Proc. 14th International Conference on Machine Learning*. [S.l.]: Morgan Kaufmann, 1997. p. 107–115.

DRUCKER, H. et al. Boosting and other ensemble methods. *Neural Computation*, v. 6, n. 6, p. 1289–1301, 1994.

EBERHART, R. C.; SHI, Y. Evolving artificial neural networks. In: *Proceeding 1998 Int. Conf. on Neural Networks and Brain*. Beijing: P.R.C., 1998. p. PL5–PL13.

EHRGOTT, M. Approximation algorithms for combinatorial multicriteria optimization problems. *International Transactions in Operational Research*, v. 7, p. 5–31, 2000.

FAWCETT, T. Using rule sets to maximize ROC performance. In: CERCONE, N.; LIN, T. Y.; WU, X. (Ed.). *ICDM*. [S.l.]: IEEE Computer Society, 2001. p. 131–138. ISBN 0-7695-1119-8.

FAWCETT, T. *ROC Graphs: Notes and Practical Considerations for Data Mining Researchers*. [S.l.], jan. 17 2003. 27 p.

FEO, T. A.; RESENDE, M. G. C. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, v. 6, p. 109–133, 1995.

FERRI, C.; FLACH, P.; HERNANDEZ-ORALLO, J. Learning decision trees using the area under the ROC curve. Morgan Kaufmann, p. 139–146, jul. 2002.

FERRI, C.; FLACH, P.; HERNANDEZ-ORALLO, J. Delegating classifiers. In: GREINEER, R.; SCHUURMANS, D. (Ed.). *Proceedings of the 21st International Conference on Machine Learning (ICML 2004)*. [S.l.]: ACM, 2004. ISBN 1-58113-838-5.

FOGEL, L. J.; OWENS, A. J.; WALSH, M. J. *Artificial Intelligence through simulated Evolution*. [S.l.]: John Wiley & Sons, 1966.

FRANK, E.; WITTEN, I. H. Generating accurate rule sets without global optimization. In: *Proc. 15th International Conf. on Machine Learning*. [S.l.]: Morgan Kaufmann, San Francisco, CA, 1998. p. 144–151.

FREITAS, A.; LAVINGTON, S. *Mining Very Large Databases with Parallel Processing*. Boston: Kluwer Academic Publishers, 1998. ISBN 0-7923-8048-7. Disponível em: <http://www.cs.kent.ac.uk/pubs/1998/1395>.

FREUND, Y.; SCHAPIRE, R. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, v. 14, n. 5, p. 771–780, 1999.

FREUND, Y.; SCHAPIRE, R. E. Experiments with a new boosting algorithm. In: *ICML*. [S.l.: s.n.], 1996. p. 148–156.

FREUND, Y.; SCHAPIRE, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. *JCSS: Journal of Computer and System Sciences*, v. 55, 1997.

FRIEDMAN, J.; HASTIE, T.; TIBSHIRANI, R. *Additive Logistic Regression: a Statistical View of Boosting*. [S.l.], jul. 24 1998.

GAMBERGER, D.; LAVRAC, N. Confirmation rule sets. In: ZIGHED, D. A.; KOMOROWSKI, H. J.; ZYTKOW, J. M. (Ed.). *PKDD*. [S.l.]: Springer, 2000. (Lecture Notes in Computer Science, v. 1910), p. 34–43. ISBN 3-540-41066-X.

GLOVER, F. R.s. barr and r.v. helgason and jeffrey l. kennington (eds). interfaces in computer science and operations research. In: ____. [S.l.]: kluwer, 1996. cap. Tabu search and adaptive memory programming - advances, applications and challenges, p. 1–75.

GOSSET, W. S. The probable error of a mean. *Biometrik*, v. 6, n. 1, p. 1–25, 1908. Published under the pseudomym Student.

GOSSET, W. S.; PEARSON, E. S.; WISHART, J. Studentś collected papers. *Cambridge: University Press*, p. 11–34, 1942.

HAND, D. *Discrimination and Classification*. Chichester, UK: Wiley, 1981.

HANSEN, M. P.; JASZKIEWICZ, A. *Evaluating the quality of approximations to the non-dominated set*. [S.l.], mar. 1998.

HARTMANN, D. *Optimierung balkenartiger Zylinderschalen aus Stahlbeton mit elastischem und plastischem Werkstoffverhalten*. Tese (Doutorado) — University of Dortmund, Dortmund, 1974.

HOLDEN, N.; FREITAS, A. A hybrid particle swarm/ant colony algorithm for the classification of hierarchical biological data. In: *Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005.* [S.l.: s.n.], 2005. p. 100 – 107.

HOLLAND, J. H. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. [S.l.]: MIT Press, 1992.

ISHIDA, C. et al. Multiobjective optimization and rule learning: Subselection algorithm or meta-heuristic algorithm? In: . [S.l.: s.n.], 2009. p. 47–70.

ISHIDA, C. Y. et al. Exploring multi-objective PSO and GRASP-PR for rule induction. In: *Eighth European Conference on Evolutionary Computation in Combinatorial Optimisation*. [S.l.]: Springer Berlin / Heidelberg, 2008.

ISHIDA, C. Y.; POZO, A. T. R. Optimization of the auc criterion for rule subset selection. In: 7TH. INTERNATIONAL CONFERENCE ON INTELLIGENT SYSTEMS DESIGN AND APPLICATION. *Intelligent Systems Design and Applications*. New York, NY, USA: IEEE Computer Society, 2007.

ISHIDA, C. Y.; POZO, A. T. R. Pareto Front Elite. In: ENIA VI ENCONTRO NACIONAL DE INTELIGENCIA ARTIFICIAL. *XXVII Congresso SBC 2007*. [S.l.]: Porto Alegre: SBC, 2007.

ISHIDA C. Y.; CAVALHEIRO, A. F. *BUSINESS INTELLIGENCE Expandindo os Benefícios do ERP.* [S.l.], 1999.

JIN, Y. *Multi-Objective Machine Learning*. Boston, MA: Springer, Berlin, 2006.

JOVANOSKI, V.; LAVRAC, N. Classification rule learning with APRIORI-C. In: BRAZDIL, P.; JORGE, A. (Ed.). *EPIA*. [S.l.]: Springer, 2001. (Lecture Notes in Computer Science, v. 2258), p. 44–51. ISBN 3-540-43030-X.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: *IEEE International Conference on Neural Networks.* [S.l.]: IEEE Press, 1995. p. 1492–1948.

KNOWLES, J.; THIELE, L.; ZITZLER, E. *A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers*. [S.l.], jul. 2005.

KNOWLES, J.; THIELE, L.; ZITZLER, E. *A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers*. [S.l.], fev. 2006. Revised version.

KOHAVI, R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: *IJCAI*. [S.l.: s.n.], 1995. p. 1137–1145.

KOZA, J. R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: MIT Press, 1992. ISBN 0-262-11170-5.

KUSIAK, A. Evolutionary Computation and Data Mining. In: GOPALAKRISHNAN, B.; GUNASEKARAN, A. (Ed.). *Proceedings of the SPIE Conference on Intelligent Systems and Advances Manufacturing*. Boston, Massachussetts: SPIE, 2000. p. 1–10.

LAGUNA, M.; MARTI, R. Grasp and path relinking for 2-layer straight line crossing minimization. *INFORMS J. on Computing*, INFORMS, Institute for Operations Research and the Management Sciences (INFORMS), Linthicum, Maryland, USA, v. 11, n. 1, p. 44–52, 1999. ISSN 1526-5528.

LAVRAC, N.; FLACH, P.; ZUPAN, B. Rule evaluation measures: A unifying view. In: DZEROSKI, S.; FLACH, P. (Ed.). *Ninth International Workshop on Inductive Logic Programming (ILP'99)*. Springer-Verlag, 1999. p. 174–185. ISBN 3-54066-109-3. Disponível em: <http://www.cs.bris.ac.uk/Publications/Papers/1000358.pdf>.

LI, M.; VITáNYI, P. *An introduction to Kolmogorov complexity and its applications (2nd ed.)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1997. ISBN 0-387-94868-6.

LI, W.; HAN, J.; PEI, J. CMAR: Accurate and efficient classification based on multiple class-association rules. In: CERCONE, N.; LIN, T. Y.; WU, X. (Ed.). *ICDM*. [S.l.]: IEEE Computer Society, 2001. p. 369–376. ISBN 0-7695-1119-8.

LIU, B.; HSU, W.; MA, Y. Integrating classification and association rule mining. In: *Knowledge Discovery and Data Mining*. [S.l.: s.n.], 1998. p. 80–86.

MITCHELL, T. M. *Machine Learning*. [S.l.]: McGraw-Hill, New York, 1997.

MITCHELL, T. M. Machine learning and data mining. *Commun. ACM*, v. 42, n. 11, p. 30–36, 1999. Disponível em: <http://doi.acm.org/10.1145/319382.319388>.

MOSTAGHIM, S.; TEICH, J. Strategies for Finding Good Local Guides in Multi-objective Particle Swarm Optimization (MOPSO). In: *2003 IEEE Swarm Intelligence Symposium Proceedings*. Indianapolis, Indiana, USA: IEEE Service Center, 2003. p. 26–33.

NEWMAN, D.; BLAKE, C.; MERZ, C. *UCI Repository of machine learning databases*. 1998. Último acesso: janeiro de 2006. Disponível em: <http://www.ics.uci.edu/∼mlearn/MLRepository.html>.

PRATI, R. C. *Novas abordagens em aprendizado de máquina para a geração de regras, classes desbalanceadas e ordenação de casos*. Tese (Doutorado) — Instituto de Ciências Matemáticas e de Computação (ICMC), 2006.

PRATI, R. C.; FLACH, P. A. ROCCER: An algorithm for rule learning based on ROC analysis. In: KAELBLING, L. P.; SAFFIOTTI, A. (Ed.). *IJCAI*. Professional Book Center, 2005. p. 823–828. ISBN 0938075934. Disponível em: <http://www.ijcai.org/papers/0621.pdf>.

PRODROMIDIS, A.; CHAN, P. *Meta-learning in distributed data mining systems: Issues and Approaches*. 2000.

PROVOST, F.; FAWCETT, T. Robust classification for imprecise environments. *Machine Learning*, v. 42, n. 3, p. 203, 2001.

PROVOST, F.; FAWCETT, T.; KOHAVI, R. The case against accuracy estimation for comparing induction algorithms. In: *Proceedings 15th International Conference on Machine Learning*. [S.l.]: Morgan Kaufmann, San Francisco, CA, 1998. p. 445–453.

PROVOST, F. J.; FAWCETT, T. Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In: *KDD*. [S.l.: s.n.], 1997. p. 43–48.

QUINLAN, J. *C4. 5: Programs for Machine Learning*. [S.l.]: Morgan Kaufmann, 1993.

QUINLAN, J. R. Generating production rules from decision trees. In: *International Joint Conference on Artificial Intelligence*. Milan: Morgan Kaufmann, 1987. p. 304–307.

RAKOTOMAMONJY, A. Optimizing area under roc curve with SVMs. In: HERNÁNDEZ-ORALLO, J. et al. (Ed.). *ROCAI*. [S.l.: s.n.], 2004. p. 71–80.

RECHENBERG, I. Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution. *Fromman-Holzboog*, 1973.

RESENDE, M.; RIBEIRO, C. Greedy randomized adaptive search procedures. In: GLOVER, F.; KOCHENBERGER, G. (Ed.). *Handbook of Metaheuristics*. [S.l.]: Kluwer Academic Publishers, 2002. p. 219–249.

RESENDE, M. G. C.; RIBEIRO, C. C. A GRASP with path-relinking for private virtual circuit routing. *NETWORKS: Networks: An International Journal*, v. 41, 2003.

RIBEIRO, C. C.; UCHOA, E.; WERNECK, R. F. F. A hybrid GRASP with perturbations for the steiner problem in graphs. *INFORMS Journal on Computing*, v. 14, n. 3, p. 228–246, 2002.

RUDIN, C.; SCHAPIRE, R. E.; DAUBECHIES, I. Boosting based on a smooth margin. In: SHAWE-TAYLOR, J.; SINGER, Y. (Ed.). *COLT*. [S.l.]: Springer, 2004. (Lecture Notes in Computer Science, v. 3120), p. 502–517. ISBN 3-540-22282-0.

SCHAPIRE. The strength of weak learnability. *MACHLEARN: Machine Learning*, v. 5, 1990.

SCHAPIRE, R. E. *The Boosting Approach to Machine Learning: An Overview*. dez. 2002.

SCHWEFEL, H. Numerische optimierung von computermodellen mittels der evolutionsstrategie. *Birkhäuser*, 1977.

SCHWEFEL, H. P. *Evolutionsstrategie und Numerische Optimierung*. Dissertação (Mestrado) — Technische Universitat Berlin, 1975.

SCHWEFEL, H.-P. *Numerical optimization of Computer models*. Chichester: John Wiley & Sons, Ltd., 1981.

SCHWEFEL, H.-P. Collective phenomena in evolutionary systems. In: CHECKLAND, P.; KISS, I. (Ed.). *Problems of Constancy and Change –The Complementarity of Systems Approaches toComplexity, Proc. 31st Annual Meeting*. Budapest: Int'l Soc. for General System Research, 1987. v. 2, p. 1025–1033.

SEBAG, M.; AZÉ, J.; LUCAS, N. Impact studies and sensitivity analysis in medical data mining with ROC-based genetic learning. In: *ICDM*. IEEE Computer Society, 2003. p. 637–640. ISBN 0-7695-1978-4. Disponível em: <http://csdl.computer.org/comp/proceedings/icdm/2003/1978/00/19780637abs.htm>.

SEBAG, M.; AZE, J.; LUCAS, N. ROC-based evolutionary learning: Application to medical data mining. In: *International Conference on Artificial Evolution, Evolution Artificielle, LNCS*. [S.l.: s.n.], 2003. v. 6.

SIGLETOS, G. et al. Meta-learning beyond classification: A framework for information extraction from the web. In: *Proceedings of the Workshop on Adaptive Text Extraction and Mining, ECML*. [S.l.: s.n.], 2003.

SOUSA, T.; SILVA, A.; NEVES, A. A particle swarm data miner. In: MOURA-PIRES, F.; ABREU, S. (Ed.). *EPIA*. [S.l.]: Springer, 2003. (Lecture Notes in Computer Science, v. 2902), p. 43–53. ISBN 3-540-20589-6.

SOUSA, T.; SILVA, A.; NEVES, A. Particle swarm-based data mining algorithms for classification tasks. *Parallel Computing*, v. 30, n. 5–6, p. 767–783, 2004. ISSN 0167-8191.

SPACKMAN, K. A. Signal detection theory: Valuable tools for evaluating inductive learning. In: SEGRE, A. M. (Ed.). *ML*. [S.l.]: Morgan Kaufmann, 1989. p. 160–163. ISBN 1-55860-036-1.

TORáCIO, A. de A. P. G. *Aprendizado de Regras de Classificação com Otimização por Nuvem de Partículas Multiobjetivo*. Dissertação (Mestrado) — Universidade Federal do Paraná, 2008.

TURBAN, E.; RAINER, R. K.; POTTER, J. E. *Administração de Tecnologia da Informação: Teoria e Prática*. 3. ed. [S.l.: s.n.], 2005. 640 p.

VESTERSTROEM, J. S.; RIGET, J. *Particle swarms: extensions for improved local, multi-modal, and dynamic search in numerical optimization.* Dissertação (Mestrado) — Department of Computer Science, University of Aarhus, 2002.

VILALTA, R.; DRISSI, Y. A perspective view and survey of meta-learning. *Artif. Intell. Rev*, v. 18, n. 2, p. 77–95, 2002.

VRIESMANN, L. M. *Boosting e Estrategias Evolucionarias na Tarefa de Regressao para a Mineracao de Dados Temporal*. Dissertação (Mestrado) — Universidade Federal do Parana, 2005.

WEISS, S.; KULIKOWSKI, C. *Computer Systems that Learn*. San Mateo, CA: Morgan Kaufmann, 1991.

WILCOXON, F. Individual comparisons by ranking methods. *Biometrics Bulletin*, International Biometric Society, v. 1, n. 6, p. 80–83, dec 1945. ISSN 0099-4987.

WITTEN, I. H.; FRANK, E. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. [S.l.]: Morgan Kaufmann, 1999. Paperback. ISBN 1558605525.

WONG, M. L.; LEUNG, K. S. *Data Mining Using Grammar Based Genetic Programming and Applications*. [S.l.]: Kluwer Academic Publishers, 2000. (Genetic Programming, v. 3).

YIN, X.; HAN, J. Cpar: Classification based on predictive association rules. *Proceedings SIM International Conference on Data Mining (SDM'03)*, p. 331–335, 2003.

ZHANG, J. et al. Learning rules from highly unbalanced data sets. In: *ICDM*. [S.l.]: IEEE Computer Society, 2004. p. 571–574. ISBN 0-7695-2142-8.

ZITZLER, E.; THIELE, L. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, v. 3, n. 4, p. 257–271, 1999.

ZITZLER, E. et al. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE-EC*, v. 7, p. 117–132, abr. 2003.

APÊNDICE A – Exemplo de comparação de conjuntos de aproximação

Na seção 3.2.5 foram apresentados dois algoritmos fictícios como exemplo da comparação de conjuntos de aproximação de algoritmos multiobjetivos. Os valores dos conjuntos de aproximação do algoritmo 1 para o exemplo fictício de comparação de fronteira estão na Tabela A.1. Os conjuntos do algoritmo 2 estão na Tabela A.2.

Tabela A.1: Conjuntos de aproximação do algoritmo 1 para o exemplo de comparação de fronteiras. Cada elemento de um conjunto corresponde a uma tupla com os valores de dois critérios de avaliação.

Conjunto 1a		Conjunto 1b		Conjunto 1c	
0,835	0,155	0,835	0,065	0,885	0,085
0,805	0,255	0,785	0,245	0,825	0,265
0,735	0,405	0,735	0,415	0,745	0,475
0,662	0,556	0,675	0,545	0,695	0,565
0,582	0,655	0,595	0,625	0,615	0,660
0,455	0,705	0,525	0,655	0,525	0,730
0,345	0,755	0,385	0,735	0,405	0,770
0,255	0,805	0,175	0,785	0,295	0,825
0,125	0,855	0,115	0,805	0,195	0,875

Tabela A.2: Conjuntos de aproximação do algoritmo 2 para o exemplo de comparação de fronteiras. Cada elemento de um conjunto corresponde a uma tupla com os valores de dois critérios de avaliação.

Conjunto 2a		Conjunto 2b		Conjunto 2c	
0,900	0,100	0,880	0,110	0,910	0,110
0,850	0,300	0,830	0,290	0,850	0,290
0,700	0,600	0,700	0,590	0,720	0,590
0,600	0,700	0,640	0,670	0,640	0,670
0,500	0,750	0,430	0,780	0,430	0,780
0,390	0,800	0,350	0,820	0,320	0,850
0,170	0,900	0,200	0,880	0,220	0,900

APÊNDICE B – Resultado das comparações estatísticas dos valores de *AUC* médios

O algoritmo *GRASP-PR Rule Learning* com $maxLoop = 100$ foi comparado com o Pareto Front Elite com os parâmetros mínimos e os *p-value*'s do teste de Wilcoxon estão na Tabela B.1.

O *GRASP-PR Rule Learning* com $maxLoop = 100$ também foi comparado com outros resultados do mesmo algoritmo e os *p-value*'s do teste de Wilcoxon estão na Tabela B.2.

Os resultados do *GRASP-PR Rule Learning* com $maxLoop = 200$ e 300 foram comparados e os *p-value*'s do teste de Wilcoxon estão na Tabela B.3. As células em cinza escuro indicam bases de dados que $maxLoop = 300$ é melhor do que $maxLoop = 200$.

GRASP-PR Rule Learning foi comparado com o algoritmo Pareto Front Elite e os *p-value*'s do teste de Wilcoxon estão na Tabela B.4.

Tabela B.1: P-value's do teste de Wilcoxon para a hipótese zero H0: AUC médio do algoritmo PFE da coluna e AUC médio do *GRASP-PR Rule Learning* com $maxLoop = 100$ ser diferente.

#	**PFE(5.1)(5.2)**	**PFE(5.3)(5.4)**	**PFE(5.5)(5.6)**
1	0,0960	0,3591	0,0960
2	0,0960	0,8651	0,0960
3	0,6104	0,7087	0,2923
4	0,8651	0,6588	(-)
5	0,1637	0,2623	0,0960
6	0,0960	0,8651	0,4756
7	0,0960	0,4756	0,9729
8	0,2620	0,2085	0,2620
9	0,1852	0,0960	(-)
10	0,0960	(-)	(-)
11	0,2923	(-)	(-)
12	0,2341	0,3587	0,3242
13	0,5636	0,0960	0,0960
14	0,3958	1,0000	0,3246
16	0,5187	0,0960	(-)

Tabela B.2: Os valores de *p-value*'s do teste de Wilcoxon para a hipótese zero H0: algoritmo da coluna e *GRASP-PR Rule Learning* com $maxLoop = 100$ serem diferentes.

#	*GPR-RL* 200	*GPR-RL* 300
1	0.0004	2.01E-05
2	0.6343	0.7512
3	0.7226	0.6294
4	0.0093	0.0097
5	0.0015	1.24E-07
6	0.412	0.6003
7	0.893	0.0080
8	1.70E-06	9.31E-10
9	1.76E-11	2.20E-16
10	2.20E-16	2.20E-16
11	2.86E-15	2.20E-16
12	2.20E-16	4.32E-05
13	2.20E-16	2.20E-16
14	0.2059	0.6616
15	8.49E-14	3.19E-15
16	2.20E-16	2.20E-16

Tabela B.3: Resultado da comparação estatística do algoritmo *GRASP-PR Rule Learning* com os valores 200 e 300 para o parâmetro *maxLoop*. Os *p-value*'s do teste de Wilcoxon para a hipótese zero H0: resultados diferentes.

#	p-value
1	0,6817
2	0,9176
3	0,7226
4	0,9890
5	0,0065
6	0,4159
7	0,0003
8	0,0153
9	0,0066
10	6,68E-07
11	0,0004
12	0,0583
13	2,20E-16
14	0,0048
15	0,0019
16	5,24E-12

Tabela B.4: Resultado da comparação estatística dos algoritmos Pareto Front Elite sem parâmetros com o algoritmo *GRASP-PR Rule Learning*. Os *p-value*'s do teste de Wilcoxon para a hipótese zero H0: algoritmo da coluna e algoritmo PFE sem parâmetros serem diferentes.

#	*GPR-RL* 100	*GPR-RL* 200	*GPR-RL* 300
1	0.09602	0.1264	0.09602
2	0.09602	0.096	0.096
3	0.2923	0.2923	0.3246
5	0.09602	0.1104	0.096
6	0.4756	0.2087	0.144
7	0.9729	0.9188	0.2088
8	0.262	1	0.5181
12	0.3242	0.1633	0.09585
13	0.09602	0.09602	0.09602
14	0.3246	0.1441	0.1104

APÊNDICE C – Comparações dos algoritmos

Nesta seção são apresentados resumos de várias comparações entre algoritmos em forma de Tabelas e Figuras. As Tabelas de comparação possuem valores já apresentados durante o texto, porém, com valores organizados e destaque para os valores que são estatisticamente diferentes do algoritmo utilizado como base de comparação.

C.1 Resumo para o PFE com os Maiores Parâmetros Mínimos

A seguir são apresentados alguns resumos dos resultados para a análise do desempenho do algoritmo Pareto Front Elite executado com os valores mais altos para os parâmetros mínimos ($MinSupport_1$ para suporte mínimo (equação 5.1) e $MinConfidence_1$ para confiança mínima (equação 5.2)). São considerados os valores de *AUC* médios do PFE apresentados na segunda coluna da Tabela 5.3 e os resultados dos trabalhos relacionados (Tabela 5.2).

A Tabela C.1 compara cada valor de *AUC* médio obtido pelo PFE com os trabalhos relacionados, destacando os valores estatisticamente diferentes do PFE. A primeira coluna contém o número do conjunto de exemplos. A segunda coluna possui *AUC* médio e o desvio-padrão (entre parênteses) para cada conjunto de exemplo (resultados primeiramente apresentados na segunda coluna da Tabela 5.3). As demais colunas contêm os resultados dos trabalhos relacionados apresentados na Tabela 5.2. As células com as cores cinza claro ou escuro mostram, respectivamente, valores que são estatisticamente melhores ou piores do que o PFE.

A Tabela C.2 é um resumo por base de dados dos valores estatisticamente diferentes destacados na Tabela C.1. A Tabela C.2 contém o número de valores de *AUC* do algoritmo Pareto Front Elite que é superior (segunda coluna), similar (terceira coluna) ou inferior (quarta coluna) aos valores de *AUC* dos trabalhos relacionados para cada base de dados. Já a Tabela C.3 contém um resumo do número de valores diferentes separados pelo trabalho relacionado. O número total de valores de *AUC* médios do PFE que é superior ao algoritmo da coluna correspondente está na segunda linha da Tabela, os totais de valores similares ao PFE estão na

Tabela C.1: Comparação dos *AUC* Médio do algoritmo Pareto Front Elite para os maiores valores de parâmetros mínimos (equações 5.1 e 5.2) (apresentados na segunda coluna da Tabela 5.3) com os *AUC* médio dos trabalhos relacionados (apresentados na Tabela 5.2). A segunda coluna (resultados do PFE) é a base para a comparação com outros algoritmos utilizando o teste T. Células em cinza escuro indicam valores estatisticamente inferiores dos resultados do PFE, células em cinza claro indicam valores estatisticamente superiores e as células sem cores indicam valores similares.

#	**PFE (5.1)(5.2)**	**ROCCER**	**C4.5**	**C4.5NP**	**CN2**	**CN2OR**	**Ripper**	**Slipper**
1	99,33	98,63	97,76	98,39	99,26	99,13	98,72	99,24
2	63,83	65,30	62,14	57,44	62,74	62,21	69,10	59,84
3	91,55	90,31	50	90,06	90,17	85,15	61,86	74,78
4	69,07	61,83	50	68,68	53,22	42,78	45,28	52,35
5	73,51	72,08	71,43	67,71	75,25	70,90	64,02	71,32
6	77,07	79,45	50	81,50	73,74	79,64	49,75	50
7	66,75	66,41	55,84	64,33	59,83	59,28	57,45	50,40
8	88,56	85,78	84,81	81,11	83,61	82,25	84,89	84,03
9	95,85	94,18	86,09	90,91	96,23	92,18	92,06	93,95
10	97,71	99,35	99,85	99,86	99,85	99,91	99,85	99,91
11	95,46	96,08	95,49	99,33	99,34	99,44	97,27	98,82
12	97,41	98,40	87,85	97,50	99,14	98,43	94,95	99,12
13	98,82	97,85	99,42	99,74	100	99,99	99,43	94,40
14	70,41	70,68	72,07	72,6	70,96	71,97	68,07	70,02
16	93,72	96,42	94,76	96,99	97,38	96,49	95,01	93,99
Med.	85,27	84,85	77,17	84,41	84,05	82,65	78,51	79,48

Tabela C.2: Número de valores de *AUC* do algoritmo Pareto Front Elite que é superior (segunda coluna), similar (terceira coluna) ou inferior (quarta coluna) aos valores de *AUC* dos trabalhos relacionados. São considerados os resultados do PFE com os maiores valores de parâmetros mínimos (equações 5.1 e 5.2) apresentados na segunda coluna da Tabela 5.3 e na segunda coluna da Tabela C.1 e os resultados dos trabalhos relacionados apresentados na Tabela 5.2 e Tabela C.1. A contagem é feita por base de dados, cujo número está na primeira coluna, e os totais gerais estão na última linha.

#	Superior	Similar	Inferior
1	2	5	0
2	0	7	0
3	3	4	0
4	4	3	0
5	2	5	0
6	3	4	0
7	5	2	0
8	2	5	0
9	3	4	0
10	0	0	7
11	0	2	5
12	1	6	0
13	2	0	5
14	0	7	0
16	0	3	4
Total	**27**	**57**	**21**

terceira linha e os totais de valores inferiores estão na última linha. A Figura C.1 traz os totais da Tabela C.3 em forma de gráfico.

C.2 Resumo do PFE com os Parâmetros Intermediários

A seguir são apresentados alguns resumos dos resultados para a análise do desempenho do algoritmo Pareto Front Elite executado com os valores intermediários para os parâmetros mínimos ($MinSupport_2$ para suporte mínimo (equação 5.3) e $MinConfidence_2$ para confiança mínima (equação 5.4)). São considerados os valores de *AUC* médios do PFE apresentados na terceira coluna da Tabela 5.3 e os resultados dos trabalhos relacionados (Tabela 5.2).

A Tabela C.4 compara cada valor de *AUC* médio obtido pelo PFE com os trabalhos relacionados, destacando os valores estatisticamente diferentes do PFE. A primeira coluna contém o número do conjunto de exemplos. A segunda coluna possui *AUC* médio e o desvio-padrão (entre parênteses) para cada conjunto de exemplo (resultados primeiramente apresentados na segunda coluna da Tabela 5.3). As demais colunas contêm os resultados dos trabalhos relacionados apresentados na Tabela 5.2. As células com as cores cinza claro ou es-

Tabela C.3: Total de bases de dados que o algoritmo Pareto Front Elite é superior (segunda linha), inferior (quarta linha) ou que não existe diferença (terceira linha) comparado com os resultados dos trabalhos relacionados. São considerados os resultados do PFE com os maiores valores de parâmetros mínimos (equações 5.1 e 5.2) apresentados na segunda coluna da Tabela 5.3 e na segunda coluna da Tabela C.1 e os resultados dos trabalhos relacionados apresentados na Tabela 5.2 e Tabela C.1. A contagem é feita com base nas comparações da Tabela C.1.

Comparação	**ROCCER**	**C4.5**	**C4.5NP**	**CN2**	**CN2OR**	**Ripper**	**Slipper**	**Total(%)**
Superior	1	7	4	1	3	6	5	27(0,26)
Similar	12	6	7	10	8	6	8	57(0,54)
Inferior	2	2	4	4	4	3	2	21(0,20)

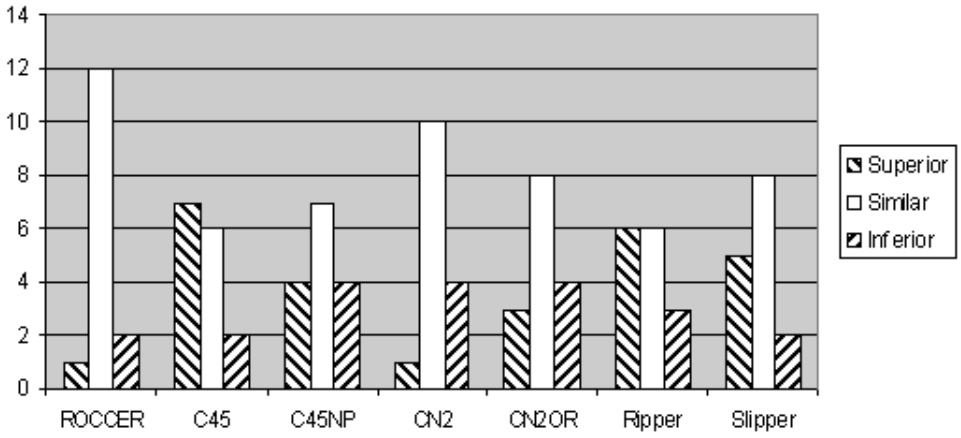


Figura C.1: Número de bases de dados que o PFE com os maiores valores de parâmetros mínimos (equações 5.1 e 5.2) é superior, similar ou inferior ao trabalho relacionado. Figura baseada nos dados da Tabela C.3.

Tabela C.4: Comparação dos *AUC* Médio do algoritmo Pareto Front Elite para os valores intermediários de parâmetros mínimos (equações 5.3 e 5.4) (apresentados na terceira coluna da Tabela 5.3) com os *AUC* médio dos trabalhos relacionados (apresentados na Tabela 5.2). A segunda coluna (resultados do PFE) é a base para a comparação com outros algoritmos utilizando o teste T. Células em cinza escuro indicam valores estatisticamente inferiores dos resultados do PFE, células em cinza claro indicam valores estatisticamente superiores e as células sem cores indicam valores similares.

#	**PFE (5.3)(5.4)**	**ROCCER**	**C4.5**	**C4.5NP**	**CN2**	**CN2OR**	**Ripper**	**Slipper**
1	99,08	98,63	97,76	98,39	99,26	99,13	98,72	99,24
2	68,08	65,30	62,14	57,44	62,74	62,21	69,10	59,84
3	90,87	90,31	50	90,06	90,17	85,15	61,86	74,78
4	70,83	61,83	50,00	68,68	53,22	42,78	45,28	52,35
5	74,95	72,08	71,43	67,71	75,25	70,90	64,02	71,32
6	82,24	79,45	50	81,50	73,74	79,64	49,75	50,00
7	58,54	66,41	55,84	64,33	59,83	59,28	57,45	50,40
8	88,77	85,78	84,81	81,11	83,61	82,25	84,89	84,03
9	96,93	94,18	86,09	90,91	96,23	92,18	92,06	93,95
12	98,55	98,4	87,85	97,5	99,14	98,43	94,95	99,12
13	99,80	97,85	99,42	99,74	100	99,99	99,43	94,40
14	71,27	70,68	72,07	72,60	70,96	71,97	68,07	70,02
16	97,96	96,42	94,76	96,99	97,38	96,49	95,01	93,99
Média	84,45	82,87	74,01	82,07	81,66	80,03	75,43	76,42

curo mostram, respectivamente, valores que são estatisticamente melhores ou piores do que o PFE com valores intermediários.

A Tabela C.5 é um resumo por base de dados dos valores estatisticamente diferentes destacados na Tabela C.4. A Tabela C.5 contém o número de valores de *AUC* do algoritmo Pareto Front Elite obtidos com os parâmetros intermediários que são superiores (segunda coluna), similares (terceira coluna) ou inferiores (quarta coluna) aos valores de *AUC* dos trabalhos relacionados para cada base de dados. Por sua vez, a Tabela C.6 contém um resumo do número de valores diferentes separados pelo trabalho relacionado. O número total de valores de *AUC* médios do PFE que é superior ao algoritmo da coluna correspondente está na segunda linha da Tabela, os totais de valores similares ao PFE estão na terceira linha e os totais de valores inferiores estão na última linha. A Figura C.2 representa os totais da Tabela C.6 em forma de gráfico.

Tabela C.5: Total de valores de *AUC* do algoritmo Pareto Front Elite que é superior (segunda coluna), similar (terceira coluna) ou inferior (quarta coluna) aos valores de *AUC* dos trabalhos relacionados. São considerados os resultados do PFE com os valores intermediários de parâmetros mínimos (equações 5.3 e 5.4) apresentados na terceira coluna da Tabela 5.3 e na segunda coluna da Tabela C.4 e os resultados dos trabalhos relacionados apresentados na Tabela 5.2 e Tabela C.4. A contagem é feita por base de dados, cujo número está na primeira coluna, e os totais gerais estão na última linha.

#	Superior	Similar	Inferior
1	1	6	0
2	5	2	0
3	3	4	0
4	5	2	0
5	5	2	0
6	4	3	0
7	1	5	1
8	5	2	0
9	5	2	0
12	1	6	0
13	4	1	2
14	0	7	0
16	6	1	0
Total	**45**	**43**	**3**

Tabela C.6: Total de bases de dados que o algoritmo Pareto Front Elite é superior (segunda linha), inferior (quarta linha) ou que não existe diferença (terceira linha) comparado com os resultados dos trabalhos relacionados. São considerados os resultados do PFE com os valores intermediários de parâmetros mínimos (equações 5.3 e 5.4) apresentados na terceira coluna da Tabela 5.3 e na segunda coluna da Tabela C.4 e os resultados dos trabalhos relacionados apresentados na Tabela 5.2 e Tabela C.4. A contagem é feita com base nas comparações da Tabela C.6.

Comparação	ROCCER	C4.5	C4.5NP	CN2	CN2OR	Ripper	Slipper	Total(%)
Superior	3	11	5	4	6	7	9	**45(0,49)**
Similar	9	2	8	8	6	6	4	**43(0,47)**
Inferior	1	0	0	1	1	0	0	**3(0,03)**

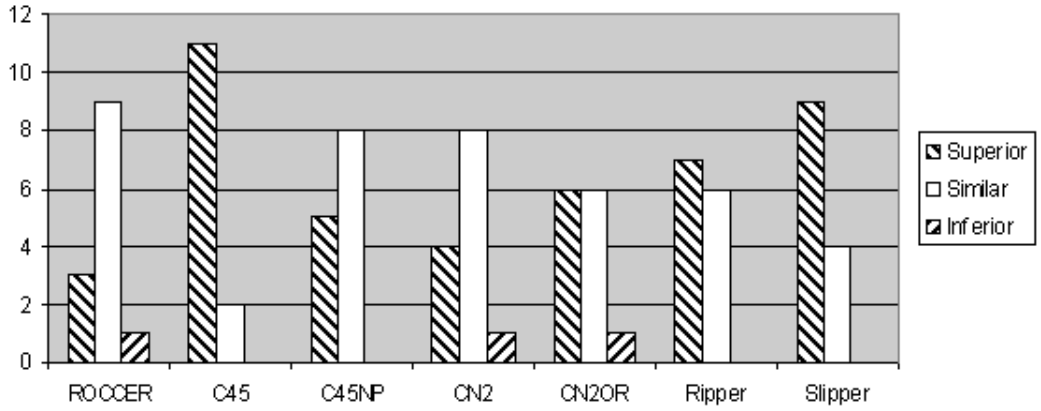


Figura C.2: Número de bases de dados que o PFE com os maiores valores de parâmetros mínimos (equações 5.3 e 5.4) é superior, similar ou inferior ao trabalho relacionado. Figura baseada nos dados da Tabela C.6.

C.3 Resumo do PFE sem os Parâmetros Mínimos

A seguir são apresentados alguns resumos dos resultados para a análise do desempenho do algoritmo Pareto Front Elite executado sem os valores para os parâmetros mínimos ($MinSupport_3$ para suporte mínimo (equação 5.5) e $MinConfidence_3$ para confiança mínima (equação 5.6)). São considerados os valores de *AUC* médios do PFE apresentados na quarta coluna da Tabela 5.3 e os resultados dos trabalhos relacionados (Tabela 5.2).

A Tabela C.7 compara cada valor de *AUC* médio obtido pelo PFE com os trabalhos relacionados, destacando os valores estatisticamente diferentes do PFE. A primeira coluna contém o número do conjunto de exemplos. A segunda coluna possui *AUC* médio e o desvio-padrão (entre parênteses) para cada conjunto de exemplo (resultados primeiramente apresentados na segunda coluna da Tabela 5.3). As demais colunas contêm os resultados dos trabalhos relacionados apresentados na Tabela 5.2. As células com as cores cinza claro ou escuro mostram, respectivamente, valores que são estatisticamente melhores ou piores do que o PFE sem os parâmetros mínimos.

A Tabela C.8 é um resumo por base de dados dos valores estatisticamente diferentes destacados na Tabela C.7. A Tabela C.8 contém o número de valores de *AUC* do algoritmo Pareto Front Elite, sem os parâmetros mínimos, que é superior (segunda coluna), similar (terceira coluna) ou inferior (quarta coluna) aos valores de *AUC* dos trabalhos relacionados para cada base de dados. A Tabela C.9 contém um resumo do número de valores diferentes separados pelo trabalho relacionado. O número total de valores de *AUC* médios do PFE que é superior ao algoritmo da coluna correspondente encontra-se na segunda linha da Tabela, os totais de valores

Tabela C.7: Comparação dos *AUC* Médio do algoritmo Pareto Front Elite obtidos sem valores para os parâmetros mínimos (equações 5.5 e 5.6) (apresentados na quarta coluna da Tabela 5.3) com os *AUC* médio dos trabalhos relacionados (apresentados na Tabela 5.2). A segunda coluna (resultados do PFE) é a base para a comparação com outros algoritmos utilizando o teste T. Células em cinza escuro indicam valores estatisticamente inferiores dos resultados do PFE e as células sem cores indicam valores similares.

#	PFE (5.5)(5.6)	ROCCER	C4.5	C4.5NP	CN2	CN2OR	Ripper	Slipper
1	99,39	98,63	97,76	98,39	99,26	99,13	98,72	99,24
2	65,40	65,3	62,14	57,44	62,74	62,21	69,1	59,84
3	92,22	90,31	50	90,06	90,17	85,15	61,86	74,78
5	75,86	72,08	71,43	67,71	75,25	70,9	64,02	71,32
6	80,86	79,45	50	81,5	73,74	79,64	49,75	50
7	59,65	66,41	55,84	64,33	59,83	59,28	57,45	50,4
8	88,61	85,78	84,81	81,11	83,61	82,25	84,89	84,03
12	97,75	98,4	87,85	97,5	99,14	98,43	94,95	99,12
13	100,00	97,85	99,42	99,74	100	99,99	99,43	94,4
14	72,01	70,68	72,07	72,6	70,96	71,97	68,07	70,02
Média	83,17	82,49	73,13	81,04	81,47	80,90	74,82	75,32

Tabela C.8: Total de valores de *AUC* do algoritmo Pareto Front Elite que é superior (segunda coluna), similar (terceira coluna) ou inferior (quarta coluna) aos valores de *AUC* dos trabalhos relacionados. São considerados os resultados do PFE sem valores para os parâmetros mínimos (equações 5.3 e 5.4) apresentados na quarta coluna da Tabela 5.3 e na segunda coluna da Tabela C.7 e os resultados dos trabalhos relacionados apresentados na Tabela 5.2 e Tabela C.7. A contagem é feita por base de dados, cujo número está na primeira coluna, e os totais gerais estão na última linha.

#	Superior	Similar	Inferior
1	2	5	0
2	0	7	0
3	3	4	0
5	3	4	0
6	3	4	0
7	1	6	0
8	2	5	0
12	1	6	0
13	6	1	0
14	0	7	0
Total	**21**	**49**	**0**

Tabela C.9: Total de bases de dados que o algoritmo Pareto Front Elite é superior (segunda linha), inferior (quarta linha) ou que não existe diferença (terceira linha) comparado com os resultados dos trabalhos relacionados. São considerados os resultados do PFE sem valores nos parâmetros (equações 5.5 e 5.6) apresentados na quarta coluna da Tabela 5.3 e na segunda coluna da Tabela C.7 e os resultados dos trabalhos relacionados apresentados na Tabela 5.2 e Tabela C.7. A contagem é feita com base nas comparações da Tabela C.7.

Comparação	**ROCCER**	**C4.5**	**C4.5NP**	**CN2**	**CN2OR**	**Ripper**	**Slipper**	**Total(%)**
Superior	1	5	4	0	3	4	4	21(0,30)
Similar	9	5	6	10	7	6	6	49(0,70)
Inferior	0	0	0	0	0	0	0	0(0,00)

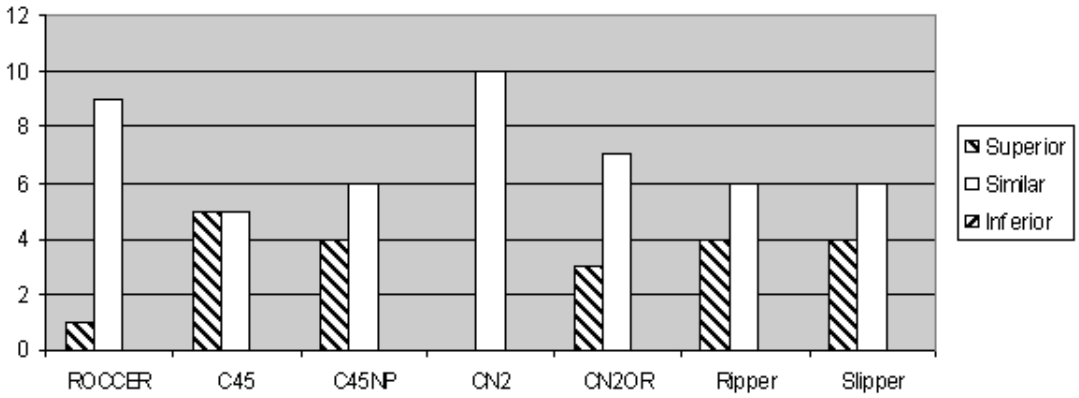


Figura C.3: Número de bases de dados que o PFE sem os valores de parâmetros mínimos (equações 5.5 e 5.6) é superior, similar ou inferior ao trabalho relacionado. Figura baseada nos dados da Tabela C.9.

similares ao PFE estão na terceira linha e os totais de valores inferiores estão na última linha. A Figura C.3 ilustra os totais da Tabela C.9 em forma de gráfico.

C.4 Resumo do MOPSO

A seguir são apresentados alguns resumos dos resultados para a análise do desempenho do algoritmo MOPSO. São considerados os valores de *AUC* médios do MOPSO apresentados na Tabela 5.5 e os resultados dos trabalhos relacionados (Tabela 5.2).

A Tabela C.11 é um resumo por base de dados dos valores estatisticamente diferentes destacados na Tabela C.10. A Tabela C.11 contém o número de valores de *AUC* do algoritmo MOPSO que é superior (segunda coluna), similar (terceira coluna) ou inferior (quarta coluna) aos valores de *AUC* dos trabalhos relacionados para cada base de dados.

Tabela C.10: Comparação pelo teste T dos *AUC* Médio do algoritmo *MOPSO* (segunda coluna da Tabela 5.5) com os *AUC* médio dos trabalhos relacionados (Tabela 5.2). Células em cinza escuro indicam valores estatisticamente inferiores dos resultados do *MOPSO* e células em cinza claro indicam valores estatisticamente superiores.

#	MOPSO	ROCCER	C4.5	C4.5NP	CN2	CN2OR	Ripper	Slipper
1	98,75	98,63	97,76	98,39	99,26	99,13	98,72	99,24
2	66,15	65,3	62,14	57,44	62,74	62,21	69,1	59,84
3	82,69	90,31	50	90,06	90,17	85,15	61,86	74,78
4	63,26	61,83	50	68,68	53,22	42,78	45,28	52,35
5	73,39	72,08	71,43	67,71	75,25	70,9	64,02	71,32
6	65,00	79,45	50	81,5	73,74	79,64	49,75	50
7	59,96	66,41	55,84	64,33	59,83	59,28	57,45	50,4
8	87,48	85,78	84,81	81,11	83,61	82,25	84,89	84,03
9	81,19	94,18	86,09	90,91	96,23	92,18	92,06	93,95
10	95,78	99,35	99,85	99,86	99,85	99,91	99,85	99,91
11	93,80	96,08	95,49	99,33	99,34	99,44	97,27	98,82
12	87,62	98,4	87,85	97,5	99,14	98,43	94,95	99,12
13	85,91	97,85	99,42	99,74	100	99,99	99,43	94,4
14	71,67	70,68	72,07	72,6	70,96	71,97	68,07	70,02
15	65,39	89,39	90,15	91,31	91,48	91,48	86,83	89,06
16	88,83	96,42	94,76	96,99	97,38	96,49	95,01	93,99
Med.	79,18	85,13	77,98	84,84	84,51	83,20	79,03	80,08

Tabela C.11: Total de valores de *AUC* do algoritmo *MOPSO* que é, segundo teste T, superior (segunda coluna), similar (terceira coluna) ou inferior (quarta coluna) aos valores de *AUC* dos trabalhos relacionados. São considerados os resultados do *MOPSO* apresentados na segunda coluna da Tabela 5.5 e os resultados dos trabalhos relacionados apresentados na Tabela 5.2 e Tabela C.10. A contagem é feita por base de dados, cujo número está na primeira coluna, e os totais gerais estão na última linha.

#	Superior	Similar	Inferior
1	1	4	2
2	2	5	0
3	2	2	3
4	4	3	0
5	2	5	0
6	3	1	3
7	3	3	1
8	3	4	0
9	0	1	6
10	0	0	7
11	0	0	7
12	0	2	5
13	0	0	7
14	0	7	0
15	0	0	7
16	0	0	7
Total	**20**	**37**	**55**

Tabela C.12: Comparação do *AUC* médios do *GPR-RL* para cada conjunto de exemplo. As cores de fundo da célula indicam diferença estatística com o algoritmo *GRASP-PR Rule Learning* com o parâmetro $maxLoop = 200$ (segunda coluna). Células em cinza escuro indicam valores melhores e as células sem cores indicam valores similares.

#	***GPR-RL* 200**	***GPR-RL* 300**
1	99,04	99,07
2	68,07	68,05
3	90,33	90,52
4	70,66	70,72
5	74,67	74,89
6	82,13	82,28
7	59,32	58,60
8	88,59	88,79
9	96,13	96,77
10	98,91	99,04
11	96,67	96,87
12	98,35	98,50
13	99,48	99,70
14	71,02	71,29
15	90,70	90,89
16	96,83	97,66
Média	**86,31**	**86,48**

C.5 Resumo do GRASP-PR Rule Learning

A seguir são apresentados alguns resumos dos resultados para a análise do desempenho do algoritmo *GRASP-PR Rule Learning*. São considerados os valores de *AUC* médios do *GPR-RL* apresentados na Tabela 5.8 e os resultados dos trabalhos relacionados (Tabela 5.2).

A Tabela C.12 indica visualmente as diferenças estatísticas, pelo teste T, dos valores de *AUC* médios do algoritmo *GRASP-PR Rule Learning* executado com o parâmetro *maxLoop* igual a 200 (segunda coluna) 300 (terceira coluna). A base de comparação é a execução do $maxLoop = 200$ e todos os resultados foram primeiramente apresentados na Tabela 5.8.

A Tabela C.13 compara cada valor de *AUC* médio obtido pelo *GPR-RL* executado com $maxLoop = 100$ e os trabalhos relacionados, destacando os valores estatisticamente diferentes do *GPR-RL*. A primeira coluna contém o número do conjunto de exemplos. A segunda coluna possui *AUC* médio e o desvio-padrão (entre parênteses) para cada conjunto de exemplo (resultados primeiramente apresentados na segunda coluna da Tabela 5.8). As demais colunas contêm os resultados dos trabalhos relacionados apresentados na Tabela 5.2. As células com as cores cinza claro ou escuro mostram, respectivamente, valores que são estatisticamente melhores ou piores do que o *GPR-RL* com $maxLoop = 100$.

Tabela C.13: Comparação dos *AUC* Médio do *GPR-RL* com $maxLoop = 100$ (apresentados na segunda coluna da Tabela 5.8) com os *AUC* médio dos trabalhos relacionados (apresentados na Tabela 5.2). A segunda coluna (resultados do *GPR-RL*) é a base para a comparação com outros algoritmos utilizando o teste T. As células com a cor cinza escuro indicam valores estatisticamente inferiores dos resultados do *GPR-RL*, as células com a cinza claro são valores superiores e células sem cores indicam valores similares.

#	***GPR-RL* 100**	**ROCCER**	**C4.5**	**C4.5NP**	**CN2**	**CN2OR**	**Ripper**	**Slipper**
1	98,75	98,63	97,76	98,39	99,26	99,13	98,72	99,24
2	67,99	65,3	62,14	57,44	57,44	62,21	69,1	59,84
3	90,27	90,31	50	90,06	90,17	85,15	61,86	74,78
4	69,15	61,83	50	68,68	53,22	42,78	45,28	52,35
5	74,30	72,08	71,43	67,71	75,25	70,9	64,02	71,32
6	82,29	79,45	50	81,5	73,74	79,64	49,75	50
7	59,27	66,41	55,84	64,33	59,83	59,28	57,45	50,4
8	88,08	85,78	84,81	81,11	83,61	82,25	84,89	84,03
9	94,43	94,18	86,09	90,91	96,23	92,18	92,06	93,95
10	98,50	99,35	99,85	99,86	99,85	99,91	99,85	99,91
11	95,89	96,08	95,49	99,33	99,34	99,44	97,27	98,82
12	98,11	98,4	87,85	97,5	99,14	98,43	94,95	99,12
13	98,90	97,85	99,42	99,74	100	99,99	99,43	94,4
14	71,18	70,68	72,07	72,6	70,96	71,97	68,07	70,02
15	89,95	89,39	90,15	91,31	91,48	91,48	86,83	89,06
16	94,24	96,42	94,76	96,99	97,38	96,49	95,01	93,99
Média	85,71	85,13	77,98	84,84	84,51	83,20	79,03	80,08

Tabela C.14: Comparação dos *AUC* Médio do *GPR-RL* com $maxLoop = 200$ (apresentados na terceira coluna da Tabela 5.8) com os *AUC* médio dos trabalhos relacionados (apresentados na Tabela 5.2). A segunda coluna (resultados do *GPR-RL*) é a base para a comparação com outros algoritmos utilizando o teste T. As células com a cor cinza escuro indicam valores estatisticamente inferiores dos resultados do *GPR-RL*, células com cinza claro são valores superiores e células sem cores indicam valores similares.

#	***GPR-RL* 200**	**ROCCER**	**C4.5**	**C4.5NP**	**CN2**	**CN2OR**	**Ripper**	**Slipper**
1	99,04	98,63	97,76	98,39	99,26	99,13	98,72	99,24
2	68,07	65,3	62,14	57,44	62,74	62,21	69,1	59,84
3	90,33	90,31	50	90,06	90,17	85,15	61,86	74,78
4	70,66	61,83	50	68,68	53,22	42,78	45,28	52,35
5	74,67	72,08	71,43	67,71	75,25	70,9	64,02	71,32
6	82,13	79,45	50	81,5	73,74	79,64	49,75	50
7	59,32	66,41	55,84	64,33	59,83	59,28	57,45	50,4
8	88,59	85,78	84,81	81,11	83,61	82,25	84,89	84,03
9	96,13	94,18	86,09	90,91	96,23	92,18	92,06	93,95
10	98,91	99,35	99,85	99,86	99,85	99,91	99,85	99,91
11	96,67	96,08	95,49	99,33	99,34	99,44	97,27	98,82
12	98,35	98,4	87,85	97,5	99,14	98,43	94,95	99,12
13	99,48	97,85	99,42	99,74	100	99,99	99,43	94,4
14	71,02	70,68	72,07	72,6	70,96	71,97	68,07	70,02
15	90,70	89,39	90,15	91,31	91,48	91,48	86,83	89,06
16	96,83	96,42	94,76	96,99	97,38	96,49	95,01	93,99
Média	86,31	85,13	77,98	84,84	84,51	83,20	79,03	80,08

A Tabela C.14 compara cada valor de *AUC* médio obtido pelo *GPR-RL* executado com $maxLoop = 200$ e os trabalhos relacionados, destacando os valores estatisticamente diferentes do *GPR-RL*. A primeira coluna contém o número do conjunto de exemplos. A segunda coluna possui *AUC* médio e o desvio-padrão (entre parênteses) para cada conjunto de exemplo (resultados primeiramente apresentados na terceira coluna da Tabela 5.8). As demais colunas contêm os resultados dos trabalhos relacionados apresentados na Tabela 5.2. As células com as cores cinza claro ou escuro mostram, respectivamente, valores que são estatisticamente melhores ou piores do que o *GPR-RL* com $maxLoop = 200$.

A Tabela C.15 compara cada valor de *AUC* médio obtido pelo *GPR-RL* executado com $maxLoop = 300$ e os trabalhos relacionados, destacando os valores estatisticamente diferentes do *GPR-RL*. A primeira coluna contém o número do conjunto de exemplos. A segunda coluna possui *AUC* médio e o desvio-padrão (entre parênteses) para cada conjunto de exemplo (resultados primeiramente apresentados na quarta coluna da Tabela 5.8). As demais colunas contêm os resultados dos trabalhos relacionados apresentados na Tabela 5.2. As células com as cores cinza claro ou escuro mostram, respectivamente, valores que são estatisticamente melhores ou

Tabela C.15: Comparação dos *AUC* Médio do *GPR-RL* com $maxLoop = 300$ (apresentados na quarta coluna da Tabela 5.8) com os *AUC* médio dos trabalhos relacionados (apresentados na Tabela 5.2). A segunda coluna (resultados do *GPR-RL*) é a base para a comparação com outros algoritmos utilizando o teste T. As células com a cor cinza escuro indicam valores estatisticamente inferiores dos resultados do *GPR-RL*, células com cinza claro são valores superiores e células sem cores indicam valores similares.

#	***GPR-RL* 300**	**ROCCER**	**C4.5**	**C4.5NP**	**CN2**	**CN2OR**	**Ripper**	**Slipper**
1	99,07	98,63	97,76	98,39	99,26	99,13	98,72	99,24
2	68,05	65,3	62,14	57,44	62,74	62,21	69,1	59,84
3	90,52	90,31	50	90,06	90,17	85,15	61,86	74,78
4	70,72	61,83	50	68,68	53,22	42,78	45,28	52,35
5	74,89	72,08	71,43	67,71	75,25	70,9	64,02	71,32
6	82,28	79,45	50	81,5	73,74	79,64	49,75	50
7	58,60	66,41	55,84	64,33	59,83	59,28	57,45	50,4
8	88,79	85,78	84,81	81,11	83,61	82,25	84,89	84,03
9	96,77	94,18	86,09	90,91	96,23	92,18	92,06	93,95
10	99,04	99,35	99,85	99,86	99,85	99,91	99,85	99,91
11	96,87	96,08	95,49	99,33	99,34	99,44	97,27	98,82
12	98,50	98,4	87,85	97,5	99,14	98,43	94,95	99,12
13	99,70	97,85	99,42	99,74	100	99,99	99,43	94,4
14	71,29	70,68	72,07	72,6	70,96	71,97	68,07	70,02
15	90,89	89,39	90,15	91,31	91,48	91,48	86,83	89,06
16	97,66	96,42	94,76	96,99	97,38	96,49	95,01	93,99
Média	86,48	85,13	77,98	84,84	84,51	83,20	79,03	80,08

piores do que o *GPR-RL* com $maxLoop = 300$.

APÊNDICE D – Avaliação multiobjetiva dos algoritmos

Nesta seção são apresentados os *p-value*'s das comparações dos conjuntos de aproximação contendo regras positivas ou negativas segundo os indicadores de hipervolume, epsilon e $R2$. Cada comparação utiliza um indicador específico e é feita entre dois algoritmos para todas as 10 partições da validação cruzada. Se o *p-value* é abaixo de $0,05$ então o algoritmo base apresenta melhor desempenho segundo o indicador avaliado.

D.1 PFE com todas as regras X *GPR-RL* maxLoop=100

Nestã seção são apresentados os resultados das comparações do algoritmo PFE com todas as regras e os conjuntos do *GRASP-PR Rule Learning* executado com o parâmetro $maxLoop = 100$. O resumo dos resultados apresentados nesta seção está na Tabela 5.16.

Resultados das comparações utilizando o indicador unário de hipervolume entre os conjuntos de aproximação do algoritmo PFE com todas as regras e os conjuntos do *GRASP-PR Rule Learning* executado com $maxLoop = 100$ em cada partição da validação cruzada encontram-se nas Tabelas D.1 e D.2, respectivamente, considerando regras positivas e negativas. Nestas Tabelas, é apresentado o *p-value* do teste de Kruskal-Wallis para a hipótese zero H0: algoritmo PFE sem parâmetros ser diferente do *GPR-RL*. Se o valor é abaixo de $0,05$ então o algoritmo PFE apresenta melhor desempenho segundo o indicador de hipervolume. A última linha contém os totais de partições que os conjuntos de aproximação do PFE têm melhor desempenho.

Os mesmo resultados dos *p-value*'s dos testes, porém, utilizando o indicador epsilon estão nas Tabelas D.3 e D.4. Os valores dos *p-value*'s para o indicador $R2$ estão nas Tabelas D.5 e D.6.

Tabela D.1: Resultado da comparação estatística utilizando o indicador de hipervolume dos conjuntos de aproximação formado pelas regras positivas do algoritmo Pareto Front Elite sem parâmetros com o algoritmo *GRASP-PR Rule Learning* com $maxLoop = 100$ para cada base de dados (a partir da segunda coluna) e cada partição (primeira coluna). Os *p-value*'s do teste de Kruskal-Wallis para a hipótese zero H0: algoritmo PFE sem parâmetros diferente do *GPR-RL*. A última linha contém o totais de partições que os conjuntos de aproximação do PFE tem melhor desempenho.

Cross	#1	#2	#3	#5	#6	#7	#8	#12	#13	#14
1	0,2539	0,0908	0,0748	0,0447	0,1372	0,2634	0,0447	0,500	0,0447	0,0447
2	0,2454	0,0755	0,0991	0,0447	0,2285	0,2195	0,0447	0,500	0,0480	0,0447
3	0,2630	0,0714	0,101	0,0447	0,1186	0,3015	0,0480	0,500	0,0447	0,0447
4	0,2630	0,0509	0,1839	0,0447	0,066	0,2196	0,0447	0,500	0,0447	0,0447
5	0,2817	0,0581	0,0297	0,0447	0,1282	0,2916	0,0447	0,500	0,0480	0,0447
6	0,2189	0,1255	0,1485	0,0447	0,1105	0,3455	0,0478	0,500	0,0480	0,0447
7	0,2539	0,1072	0,0448	0,0447	0,0396	0,1875	0,0514	0,500	0,0515	0,0447
8	0,1873	0,0853	0,0458	0,0447	0,1657	0,3017	0,0447	0,500	0,0515	0,0447
9	0,2191	0,1248	0,1627	0,0447	0,1006	0,2814	0,0447	0,500	0,0447	0,0447
10	0,1410	0,0855	0,0493	0,0447	0,0850	0,2817	0,0480	0,4438	0,0480	0,0447
Total	0	0	4	10	1	0	9	0	8	10

Tabela D.2: Resultado da comparação estatística utilizando o indicador de hipervolume dos conjuntos de aproximação formado pelas regras negativas do algoritmo Pareto Front Elite sem parâmetros com o algoritmo *GRASP-PR Rule Learning* com $maxLoop = 100$ para cada base de dados (a partir da segunda coluna) e cada partição (primeira coluna). Os *p-value*'s do teste de Kruskal-Wallis para a hipótese zero H0: algoritmo PFE sem parâmetros diferente do *GPR-RL*. A última linha contém o totais de partições que os conjuntos de aproximação do PFE tem melhor desempenho.

Cross	#1	#2	#3	#5	#6	#7	#8	#12	#13	#14
1	0,1518	0,0469	0,1581	0,0447	0,0446	0,2075	0,0447	0,5000	0,4438	0,0447
2	0,2455	0,0455	0,2177	0,0447	0,0475	0,3111	0,0447	0,4013	0,3336	0,0447
3	0,1381	0,0500	0,2540	0,0447	0,0425	0,3226	0,0447	0,4200	0,4199	0,0446
4	0,1310	0,0327	0,2025	0,0447	0,0436	0,3226	0,0447	0,5000	0,3577	0,0445
5	0,1191	0,0426	0,2025	0,0447	0,0547	0,3576	0,0447	0,4199	0,5000	0,0444
6	0,1111	0,0417	0,2452	0,0447	0,0470	0,3220	0,0447	0,4438	0,3575	0,0551
7	0,1119	0,0539	0,1872	0,0447	0,0445	0,3337	0,0447	0,4013	0,4200	0,0503
8	0,1072	0,0431	0,1880	0,0447	0,0445	0,2111	0,0447	0,4200	0,3454	0,0447
9	0,1014	0,0412	0,2036	0,0447	0,0445	0,0053	0,0447	0,4438	0,3709	0,0445
10	0,2915	0,0433	0,2363	0,0447	0,0442	0,3578	0,0447	0,4200	0,4438	0,0446
Total	0	9	0	10	9	1	10	0	0	8

Tabela D.3: Resultado da comparação estatística utilizando o indicador epsilon dos conjuntos de aproximação formado pelas regras positivas do algoritmo Pareto Front Elite sem parâmetros com o algoritmo *GRASP-PR Rule Learning* com $maxLoop = 100$ para cada base de dados (a partir da segunda coluna) e cada partição (primeira coluna). Os *p-value*'s do teste de Kruskal-Wallis para a hipótese zero H0: algoritmo PFE sem parâmetros diferente do *GPR-RL*. A última linha contém o totais de partições que os conjuntos de aproximação do PFE tem melhor desempenho.

Cross	#1	#2	#3	#5	#6	#7	#8	#12	#13	#14
1	0,2524	0,0882	0,0669	0,0444	0,1372	0,2633	0,0423	0,5000	0,0426	0,0420
2	0,2446	0,0737	0,0943	0,0445	0,2278	0,2193	0,0398	0,5000	0,0473	0,0424
3	0,2618	0,0692	0,0947	0,0444	0,1168	0,3015	0,0405	0,5000	0,0434	0,0424
4	0,2618	0,0450	0,1109	0,0429	0,0487	0,2194	0,0433	0,5000	0,0423	0,0357
5	0,2804	0,0553	0,0128	0,0443	0,1230	0,2914	0,0421	0,5000	0,0464	0,0416
6	0,2146	0,1238	0,1458	0,0445	0,1075	0,3454	0,0447	0,5000	0,0452	0,0440
7	0,2532	0,1059	0,0115	0,0445	0,0385	0,1864	0,0475	0,5000	0,0506	0,0418
8	0,1840	0,0839	0,0188	0,0474	0,1653	0,3015	0,0408	0,5000	0,0497	0,0403
9	0,2163	0,1241	0,1624	0,0435	0,0961	0,2814	0,0414	0,5000	0,0434	0,0415
10	0,1291	0,0635	0,0435	0,0445	0,0822	0,2817	0,0446	0,4438	0,0464	0,0424
Total	0	1	4	10	2	0	10	0	9	10

Tabela D.4: Resultado da comparação estatística utilizando o indicador epsilon dos conjuntos de aproximação formado pelas regras negativas do algoritmo Pareto Front Elite sem parâmetros com o algoritmo *GRASP-PR Rule Learning* com $maxLoop = 100$ para cada base de dados (a partir da segunda coluna) e cada partição (primeira coluna). Os *p-value*'s do teste de Kruskal-Wallis para a hipótese zero H0: algoritmo PFE sem parâmetros diferente do *GPR-RL*. A última linha contém o totais de partições que os conjuntos de aproximação do PFE tem melhor desempenho.

Cross	#1	#2	#3	#5	#6	#7	#8	#12	#13	#14
1	0,1426	0,0409	0,0911	0,0443	0,0381	0,2075	0,0370	0,290073	0,443769	0,040852
2	0,2452	0,0446	0,1350	0,0436	0,0325	0,3111	0,0408	0,253509	0,333114	0,042742
3	0,1351	0,0492	0,1411	0,0442	0,0417	0,3226	0,0342	0,262503	0,419946	0,040482
4	0,1273	0,0362	0,1138	0,0436	0,0420	0,3224	0,0357	0,21628	0,3575	0,035963
5	0,1163	0,0433	0,0946	0,0443	0,0490	0,3576	0,0354	0,280416	0,5	0,042394
6	0,1036	0,0412	0,1374	0,0437	0,0452	0,3220	0,0397	0,322535	0,3575	0,051867
7	0,1108	0,0389	0,0871	0,0432	0,0214	0,3337	0,0379	0,244373	0,419946	0,047494
8	0,1034	0,0393	0,0981	0,0441	0,0389	0,2109	0,0374	0,253594	0,344997	0,04099
9	0,0995	0,0414	0,1306	0,0441	0,0404	0,3338	0,0091	0,243941	0,370923	0,041208
10	0,2915	0,0394	0,1348	0,0428	0,0214	0,3577	0,0354	0,271427	0,443769	0,042042
Total	0	10	0	10	10	0	10	0	0	9

Tabela D.5: Resultado da comparação estatística utilizando o indicador *R*2 dos conjuntos de aproximação formado pelas regras positivas do algoritmo Pareto Front Elite sem parâmetros com o algoritmo *GRASP-PR Rule Learning* com $maxLoop = 100$ para cada base de dados (a partir da segunda coluna) e cada partição (primeira coluna). Os *p-value*'s do teste de Kruskal-Wallis para a hipótese zero H0: algoritmo PFE sem parâmetros diferente do *GPR-RL*. A última linha contém o totais de partições que os conjuntos de aproximação do PFE tem melhor desempenho.

Cross	#1	#2	#3	#5	#6	#7	#8	#12	#13	#14
1	0,2539	0,1873	0,1055	0,0514	0,4438	0,3709	0,1246	0,5	0,112904	0,270525
2	0,2454	0,0688	0,1153	0,0320	0,3708	0,3220	0,0992	0,5	0,131048	0,027179
3	0,2630	0,0862	0,1260	0,0411	0,2014	0,3708	0,0880	0,5	0,084259	0,017724
4	0,2630	0,1859	0,1176	0,0515	0,4199	0,3008	0,1176	0,5	0,113077	0,249481
5	0,2817	0,0102	0,0229	0,0499	0,3708	0,3575	0,0935	0,5	0,095024	0,009268
6	0,2189	0,1699	0,3000	0,0494	0,4199	0,3852	0,2917	0,5	0,107867	0,202721
7	0,2539	0,2353	0,0433	0,0401	0,5000	0,2705	0,0922	0,5	0,158381	0,000135
8	0,1873	0,1311	0,0464	0,0567	0,3708	0,4013	0,0993	0,5	0,120122	0,000498
9	0,2191	0,1424	0,1604	0,0405	0,4199	0,3853	0,0836	0,5	0,124149	0,133481
10	0,1410	0,1057	0,3220	0,0394	0,1099	0,4438	0,1362	0,443769	0,183156	0,234998
Total	0	1	3	7	0	0	0	0	0	5

Tabela D.6: Resultado da comparação estatística utilizando o indicador *R*2 dos conjuntos de aproximação formado pelas regras negativas do algoritmo Pareto Front Elite sem parâmetros com o algoritmo *GRASP-PR Rule Learning* com $maxLoop = 100$ para cada base de dados (a partir da segunda coluna) e cada partição (primeira coluna). Os *p-value*'s do teste de Kruskal-Wallis para a hipótese zero H0: algoritmo PFE sem parâmetros diferente do *GPR-RL*. A última linha contém o totais de partições que os conjuntos de aproximação do PFE tem melhor desempenho.

Cross	#1	#2	#3	#5	#6	#7	#8	#12	#13	#14
1	0,2361	0,0370	0,1682	0,0862	0,2632	0,2301	0,0298	0,2901	0,4438	0,0878
2	0,3455	0,0148	0,1770	0,0512	0,2449	0,4438	0,0309	0,2622	0,3336	0,0896
3	0,3118	0,0093	0,1950	0,0585	0,2613	0,5000	0,0473	0,2714	0,4199	0,0417
4	0,3016	0,0186	0,1636	0,0666	0,2543	0,5000	0,0477	0,2163	0,3577	0,0564
5	0,3578	0,0093	0,1438	0,0513	0,2020	0,5000	0,0366	0,3000	0,5000	0,1246
6	0,3338	0,0157	0,1951	0,0669	0,1938	0,5000	0,0250	0,3224	0,3575	0,0856
7	0,2915	0,0374	0,1908	0,0869	0,2265	0,5000	0,0441	0,2714	0,4200	0,2197
8	0,3578	0,0521	0,1747	0,0444	0,2177	0,5000	0,0362	0,2626	0,3454	0,0783
9	0,2722	0,0288	0,2100	0,1308	0,2104	0,4013	0,0504	0,2439	0,3709	0,1298
10	0,3710	0,0356	0,1564	0,0549	0,2804	0,5000	0,0770	0,2804	0,4438	0,0983
Total	0	9	0	1	0	0	8	0	0	1

Tabela D.7: Resultado da comparação estatística utilizando o indicador de hipervolume dos conjuntos de aproximação formado pelas regras positivas do algoritmo Pareto Front Elite sem parâmetros com o algoritmo *GRASP-PR Rule Learning* com $maxLoop = 200$ para cada base de dados (a partir da segunda coluna) e cada partição (primeira coluna). Os p-value's do teste de Kruskal-Wallis para a hipótese zero H0: algoritmo PFE sem parâmetros diferente do *GPR-RL*. A última linha contém o totais de partições que os conjuntos de aproximação do PFE tem melhor desempenho.

Cross	#1	#2	#3	#5	#6	#7	#8	#12	#13	#14
1	0,3710	0,1953	0,2015	0,0447	0,2495	0,4013	0,0445	0,5000	0,0672	0,0446
2	0,3710	0,2036	0,1717	0,0447	0,3853	0,4438	0,0447	0,5000	0,0669	0,0447
3	0,3577	0,1654	0,2104	0,0447	0,3225	0,4199	0,0807	0,5000	0,0627	0,0447
4	0,3452	0,1429	0,3575	0,0447	0,1746	0,4013	0,0446	0,5000	0,0551	0,0444
5	0,3852	0,0910	0,0002	0,0447	0,2799	0,3576	0,0438	0,5000	0,0589	0,0441
6	0,4438	0,2540	0,3223	0,0447	0,2274	0,5000	0,0492	0,5000	0,1020	0,0445
7	0,4199	0,2725	0,0284	0,0447	0,0409	0,4438	0,0858	0,5000	0,1135	0,0445
8	0,3453	0,1727	0,0234	0,0871	0,4438	0,4438	0,0447	0,5000	0,0590	0,0446
9	0,3852	0,1951	0,3709	0,0447	0,2818	0,4438	0,1134	0,5000	0,0630	0,0446
10	0,3709	0,2032	0,1470	0,0447	0,1861	0,4438	0,0963	0,5000	0,0669	0,0447
Total	0	0	3	9	1	0	6	0	0	10

D.2 PFE com todas as regras X *GPR-RL* maxLoop=200

Nestã seção são apresentados os resultados das comparações do algoritmo PFE com todas as regras e os conjuntos do *GRASP-PR Rule Learning* executado com $maxLoop = 200$. O resumo dos resultados apresentados nesta seção está na Tabela 5.17.

Resultados das comparações utilizando o indicador unário de hipervolume entre os conjuntos de aproximação do algoritmo com todas as regras e os conjuntos do *GRASP-PR Rule Learning* executado com $maxLoop = 200$ em cada partição da validação cruzada encontram-se nas Tabelas D.7 e D.8, respectivamente, considerando regras positivas e negativas. Nestas Tabelas, é apresentado o *p-value* do teste de Kruskal-Wallis para a hipótese zero H0: algoritmo PFE sem parâmetros ser diferente do *GPR-RL*. Se o valor é abaixo de $0,05$ então o algoritmo PFE apresenta melhor desempenho segundo o indicador de hipervolume. A última linha contém os totais de partições que os conjuntos de aproximação do PFE têm melhor desempenho.

Os mesmo resultados dos *p-value*'s dos testes, porém, utilizando o indicador epsilon estão nas Tabelas D.9 e D.10. Os valores dos *p-value*'s para o indicador $R2$ estão nas Tabelas D.11 e D.12.

Tabela D.8: Resultado da comparação estatística utilizando o indicador de hipervolume dos conjuntos de aproximação formado pelas regras negativas do algoritmo Pareto Front Elite sem parâmetros com o algoritmo *GRASP-PR Rule Learning* com $maxLoop = 200$ para cada base de dados (a partir da segunda coluna) e cada partição (primeira coluna). Os *p-value*'s do teste de Kruskal-Wallis para a hipótese zero H0: algoritmo PFE sem parâmetros diferente do *GPR-RL*. A última linha contém o totais de partições que os conjuntos de aproximação do PFE tem melhor desempenho.

Cross	#1	#2	#3	#5	#6	#7	#8	#12	#13	#14
1	0,2424	0,0628	0,3576	0,0447	0,0429	0,4438	0,0446	0,5000	0,5000	0,0660
2	0,3854	0,0407	0,4013	0,0447	0,0559	0,5000	0,0477	0,5000	0,4438	0,0580
3	0,3008	0,0431	0,4013	0,0447	0,0282	0,4200	0,0447	0,5000	0,4438	0,0465
4	0,2618	0,0287	0,4199	0,0447	0,0267	0,4200	0,0446	0,5000	0,4199	0,0802
5	0,3854	0,0356	0,4200	0,0447	0,1066	0,5000	0,0471	0,5000	0,5000	0,0516
6	0,3223	0,0192	0,4438	0,0447	0,0513	0,4438	0,0446	0,5000	0,4438	0,0712
7	0,2804	0,1108	0,3710	0,0447	0,0402	0,5000	0,0476	0,5000	0,5000	0,1120
8	0,3452	0,0275	0,3854	0,0447	0,0445	0,4013	0,0441	0,5000	0,5000	0,0587
9	0,2524	0,0132	0,3710	0,0447	0,0510	0,0000	0,0447	0,4438	0,5000	0,0668
10	0,3854	0,0231	0,3576	0,0447	0,0299	0,5000	0,0447	0,5000	0,4438	0,0618
Total	0	8	0	10	6	1	10	0	0	1

Tabela D.9: Resultado da comparação estatística utilizando o indicador epsilon dos conjuntos de aproximação formado pelas regras positivas do algoritmo Pareto Front Elite sem parâmetros com o algoritmo *GRASP-PR Rule Learning* com $maxLoop = 200$ para cada base de dados (a partir da segunda coluna) e cada partição (primeira coluna). Os *p-value*'s do teste de Kruskal-Wallis para a hipótese zero H0: algoritmo PFE sem parâmetros diferente do *GPR-RL*. A última linha contém o totais de partições que os conjuntos de aproximação do PFE tem melhor desempenho.

Cross	#1	#2	#3	#5	#6	#7	#8	#12	#13	#14
1	0,3708	0,1931	0,2005	0,0502	0,2495	0,4013	0,0408	0,5000	0,0663	0,0213
2	0,3708	0,2031	0,1659	0,0315	0,3852	0,4438	0,0351	0,5000	0,0639	0,0427
3	0,3575	0,1604	0,2088	0,0431	0,3225	0,4199	0,0765	0,5000	0,0623	0,0295
4	0,3450	0,1409	0,3225	0,0441	0,1656	0,4013	0,0429	0,5000	0,0529	0,0348
5	0,3852	0,0910	0,0002	0,0811	0,2793	0,3576	0,0359	0,5000	0,0578	0,0174
6	0,4438	0,2536	0,3223	0,0540	0,2265	0,5000	0,0462	0,5000	0,1011	0,0336
7	0,4199	0,2724	0,0052	0,0910	0,0409	0,4438	0,0819	0,5000	0,1130	0,0365
8	0,3452	0,1698	0,0006	0,2916	0,4438	0,4438	0,0387	0,5000	0,0570	0,0426
9	0,3852	0,1944	0,3709	0,0805	0,2808	0,4438	0,1122	0,5000	0,0620	0,0417
10	0,3708	0,1718	0,1470	0,0715	0,1853	0,4438	0,0923	0,5000	0,0585	0,0409
Total	0	0	3	3	1	0	6	0	0	10

Tabela D.10: Resultado da comparação estatística utilizando o indicador epsilon dos conjuntos de aproximação formado pelas regras negativas do algoritmo Pareto Front Elite sem parâmetros com o algoritmo *GRASP-PR Rule Learning* com $maxLoop = 200$ para cada base de dados (a partir da segunda coluna) e cada partição (primeira coluna). Os *p-value*'s do teste de Kruskal-Wallis para a hipótese zero H0: algoritmo PFE sem parâmetros diferente do *GPR-RL*. A última linha contém o totais de partições que os conjuntos de aproximação do PFE tem melhor desempenho.

Cross	#1	#2	#3	#5	#6	#7	#8	#12	#13	#14
1	0,2398	0,0443	0,2914	0,0432	0,0232	0,4438	0,0386	0,3852	0,5000	0,0587
2	0,3853	0,0397	0,2608	0,0414	0,0106	0,5000	0,0398	0,3708	0,4438	0,0311
3	0,3004	0,0431	0,2699	0,0433	0,0272	0,4200	0,0296	0,3852	0,4438	0,0145
4	0,2613	0,0287	0,2322	0,0430	0,0194	0,4200	0,0401	0,4013	0,4199	0,0726
5	0,3854	0,0355	0,3220	0,0434	0,0913	0,5000	0,0376	0,3452	0,5000	0,0454
6	0,3223	0,0036	0,2714	0,0429	0,0486	0,4438	0,0398	0,4013	0,4438	0,0632
7	0,2804	0,1043	0,2529	0,0412	0,0012	0,5000	0,0433	0,4200	0,5000	0,1094
8	0,3450	0,0090	0,2238	0,0497	0,0103	0,4013	0,0266	0,3708	0,5000	0,0515
9	0,2518	0,0158	0,2717	0,0425	0,0106	0,4199	0,0303	0,3335	0,5000	0,0511
10	0,3853	0,0193	0,2625	0,0436	0,0000	0,5000	0,0407	0,3708	0,4438	0,0563
Total	0	9	0	10	9	0	10	0	0	3

Tabela D.11: Resultado da comparação estatística utilizando o indicador $R2$ dos conjuntos de aproximação formado pelas regras positivas do algoritmo Pareto Front Elite sem parâmetros com o algoritmo *GRASP-PR Rule Learning* com $maxLoop = 200$ para cada base de dados (a partir da segunda coluna) e cada partição (primeira coluna). Os *p-value*'s do teste de Kruskal-Wallis para a hipótese zero H0: algoritmo PFE sem parâmetros diferente do *GPR-RL*. A última linha contém o totais de partições que os conjuntos de aproximação do PFE tem melhor desempenho.

Cross	#1	#2	#3	#5	#6	#7	#8	#12	#13	#14
1	0,3710	0,3331	0,2601	0,0360	0,5000	0,4438	0,1874	0,5000	0,1475	0,5000
2	0,3710	0,2109	0,2009	0,0306	0,5000	0,5000	0,2105	0,5000	0,3107	0,0192
3	0,3577	0,2495	0,3334	0,0500	0,3575	0,5000	0,2915	0,5000	0,2022	0,1066
4	0,3452	0,2896	0,3225	0,1093	0,5000	0,4438	0,2111	0,5000	0,2065	0,3331
5	0,3852	0,1066	0,0002	0,1060	0,5000	0,3708	0,1946	0,5000	0,2896	0,0000
6	0,4438	0,3000	0,4199	0,0513	0,4438	0,5000	0,4438	0,5000	0,2276	0,3453
7	0,4199	0,3578	0,0269	0,0469	0,5000	0,5000	0,2055	0,5000	0,3223	0,0000
8	0,3453	0,2398	0,0132	0,0650	0,5000	0,4438	0,2116	0,5000	0,1369	0,0000
9	0,3852	0,2205	0,3709	0,0409	0,4438	0,5000	0,2632	0,5000	0,1317	0,3000
10	0,3709	0,1860	0,4438	0,0138	0,2495	0,5000	0,2917	0,5000	0,2912	0,3450
Total	0	0	3	6	0	0	0	0	0	4

Tabela D.12: Resultado da comparação estatística utilizando o indicador *R*2 dos conjuntos de aproximação formado pelas regras negativas do algoritmo Pareto Front Elite sem parâmetros com o algoritmo *GRASP-PR Rule Learning* com $maxLoop = 200$ para cada base de dados (a partir da segunda coluna) e cada partição (primeira coluna). Os *p-value*'s do teste de Kruskal-Wallis para a hipótese zero H0: algoritmo PFE sem parâmetros diferente do *GPR-RL*. A última linha contém o totais de partições que os conjuntos de aproximação do PFE tem melhor desempenho.

Cross	#1	#2	#3	#5	#6	#7	#8	#12	#13	#14
1	0,4013	0,0631	0,3013	0,3150	0,4438	0,4438	0,0416	0,3852	0,5000	0,1359
2	0,5000	0,0012	0,2608	0,1732	0,3452	0,5000	0,0525	0,3708	0,4438	0,1789
3	0,4438	0,0357	0,2793	0,3913	0,3852	0,5000	0,0605	0,3852	0,4438	0,0426
4	0,4438	0,0401	0,2322	0,0996	0,4200	0,5000	0,0522	0,4013	0,4199	0,1186
5	0,4438	0,0154	0,3331	0,0758	0,3576	0,5000	0,0313	0,3452	0,5000	0,2630
6	0,4438	0,0022	0,2808	0,2807	0,4013	0,5000	0,0424	0,4013	0,4438	0,2451
7	0,4438	0,1356	0,2804	0,1605	0,3709	0,5000	0,0438	0,4200	0,5000	0,3708
8	0,5000	0,0089	0,2495	0,0438	0,3453	0,5000	0,0400	0,3708	0,5000	0,1561
9	0,4438	0,0166	0,3008	0,2918	0,3450	0,5000	0,0522	0,3335	0,5000	0,1933
10	0,4438	0,0409	0,2908	0,1534	0,4013	0,5000	0,1174	0,3708	0,4438	0,2010
Total	0	8	0	1	0	0	5	0	0	1

D.3 PFE com todas as regras X *GPR-RL* maxLoop=300

Nestã seção são apresentados os resultados das comparações do algoritmo PFE com todas as regras e os conjuntos do *GRASP-PR Rule Learning* executado com o parâmetro $maxLoop = 300$. O resumo dos resultados apresentados nesta seção está na Tabela 5.18.

Resultados das comparações utilizando o indicador unário de hipervolume entre os conjuntos de aproximação do algoritmo com todas as regras e os conjuntos do *GRASP-PR Rule Learning* executado com $maxLoop = 300$ em cada partição da validação cruzada encontram-se nas Tabelas D.13 e D.14, respectivamente, considerando regras positivas e negativas. Nestas Tabelas, é apresentado o *p-value* do teste de Kruskal-Wallis para a hipótese zero H0: algoritmo PFE sem parâmetros ser diferente do *GPR-RL*. Se o valor é abaixo de $0,05$ então o algoritmo PFE apresenta melhor desempenho segundo o indicador de hipervolume. A última linha contém os totais de partições que os conjuntos de aproximação do PFE têm melhor desempenho.

Os mesmo resultados dos *p-value*'s dos testes, porém, utilizando o indicador epsilon estão nas Tabelas D.15 e D.16. Os valores dos *p-value*'s para o indicador *R*2 estão nas Tabelas D.17 e D.18.

Tabela D.13: Resultado da comparação estatística utilizando o indicador de hipervolume dos conjuntos de aproximação formado pelas regras positivas do algoritmo Pareto Front Elite sem parâmetros com o algoritmo *GRASP-PR Rule Learning* com $maxLoop = 300$ para cada base de dados (a partir da segunda coluna) e cada partição (primeira coluna). Os *p-value*'s do teste de Kruskal-Wallis para a hipótese zero H0: algoritmo PFE sem parâmetros diferente do *GPR-RL*. A última linha contém o totais de partições que os conjuntos de aproximação do PFE tem melhor desempenho.

Cross	#1	#2	#3	#5	#6	#7	#8	#12	#13	#14
1	0,4400	0,2800	0,2800	0,0500	0,3300	0,5000	0,0500	0,5000	0,1200	0,0500
2	0,4400	0,2700	0,3000	0,0400	0,4400	0,5000	0,0500	0,5000	0,0900	0,0400
3	0,4200	0,2900	0,3900	0,0400	0,4000	0,5000	0,1300	0,5000	0,1600	0,0400
4	0,4400	0,2700	0,3900	0,0600	0,2400	0,5000	0,0500	0,5000	0,0900	0,0400
5	0,5000	0,1300	0,0000	0,0400	0,3300	0,4400	0,0600	0,5000	0,1500	0,0400
6	0,4400	0,3500	0,3500	0,0400	0,3900	0,5000	0,0800	0,5000	0,2500	0,0400
7	0,4200	0,3500	0,0200	0,0500	0,0300	0,5000	0,1200	0,5000	0,2100	0,0400
8	0,4400	0,2700	0,0400	0,2300	0,5000	0,5000	0,0500	0,5000	0,1700	0,0400
9	0,4200	0,3300	0,5000	0,0400	0,3600	0,5000	0,2000	0,5000	0,1200	0,0500
10	0,4400	0,2600	0,2500	0,1000	0,2500	0,5000	0,1500	0,5000	0,1500	0,0500
Total	0	0	3	5	1	0	0	0	0	7

Tabela D.14: Resultado da comparação estatística utilizando o indicador de hipervolume dos conjuntos de aproximação formado pelas regras negativas do algoritmo Pareto Front Elite sem parâmetros com o algoritmo *GRASP-PR Rule Learning* com $maxLoop = 300$ para cada base de dados (a partir da segunda coluna) e cada partição (primeira coluna). Os *p-value*'s do teste de Kruskal-Wallis para a hipótese zero H0: algoritmo PFE sem parâmetros diferente do *GPR-RL*. A última linha contém o totais de partições que os conjuntos de aproximação do PFE tem melhor desempenho.

Cross	#1	#2	#3	#5	#6	#7	#8	#12	#13	#14
1	0,3900	0,0900	0,4400	0,0400	0,0300	0,4000	0,0400	0,5000	0,5000	0,0900
2	0,5000	0,0200	0,4400	0,0400	0,0400	0,5000	0,0500	0,5000	0,5000	0,1000
3	0,2900	0,0600	0,4400	0,0500	0,0100	0,5000	0,0400	0,5000	0,5000	0,0400
4	0,4200	0,0100	0,5000	0,0700	0,0100	0,5000	0,0400	0,5000	0,5000	0,1000
5	0,3900	0,0200	0,5000	0,0400	0,1400	0,5000	0,0400	0,5000	0,5000	0,1300
6	0,3100	0,0200	0,4400	0,0400	0,0300	0,5000	0,0400	0,5000	0,4400	0,1700
7	0,3300	0,1500	0,5000	0,0400	0,0200	0,5000	0,0500	0,5000	0,5000	0,2100
8	0,3600	0,0300	0,5000	0,0400	0,0100	0,5000	0,0400	0,5000	0,5000	0,1000
9	0,3700	0,0000	0,4200	0,0500	0,0400	0,0000	0,0400	0,5000	0,5000	0,1300
10	0,5000	0,0300	0,4400	0,0400	0,0100	0,5000	0,0500	0,5000	0,5000	0,1100
Total	0	7	0	7	9	1	7	0	0	1

Tabela D.15: Resultado da comparação estatística utilizando o indicador epsilon dos conjuntos de aproximação formado pelas regras positivas do algoritmo Pareto Front Elite sem parâmetros com o algoritmo *GRASP-PR Rule Learning* com $maxLoop = 300$ para cada base de dados (a partir da segunda coluna) e cada partição (primeira coluna). Os *p-value*'s do teste de Kruskal-Wallis para a hipótese zero H0: algoritmo PFE sem parâmetros diferente do *GPR-RL*. A última linha contém o totais de partições que os conjuntos de aproximação do PFE tem melhor desempenho.

Cross	#1	#2	#3	#5	#6	#7	#8	#12	#13	#14
1	0,4400	0,2800	0,2800	0,2800	0,3300	0,5000	0,0400	0,5000	0,1200	0,0200
2	0,4400	0,2700	0,3000	0,0200	0,4400	0,5000	0,0200	0,5000	0,0800	0,0400
3	0,4200	0,2900	0,3900	0,0500	0,4000	0,5000	0,1200	0,5000	0,1600	0,0200
4	0,4400	0,2700	0,2900	0,0700	0,2400	0,5000	0,0400	0,5000	0,0800	0,0200
5	0,5000	0,1300	0,0000	0,0600	0,3300	0,4400	0,0400	0,5000	0,1400	0,0100
6	0,4400	0,3500	0,3500	0,1400	0,3900	0,5000	0,0800	0,5000	0,2500	0,0100
7	0,4200	0,3500	0,0100	0,1400	0,0300	0,5000	0,1100	0,5000	0,2100	0,0100
8	0,4400	0,2700	0,0300	0,5800	0,5000	0,5000	0,0300	0,5000	0,1700	0,0400
9	0,4200	0,3300	0,5000	0,1400	0,3600	0,5000	0,1900	0,5000	0,1200	0,0300
10	0,4400	0,2500	0,2500	0,4200	0,2500	0,5000	0,1400	0,5000	0,1400	0,0300
Total	0	0	3	1	1	0	5	0	0	10

Tabela D.16: Resultado da comparação estatística utilizando o indicador epsilon dos conjuntos de aproximação formado pelas regras negativas do algoritmo Pareto Front Elite sem parâmetros com o algoritmo *GRASP-PR Rule Learning* com $maxLoop = 300$ para cada base de dados (a partir da segunda coluna) e cada partição (primeira coluna). Os *p-value*'s do teste de Kruskal-Wallis para a hipótese zero H0: algoritmo PFE sem parâmetros diferente do *GPR-RL*. A última linha contém o totais de partições que os conjuntos de aproximação do PFE tem melhor desempenho.

Cross	#1	#2	#3	#5	#6	#7	#8	#12	#13	#14
1	0,3900	0,0700	0,3500	0,0400	0,0200	0,4000	0,0400	0,4400	0,5000	0,0800
2	0,5000	0,0200	0,3300	0,0400	0,0000	0,5000	0,0400	0,5000	0,5000	0,0600
3	0,2900	0,0600	0,3100	0,0400	0,0100	0,5000	0,0400	0,5000	0,5000	0,0000
4	0,4200	0,0100	0,3900	0,0800	0,0100	0,5000	0,0400	0,4400	0,5000	0,0800
5	0,3900	0,0200	0,3300	0,0400	0,1200	0,5000	0,0300	0,3700	0,5000	0,1200
6	0,3100	0,0100	0,3300	0,0400	0,0300	0,5000	0,0300	0,4400	0,4400	0,1600
7	0,3300	0,1400	0,3300	0,0500	0,0000	0,5000	0,0400	0,4400	0,5000	0,2100
8	0,3600	0,0100	0,3100	0,0800	0,0000	0,5000	0,0000	0,4200	0,5000	0,0900
9	0,3700	0,0000	0,3200	0,0400	0,0000	0,5000	0,0400	0,3900	0,5000	0,1100
10	0,5000	0,0200	0,3700	0,0400	0,0000	0,5000	0,0400	0,5000	0,5000	0,1100
Total	0	7	0	7	9	0	10	0	0	1

Tabela D.17: Resultado da comparação estatística utilizando o indicador *R*2 dos conjuntos de aproximação formado pelas regras positivas do algoritmo Pareto Front Elite sem parâmetros com o algoritmo *GRASP-PR Rule Learning* com $maxLoop = 300$ para cada base de dados (a partir da segunda coluna) e cada partição (primeira coluna). Os *p-value*'s do teste de Kruskal-Wallis para a hipótese zero H0: algoritmo PFE sem parâmetros diferente do *GPR-RL*. A última linha contém o totais de partições que os conjuntos de aproximação do PFE tem melhor desempenho.

Cross	#1	#2	#3	#5	#6	#7	#8	#12	#13	#14
1	0,4400	0,3700	0,3000	0,0300	0,5000	0,5000	0,2800	0,5000	0,2600	0,5000
2	0,4400	0,2900	0,3100	0,0400	0,5000	0,5000	0,3200	0,5000	0,3500	0,0100
3	0,4200	0,3000	0,4200	0,0400	0,4000	0,5000	0,2700	0,5000	0,3600	0,1100
4	0,4400	0,4200	0,2900	0,1200	0,5000	0,5000	0,2600	0,5000	0,2000	0,4200
5	0,5000	0,1200	0,0000	0,1000	0,5000	0,5000	0,2700	0,5000	0,3200	0,0100
6	0,4400	0,4000	0,4400	0,0600	0,5000	0,5000	0,5000	0,5000	0,3000	0,4000
7	0,4200	0,5000	0,0300	0,0400	0,5000	0,5000	0,2200	0,5000	0,4000	0,0000
8	0,4400	0,3200	0,0400	0,0800	0,5000	0,5000	0,3300	0,5000	0,2900	0,0000
9	0,4200	0,3300	0,5000	0,0400	0,5000	0,5000	0,2900	0,5000	0,2800	0,4200
10	0,4400	0,2500	0,5000	0,0000	0,3200	0,5000	0,3900	0,5000	0,3500	0,4400
Total	0	0	3	6	0	0	0	0	0	4

Tabela D.18: Resultado da comparação estatística utilizando o indicador *R*2 dos conjuntos de aproximação formado pelas regras negativas do algoritmo Pareto Front Elite sem parâmetros com o algoritmo *GRASP-PR Rule Learning* com $maxLoop = 300$ para cada base de dados (a partir da segunda coluna) e cada partição (primeira coluna). Os *p-value*'s do teste de Kruskal-Wallis para a hipótese zero H0: algoritmo PFE sem parâmetros diferente do *GPR-RL*. A última linha contém o totais de partições que os conjuntos de aproximação do PFE tem melhor desempenho.

Cross	#1	#2	#3	#5	#6	#7	#8	#12	#13	#14
1	0,5000	0,0800	0,3500	0,6400	0,4200	0,4000	0,0400	0,4400	0,5000	0,2300
2	0,5000	0,0000	0,3400	0,2300	0,4400	0,5000	0,0600	0,5000	0,5000	0,2500
3	0,5000	0,0400	0,3200	0,6400	0,4200	0,5000	0,0800	0,5000	0,5000	0,0200
4	0,5000	0,0400	0,3900	0,1600	0,3700	0,5000	0,0700	0,4400	0,5000	0,1100
5	0,5000	0,0200	0,3300	0,0900	0,3900	0,5000	0,0500	0,3700	0,5000	0,4000
6	0,5000	0,0100	0,3300	0,3500	0,4200	0,5000	0,0500	0,4400	0,4400	0,2100
7	0,5000	0,1600	0,3300	0,2500	0,4400	0,5000	0,0400	0,4400	0,5000	0,5000
8	0,5000	0,0400	0,3100	0,0600	0,3900	0,5000	0,0600	0,4200	0,5000	0,2200
9	0,5000	0,0000	0,3400	0,3900	0,4200	0,5000	0,0700	0,3900	0,5000	0,2600
10	0,5000	0,0800	0,3900	0,4000	0,4400	0,5000	0,2800	0,5000	0,5000	0,2400
Total	0	7	0	0	0	0	2	0	0	1